Advancing Model Explainability in Pervasive Computing

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

M.Sc. Yiran Huang

Tag der mündlichen Prüfung: 28. Nov. 2024

1. Referent: Prof. Dr. Michael Beigl
2. Referent: Prof. Dr. Oliver Amft

This document is licensed under a Creative Commons
Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0):
https://creativecommons.org/licenses/by-sa/4.0/deed.en

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Dr. Michael Beigl, for his invaluable guidance, encouragement, and unwavering support throughout my PhD journey. His expertise and dedication have been instrumental in shaping both my research and my professional development. I am also immensely grateful to my co-supervisor, Prof. Dr. Oliver Amft, for his insightful feedback and continuous support, which have greatly enriched my work.

I would like to extend my heartfelt thanks to Dr. Till Riedel, the head of the SDSC project, for providing me with the opportunity to participate in such a significant research initiative. His leadership and advice have been invaluable throughout the course of the project.

A special thank you goes to my office colleagues and friends, Dr. Haibin Zhao, Dr. Yexu Zhou and Chaofan Li. Our countless discussions, collaborations, and shared experiences have greatly contributed to the success of my research. I also wish to acknowledge Likun Fang, with whom I have had the pleasure of co-authoring several papers, for her collaborative spirit and insightful contributions.

I would also like to thank Alexander Studt and Ploypleарn Ravivanpong, with whom I worked on the SDSC project, for their helpful insights and teamwork. Long Wang and Dr. Michael Hefenbrock deserve special mention for their critical feedback on my writing, as well as for co-authoring a paper with me. Their mentorship has been invaluable.

I am deeply thankful to Bowen Wang, who introduced me to TECO and set me on this path, as well as to Ployplearn Ravivanpong, who offered me tremendous support during my early days at TECO. My sincere thanks also go to Melissa Alpman and Zina Tsiouma for their kind administrative support and assistance throughout my time at the institute.

On a more personal note, I would like to express my deepest gratitude to my wife, Qiuyu Gong, and my daughter, Keying Huang, for their endless love, patience, and encouragement. They have been my source of strength throughout this journey. I also wish to thank my parents, Junjie Huang and Ruirong Wang, for their unwavering support and belief in me.

Finally, I am grateful to all the colleagues and friends who have shared this journey with me. Without your support, this thesis would not have been possible.

Abstract

Pervasive computing technologies are increasingly integral to numerous domains, necessitating enhanced explainability due to their human-centered design. However, task difficulty and data complexity pose substantial challenges for Explainable Artificial Intelligence (XAI) methodologies. Particularly, the predictive performance of interpretable models is often limited within these domains. Furthermore, the inherent complexities of pervasive computing, characterized by noisy, high-volume, and temporally dependent data, exacerbate the difficulty in developing effective explanatory methods. Prevalent XAI approaches, especially those dependent on saliency maps, typically fail to adequately elucidate the underlying decision-making processes of complex models.

This dissertation aims to advance the field of XAI within pervasive computing by addressing these critical challenges. It is structured around three principal objectives: First, it seeks to improve the predictive performance of interpretable models through the introduction of an innovative automatic feature engineering framework, coupled with an optimization algorithm specifically designed for pervasive environments. Second, the dissertation conducts a comprehensive literature review and model architecture analysis to systematically identify key data elements leveraged by high-quality models in the domain. Third, utilizing these key data elements, it proposes novel post hoc explanation methods, including a tree-search approach for exploring model decisions and a Markov chain-based technique to elucidate temporal dynamics. These contributions collectively represent a significant advancement in the field of XAI for pervasive computing, facilitating the development of more effective and interpretable artificial intelligence systems within this complex domain.

Zusammenfassung

Pervasive computing-Technologien spielen in zahlreichen Branchen eine zunehmend wichtige Rolle und erfordern aufgrund ihres menschenzentrierten Designs verbesserte Erklärbarkeit. Die Komplexität domänenspezifischer Daten und Aufgaben stellt erhebliche Herausforderungen für Methoden der erklärlichen künstlichen Intelligenz (XAI) dar. Insbesondere ist die Vorhersageleistung interpretierbarer Modelle in diesen Domänen oft begrenzt. Darüber hinaus, die inhärenten Komplexitäten des Pervasive Computing, gekennzeichnet durch lärmende, umfangreiche und zeitabhängige Daten, verschärfen die Schwierigkeit, wirksame Erklärungsmethoden zu entwickeln. Die gängige XAI-Ansätze schaffen, insbesondere jene, die auf Saliency Maps basieren, typischerweise nicht, die zugrundeliegenden Entscheidungsprozesse ausreichend zu erhellen.

Diese Dissertation zielt darauf ab, das Feld der XAI innerhalb des Pervasive Computing voranzutreiben, indem diese kritischen Herausforderungen angegangen werden. Die Forschung ist um drei Hauptziele strukturiert: Erstens strebt sie danach, die Vorhersageleistung interpretierbarer Modelle durch die Einführung eines innovativen automatischen Feature-Engineering-Frameworks zu verbessern, ergänzt durch einen speziell für pervasive Umgebungen entwickelten Optimierungsalgorithmus. Zweitens führt die Arbeit eine umfassende Literaturrecherche und eine Analyse der Modellarchitektur durch, um systematisch Schlüsseldatenelemente zu identifizieren, die von qualitativ hochwertigen Modellen genutzt werden. Drittens schlägt sie, basierend auf diesen Schlüsseldatenelementen, neue post-hoc-Erklärungsmethoden vor, einschließlich eines Baumsuchverfahrens zur Erkundung von Modellentscheidungen und einer auf Markov-Ketten basierenden Technik zur Erläuterung zeitlicher Dynamiken. Diese Beiträge stellen insgesamt eine bedeutende Weiterentwicklung im Bereich der XAI für Pervasive Computing dar, die die En-

twicklung effektiverer und interpretierbarer KI-Systeme in diesem komplexen Bereich erleichtert.

Contents

1. Introduction **1**

1.1. Pervasive Computing and Explainable Artificial Intelligence . 1

1.1.1. Significance of Pervasive Computing 2

1.1.2. The Shift Towards Complex Models 3

1.1.3. The Role of Explainable Artificial Intelligence 4

1.2. Importance of Interpretable Models 7

1.3. Challenges and Goal 9

1.4. Dissertation Outline and Contribution 11

1.4.1. Part I: Advancing Interpretable Models 11

1.4.2. Part II: Unveiling Key Decision Elements 14

1.4.3. Part III: Innovating Post hoc Explanation Techniques . 15

1.5. Publication List 16

2. Background **19**

2.1. Overview of Explainable Artificial Intelligence (XAI) methods 19

2.2. Genetic Algorithm 21

2.3. Monte Carlo Tree Search 22

2.4. Deep Learning Model 24

2.4.1. Overview of Deep Learning Models 24

2.4.2. The Black-Box Nature of Deep Learning Models . . . 25

I. Advancing Interpretable Models **27**

3. Improve Model Performance with Feature Engineering **29**

3.1. Introduction 30

3.2. Related Work 31

3.3. Method 32
  3.3.1. The Transformation Tree 33
  3.3.2. The Selection Policy 34
  3.3.3. The Expansion Policy 37
  3.3.4. The mCAFE Algorithm 39
3.4. Evaluation 41
  3.4.1. Performance of mCAFE 43
  3.4.2. Ablation Study 44
    3.4.2.1. Selection Policy 44
    3.4.2.2. Expansion Policy 46
  3.4.3. Length of Feature Engineering Pipeline 47
  3.4.4. Performance of mCAFE on Different Predictive Models 51
3.5. Discussion 51

4. Improve Model Performance with Parameter Optimization 53
4.1. Introduction 54
4.2. Related Work 57
4.3. Method 58
  4.3.1. Individual Coding Design 59
  4.3.2. Populations Generation 59
  4.3.3. Individual Growth and Fitness Calculation 60
  4.3.4. Individual Selection 61
  4.3.5. Crossover and Mutation 63
  4.3.6. New Individual Generation 64
  4.3.7. Neural Network Feature Extraction 66
4.4. Evaluation 66
  4.4.1. Benchmark Models 68
  4.4.2. Benchmark Datasets 68
  4.4.3. Experiment Setup 69
  4.4.4. Result 70
4.5. Discussion 70

5. Explain with Large Language Model 73
5.1. Introduction 74

5.2. Related Work 75
5.3. Method 76
    5.3.1. Components of Prompt 76
    5.3.2. Extract Classification Rules 77
    5.3.3. Generate Lookup Table 78
    5.3.4. Translate Data to Description 80
5.4. Evaluation 81
    5.4.1. Experiment Setup 81
    5.4.2. Result 83
5.5. Discussion 84
    5.5.1. Limitation 84
    5.5.2. Summary 86

II. Unveiling Key Decision Elements 89

6. Summarize the Decision-making Elements of High-quality Models 91
6.1. Introduction 92
6.2. Observation 93
    6.2.1. Data Processing 93
        6.2.1.1. 1D Transformation 94
        6.2.1.2. 2D Transformation 94
        6.2.1.3. 3D Transformation 95
        6.2.1.4. Summary of Key Data Process Trends . . . 95
    6.2.2. Model Architecture 96
        6.2.2.1. Traditional Machine Learning Approaches . 96
        6.2.2.2. Deep Learning Architectures 97
        6.2.2.3. Hybrid and Advanced Architectures 98
        6.2.2.4. Ensemble Methods 99
        6.2.2.5. Summary of Key Architectural Trends . . . 99
6.3. Discussion 100

7. Prove Element Importance through Innovative Model Design 101
7.1. Introduction 102

7.2. Observation 103
  7.2.1. Automatic Remaining Useful Life Estimation Framework with Embedded Convolutional LSTM as the Backbone 103
  7.2.2. TinyHAR: A Lightweight Deep Learning Model Designed for HAR 105
7.3. Discussion 108

III. Innovating Post hoc Explanation Techniques 111

8. Explain with Spatial Information 113
8.1. Introduction 114
8.2. Related Work 116
  8.2.1. Explainable Artificial Intelligence 116
  8.2.2. Monte Carlo Tree Search 117
8.3. Method 117
  8.3.1. Tree Representation and Algorithm Framework 118
  8.3.2. Monte Carlo Tree Generation 120
  8.3.3. Action Set Selection and Refinement 123
8.4. Evaluation 124
  8.4.1. Classification Game: Comparing Local Patterns with Positive Impact 124
  8.4.2. Misclassification Game: Testing the Improvement of Complex Model through Retraining 127
  8.4.3. Extracting Explanation from MCT 129
8.5. Discussion 132

9. Explain with Temporal Information 135
9.1. Introduction 136
9.2. Related Work 137
9.3. Method 138
  9.3.1. Segment & Clustering Module 138
  9.3.2. Perturbation Module 141

9.3.3. Explanation Module 141
9.4. Evaluation 143
9.5. Discussion 147

10. Explain with other Information 149
10.1. Introduction 150
10.2. Related Work 151
10.3. Method 152
10.3.1. Problem Definition and Individual Coding 152
10.3.2. Population Generation 154
10.3.3. Fitness Function Design 155
10.3.4. Growth 158
10.3.5. Crossover and Mutation 159
10.3.6. Explanation 160
10.4. Evaluation 162
10.4.1. Benchmark Dataset 163
10.4.2. Target Models 164
10.4.3. Benchmark Algorithms 164
10.4.4. Experiments Design 164
10.4.5. Experiment Setup 165
10.4.6. Evaluation 165
10.5. Discussion 170

11. Final Discussion and Future Work 171
11.1. Discussion on Improving Interpretable Model Performance . . 171
11.2. Discussion on Extracting Model Decision Elements 173
11.3. Discussion on Post-hoc Explanation Methods 174
11.4. Conclusion 175
11.5. Future Work 176

Bibliography 179

List of Figures

1.1. An example to demonstrate the function of XAI. The left subplot (a) shows the original input signal. The right subfigure (b) shows the interpretation of our proposed method. 6

1.2. The structure, content, and contributions of the dissertation chapters and their interrelationships. 17

3.1. Representation of feature transformation with a tree structure. Here, each node corresponds to a state *D* and each edge corresponds to a transformation (action). The distributions on the edges show the distribution over the mean success (reward = 1) probability when taking the action in the parent state. 35

3.2. The Surrogate network consist of 2 Long Short-Term Memory (LSTM) layers of size 32 and a two fully connected layers of size 32 with a ReLU activation function. 38

3.3. The mCAFE framework: each iteration (episode) includes four phases: selection, expansion, roll-out, back-propagation. *B* is the number of iterations. 39

3.4. Example of an episode of mCAFE. The beta distributions of the edges in the selected path are displayed next to the corresponding edge. Blue denotes the distribution before back-propagation and orange after back-propagation. 41

3.5. Comparing the performance between mCAFE-ucb and mCAFE. 46

3.6. Comparing the performance of Multi-Layer Perceptron (MLP) and LSTM model in predicting the Q value. 46

3.7. Comparing performance of mCAFE with neural network expansion policy (with nn), mCAFE with random expansion policy (with random) and mCAFE with fix expansion policy (with fix) on all the regression dataset. Classification task is evaluated with F1-score and regression task is evaluated with (1-relative absolute error). 48

4.1. The pipeline of the proposed algorithm. 58

5.1. Extract classification rules from train dataset 76

5.2. the pipeline of generating lookup table 80

5.3. The example of ask GPT to explain the given feature 80

5.4. Explanation of shake hand 85

8.1. Illustration of action selection (a) and refinement (b). 123

8.2. (a) Ranks of local pattern importance and masked image for each method. (A) Shows the input instance. (B) Shows the explanation created by each algorithm. The local patterns colored in blue have a positive local pattern importance according to each method. (C) Shows the masked image which is no longer predicted as 7. (b) Shows the Monte-Carlo Tree (MCT) created by McXai from the example of (a). The state x is the input instance. The value of each edge is written beside the corresponding edge. The path $[a_1, a_4, a_7]$ is the best path containing the actions with the highest expected value of the considered states. 127

8.3. Two examples of complex model prediction probabilities . . . 132

9.1. Explanation for the example data entry given the TS-MULE, which 1). only contains the feature importance, 2). only shows the original data instead of summarizing the characteristic of the trend. 145

10.1. The pipeline of the proposed ExTea algorithm. 153

10.2. Comparison of ExTea performance with different parameter setting. The line next to the violin describes the value range of the corresponding parameter. The violin describes the kernel density estimate of achieving the best result. 169

List of Tables

1.1. Abstract framework of the dissertation. 12

3.1. Average percentage of time for each process in the first 20 episodes. 43
3.2. Comparing performance of without feature engineering (Base), reinforcement-based model (RBM) [91], Cognito [90], random selection, and mCAFE in 100 episodes using 15 open source datasets. Classification tasks (C) are evaluated with the F1 score, and regression tasks (R) are evaluated with (1-relative absolute error). 45
3.3. The performances of mCAFE with different predictive models on Automatic Machine Learning (AutoML) benchmark dataset [51]. The improvements brought by the mCAFE are shown in parentheses. Classification task is evaluated with F1-score and regression task is evaluated with (1-relative absolute error) 49
3.4. Comparing performance of mCAFE with different maximum pipeline length on 3 classification datasets (F1-score) and 3 regression datasets (1-relative absolute error). 50

4.1. The processes executed in process pool. 62
4.2. The neural network for feature extraction. 67
4.3. Accuracy performance of the proposed algorithm 70

5.1. Translate data to description 82
5.2. The accuracy of experiments on 3 datasets 84

7.1. In the early fusion convolution, the kernel height is fixed, that is, the same as the number of features. The sliding direction of the convolution kernel is along the time axis. In the late fusion convolution, the kernel height is 1. Each feature has its own convolution kernel. The convolution kernel also has only one sliding direction, namely the time axis. In hybrid fusion convolution, the kernel height is also 1. But it has two sliding directions, one is the time axis and the other is the feature axis. Because of weight sharing, it can save many parameters. It should be noted that when the number of filters is greater than 1, the output of the early fusion convolution is 2-dimensional. The outputs of the two remaining convolutions are 3-dimensional. 106
7.2. Overview of the proposed Algorithm. 107

8.1. A framework of the McXai algorithm. 119
8.2. Comparing average number of steps (NoS) needed to take to change the prediction of complex model according to the suggestion of the LIME, SHAP and McXai methods. 126
8.3. Comparing performance of complex model: mnasnet0_5, mnasnet1_0, DenseNet121, WideResNet and GoogleNet in these three different situations: (1) trained with training set $\mathscr{D}_{train}$ (base_score) (2) trained with training set $\mathscr{D}_{train}$ and $\mathscr{D}_{both}$ (score_both) (3) trained with training set $\mathscr{D}_{train}$ and $\mathscr{D}_{mis}$ (score_mis) 129
8.4. An example of the explanation of the three different methods: McXai, Grad-cam, LIME. The first row shows the original image, McXai's explanation, Grad-cam's explanation, LIME's explanation in order. The lower right part is the MCT generated by the McXai and the lower left part is the corresponding 'performance' and 'value' attributes of each edge in the MCT. 130

9.1. (a). Description of the Whole Model Pipeline. (b). Details of the Hyperparameter Optimization Procedure for the Segment & Clustering Module. 139

9.2. (a). Transitions graph of HMM, showing the importance of different state transitions. (b). The state sequence of the interested data entry (IDE) and the background color showing the feature importance. (c). The Correspondence between the state and the cluster core. The x-coordinates in Figures (b), (c) both represent timestamp. 144
9.3. Attack Success Rate (ASR) of different modification methods. 146

10.1. An example of the ExTea explanation. 161
10.2. Summary of the datasets used in the experiments. 163
10.3. Ratio of information needed to support the model decision, the smaller the better. The bold numbers denote the smallest ratio in the corresponding groups. 166
10.4. Comparison of ExTea performance with and without growth processes. The x-axis represents the number of generation (epoch) and the y-axis indicates the ratio between the final cognitive block length and the input sequence length. 168

1. Introduction

1.1 Pervasive Computing and Explainable Artificial Intelligence

Pervasive computing, also known as ubiquitous computing, refers to the embedding of computational capabilities into everyday objects and environments, allowing them to communicate and perform useful tasks without requiring direct human interaction [170]. This concept, originally introduced by Mark Weiser in the early 1990s, envisions a world where computers become seamlessly integrated into the fabric of our lives [175]. As pervasive computing is increasingly integrated into various domains such as smart grids and healthcare [34; 171], the growing task difficulty and data complexity necessitate the use of increasingly complex Machine Learning (ML) models [128]. This evolution from simple rule-based systems to complex deep learning networks [100] has significantly enhanced their capabilities, allowing them to perform more complex and varied tasks with higher predictive performance [1]. However, this evolution also introduces greater opacity in the decision-making processes of these systems [38], prompting vital questions regarding the transparency and trustworthiness of Artificial Intelligence (AI) decisions in pervasive environments [101], especially in applications where decisions significantly impact human property, lives, and safety [10].

To address these challenges, XAI is introduced. It is a collection of techniques and methods designed to make the decision-making process of AI systems transparent and understandable. However, related research in the field of pervasive computing remains limited [131]. On the one hand, because of the task complexity, current interpretable models often struggle to achieve the

---

[1] Predictive performance refers to how well a model or algorithm can predict outcomes based on given input data. Common metrics for assessing predictive performance include accuracy, precision, mean squared error, etc.

same level of accuracy as their more complex counterparts. On the other hand, existing post hoc explanation methods [2] frequently fail to provide sufficient domain-specific insight. Based on these challenges, this dissertation aims to advance the field of XAI within pervasive computing by improving the predictive performance of interpretable models and developing novel domain-specific post hoc explanation methods. By enhancing both model predictive performance and explainability, this research seeks to make AI-driven systems in pervasive computing more transparent, trustworthy, and effective.

1.1.1 Significance of Pervasive Computing

The significance of pervasive computing is particularly evident in its diverse applications, which span across numerous sectors, enhancing efficiency, safety, and user experience [1].

In energy applications, the integration of pervasive computing within smart grids has revolutionized energy management. Smart grids equipped with pervasive computing technologies can optimize energy distribution, reduce losses, and enhance reliability. For example, in Hamburg's central renewable smart grid, the excess capacity necessary to prevent blackouts was significantly reduced from 95% to 65% through advanced monitoring and predictive analytics [22]. This optimization not only ensures a stable energy supply, but also contributes to more sustainable energy consumption practices.

In addition, Human Activity Recognition (HAR) represents another crucial application of pervasive computing. HAR involves the use of ML algorithms to identify and predict human activity based on data from wearable devices, smartphones, or ambient sensors [149]. This technology has profound implications for various fields, including healthcare [20], fitness [62], and elderly care [172]. For example, in elderly care, HAR systems can detect falls or unusual patterns of movement, triggering alerts to caregivers or emergency services promptly [96]. This capability enhances the safety and independence of elderly individuals living alone, providing peace of mind to their families. Moreover, fitness applications use HAR to track and analyze physical activi-

---

[2] Post hoc explanation method refers to explainable artificial intelligence techniques that are applied after a model has been trained and is making predictions [5].

ties such as running, walking, or cycling, offering users detailed insight and personalized recommendations to improve their health and predictive performance [118].

In summary, pervasive computing, through its wide-ranging applications, has become a cornerstone of modern technological advancement. Its integration into various sectors underscores its potential to transform everyday life by making environments smarter, more responsive, and more efficient. However, this ubiquitous characteristic also fosters mutual trust between humans and the AI system. This necessitates a continuous focus on enhancing transparency and trustworthiness, especially as these embedded systems increasingly rely on complex models.

1.1.2 The Shift Towards Complex Models

Initially, pervasive computing systems relied on simple and transparent models, such as rule-based systems and linear networks [159]. These models had understandable operations and decision-making processes, which made them highly interpretable and easy to trust [141]. However, as the range of applications for pervasive computing expanded and the complexity of tasks increased, the models themselves evolved to handle these new demands [100]. Modern models, such as deep learning networks and ensemble methods, exhibit significant increases in complexity to address the sophisticated requirements of new applications. These advanced models often involve numerous layers of nonlinear transformations and a large number of parameters [86], which collectively enhance their predictive performance in complex tasks such as image recognition, natural language processing, and autonomous decision-making.

Despite these advancements, the increased complexity of these models has introduced new challenges, particularly concerning transparency and explainability. Unlike their simpler predecessors, deep learning model operates as "black boxes," making it difficult to understand how they arrive at specific decisions. This opacity has raised significant concerns regarding the trustworthiness and accountability of AI-driven decisions in pervasive computing environments [3; 104].

1.1.3 The Role of Explainable Artificial Intelligence

XAI is a critical approach in AI that aims to clarify the predictive outcomes and decision-making processes of ML models. This transparency is essential in critical applications like predictive maintenance [152], product quality assessments [150], and medical assistance [113]. For example, in predictive maintenance, XAI enables engineers to understand the reasons behind predicted failures of machine components, facilitating timely and targeted interventions. In product quality assessments, XAI sheds light on the factors leading to product defects, allowing manufacturers to enhance production processes. In brainwave analysis, XAI helps researchers and clinicians interpret AI-driven insights from Electroencephalogram (EEG) data, crucial for diagnosing neurological conditions and personalizing treatments. In these contexts, explainability ensures that AI systems are transparent, fair, and credible, which is vital for gaining user trust and enabling informed decision-making.

In addition, legal frameworks like the General Data Protection Regulation mandate clarity in automated decision-making, ensuring that individuals can understand and contest decisions made by AI systems. The Artificial Intelligence Act proposed by the European Commission in 2021 further emphasizes the necessity of explainability, identifying it as one of the seven key requirements to build trustworthy AI systems [40].

XAI serves three primary functions in pervasive computing environments namely justification, validation, and discovery [121].

Justification. This answers the question "Why did the model give the decision?". It involves providing clear explanations for AI decisions, particularly in high-stakes scenarios such as predictive maintenance or medical interventions, where erroneous decisions can have severe consequences. For example, in medical support systems, understanding the rationale behind a diagnosis or treatment recommendation is crucial for healthcare providers to make informed decisions and ensure patient safety [10].

Validation. This answers the question "Is the model right for the right reason?". It focuses on ensuring that AI models make accurate decisions based on sound reasoning, adapting effectively to pervasive computing environments that are often characterized by incomplete or noisy data. The wolf vs. husky classification problem [134] highlights this necessity by showing how a complex model can mistakenly learn to differentiate wolves from huskies based on irrelevant features, such as snow in the background of wolf images, rather than the animals' actual characteristics.

Discovery. This answers the question "Do the explanation reveal new information?". It involves using explanations to uncover new insights or unintended biases in AI models. This function is vital for continuous improvement and innovation, as it allows developers to understand the underlying patterns and biases in the data, leading to the development of more accurate and fair models. For example, XAI can reveal biases in predictive policing algorithms, prompting corrective measures to ensure fairness and equity [57].

To demonstrate the three core functions discussed, we employ a HAR example from a Wii [3] system, specifically the recognition of a clockwise circular motion using sensor data from the Wii accelerometer, as illustrated in Figure 1.1-(a). In this figure, the x-axis represents the timestamps, while the y-axis denotes the corresponding acceleration values. For classification, we utilize the TinyHAR model [187], which accurately classifies the input as a clockwise circular motion.

Figure 1.1-(b) visualizes part of the explanation of the proposed ExTea algorithm [78]. The orange-highlighted regions in the figure represent data segments identified by ExTea as critical to the model's decision. The arrow above these segments, along with the "X" marks, indicates that the relative positions of these segments are crucial, reversing their order would result in a misclassification by the model. The cyan regions adjacent to the orange segments show areas where the data segments can shift forward or backward without affecting the model decision. In addition, the yellow areas within the orange segments indicate the allowed range of perturbation within these segments.

From this analysis, several insights can be derived:

---

[3] The Wii is a home video game console developed by Nintendo, featuring innovative motion control technology. The players use a handheld controller, the Wii Remote, equipped with an accelerometer and infrared sensors to detect motion and position, enabling an intuitive and immersive gaming experience by translating physical movements into in-game actions.

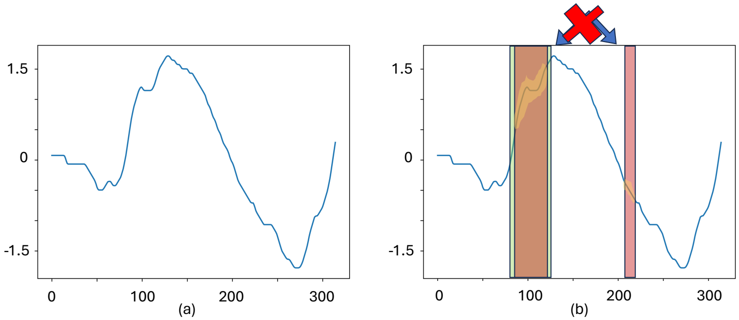


Figure 1.1.: An example to demonstrate the function of XAI. The left subplot (a) shows the original input signal. The right subfigure (b) shows the interpretation of our proposed method.

- The orange data segments, and the fact that their sequence cannot be reversed, suggest that the model bases its decision on the order in which the acceleration increases in both the positive and negative directions.
- By comparing this with our common sense, we can verify that the model's reasoning aligns with the expected interpretation.
- The small move region allowed for the data segment implies that the participant that executed the clockwise circular motion during data collection performed the action at a relatively consistent speed. This leads the model to recognize the time interval between the two data segments (the increase in acceleration in both directions) as important element for decision making, which contradicts our expectations. To improve the model, we could consider instructing participants to vary their speed during motion capture. Alternatively, applying targeted data augmentation techniques could enhance data diversity and mitigate this effect.

In summary, XAI is integral to the advancement and deployment of AI in pervasive computing. It not only enhances transparency and trust, but also ensures that the AI systems are accountable and aligned with ethical and legal standards. As AI continues to permeate various aspects of daily life, the role of

XAI will become increasingly critical in fostering trustworthy and responsible AI development.

1.2 Importance of Interpretable Models

The rapid evolution of complex models within the field of pervasive computing has significantly enhanced researchers' ability to process and analyze intricate datasets, thereby facilitating the creation of intelligent and efficient computing systems [180]. These advanced methods possess the capability to autonomously extract intricate features from the data, consequently surpassing the predictive performance of interpretable models. Concurrently, the emergence of XAI approach has provided a partial explanation of the decision-making processes of these complex methods. This development raises a pivotal question: Why should the deployment of interpretable models continue to be a consideration in the realm of pervasive computing?

We can approach this question from several perspectives. Firstly, a key advantage of interpretable models is their high level of transparency in both training and decision-making processes. In pervasive computing environments, systems often handle sensitive personal data, including health monitoring, smart home, and personal assistant applications [43]. In these contexts, user trust is essential. Interpretable models offer a transparent decision-making process that enables users to understand and trust the system's output. For instance, in a health monitoring system, it is crucial for doctors and patients to understand how the model arrives at a particular diagnosis to make informed treatment decisions [165]. XAI Methods designed to explain these models, such as LIME [134] and SHAP [110], often simplify or locally approximate the model's behavior to provide explanations. However, these explanations often fall short in accurately representing the model's true reasoning, as they may only capture local behaviors and not the global decision-making process. Studies [4; 104; 138] have shown that these explanation methods can produce explanations that do not fully align with the actual workings of the model. This discrepancy highlights the importance of developing inherently interpretable models to ensure accountability and trust in the use of complex systems.

Additionally, in pervasive computing environments, many applications necessitate real-time responses. For example, intelligent traffic management systems must process large amounts of sensor data and make decisions in very short time frames [183]. Interpretable models, typically lighter in computational load than complex deep learning models, can provide faster predictions. In contrast, deep learning models, while powerful, often involve extensive computational requirements and longer processing times due to their numerous parameters and complex architectures. This can lead to latency issues in time-sensitive applications, making them less suitable for scenarios where prompt decision-making is critical [147].

Besides, in some pervasive computing applications, data may be scarce or incomplete. For instance, some environmental monitoring systems may only have access to limited sensor data. Interpretable models can utilize scarce data more effectively due to their reliance on simpler assumptions and structures that can be trained and validated with fewer data. For example, in agricultural systems with limited weather and soil data, simple linear models can effectively predict crop growth [84]. In contrast, deep learning models generally require large amounts of high-quality data to achieve optimal predictive performance. Their complex architectures and high number of parameters make them prone to overfitting when trained on limited or incomplete datasets, leading to unreliable predictions and poor generalization [53]. This makes them less suitable for applications where data availability is limited.

Pervasive computing systems are typically distributed across various environments and require efficient deployment and maintenance [56]. Interpretable models are easier to deploy and run on resource-constrained devices due to their lower complexity and computational requirements. Additionally, these models are easier to debug and maintain. For instance, in a smart home system, rule-based models can be quickly deployed to various devices, and when system anomalies occur, they can be reviewed to quickly locate them. In contrast, deep learning models often face significant challenges in these areas. Their high computational demands and complex architectures make them difficult to deploy on devices with limited resources. Furthermore, debugging and maintaining deep learning models is more complicated due to their "black box" nature, which obscures the reasoning behind their decisions and makes identifying the source of errors more challenging [104; 138].

In summary, the necessity of interpretable models in pervasive computing has been demonstrated from four key perspectives: explainability, real-time requirements, data scarcity, and deployability. Therefore, despite some limitations in predictive performance, interpretable models possess irreplaceable value in the field of pervasive computing and cannot be ignored.

Considering the critical role of XAI in pervasive computing, it is essential to either enhance the predictive performance of existing interpretable models or to develop new robust models that maintain interpretability. Such efforts are pivotal in addressing existing disparities between predictive performance and explainability, thereby ensuring that pervasive computing systems are both effective and comprehensible. This strategic focus is vital for sustaining the integration of XAI into pervasive computing applications, promoting transparency and trustworthiness.

1.3 Challenges and Goal

In pervasive computing, XAI approach faces multiple challenges due to the complexities associated with pervasive environments and the inherent limitations of current XAI methods. Specifically, it includes the following five challenges.

(i) **C1: The complexity of tasks and data in the pervasive computing domain poses a huge challenge for interpretable models, which demands advanced interpretability without compromising performance.** Task complexity necessitates complex techniques in feature extraction and optimization, which often clash with the need for simplicity in interpretable models. For instance, decision trees provide clearer insights but lack scalability and robustness in managing large, noisy datasets. Conversely, more complex models like random forests offer improved predictive performance, but at the expense of explainability.

(ii) **C2: Pervasive computing imposes stringent constraints on model deployment, which further complicates the optimization of interpretable models.** Models deployed in such environments are frequently subject to constraints such as limited computational resources, reduced power availability,

and the need for low-latency inference [4].

(iii) **C3: The specific characteristics of pervasive computing, such as diverse data sources, large volumes of noisy data, and temporal dependencies, further complicate the application of XAI.** For instance, interpreting data from pervasive systems, especially time-series data, presents distinct challenges that differ from those encountered in more static domains like image recognition. Unlike image data, where domain experts can easily identify critical information, time series data often lack intuitive features that clearly influence model predictions. This is evident in specific applications, such as the classification of insect species using field-deployed sensors, where frequency and temporal patterns are critical but difficult to explain in an interpretable manner. Such challenges highlight the need for explanation methods that can handle the frequent and temporal patterns present in pervasive computing data.

(iv) **C4: Generic saliency-based explanations are insufficient to explain the decision-making process in pervasive computing domain.** Currently, most XAI techniques, especially those in the image domain, rely on saliency maps to explain complex models. While saliency maps are effective in highlighting which parts of the input data influence the model decision, they often fall short in explaining the decision-making process itself. This lack of clarity can lead to oversimplified explanations that do not adequately capture the model's reasoning. In pervasive computing, where domain-specific knowledge plays a crucial role, generic saliency-based explanations are insufficient. The field requires more advanced explanation methods that can not only indicate "where" in the data the model is focusing but also provide insights into "how" the model processes this information to arrive at its decision.

(v) **C5: The reliability of some existing post hoc XAI methods, such as those based on local surrogate models, is doubtful.** While these methods are popular for explaining complex models, they sometimes fail to faithfully represent the internal workings of the model across all data points [134]. Besides, their explanations are local by nature, which means they only approximate the model's behavior for specific instances. This limitation becomes problematic in pervasive computing, where continuous streams of dynamic and often highly

---

[4] Inference time in model prediction refers to the amount of time it takes for a machine learning or deep learning model to process a single input and generate a prediction or output.

interdependent data are the norm. As a result, explanations generated by these methods may not fully encapsulate the complex, temporally-aware reasoning processes inherent in advanced AI systems.

These challenges have significantly influenced the direction of my research. This dissertation aims to enhance XAI performance in pervasive computing by not only enhancing the predictive performance of interpretable models but also developing innovative explanation methods targeting the aforementioned challenges. The goal is to overcome the limitations of current XAI approaches and to derive more comprehensive insights into the models used in pervasive computing environments.

1.4 Dissertation Outline and Contribution

Figure 1.1 illustrates the abstract framework of this dissertation. When modeling the collected pervasive computing data, two primary approaches are available. The first approach involves using interpretable models, which, due to the inherent complexity of the data and the task, typically result in high explainability but suboptimal performance (as depicted in the upper part of the Figure 1.1). The second approach employs more complex models, such as deep learning architectures, which generally offer superior performance but at the expense of reduced explainability (shown in the lower part of the Figure 1.1). The objective of this dissertation is to address this trade-off and strive for a balance between model performance and interpretability (right side of the Figure 1.1). To achieve this, the overall objective has been divided into three sub-goals, each corresponding to a part of the dissertation: Part I: "Advancing Interpretable Models", Part II: "Unveiling Key Decision Elements", and Part III: "Innovating Post hoc Explanation Techniques" (center of the Figure 1.1). A detailed discussion of each part follows.

1.4.1 Part I: Advancing Interpretable Models

In pervasive computing, the complex nature of data (**C1**) and deployment constraint (**C2**) pose substantial challenges to the predictive performance of interpretable models. Despite these difficulties, the necessity for interpretable

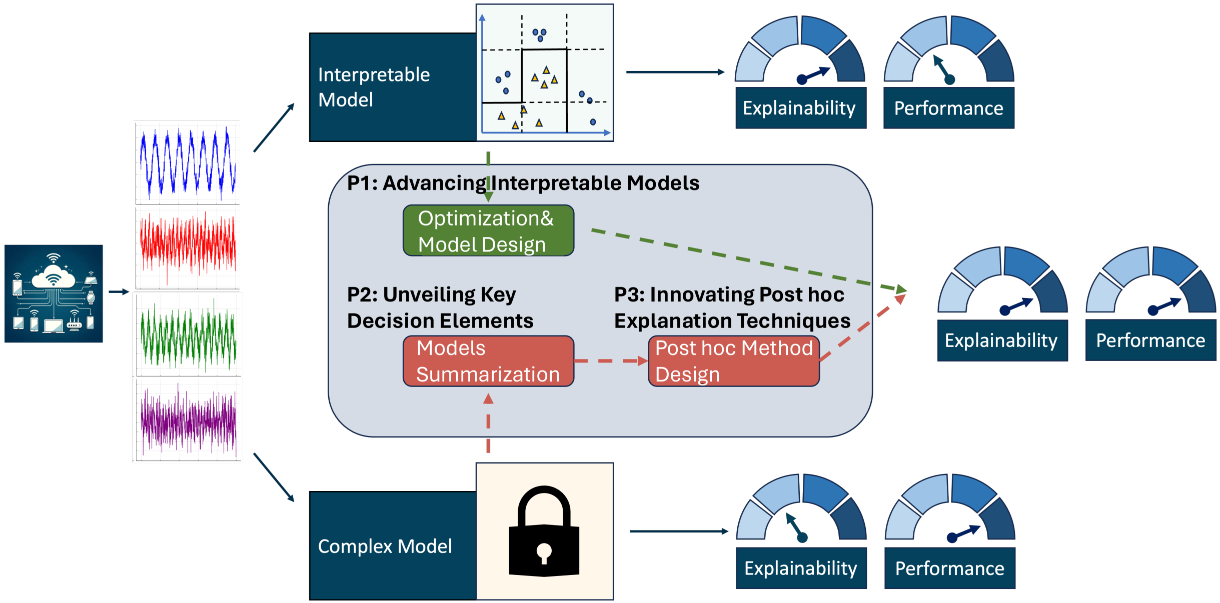


Table 1.1.: Abstract framework of the dissertation.

models in this field remains critical, prompting the first research question:

> **Q1:** How can the predictive performance of interpretable models within the pervasive computing domain be enhanced?

One primary issue affecting the predictive performance of interpretable models is the inadequacy of the features generated through traditional feature extraction processes. These features often fail to address the target problem comprehensively, and the models themselves have limited capabilities in generating new, effective features. To address these issues, we have developed a novel automatic feature engineering approach that incorporates advanced exploratory techniques and emphasizes temporal dependencies that are often overlooked by traditional methods. This led to the creation of an automatic feature engineering framework, mCafe, which outperforms state-of-the-art methods on common benchmark datasets.

Beyond feature engineering, the predictive performance of interpretable models also depends on appropriate feature selection, model selection, and parameter tuning. To address these aspects, we introduce a novel ML model optimization algorithm designed for constrained pervasive computing environments characterized by uneven search spaces and complex parameter interdependencies.

Conversely, understanding features in the pervasive computing domain frequently demands specialized knowledge, such as interpreting changes in energy over time. Large Language Models (LLMs) exhibit significant potential in knowledge storage, causal inference, and text generation. However, their limitations in processing raw signals present a substantial challenge for employing LLMs as interpretable model. To overcome this obstacle, we propose an LLMs proxy scheme that integrates traditional feature extraction methods with LLMs. This approach not only enables accurate classification of a given sample, but also provides clear explanations for classification decisions.

Based on these three studies, we make the following contribution:

> **Contribution 1:** Enhancement of interpretable model predictive performance through a feature engineering, constrained AutoML methodology and novel interpretable model design.

1.4.2 Part II: Unveiling Key Decision Elements

To dig deeper into the cause of Challenge **C3**, **C4** and advance the development of post hoc XAI methods to address these challenges in pervasive computing, it is essential to first comprehend the models used in this field and the data they leverage for decision-making. This leads to the formulation of the second research question:

> **Q2:** What data elements do high-quality models utilize for decision-making in the pervasive computing domain?

In response, we conducted a systematic literature analysis of 108 peer-reviewed research articles. We categorized these articles according to their training methodologies employed and extracted the relevant data features utilized in each. In addition, we analyze the model architecture to determine the emphasized features. These dual analyses have elucidated the critical data elements leveraged by high-quality models in this domain.

Our ongoing research extends to the design and testing of new model architectures to refine how models focus on crucial data elements. These efforts improve the predictive performance of the model and prove the importance of the discovered data elements in decision-making processes. This leads to our second significant contribution:

> **Contribution 2:** Summarization of training processes and model architectures, identifying the key data elements utilized by high-quality models in the pervasive computing domain.

1.4.3 Part III: Innovating Post hoc Explanation Techniques

Understanding the bases of high-quality model decisions informs our approach to addressing XAI challenges (**C3**, **C4**, and **C5**), leading to the third research question.

> **Q3:** How can the data elements uncovered be utilized in post hoc XAI method design?

In contrast to image-based classification, where information is typically more centralized, sensor-based classification often involves extended time periods and multiple data pipelines. In this context, the interactions between features of different sensors at various timestamps play a crucial role in the decisions made by the model. However, current XAI methods struggle to explain this complexity effectively.

To solve the problem, we conceptualize the explanation of model decisions as a tree-search problem. Given samples and their corresponding classifications, the root node of the tree represents the data samples to be interpreted. Each node in the tree corresponds to the data sample with certain masked information. Consequently, each leaf node provides an explanation, and the change in logits between parent and child nodes reflects the relationship between different data segments. This tree-search approach enables us to systematically explore the relationships among various data segments and link them to the final explanation.

In addition, to incorporate time series information into the explanation, we represent the original samples as state sequences by clustering data segments into distinct states. This method leverages state transitions to capture the temporal dynamics of the original signal, reframing the explanation of decisions as the task of identifying the state transition graph that most likely generates a given sample.

Part II outlined the seven key decision-making elements essential to a high-quality model. Developing an explanation method for each element individually is both tedious and time-consuming. To address this, we propose a unified approach that explores all these elements simultaneously using a single algorithm, and identify an effective method for presenting the exploration results. Given the multi-objective nature of this problem, we have chosen a genetic algorithm as the most appropriate solution. We incorporate the exploration of each element into the fitness function, categorize the potential of individuals within a pyramidal pool, and employ a Baldwin-effect-like operation to optimize individuals within the same generation. Furthermore, we establish logical rules for generating explanatory texts, enhancing the clarity and comprehensibility of the final explanations. Our contribution is summarized as follows:

> **Contribution 3:** Integration of local data pattern, temporal dependence, noise, and other data elements discovered in Part II into explanations using novel proposed post hoc techniques.

Figure 1.2 provides a summary of the dissertation's structure, main content, and contributions of each chapter. Their content and contributions align with the above-mentioned overview. Structurally, Chapters 3, 4, 5, 8, 9, and 10 follow the IRMED format, where each chapter is structured into five sections: introduction, related work, methodology, evaluation, and discussion. These chapters correspond to individual research papers. Meanwhile, Chapters 6 and 7 are organized using the IOD format, comprising three sections: introduction, observations, and discussion. These chapters correspond to two distinct papers, with the observations section summarizing the findings of multiple researches.

1.5 Publication List

Most of the chapters in this dissertation are based on published papers. The content of these chapters is largely consistent with that of the original paper, so each chapter can be read independently. The following section summarizes the author's published works that are directly related to the contributions of this dissertation.

Part I: Insight into the Interpretable Model

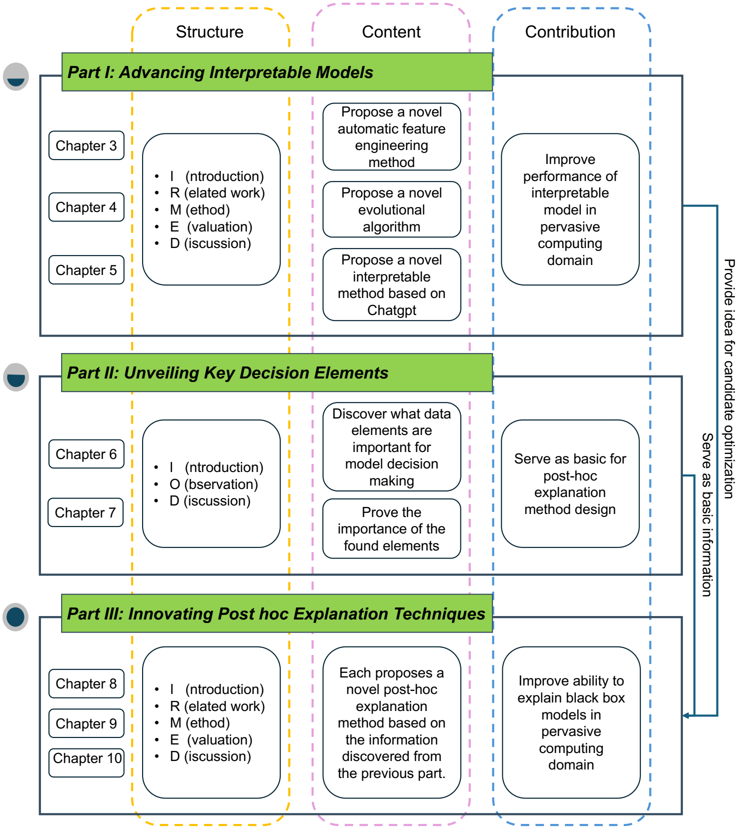

Structure
Content
Contribution
Part I: Advancing Interpretable Models
Chapter 3
Chapter 4
Chapter 5
I (ntroduction)
R (elated work)
M (ethod)
E (valuation)
D (iscussion)
Propose a novel automatic feature engineering method
Propose a novel evolutional algorithm
Propose a novel interpretable method based on Chatgpt
Improve performance of interpretable model in pervasive computing domain
Part II: Unveiling Key Decision Elements
Chapter 6
Chapter 7
I (ntroduction)
O (bservation)
D (iscussion)
Discover what data elements are important for model decision making
Prove the importance of the found elements
Serve as basic for post-hoc explanation method design
Part III: Innovating Post hoc Explanation Techniques
Chapter 8
Chapter 9
Chapter 10
I (ntroduction)
R (elated work)
M (ethod)
E (valuation)
D (iscussion)
Each proposes a novel post-hoc explanation method based on the information discovered from the previous part.
Improve ability to explain black box models in pervasive computing domain
Provide idea for candidate optimization
Serve as basic information

Figure 1.2.: The structure, content, and contributions of the dissertation chapters and their interrelationships.

Y. Huang, Y. Zhou, M. Hefenbrock, T. Riedel, L. Fang, and M. Beigl. Automatic feature engineering through monte carlo tree search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 581–598. Springer, 2022

Y. Huang, Y. Zhou, H. Zhao, T. Riedel, and M. Beigl. Optimizing automl for tiny edge systems: A baldwin-effect inspired genetic algorithm. In *22nd IEEE International Conference on Pervasive Computing and Communicaitons (PerCom 2024)*, 2024

Y. Huang, Z. Xue, H. Ma, and M. Beigl. Generate explanations for time-series classification by chatgpt. *Explainable Artificial Intelligence, Malta, 17th–19th June 2024*, 2024

Part II: Insight into Model Decision Elements

Y. Huang, H. Zhao, Y. Zhou, T. Riedel, and M. Beigl. Standardizing your training process for human activity recognition models: A comprehensive review in the tunable factors. *EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2024

Y. Huang, Y. Zhou, H. Zhao, T. Riedel, and M. Beigl. Optimizing automl for tiny edge systems: A baldwin-effect inspired genetic algorithm. In *22nd IEEE International Conference on Pervasive Computing and Communicaitons (PerCom 2024)*, 2024

Y. Zhou, M. Hefenbrock, Y. Huang, T. Riedel, and M. Beigl. Automatic remaining useful life estimation framework with embedded convolutional lstm as the backbone. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part IV*, pages 461–477. Springer, 2021

Y. Zhou, H. Zhao, Y. Huang, T. Riedel, M. Hefenbrock, and M. Beigl. Tinyhar: A lightweight deep learning model designed for human activity recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, pages 89–93, 2022

Part III: Insight into the Post hoc Technologies

Y. Huang, N. Schaal, M. Hefenbrock, Y. Zhou, T. Riedel, and M. Beigl. Mcxai: local model-agnostic explanation as two games. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2023

Y. Huang, C. Li, H. Lu, T. Riedel, and M. Beigl. State graph based explanation approach for black-box time series model. In *World Conference on Explainable Artificial Intelligence*, pages 153–164. Springer, 2023

Y. Huang, Y. Zhou, H. Zhao, L. Fang, T. Riedel, and M. Beigl. Extea: An evolutionary algorithm-based approach for enhancing explainability in time-series models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 429–446. Springer, 2024

2. Background

2.1 Overview of XAI methods

Depending on the techniques used, XAI can be broadly classified into the following categories:

(i) **Feature Importance Methods**: These methods determine the significance of individual features in model predictions. They include model-based methods, such as the Gini index or mean-decrease impurity methods, which employ decision trees or random forests to assess feature importance. Additionally, model-independent methods rank the importance of features without relying on a specific model structure.

(ii) **Locally Interpretable Methods**: These methods focus on explaining the predictions of complex models at individual data points by constructing local models. For example, LIME (Local Interpretable Model-agnostic Explanations) [134] provides such local explanations. SHAP (SHapley Additive exPlanations) [110] uses a game-theoretic approach to quantify the contribution of each feature to a prediction, based on Shapley values.

(iii) **Visualization Methods**: Graphs and charts illustrate the internal structure of the model and its prediction results, aiding in understanding the model. The Partial Dependence Plot (PDP) [44] demonstrates how the model's prediction results change with a target feature while keeping other feature values fixed, thus revealing the overall impact of specific features on model output [44]. The Individual Conditional Expectation (ICE) [52] plot, an extension of PDP, offers a more detailed interpretation by showing how predictions vary with features across different data points. By comparing trends across these data points, the model's sensitivity and inconsistency to feature changes can be assessed [9].

(iv) **Rule-Based Methods**: These methods explain the behavior of a model

by extracting rules and logical conditions. For example, anchor point methods explain model predictions by generating highly accurate "anchor points" [135]. An anchor point is a set of sufficient conditions ensuring that the model's predictions remain unchanged under those conditions. Initially, features are randomly selected as anchors, which are then expanded by adding more features and evaluating their coverage and accuracy. The anchor with the best coverage and accuracy is selected as the final interpretation. The RuleFit method combines decision trees and linear regression. It extracts rules from decision trees and inputs these rules as features into a linear model for training [44]. This approach creates sparse, easily interpretable linear models while maintaining the predictive power of complex models. Association rule learning [92] involves converting high-dimensional data into low-dimensional representations through dimensionality reduction techniques. Association rules are then generated from these low-dimensional representations to explain the features and patterns of the original high-dimensional data.

(v) **Prototype-Based Methods (PBM)**: These methods explain the behavior of a model by comparing examples using prototypes and data [93]. Prototypes are typical data points representing a particular class, while critiques are anomalous data points that deviate from the prototypes and do not meet the model's expectations. The algorithm first selects a set of typical data points from the dataset that best represent each category as prototypes. Then, it identifies data points significantly different from the prototypes as critique instances, revealing the model's limitations and anomalous behavior. During model training, a set of prototype images is learned, representing different categories. In the prediction phase, the similarity between the input images and each prototype image is calculated. The model decision-making process is explained by demonstrating the similarity between the input images and the prototype images.

(vi) **Counterfactual-Based Approaches**: These methods explain the model predictions by generating instances that differ from, but are close to, the current instance [169]. The process begins by identifying the desired change in the model's prediction. Counterfactual instances are then generated by altering the input data characteristics. The goal is to find the smallest feature

change that causes the desired change in the prediction of the model. The differences between the original and counterfactual instances are displayed to explain the model's prediction basis. For instance, Pawelczyk *et al.* [126] propose a method to generate counterfactual explanations using adversarial attacks. Goyal *et al.* [55] employ an optimization algorithm to generate counterfactual instances that achieve the target prediction results with minimal modification of features. Visualizing the original and counterfactual instances intuitively demonstrates the feature changes and their impact on the prediction results.

(vii) There are several approaches in the field of XAI that combine multiple techniques to provide comprehensive explanations. Assaf *et al.* [11] introduced an attention layer to a deep neural network to generate an importance score for each feature. This mechanism creates a feature importance mapping by calculating the contribution of each input feature to the prediction result through the attention layer. The importance of features and their impact on the prediction results are then visually represented, often using heat maps. Causal inference methods are employed to understand the causal relationships between features and explain the predictive results of the model. For instance, causal diagrams and intervention methods are used to identify which changes in feature lead to changes in predicted outcomes.

This section provides a broad categorization of existing XAI methods which are essential to understand how model transparency can be achieved in various domains, including pervasive computing. These methods form the foundation upon which we develop our novel post hoc techniques, specifically tailored to the unique challenges of pervasive computing environments. Understanding these XAI methods also underpins our work in identifying critical data elements and advancing interpretability, as discussed in Parts II and III.

2.2 Genetic Algorithm

The Genetic Algorithm (GA) [67] is a heuristic optimization technique inspired by the principles of natural selection and genetics, which is particularly effective in solving complex optimization problems where the search space is large, discrete, nonlinear, or poorly understood.

To apply GA, potential solutions to a problem are first encoded as chromosomes, which are typically represented as binary strings. In addition, a fitness function is designed to measure how well a chromosome solves the target problem. The algorithm begins with an initial population of randomly generated chromosomes. It includes three core operations namely selection, crossover, and mutation:

- Selection: This process simulates natural selection, where chromosomes with higher fitness scores are more likely to be selected to pass their genes to the next generation.

- Crossover: This operation combines the genetic information of two parent chromosomes to produce offspring. It is typically done by exchanging segments of the parent chromosomes at one or more crossover points.

- Mutation: This operation introduces random changes to the gene of a chromosome to maintain the diversity among the chromosome within the population and prevent premature convergence to a local optimum.

The GA iterates through these steps, creating successive generations of solutions. The population is expected to evolve toward an optimal or near-optimal solution to the problem.

Genetic algorithms offer an optimization approach that simulates natural selection, particularly useful for complex search spaces such as those encountered in model optimization tasks. Their ability to explore uneven, high-dimensional search spaces is directly related to the optimization challenges faced in constrained pervasive computing environments, as highlighted in Part I. These principles are later employed in our novel approach to explainability optimization.

2.3 Monte Carlo Tree Search

Monte-Carlo Tree Search (MCTS) [30] is a powerful algorithm for decision-making under uncertainty in domains with large or complex state spaces. MCTS has gained significant attention due to its success in various board games, e.g., Go and in complex real-world applications like robotics and autonomous systems.

MCTS operates by constructing a search tree incrementally. In the tree, each node represents a state in the decision process, and the edges represent actions leading to subsequent states. The core idea is to use Monte Carlo simulations to estimate the value of decisions, gradually refining these estimates as more simulations are performed. It consists of four main steps:

- Selection: Starting from the root node, the algorithm selects child nodes to explore based on a policy that balances exploration and exploitation. A commonly used policy is Upper Confidence Bound for Trees (UCT), which selects nodes that maximize the potential reward while considering the uncertainty in their value estimates.

- Expansion: Once a node is selected, it is expanded with one or more child nodes that generated correspond to possible actions from that state. If a node is fully expanded (i.e., all possible actions have been considered), the algorithm moves to the next step.

- Simulation: Also known as rollout, this step involves running a simulation from the newly added node to a terminal state (or a pre-defined maximal depth) using a default policy. The outcome of the simulation provides an estimate of the value of the decision path that led to the expanded node.

- Backpropagation: The result of the simulation is then propagated back up the tree, updating the parameters of all nodes along the path from the expanded node to the root. This process reinforces good decisions (those leading to high rewards) and de-emphasizes poor ones.

MCTS is a decision-making algorithm that balances exploration and exploitation, particularly useful in complex environments with uncertain outcomes. The explanation of model decisions as a tree search problem, as proposed in Part III, is based heavily on these principles. Understanding MCTS allows us to systematically explore relationships between data segments, ultimately improving explainability in sensor-based classification.

2.4 Deep Learning Model

Deep learning represents a class of algorithms that leverage multilayer neural networks to model complex patterns in data. These models have revolutionized numerous fields, such as computer vision, natural language processing, and speech recognition, by achieving unprecedented levels of predictive performance. However, one of the most significant and often criticized characteristics of deep learning models is their black-box nature, which refers to the difficulty of explaining how these models make decisions.

2.4.1 Overview of Deep Learning Models

At the core of deep learning are artificial neural networks, which are inspired by the structure of human brain. These networks consist of multiple layers of interconnected nodes (neurons), where each neuron processes input data, applies a non-linear transformation via an activation function, and passes the output to the next layer. The primary structure of deep learning models includes the following:

(i) **Feedforward Neural Networks** [136]: It consists of multi-layer and each layer contains multiple neuraons. Information flows in one direction, from the input layer through hidden layers to the output layer. This structure is effective for basic tasks. However they do not inherently model temporal or spatial dependencies within data.

(ii) **Convolutional Neural Networks** [99]: It is designed for processing grid-like data such as images. It emploies convolutional layers to detect local patterns (like edges and textures) in the data. Their success in image-related tasks stems from their ability to learn hierarchical features, from low-level edges to high-level concepts like objects.

(iii) **Recurrent Neural Networks** [139]: It is particularly designed for sequential data as they maintain a memory of previous inputs through hidden states. Variants like LSTM [66] networks and Gated Recurrent Unit (GRU) [26] networks address issues like vanishing gradients, allowing the model to capture long-term dependencies in sequences.

(iv) **Transformers** [166]: Since 2017, Transformers have become the foun-

dation of modern natural language processing. They use self-attention mechanisms to capture relationships between words in a sentence, regardless of their distance from each other, making them more effective than Recurrent Neural Network (RNN) for many tasks.

2.4.2 The Black-Box Nature of Deep Learning Models

Despite their impressive capabilities, deep learning models are often criticized for their black-box nature, which describes the opacity of the models' decision-making processes. This opacity comes mainly from the following sources:

(i) **Complexity and Non-linearity**: Deep neural networks rely on numerous layers and millions of parameters. Each neuron in a layer applies a weighted sum of its inputs, followed by a non-linear activation function. These non-linearity, combined with the depth of the network, leads to highly complex decision boundaries that are difficult to disentangle or explain. This complexity makes it nearly impossible to directly interpret how the model arrived at a particular decision.

(i) **Lack of Transparency** The decision process of traditional ML models such as linear regression or decision trees, can often be traced back to individual features or rules. Deep learning models, however, obscure the relationship between input features and predictions. The interactions between neurons and layers are intricate, and the contribution of each feature to the final output is not straightforwardly discernible.

(i) **Challenge of Interpretation** Current deep learning explanation methods often rely on post hoc techniques, such as feature importance scores, attention mechanisms, or visualization methods like saliency maps. These approaches provide some insight into what the model may be focusing on, but they do not fully open the black box. Moreover, most of these explanation are based on approximations and can sometimes be misleading.

The black-box nature of deep learning models complicates model decision interpretation, especially in applications that involve pervasive computing. This section explains why existing deep learning models fail to provide transparent explanations, setting the stage for the novel interpretability techniques we propose in Part III. These challenges motivate our contribution to improve interpretability without sacrificing predictive performance.

Part I.

Advancing Interpretable Models

3. Improve Model Performance with Feature Engineering

While recent advances in deep learning have achieved superior performance relative to traditional models, the explainability of deep learning models remains a significant challenge. Specifically, explanations derived from deep learning models often fail to accurately reflect the underlying reasoning of the model [4; 104; 138], limiting their utility in fields where transparency is crucial, such as healthcare in the pervasive computing domain. For decades, interpretable ML models, such as decision trees and logistic regression, have been preferred for their transparency and ease of comprehension [173]. Today, traditional interpretable models continue to be relied upon. However, performance limitations remain the biggest obstacle to generalizing interpretable models in the pervasive domain. To expand the applicability of these interpretable models, it is crucial to enhance their performance while respecting the constraints imposed by the specific tasks and data. Feature engineering, which involves the extraction of meaningful features from raw data, plays a pivotal role in this enhancement. In this section, we explore how feature engineering can be leveraged to improve the performance of traditional interpretable ML models.

Corresponding Publication:. Y. Huang, Y. Zhou, M. Hefenbrock, T. Riedel, L. Fang, and M. Beigl. Automatic feature engineering through monte carlo tree search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 581–598. Springer, 2022

3.1 Introduction

In automated, highly integrated systems, it is crucial for users to understand and trust the decision-making process of models to confidently use these systems in critical applications [134]. Moreover, legal and ethical standards necessitate a certain level of transparency [54]. Consequently, interpretable models are often preferred over deep models due to the inadequate explanatory power of post hoc explanation techniques. However, pervasive computing, characterized by extensive environment sensing, data collection, and processing, often results in diverse and dynamic data. This complexity limits the predictive performance of interpretable ML models, thereby affecting their effectiveness in complex environments. Thus, improving the performance of interpretable ML models is essential to advance the explainability in pervasive computing.

Predictive performance improvement often hinges on the combination of sophisticated algorithms and extensive domain knowledge. This expertise is particularly evident in data preprocessing, where raw data is systematically transformed to optimize ML workflows, a process known as feature engineering. Predominantly driven by heuristic searches by domain experts, feature engineering is crucial for enhancing model performance.

However, the complexity of pervasive computing data often leads to information loss and high labor costs in expert-driven feature extraction. Automated Feature Engineering (AFE) addresses these issues by automatically generating, extracting, and selecting features from raw data using algorithms and automated tools. It aims to improve the efficiency and quality of feature generation by reducing human intervention, thereby enhancing the performance of ML models.

Despite its potential, AFE faces significant challenges. The search space for potential feature transformations, such as addition, logarithmic, or trigonometric operations, expands rapidly, making the search process computationally intensive. Additionally, assessing the efficacy of these transformations requires extensive model training and evaluation, further complicating the process. To address these challenges, various strategies have been developed. For instance, Cognito [90] employs a transformation tree to model the exploration space, utilizing heuristic strategies like depth-first and balanced traversal for efficient search. A reinforcement learning approach, using a Q-learning algorithm with linear approximation, has also been introduced to refine the automation of feature engineering.

Although these methods have been successful, further enhancements are needed to address two key issues:

- Choice of search hyperparameters and dynamic adaptation of heuristic strategies: Fully heuristic approaches risk convergence to local optima. Strategies such as epsilon-greedy and Upper Confidence Bound (UCB) [12] mitigate this risk but require meticulous tuning of hyperparameters to adapt dynamically.

- Sequential information in composite transformations: Current methods do not fully leverage the order sensitivity of feature transformations, often simplifying the sequence's representation with linear or convolutional neural models [91; 181].

To address these shortcomings, we present a novel algorithm called Monte Carlo tree search for Automatic Feature Engineering (mCAFE). We choose Thompson sampling as an automatically adjusting selection policy, in combination with an LSTM network to capture the sequential information in the feature transformation sequences, while the main structure follows a Monte Carlo Tree Search (MCTS) [30].

3.2 Related Work

In recent years, substantial research has focused on domain-specific feature engineering. For instance, Guo *et al.* [61] investigate how to share information through feature engineering in multi-task learning tasks, while Schelling *et al.* [142] aim to find suitable features to improve class separation. However, there has been limited new research on feature engineering methods applicable to all types of data.

FCTree, proposed by Fan *et al.* [41], utilizes both original and constructed features as splitting points in a decision tree to partition the data. It constructs local features in regions where the local error is high and the existing features are not sufficiently predictive. FEADIS [37] employs a random combination of mathematical functions, including ceiling, modulus, and sine, alongside feature selection methods to construct new features, which are then greedily selected and added to the original feature set. The Data Science Machine (DSM) [85] applies transformations to all features simultaneously, followed by the selection of features and optimization of the model in the generated dataset. A similar approach was used by Lam *et al.* [98].

In contrast, ExploreKit [88] iteratively increases the constructed features. To manage the exponential growth of the feature space, ExploreKit employs a novel ML-based feature selection approach to predict the usefulness of new candidate features. Similarly, Cognito [90] introduces a tree-like exploration of the transformation space using hand-crafted heuristic traversal strategies, such as depth-first and global-first. However, these strategies do not consider factors such as episode budget constraints. An improvement in this area was proposed by Khurana *et al.* [91], which uses a reinforcement learning-based feature engineering method to explore the available feature engineering options within a given budget. LFE [120] treats each feature individually, predicting the best transformation for each through a learning-based method. Nevertheless, none of these methods account for the order of feature transformations.

More recently, a graph-based method was proposed by Zhang *et al.* [181], which guides exploration of the transformation space using a deep neural network.

3.3 Method

We model the feature engineering problem as a classic episode-based reinforcement learning problem, where an agent interacts with the environment. The search begins from the initial state representing the original dataset $D_0 \in \mathscr{D}$, where $\mathscr{D}$ denotes the state space. From D_0, a transformation $t \in \mathscr{T}$ (action) can be chosen to transform the dataset, applying t to all the features contained within it. The new state D' is then obtained by concatenating the data of the old state D with $t(D)$, i.e., $D' = [D, t(D)]$. This ensures that the new state retains all information from previous states, reflecting the Markov property.

For each state D, a ML model can be trained to obtain its n-fold cross-validation performance. Since our goal is to find the optimal sequence of length L, we define a feature engineering pipeline as an ordered sequence $(t_1, \cdots, t_i, \cdots, t_L)$ consisting of L transformations. The i-th entry of the sequence represents the decision in the i-th step, i.e., the transformation applied to the data to generate new features.

Overall, the environment can be summarized by the 3-tuple $(\mathscr{D}, \mathscr{T}, r)$, where $\mathscr{D}$ represents the state space, $\mathscr{T}$ represents the transformation (action) space and $r \in \{0, 1\}$ represents the rewards. The reward r indicates whether a transformation pipeline of length L has improved over the best-performing model found so far.

MCTS refers to a category of sampling-based algorithms designed for tree search, which are utilized to identify optimal decisions within extensive search domains. This method has been effectively implemented in various related issues, such as feature subset selection [46]. To manage large search spaces, MCTS represents the search area as a tree structure and explores it iteratively. It incrementally prioritizes the most promising sections of the search tree based on a given evaluation function.

Evidently, our search space for feature transformations can be represented as a tree, enabling the application of MCTS to find an optimal transformation pipeline for constructing features on the original dataset. We will discuss the construction of this tree, the selection policy (Thompson sampling), and a surrogate model-based expansion policy using an LSTM. Finally, we will outline the overall mCAFE algorithm.

3.3.1 The Transformation Tree

We illustrate the reinforcement task using a transformation tree with a maximum depth L, where each node represents a state (dataset), each edge represents an action (transformation), and each path from the root to a leaf node represents a feature engineering pipeline. Each edge in the tree is associated with a distribution that indicates the mean success probability (reward = 1) of taking the action in its parent state.

The nodes in the tree are divided into two categories:

- **Root node** D_0: This is the initial state for each pipeline and represents the original dataset.

- **Derived nodes** D_i, where $i > 0$: Each derived node has one parent node D_j, where $i > j \geq 0$. The connecting edge corresponds to the action $t \in \mathscr{T}$ applied to the parent node, i.e., $D_i = [D_j, t(D_j)]$.

In this way, we translate the feature engineering problem into the task of exploring the transformation tree to find the node that maximizes the expected reward.

Figure 3.1 shows a complete transformation tree for a pipeline with $L = 2$ and two available actions, $\mathscr{T} = \{\log, \text{add}\}$. Each node in the tree represents a candidate dataset for the feature engineering problem. For example, the derived nodes D_4 and D_5 represent the following datasets:

$$D_4 = \{D_0, \text{add}(D_0), \log(D_0), \log(\text{add}(D_0))\},$$
$$D_5 = \{D_0, \log(D_0), \text{add}(D_0), \text{add}(\log(D_0))\}.$$

Because the order in which the transformations are applied is different, although the transformations in D_4 and D_5 are the same, the resulting datasets are not identical.

We can find the optimal node by traversing this tree. However, the complexity of this task grows exponentially as L and the number of available transformations $|\mathscr{T}|$ increases. Since traversing all possible nodes of the tree is prohibitive, mCAFE focuses on optimizing the selection policy π_s and the expansion policy π_e to reduce the number of evaluations required to find a good transformation sequence.

3.3.2 The Selection Policy

The selection policy π_s determines the balance between exploration and exploitation, guiding the selection of known parts of the MCTS. The Upper Confidence Bound (UCB) and ε-greedy are the two most commonly used selection policies, both of which come with strong theoretical guarantees on regret [1].

---

[1] the amount lost for not selecting the optimal action in each state

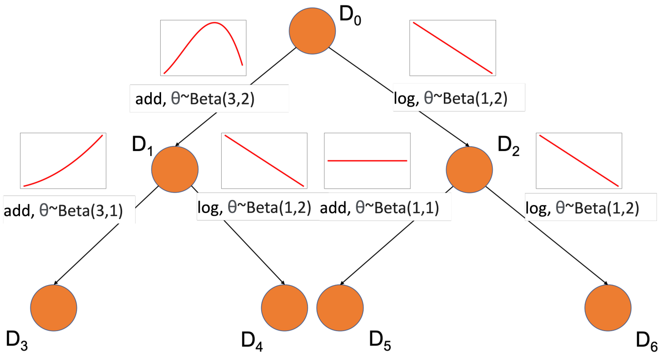


Figure 3.1.: Representation of feature transformation with a tree structure. Here, each node corresponds to a state D and each edge corresponds to a transformation (action). The distributions on the edges show the distribution over the mean success (reward = 1) probability when taking the action in the parent state.

While these policies have proven successful in various reinforcement learning settings, they are not ideal for feature engineering applications. This is mainly due to their requirement to explicitly define the exploration-exploitation trade-off through parameters like ε in ε-greedy and λ in UCB. Furthermore, ε-greedy does not dynamically adapt this trade-off, always maintaining a fixed $\varepsilon\%$ exploration rate.

To address these problems, we use Thompson sampling as the selection policy. Here, we introduce Thompson sampling and adapt it to the feature engineering case.

Consider the state space $\mathscr{D}$, the action space $\mathscr{T}$ and rewards $r \in \{0,1\}$. Thompson sampling selects an action based on the probability of it being the optimal action. Representing the set of N observations $\mathscr{O} = \{(r,t,D)\}^N$, where $D \in \mathscr{D}$, $t \in \mathscr{T}$, we model the probability of different rewards of each action with a parametric likelihood distribution $p(r|t,D,\theta)$ depending on the parameters θ. The prior distribution of these parameters is denoted by $p(\theta)$. Consequently, the posterior distribution given a set of observations $\mathscr{O}$ can be calculated using Bayes rule, i.e., $p(\theta|\mathscr{O}) \propto p(\mathscr{O}|\theta)p(\theta)$. Thompson sampling implements the selection policy π_s by sampling a parameter θ from the posterior distribution $p(\theta|\mathscr{O})$, and taking the action that maximizes the expected reward. Hence,

$$\pi_s(D) = \underset{t \in \mathscr{T}}{\operatorname{argmax}} \quad \mathbb{E}\left[r|t, D, \theta\right] \quad \text{where} \quad \theta \sim p\left(\theta|\mathscr{O}\right). \tag{3.1}$$

Since, in the case of feature engineering, each state $D \in \mathscr{D}$ satisfies the Markov property, we can simplify the problem of which action to take on state D, to whether taking the action $t \in \mathscr{T}$ leads to a performance improvement. This can be modeled as a classic Bernoulli bandit problem, where the variable $\theta = (\theta_1, \theta_2, \cdots)$ denotes the expected values of a Bernoulli random variable expressing the probability of taking the selected action in a given state (and obtaining a reward of one). The distribution of the parameter θ_t can be modeled through a beta distribution

$$p(\theta_t|\alpha, \beta) = \tfrac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta_t^{\alpha-1}(1-\theta_t)^{\beta-1},$$

where Γ is the Gamma function. $\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}$ serves as a normalization constant that ensures that the integration of the density function over (0,1) is 1. The parameters α and β control the shape of the distribution and the mean of the distribution is $\frac{\alpha}{\alpha+\beta}$. It denotes the expectation that taking the corresponding action will lead to a performance improvement. The higher α, the larger the mean and therefore the probability of the action being selected. On the other hand, the larger β, the lower the probability.

The beta distribution is conjugate to the Bernoulli distribution (i.e., the posterior distribution $p(\theta|\mathscr{O})$ inherits the functional form of the prior distribution $p(\theta)$).

Given an observed sample $O = (r, t, D)$, the posterior distribution of the parameters θ is given by

$$\theta_{t'} \sim \text{Beta}\left(\alpha + \mathbb{1}_{r=1,t'=t}, \beta + \mathbb{1}_{r=0,t'=t}\right), \quad t' \in \mathscr{T}. \tag{3.2}$$

The parameter α is incremented when the action leads to an improvement in performance. Otherwise, the parameter β is incremented. In this view, α represents the number of successes in the Bernoulli trial and β represents the number of failures.

Furthermore, the support of the beta distribution is $(0,1)$, independent of parameterization. This ensures that there is always a non-zero probability for each action to be selected.

Consequently, there is always a non-zero probability to take each path in the tree.

Figure 3.1 shows an example of the tree representation. Each edge in the tree maintains a beta distribution Beta(α,β). By comparing the two transformations on D_0 with the same β, we can see that the higher the value of α the more the distribution is shifted towards sampling larger values (higher probabilities of success).

In each step, an edge is selected based on the sampling result. This ensures the priority of high-quality edges while also allowing inferior edges to be selected occasionally.

By using Thompson sampling as the selection policy, we avoid choosing hyperparameters to balance the exploration and exploitation trade-off. In contrast, the trade-off is adjusted dynamically through the posterior distribution of the parameter θ, which is updated along with the observation. Even though α and β represent hyperparameters, their choice is arguably more intuitive as $\alpha = \beta = 1$ describes a uniform distribution.

The requirement to construct and sample from a beta distribution for each action may rise efficiency concerns, as this process is slow compared to an ε-greedy selection.

However, this is not an issue for feature engineering as in each episode, the selection phase takes little time compared to the other phases of the algorithm. This will be further explored in Section 3.4 (see Table 3.1).

3.3.3 The Expansion Policy

The selection policy π_s guides the selection of actions in parts of the search space that have been explored. Outside of the explored search space and beyond the leaves of the MCTS, the expansion policy π_e guides the selection of actions t. It expands the child nodes to the tree and selects the one with the maximum expectation reward (Q value) as the next exploration candidate

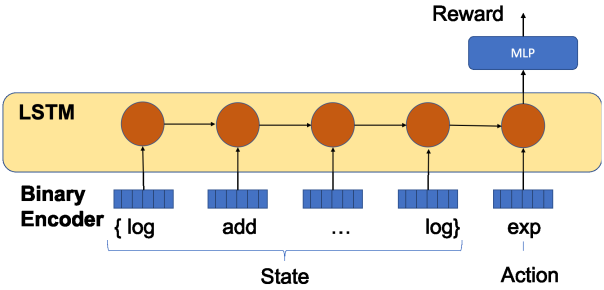


Figure 3.2.: The Surrogate network consist of 2 LSTM layers of size 32 and a two fully connected layers of size 32 with a ReLU activation function.

$$\begin{aligned} \pi_e(D) &= \underset{t\in\mathscr{T}}{\operatorname{argmax}} \quad \mathbb{E}\left[r|t,D,\theta\right] \\ &= \underset{t\in\mathscr{T}}{\operatorname{argmax}} \quad \hat{Q}(D,t). \end{aligned} \tag{3.3}$$

Given the vast state space $\mathscr{D}$, directly calculating the expectation is infeasible. To address this, mCAFE employs a surrogate network $\hat{Q}(D,t)$ as depicted in Figure 3.2. This network takes the selected action t and the action sequence used to generate the leaf node state D as input, and outputs the expected reward of taking this action in the given state.

The surrogate network comprises three key components:

- **Binary Encoder**: This component encodes an action into a binary code.
- **LSTM Layer**: Featuring a hidden size of 32, this layer handles varying input sequence lengths and captures their sequential information.
- **Fully Connected Layer**: With an input size of 32 and a ReLU activation function, this layer maps the LSTM output to the expected reward.

Each edge in the tree maintains a beta distribution. We collect training data from all existing edges in the tree and update the surrogate model after each iteration (episode).

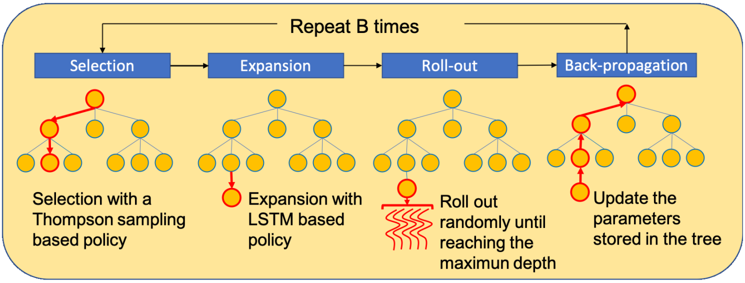

Repeat B times
Selection
Expansion
Roll-out
Back-propagation
Selection with a Thompson sampling based policy
Expansion with LSTM based policy
Roll out randomly until reaching the maximun depth
Update the parameters stored in the tree

Figure 3.3.: The mCAFE framework: each iteration (episode) includes four phases: selection, expansion, roll-out, back-propagation. B is the number of iterations.

Utilizing the surrogate model $\hat{Q}(D,t)$, the expansion policy is defined by selecting the action t that maximizes the expected reward as predicted by the surrogate model.

3.3.4 The mCAFE Algorithm

The mCAFE algorithm employs MCTS to explore the target space, with the selection policy gradually favoring actions that lead to more promising regions of the search space. This approach aims to find the optimal sequence of actions. The four main phases of the general MCTS scheme have been modified as follows (Figure 3.3):

- **Selection** Starting from the root node, mCAFE iteratively selects the child nodes according to the selection policy π_s until a leaf node is reached.

- **Expansion** At a leaf node of the transformation tree, all available child nodes are expanded. One of these nodes is then selected for further exploration according to the expansion policy π_e.

- **Roll-out** Rather than evaluating the performance of the current node, mCAFE focuses on whether the expected performance of its descendant nodes exceeds the best performance achieved so far. To achieve this, mCAFE combines n-fold cross-validation with the general roll-out process as follows. Assuming that the current node is at depth l in the MCTS tree, mCAFE completes the feature engineering pipeline by sampling $L-l$ transformations from $\mathscr{T}$ randomly with replacement, where L is the predefined length of the pipeline (transformation sequence). This process is repeated n times to generate n different pipelines, corresponding to the number of iterations in n-fold cross-validation. A reward $r = 1$ is returned if the mean evaluation score of these transformation sequences is higher than the best performance so far; otherwise, $r = 0$.

- **Back-propagation** The reward from the roll-out process is back-propagated along the path from the node selected in the expansion process to the root node. During this back-propagation, the parameters α and β in each edge on the path are updated according to the update rule described in Section 3.3.2.

The algorithm stops after the computational budget is exhausted, e.g. the algorithm stops when the number of episodes reaches 100 in the experiment.

Figure 3.4 illustrates an episode of the mCAFE algorithm. Starting from the root node D_0, the algorithm selects the explored nodes according to the selection policy π_s until it reaches the leaf node D_4. Next, an unexplored node D_7 is selected and expanded into the tree according to the expansion policy π_e.

If the depth l of the current node (expanded node D_7) is smaller than the predefined pipeline length L (maximum depth), an action is selected randomly and applied to the current node to create a new node, which then becomes the new current node. This process is repeated until the depth l of the new node exceeds the pipeline length L.

Finally, the current node is evaluated and its reward is back-propagated, updating the parameters of the beta distributions along the path from D_0 to D_7. Since $r = 1$, the α parameters of all edges along the path are incremented, while the β parameters remain unchanged. Consequently, the beta distribution of each edge in the path shifts slightly to the right, increasing the probability of selecting the corresponding actions in future iterations.

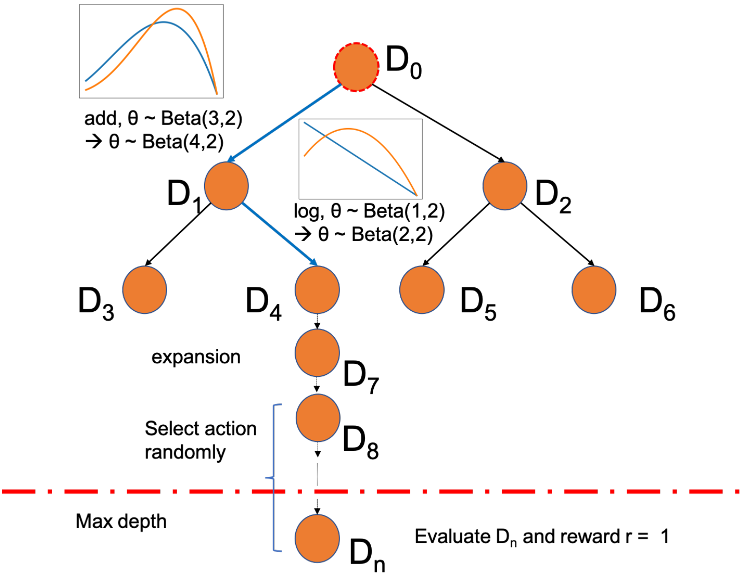


Figure 3.4.: Example of an episode of mCAFE. The beta distributions of the edges in the selected path are displayed next to the corresponding edge. Blue denotes the distribution before back-propagation and orange after back-propagation.

3.4 Evaluation

In this section, we design six different experiments to address the following questions: *(i)* How well does the mCAFE approach compare to the state-of-the-art [91]? *(ii)* Is the sampling-based selection policy necessary for the mCAFE algorithm? *(iii)* Is the sequential information of the transformation sequence important for the prediction of the Q value? *(iv)* Is the surrogate-based expansion policy necessary for the mCAFE algorithm? *(v)* How should the hyperparameter L (pipeline length) be chosen in the mCAFE algorithm? *(vi)* How does mCAFE perform with different predictive models?

For the first five experiments, we use the same benchmarks as in [91]. We attempted to reproduce their work, but some datasets were excluded from the experiment because the results of the base model differed significantly from those reported by [91], such as 'Amazon Employ' and 'Wine Quality Red'. Additionally, datasets like 'Wine Quality White', 'Higgs Boson', 'SVMGuide3', and 'Bikeshare DC' were removed because their sizes differed from those cited in the original paper [91]. To mitigate this issue in future work, we have published our code and datasets in `https://github.com/HuangYiran/MonteCarlo-AFE.git`.

The last experiment is conducted on the AutoML benchmark datasets [51], maintaining the same hyperparameter settings as in the first five experiments for both our work and the baseline.

We use episode budgets instead of time budgets for the following three reasons:

- Dominance of Candidate Evaluation Time: In feature engineering tasks, the time spent on candidate evaluation predominates over the overall time spent. This evaluation time is inevitable for all evaluation-oriented optimization methods in feature engineering tasks. Table 3.1 shows the average percentage of time taken by mCAFE for each step in the first 20 episodes, with the roll-out phase (random transformation selection and candidate evaluation) consuming an average of 97% of the total time.
- Variability Across Datasets: The run time varies significantly across datasets due to differences in data size and sensitivity to various transformations.
- Impact of Implementation and Environment: The algorithm's implementation and the operating environment greatly influence run time.

By addressing these questions through our designed experiments, we aim to thoroughly evaluate the effectiveness and efficiency of the mCAFE algorithm across different scenarios and conditions.

For the first five experiments, we use the random forest model from the sklearn package (version 0.24) with default parameters and an episode budget of $B = 100$, following [91] to ensure comparability of results.

The pipeline length is set to $L = 4$ based on the results of the third experiment, and all beta distributions are initialized with $(1,1)$ to represent a uniform

Table 3.1.: Average percentage of time for each process in the first 20 episodes.

	Size		Time spent in percentage (%)			
Dataset	Rows	Feat.	Selection	Expansion	Roll-out	Back-propagation
SpecFact	267	44	0.01	0.04	96.94	3.01
PimaIndian	768	8	0.02	0.05	97.29	2.66
Lymphography	148	18	0.01	0.04	96.98	2.97
Ionosphere	351	34	0.01	0.06	96.37	3.56
AP-omentum-ovary	275	10936	0.01	0.02	98.57	1.40
SpamBase	4601	57	0.01	0.01	98.74	1.24

prior. To reduce computation time, we sub-sample datasets with a large number of data points, considering up to 10^4 data points per dataset. To maintain comparability, we did not tune any hyperparameters of the feature engineering algorithms for specific datasets or prediction models, except for the last experiment.

For unbalanced datasets, we apply the F1-score to assess classification performance and use 1 - Relative Absolute Error (RAE) as the metric for regression tasks, following [91]. All performances are obtained under 5-fold cross-validation, meaning the parameter n in the roll-out process is set to 5.

This setup ensures that our experiments are comparable to the baseline and provides a consistent evaluation framework for assessing the effectiveness of the mCAFE algorithm.

In the experiment, we used the transformation functions $\mathscr{T} = \{$ Log, Exp, Square, Sin, Cos, TanH, Sigmoid, Abs, Negative, Radian, K-term, Difference, Add, Minus, Product, Div, NormalExpansion, Aggregation, Normalization, Binning $\}$.

3.4.1 Performance of mCAFE

We evaluated the improvement of the mCAFE algorithm by comparing it with the following methods: *(i)* Original Dataset (Base): No feature engineering applied. *(ii)* Reinforcement-Based Model (RBM): Uses a discount factor of 0.99, a learning rate of 0.05, and an episode budget $B = 100$. *(iii)* Tree-Heuristic Model (Cognito): Employs a global search heuristic for 100 nodes. *(iv)* Random Selection: Selects and applies transformations randomly from the available transformation set to one or more features in the original dataset. If the addition of new features improves performance, the new feature is retained. This process is repeated 100 times to generate the final dataset.

The performance of these methods is summarized in Table 3.2. The results show that mCAFE achieves the best scores on all regression datasets compared to the reinforcement-based model and yields superior results on most classification datasets. However, mCAFE performs worse than the reinforcement-based model on the 'Credit Default' and 'SpamBase' datasets, with the difference on 'SpamBase' being not significant.

3.4.2 Ablation Study

The proposed selection and expansion strategies are crucial components that support the performance of the mCAFE algorithm. To verify their importance, we designed two ablation experiments.

3.4.2.1 Selection Policy. We compare the performance of the traditional UCB with ε-greedy policy (mCAFE-ucb) to the proposed model using a Thompson sampling-based selection policy (mCAFE-ts). The parameter λ of UCB is set to 1.412 as proposed by [29], and ε is set to 0.1. The mCAFE algorithm keeps the same settings as in the previous experiments. Performance for classification tasks is measured with the F1 score, and regression tasks are measured with (1 - Relative Absolute Error).

Figure 3.5 categorizes the comparison results into four categories: *(i)* mCAFE-ts achieves better results: mCAFE-ts outperforms mCAFE-ucb in terms of measured performance. *(ii)* mCAFE-ts is faster: mCAFE-ts requires fewer episodes to achieve the same result as mCAFE-ucb. *(iii)* Tie: mCAFE-ucb and mCAFE-ts achieve the same result and require a similar number of episodes (difference smaller than 5). *(iv)* mCAFE-ucb achieves better results: mCAFE-ucb obtains the same results and requires fewer episodes than mCAFE-ts.

The results in Figure 3.5 demonstrate the significance of the selection strategy. mCAFE-ts achieved better performance in 64.7% of the datasets and tied on 13.3%.

Table 3.2.: Comparing performance of without feature engineering (Base), reinforcement-based model (RBM) [91], Cognito [90], random selection, and mCAFE in 100 episodes using 15 open source datasets. Classification tasks (C) are evaluated with the F1 score, and regression tasks (R) are evaluated with (1-relative absolute error).

Dataset	C/R	Rows	Feat.	Base	RBM	Cognito	Random	mCAFE
SpecFact	C	267	44	0.686	0.788	0.790	0.748	**0.855** ± **0.036**
PimaIndian	C	768	8	0.721	0.756	0.732	0.709	**0.773** ± **0.026**
German Credit	C	1001	21	0.661	0.724	0.662	0.655	**0.764** ± **0.026**
Lymphography	C	148	18	0.832	0.895	0.849	0.680	**0.967** ± **0.016**
Ionosphere	C	351	34	0.927	0.941	0.941	0.934	**0.962** ± **0.014**
Credit Default	C	30000	25	0.797	**0.831**	0.799	0.766	0.796 ± 0.006
AP-omentum-ovary	C	275	10936	0.615	0.820	0.758	0.710	**0.831** ± **0.036**
SpamBase	C	4601	57	0.955	**0.961**	0.959	0.937	0.953 ± 0.016
Openml_618	R	1000	50	0.428	0.589	0.532	0.428	**0.743** ± **0.015**
Openml_589	R	1000	25	0.542	0.687	0.644	0.571	**0.776** ± **0.018**
Openml_616	R	500	50	0.343	0.559	0.450	0.343	**0.622** ± **0.010**
Openml_607	R	1000	50	0.380	0.647	0.629	0.411	**0.803** ± **0.010**
Opemml_620	R	1000	25	0.524	0.683	0.583	0.524	**0.765** ± **0.012**
Openml_637	R	500	50	0.313	0.585	0.582	0.313	**0.637** ± **0.021**
Openml_586	R	1000	25	0.547	0.704	0.647	0.549	**0.783** ± **0.020**

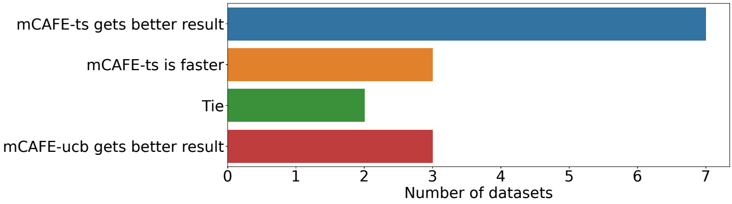


Figure 3.5.: Comparing the performance between mCAFE-ucb and mCAFE.

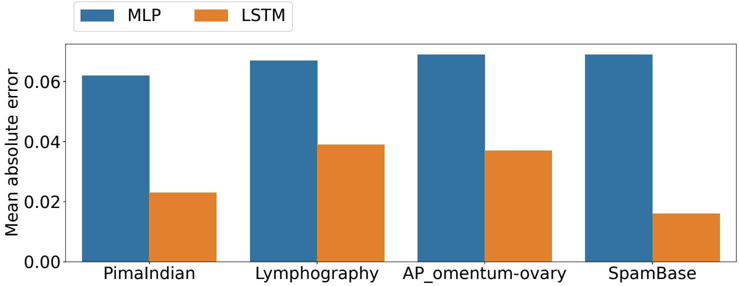


Figure 3.6.: Comparing the performance of MLP and LSTM model in predicting the Q value.

3.4.2.2 Expansion Policy. In the expansion process, we use an LSTM neural network to approximate the expectation reward (Q value) of taking an action because it can capture the sequential information of the transformation sequence. To demonstrate the importance of this sequential information for the prediction of Q values, we designed an experiment to compare the performance of using a MLP and an LSTM as the surrogate model in mCAFE.

To ensure comparability, the MLP model used here contains two hidden layers, each with 76 dimensions, resulting in a similar number of parameters as the LSTM surrogate model mentioned earlier. The evaluation criterion is the Mean Absolute Error (MAE), with a lower value indicating a better model. Both models are trained for 100 epochs. Figure 3.6 shows that the LSTM model achieves significantly better results than the MLP model across all datasets.

To assess the contribution of different expansion policies to mCAFE, we compared the performance of three models: mCAFE with an LSTM-based expansion policy, mCAFE with a random expansion policy, and mCAFE with a greedy expansion policy, which consistently selects the best action explored.

Each model utilized the same initial parameters as the previous experiment and was evaluated ten times in each dataset. The performance results for the regression datasets are illustrated using a box plot in Figure 3.7. Our analysis reveals that mCAFE with the neural network expansion policy achieved the best performance on all datasets except two. Specifically, mCAFE with the random policy outperformed on the 'Openml_618' dataset, while mCAFE with the fixed expansion policy excelled on the 'Openml_586' dataset. Additionally, mCAFE with the neural network expansion policy was outperformed by the mCAFE with the random expansion policy on the 'AP-omentum-ovary' dataset.

The primary distinctions among these three expansion approaches lie in their utilization of previous observations and the dispersion of selected actions. mCAFE with a greedy expansion policy selects actions based solely on their performance in the initial layer. While this method provides stability, it restricts the exploration of new transformations, likely contributing to its underperformance in most cases. Conversely, mCAFE with a neural network expansion policy leverages performance information from previous observations to predict the expected rewards of future actions, facilitating more informed and potentially advantageous decision-making.

3.4.3 Length of Feature Engineering Pipeline

The length of the feature engineering pipeline, denoted as L, determines the number of actions selected in each roll-out step and the length of the final transformation sequence. This parameter affects the mCAFE algorithm's performance, not only in terms of final results but also in time and memory consumption. Generally, a larger L increases both time and memory usage while also producing a larger number of features after transformation.

To identify an optimal L given resource constraints, we conducted an experiment comparing the performance of the algorithm with different L values.

Figure 3.4 illustrates the relationship between L and the best performance achieved by the mCAFE algorithm for six datasets. Here, $L = 0$ represents the

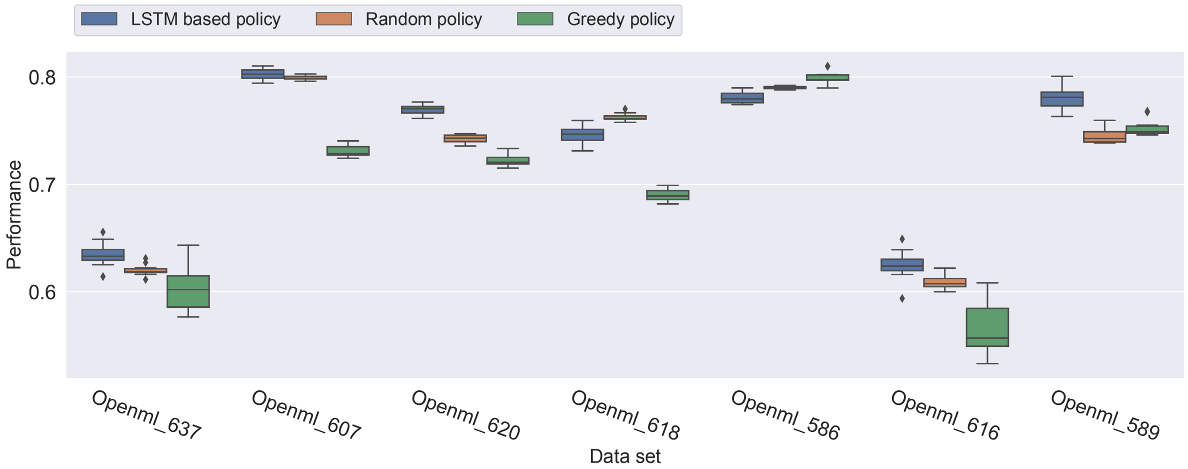


Figure 3.7.: Comparing performance of mCAFE with neural network expansion policy (with nn), mCAFE with random expansion policy (with random) and mCAFE with fix expansion policy (with fix) on all the regression dataset. Classification task is evaluated with F1-score and regression task is evaluated with (1-relative absolute error).

Table 3.3.: The performances of mCAFE with different predictive models on AutoML benchmark dataset [51]. The improvements brought by the mCAFE are shown in parentheses. Classification task is evaluated with F1-score and regression task is evaluated with (1-relative absolute error)

	Base performance				Performance with mCafe			
AutoML benchmark datasets	Rbf-svm	Linear-svm	Linear model	Decision tree	Rbf-svm	Linear-svm	Linear model	Decision tree
shuttle	0.901	0.996	0.859	**0.998**	0.996 (0.095)	0.998 (0.002)	0.996 (0.137)	**0.998** (0.001)
phpZLgL9q	0.407	0.432	**0.503**	0.454	0.407 (0.000)	0.432 (0.000)	**0.503** (0.000)	0.454 (0.000)
phpyM5ND4	0.563	0.848	0.775	**0.920**	0.752 (0.189)	0.943 (0.095)	0.893 (0.118)	**0.959** (0.039)
phpvcoG8S	0.471	0.426	0.468	**0.572**	0.561 (0.090)	**0.579 (0.153)**	0.529 (0.061)	0.578 (0.006)
phpQOf0wY	0.320	0.385	0.457	**0.698**	0.620 (0.300)	0.493 (0.108)	0.682 (0.225)	**0.698** (0.000)
phpnBqZGZ	0.016	**0.768**	0.700	0.747	0.467 (0.451)	**0.768** (0.000)	0.760 (0.060)	0.750 (0.003)
phpmPOD5A	**0.919**	0.749	0.869	0.912	**0.919** (0.000)	0.908 (0.159)	0.908 (0.039)	0.913 (0.001)
phpmcGu2X	0.930	**0.953**	0.941	0.854	**0.970** (0.040)	0.953 (0.000)	0.941 (0.000)	0.854 (0.001)
phpMawTba	0.650	0.589	0.716	**0.797**	0.802 (0.152)	0.787 (0.198)	0.787 (0.071)	**0.820** (0.023)
phpkIxskf	0.833	0.767	0.847	**0.877**	0.883 (0.050)	0.883 (0.116)	0.870 (0.023)	**0.892** (0.015)

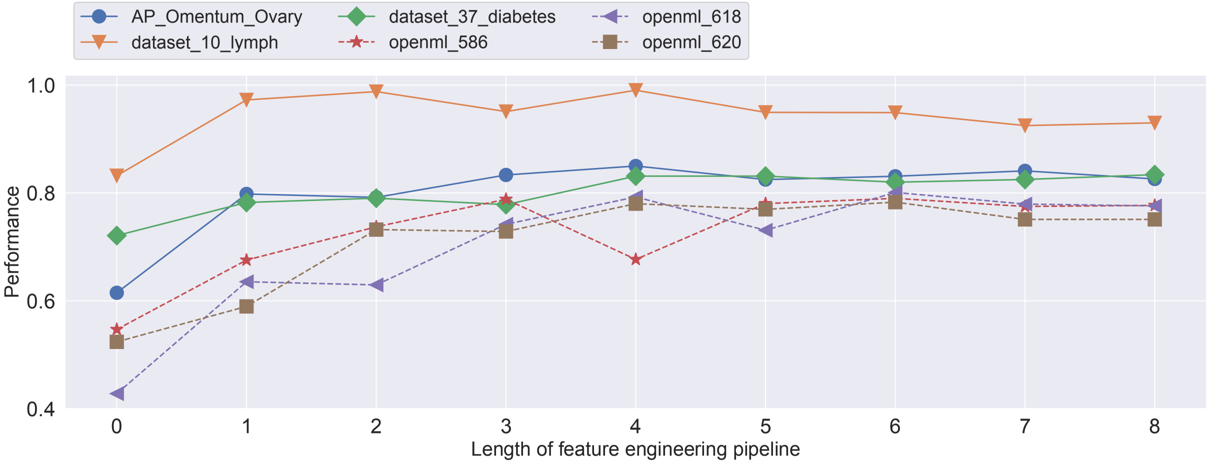


Table 3.4.: Comparing performance of mCAFE with different maximum pipeline length on 3 classification datasets (F1-score) and 3 regression datasets (1-relative absolute error).

performance of the random forest model on the base dataset. The results show that while some datasets perform well with $L = 1$, increasing L generally enhances performance. Most datasets reach peak performance at $L = 4$, though a few exhibit decreased performance, possibly due to randomness in initial selection. Notably, performance on 'Dataset_10_lymph' deteriorates with higher L, likely due to overfitting.

From this experiment, we conclude that the optimal L varies by dataset. However, $L = 4$ is generally a suitable choice for most cases.

3.4.4 Performance of mCAFE on Different Predictive Models

Different predictive models exhibit varying performance and sensitivity to mCAFE on the same dataset. To test this hypothesis, we evaluated the performance of mCAFE using the following predictive models on the AutoML benchmark datasets: Rbf-SVM, Linear-SVM, Linear Model, and Decision Tree.

Table 3.3 summarizes the experimental results. The data indicate that mCAFE enhances performance across most datasets, with feature engineering proving particularly beneficial for linear and SVM models. Notably, despite significant differences in each model's performance on the original datasets, the performance after applying mCAFE tends to converge, suggesting that mCAFE helps standardize the efficacy of different models.

3.5 Discussion

In automated and highly integrated systems, there is an increasing demand for explainability and transparency in model decision-making, especially in critical applications. However, the complexity of data in pervasive computing limits the performance of interpretable ML models, making them less effective in such environments. Additionally, traditional expert-driven feature engineering is prone to information loss and high labor costs. While AFE offers the potential to reduce human intervention and improve efficiency, it faces significant challenges in handling complex transformation sequences and dynamic search strategies. Therefore, developing new methods to enhance the performance and explainability of AFE, particularly in complex data environments, is crucial.

In this chapter, we demonstrate that existing automatic feature engineering methods can be significantly enhanced by leveraging two key insights. Our findings suggest that effective feature engineering should incorporate sequence information and composite transformations into the surrogate model, alongside an appropriate selection policy. The proposed novel MCTS-based framework employs an LSTM neural network for the expansion policy to efficiently navigate the search space. Additionally, Thompson sampling is utilized to balance exploration and exploitation in the selection policy. As a result, our approach achieves superior performance compared to state-of-the-art methods for automatic feature engineering on the majority of commonly used benchmarks. We believe that further improvements could be made by incorporating transformations that reduce redundant and irrelevant features during the construction process.

The operations used by the algorithm are straightforward mathematical operations and the tree structure effectively illustrates their interactions. Experimentally, we have shown that the algorithm achieves optimal performance at a depth of 4, making the new features generated easy to track and understand. This indicates that performance improvement does not come at the cost of explainability. By integrating this automated ML algorithm with an interpretable model, we can enhance the effectiveness of the interpretable model in handling complex tasks, allowing accurate decision interpretation.

Beyond the features used, it is understood that the choice of interpretable model and its parameters also impact the model's final performance. The pervasive computing domain imposes additional requirements on deployed models, such as specific modeling needs and performance improvements through means other than feature engineering, which we will explore in the next chapter.

4. Improve Model Performance with Parameter Optimization

In the previous chapter, we explored how Automated Feature Engineering can mitigate the challenges posed by complex data in pervasive computing (**C1**), thereby improving model predictive performance and interperatability. Building on this foundation, this chapter expands our investigation to encompass other crucial aspects of the automated ML pipeline, including feature selection, model selection, and hyperparameter tuning. Furthermore, the deployment of models in pervasive computing environments, particularly on battery-powered edge devices, is often hindered by constraints such as limited battery life, the demand for low-latency processing, and the requirement for compact model sizes (**C2**). In this chapter, we examine the use of GA to optimize feature selection, model selection, and hyperparameter tuning, taking into account these operational constraints.

Corresponding Publication:. Y. Huang, Y. Zhou, H. Zhao, T. Riedel, and M. Beigl. Optimizing automl for tiny edge systems: A baldwin-effect inspired genetic algorithm. In *22nd IEEE International Conference on Pervasive Computing and Communicaitons (PerCom 2024)*, 2024

4.1 Introduction

In the realm of pervasive computing, the integration of AI capabilities within compact edge systems is catalyzing digital transformation across diverse industries. These systems, notably Internet of Things (IoT) devices, wearables, and smart textiles, are progressively advancing and are manifesting substantial trends across various sectors. The potential applications for AI-enhanced sensors and actuators appear boundless. Nevertheless, edge devices, which are predominantly battery-operated, face significant challenges. These include the capability to run multiple applications simultaneously with constrained resources, such as limited battery life. Furthermore, the imperative for real-time services demands the deployment of low-latency models on these devices.

These operational constraints render interpretable models more advantageous compared to deep learning models due to their lower computational requirements. Despite this advantage, the practical deployment of interpretable models is hampered by their limited effectiveness. Optimization of these models generally requires extensive expert knowledge and involves labor-intensive processes such as manual hyperparameter tuning, which is not only time consuming but also frequently suboptimal.

AutoML represents a pivotal advancement in automating the ML pipeline, particularly by optimizing model performance within specific constraints. This is especially applicable to diminutive edge systems, where AutoML systematically selects features, models, and model parameters tailored to meet the stringent software and hardware constraints of these systems, which include computational cost, latency, and memory limitations.

The methodologies employed in AutoML can be broadly categorized into two distinct types: gradient- and heuristic-based search strategies. Gradient-based approaches, exemplified by Neural Architecture Search (NAS), integrate hardware considerations directly into a loss function. These systems refine the model architecture under constraints using optimization techniques such as gradient descent. This approach is evident in methodologies like NEAT [158]. Conversely, heuristic search strategies utilize a process of iterative candidate generation and evaluation. As it is often complicated to formulate the valid optimization space (such as complex resource constraints on the target hardware) and complex objective functions in NAS. Our research focuses on highly extensible methods based on GA. GAs have the additional advantage that parallelism can be easily exploited on any training hardware [115].

Despite the breadth of research conducted on GA, their application to small-scale embedded systems remains fraught with challenges. This section elucidates several primary obstacles:

- Multimodal Data Streams and Feature Complexity: Multimodal data streams, characteristic of applications like human activity recognition, often generate voluminous feature sets. For example, using tools such as TsFel [15], up to 198 features may be extracted from each channel. In the context of the PAMAP2 benchmark dataset [132], TsFel can identify as many as 3564 distinct features from a single data sample. The proliferation of features presents significant challenges for conventional GA methodologies, primarily:
  - The tendency to disproportionately focus on feature selection, thereby sidelining the optimization of other crucial parameters.
  - A notable increase in both the evaluation time and the variability of this time across different candidates. This becomes particularly evident when contrasting the evaluation times of simpler models such as K Nearest Neighbor (KNN) with more complex models like stochastic gradient descent. The latter requires approximately 25 seconds for a single evaluation on the PAMAP2 dataset, nearly 100 times longer than the former.
- Balancing Multiple Objectives: The optimization of embedded systems for target hardware requires a careful balance between several often contradictory objectives, such as model performance and latency. This balancing act tends to generate a multitude of non-dominated solutions, thereby diminishing the search efficiency typically observed with conventional GAs.
- Complex Feature Sets for Edge Tasks: Edge computing tasks demand comprehensive and intricate feature sets due to the complexity of the

tasks they are designed to perform. Daily life activities such as brushing teeth or washing dishes involve complex patterns and significant variations among subjects over prolonged durations. These activities necessitate the integration of features from diverse domains, including temporal, spatial, and frequency aspects. Traditional time series feature extraction methods, such as TsFel, struggle to adequately capture these multidimensional features, rendering them insufficient for effectively processing such complex data.

- Parameter Optimization in Embedded ML Architectures: The optimization of parameters within embedded ML architectures involves an intricate interplay of diverse parameter types and their interdependencies. Parameters specific to certain models may vary widely; while many are numerical, choices related to feature and model selection often fall into categorical types. Consequently, these differing parameter types necessitate tailored optimization strategies to ensure effective handling and integration within GA frameworks.

These challenges underscore the need for refined GA methodologies that can adapt to the constraints and complexities inherent in embedded system applications.

Building on the identified challenges, this chapter presents a novel adaptation of GA tailored to AutoML challenges within constrained edge environments. This refined approach acknowledges and leverages the interdependence of model parameters. It conceptualizes the optimization of individual parameters as analogous to the evolutionary development witnessed in natural populations, mirroring the Baldwin effect where individual learning can influence genetic evolution. This methodology ensures efficient resource use by promptly eliminating suboptimal candidates early in the process and employs techniques such as process pooling and early selection to markedly reduce the duration of fitness evaluations, thus curtailing prolonged waiting times.

Furthermore, the proposed method resolves conflicts between multiple optimization objectives by integrating them into a coherent set of goals with varied priorities. To overcome the traditional constraints of feature extraction in ML, we have developed a specialized neural network designed to extract and leverage features across spatial, temporal and frequency domains. This enhancement not only broadens the scope of detectable features, but also substantially improves the overall capability of the feature extraction process, making it more suited for complex edge computing tasks. This integration of innovative GA strategies and advanced neural network designs represents a significant step forward in optimizing ML applications for tiny embedded systems.

4.2 Related Work

GA are algorithmic frameworks inspired by the principles of natural biological evolution, aimed at discovering optimal solutions by emulating natural selection and genetic variation. A typical GA involves several critical stages: coding, population initialization, fitness evaluation, selection, crossover, and mutation. These algorithms utilize a genetic space to represent potential solutions indirectly, since they cannot interact directly with the actual parameters of the problem space. For instance, in feature selection tasks, individual features might be represented by Boolean values to denote their presence or absence in the model training process.

Recent advances have demonstrated the versatility of GAs across various applications. Magdum *et al.* [112] developed an enhanced GA to optimize Artificial Neural Network (ANN) performance for Optimal Power Flow (OPF) applications, focusing on selecting the most effective weights and biases to reduce error rates and operational costs. Similarly, Ali *et al.* [6] implemented a hybrid approach combining filter-based feature selection methods such as information gain with GAs to refine feature selection further, significantly improving cancer classification accuracy in complex microarray datasets. These applications, however, are highly specialized and might not be readily transferable to other contexts due to their task-specific designs.

At the frontier of algorithmic innovation, Blanchard *et al.* [21] introduced a novel approach utilizing masked language models to facilitate mutation generation in GAs. This technique optimizes the generation of molecular string representations for applications in drug development, identifying common subsequences within genomes to create a vocabulary that aids in tokenizing and rearranging genomes. Although this method has shown promise in molecu-

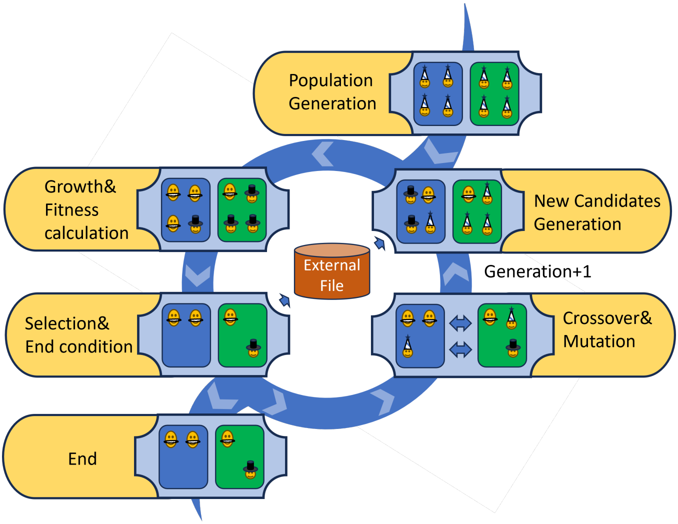


Figure 4.1.: The pipeline of the proposed algorithm.

lar applications, its adaptation to time series data remains challenging due to the absence of consistent patterns and the intensive computational resources required, suggesting the need for further investigation.

The task-dependent nature of GAs, underscored by the necessity to tailor the coding of individuals to specific applications, highlights the importance of developing specialized GA designs, particularly for applications in constrained environments like tiny edge systems.

4.3 Method

The proposed algorithm represents a specialized adaptation of GA tailored for tiny edge systems, adhering to the conventional GA process. Figure 4.1 delineates the algorithm workflow. In this section, we will systematically unveil the method, aligning with the sequential stages of the GA's workflow.

4.3.1 Individual Coding Design

The algorithm proposed here is designed to augment the effectiveness of computational models within small-scale edge devices by optimizing the entire training process. This optimization involves the meticulous selection and tuning of features, models, and their respective parameters.

Traditionally, in GA, each entity is characterized as a sequence whose elements correspond to parameter values within a defined problem space. However, when such algorithms are employed in specific applications like Wearable Human Activity Recognition (WHAR), they face notable challenges. These challenges include the variability in sequence length, which depends on the model chosen, and an expansive search space for feature selection, which arises from handling multimodal data sources.

In the refined approach, the optimization of model parameters is explicitly separated from other parameters. In this method, individuals are encoded using a Boolean array to manage feature selection and an integer that specifies the model choice. Building on this framework, each individual is equipped with a mechanism that adjusts the parameters of the selected model and computes a fitness score. This score evaluates the performance of the model in relation to its designated objective. Importantly, the tuning of model parameters is not incorporated into the individual's genetic makeup, and hence is excluded from the genetic operations of crossover and mutation. Rather, it is considered an external reflection of individual development. As the algorithm progresses, these model parameters are refined incrementally. Moreover, each individual includes an additional attribute: a 'result' table. This table consists of two columns that record the frequency of parameter adjustments within the individual and the corresponding peak fitness scores obtained.

4.3.2 Populations Generation

The optimization of models for tiny edge devices involves inherent complexities due to their multi-objective nature, which often encompasses conflicting goals. To address this challenge, one viable strategy is the normalization and integration of these sub-objectives into a unified framework. Drawing an analogy to the categorization of academic disciplines into arts and sciences in China, we propose the creation of distinct groups within the population. Each group is directed by a unique combination of sub-objectives, prioritized differently to focus on particular performance aspects.

This methodology allows specific groups to concentrate on individual sub-objectives, thereby maintaining an overall balance within the system. Furthermore, by differentiating these groups at the outset and promoting interactions during the crossover phase of the GA, the approach aims to foster a synergistic effect. This synergy is anticipated to yield progeny with enhanced traits, simultaneously ensuring the retention of genetic diversity across the population.

4.3.3 Individual Growth and Fitness Calculation

In contrast to conventional GA, where an individual's fitness score is determined at inception, the proposed methodology facilitates the dynamic optimization (growth) of individuals. This is accomplished through the continuous refinement of the model parameters, predominantly numerical, using Bayesian Optimization (BO). BO leverages a prior probability distribution to model the behavior of the objective function, which is iteratively updated by assessing candidate points within the search space, thereby gradually aligning the model with the actual distribution of the objective function.

Furthermore, we have integrated the concept of 'age' to monitor the number of evaluations each individual undergoes via BO. All individuals initially have an age of zero. During each iteration of our method, individuals younger than a predefined age threshold are subjected to the optimization process. The evolution of fitness scores alongside the corresponding ages are meticulously recorded in a designated fitness table termed 'results'.

Given the extensive data samples and the high number of features per sample characteristic of tasks in edge devices, the BO optimization process for each individual is notably prolonged. To efficiently manage this, we employ a process pool that parallels individual growth, aligning the size of the pool with the number of CPUs on the device. As shown in Figure 4.1, individuals are sequentially enqueued into the process pool, and as computing resources become available, the next individual is processed. This queue order is predicated on their initial order of entry into the pool.

It is critical to recognize that not all individuals are subjected to the selection process in each iteration. The substantial variance in growth durations among individuals suggests that waiting for the completion of all growth processes before advancing to the selection phase would be impractical. Therefore, we implement an 'early selection' strategy within the process pool. Specifically, a threshold is established: once the number of individuals that have completed their growth surpasses this threshold, they are immediately advanced to the selection stage. Those that remain in the pool are deferred to the subsequent selection cycle, ensuring an efficient progression of the optimization process.

4.3.4 Individual Selection

In this study, each participant is characterized by a unique 'results' fitness table, the dimensions of which vary due to the differing experiential backgrounds of the individuals involved. To facilitate a fair comparison and ranking of these variably-sized matrices, a methodological preference is instituted. Specifically, enhanced scores are given priority among younger participants, while a wealth of experience benefits older participants. This approach aligns with the ranking protocol delineated in Algorithm 1. Subsequent to the ranking process, individuals positioned lower in the hierarchy are archived. This archiving process is thorough, encapsulating all relevant data, such as feature and model selections, optimal parameters, and peak fitness scores, which are meticulously documented in an external file for subsequent utilization. This comprehensive repository of data is crucial for the creation of new individuals in the study.

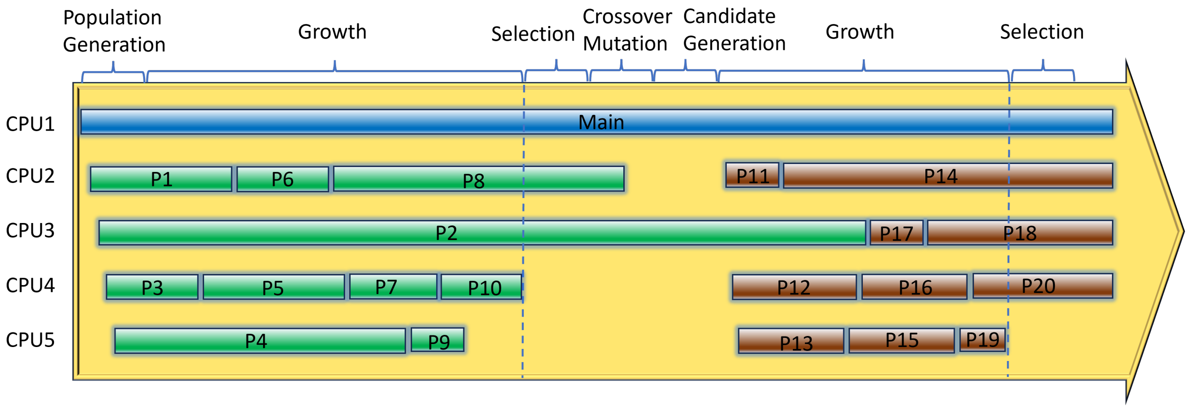


Table 4.1.: The processes executed in process pool.

Algorithm 1 Individual Selection

```
Require: $results$: A dictionary with DataFrames, $ratio$: Ratio of Individual to keep
Ensure: $li\_keep$: list containing individual ids to keep
$all\_ages$ ← sorted unique ages from $results$
Initialize $rank\_df$ as a DataFrame with index as $all\_ages$
for each $age$ in $all\_ages$ do
    Initialize lists: $scores$, $ids$
    for each $id, df$ in $results$ do
        if $age$ is in $df['age']$ then
            $score$ ← score for the given $age$ in $df$
            Append $score$ to $scores$ and $id$ to $ids$
        end if
    end for
    $ranks$ ← rank $scores$ in descending order
    for each $id, rank$ in $ranks$ do
        Assign $rank$ to $rank\_df$ for the corresponding $age$ and $id$
    end for
end for
$average\_ranks$ ← mean of $rank\_df$
$li\_keep$ ← top items of $average\_ranks$ based on $ratio$
return $li\_keep$
```

4.3.5 Crossover and Mutation

In this research, we implement a dual-tier crossover strategy, integrating both intra-population and inter-population exchanges, as specified in Algorithm 2. Candidate selection within each population is conducted through a randomized process, and the transfer of characteristic and model values follows a predetermined probability distribution. This stochastic approach is preferred over a deterministic half-swapping method. The rationale for this preference is based on observations from the TsFel framework, where there is a tendency to cluster correlated features. Such clustering results in the concentration of high-quality features within specific regions of the feature array, which could reduce the efficacy of a straightforward half-swapping strategy.

Algorithm 2 Crossover for Feature

```
Require: $individual1, individual2$
Ensure: $child1, child2$: Two new feature arrays derived from the parents
$parent1 \leftarrow individual1.features$
$parent2 \leftarrow individual2.features$
Initialize $child1$ and $child2$ as zero arrays of the same shape as $parent1$ and $parent2$
Generate a $mask$ bool array with random value of the same length as $parent1$
for each index $i$ in the range of length of $parent1$ do
    if $mask[i]$ is 1 then
        $child1[i] \leftarrow parent2[i]$
        $child2[i] \leftarrow parent1[i]$
    else
        $child1[i] \leftarrow parent1[i]$
        $child2[i] \leftarrow parent2[i]$
    end if
end for
return $child1, child2$
```

The mutation process in our GA is intricately designed to complement the crossover strategy, operating under a predefined probability that dictates the occurrence of feature and model mutations. When activated, this mutation induces deviations in the selected feature or model values from their original configurations. Such deviations introduce novel variations into the GA search process, thereby enhancing the exploratory capabilities of the algorithm and potentially improving solution diversity and robustness.

4.3.6 New Individual Generation

Given the expansive search space inherent in GA, relying solely on the crossover and mutation mechanisms often results in suboptimal exploration efficiency. To address this limitation, we incorporate heuristic methods that leverage data from the selection process, thereby facilitating the generation of new individuals and significantly enhancing search efficiency.

Moreover, we have developed tailored strategies for the creation of feature and model candidates, which are based on the unique characteristics of individual components. Drawing inspiration from [72], we employ two distinct MLP networks for the generation of feature candidates. The first network acts as a generator, employing Gaussian noise to produce a one-dimensional float array of values within the [0,1] interval. A predefined threshold is then applied to this output to formulate a candidate feature array. The second network serves as an evaluator, which receives a feature array and a model selection value as input, subsequently producing a single float value that estimates the fitness score of the specific feature-model combination. This bifurcated approach allows for a more nuanced and effective generation and evaluation of potential candidates within the GA framework.

We refine the functionality of our GA by integrating two specialized networks: the generator and the evaluator, the latter being trained using data from an external file and a Mean Standard Error (MSE) loss function. Upon completing the training of the evaluator network, its weights are fixed to maintain consistency in fitness estimation. The generator network, which inputs Gaussian noise to produce candidate features, is then coupled to the fixed evaluator. In this setup, the output of the generator, combined with a randomly chosen model value, forms the input for the evaluator. We enhance the generator's performance by optimizing it to maximize the output of the evaluator, utilizing the straight-through estimator to facilitate effective gradient descent between the two networks. This strategy ensures a diverse array of feature candidates due to the stochastic nature of the generator's input.

For model selection, we employ the UCB strategy, detailed in Huang *et al.* (2023). This approach utilizes aggregated data from the external file, including the total number of trials v, the number of times each model i has been assessed (v_i), and their mean fitness scores f. The selection score for each model i is calculated as follows:

$$score_i = f + \gamma\sqrt{\frac{\log v}{v_i}},$$

where γ represents the exploration weight. This formula facilitates a strategic balance between exploiting known models and exploring those less frequently used, optimizing the search for the most effective models within the GA framework.

The proposed method iterates the above processes, growth, selection, crossover, mutation, and individual generation, until a predefined condition is reached.

4.3.7 Neural Network Feature Extraction

The complexity of multimodal data processed by diminutive edge systems necessitates robust feature extraction capabilities. To address this challenge, we devised a streamlined neural network architecture, depicted in Figure 4.2. This architecture comprises three distinct branches, each equipped with a dual-layer convolutional setup, where the output channel size is fixed at 16. The network is engineered to extract features reflecting the temporal, spatial, and frequency dimensions of the multimodal data. These extracted features are then integrated to facilitate accurate target prediction. As illustrated, the parameters T, S, and c represent the time duration for each prediction sample, the number of sensor channels, and the number of channels output by each convolutional layer, respectively. Post-training, the features harvested from each branch are combined with those derived via ML techniques, forming a comprehensive set of candidate features for subsequent training of the target model.

4.4 Evaluation

In this section, we outline an experimental framework designed to assess the effectiveness of our proposed algorithm. The primary objectives of the experiment are twofold: first, to benchmark the performance of our algorithm against contemporary state-of-the-art GA; and second, to dissect the influence of each discrete component within our algorithm. This systematic decomposition will facilitate a detailed understanding of how individual elements contribute to the aggregate efficacy of the algorithm. This approach not only substantiates the algorithm's performance, but also elucidates the functional importance of its

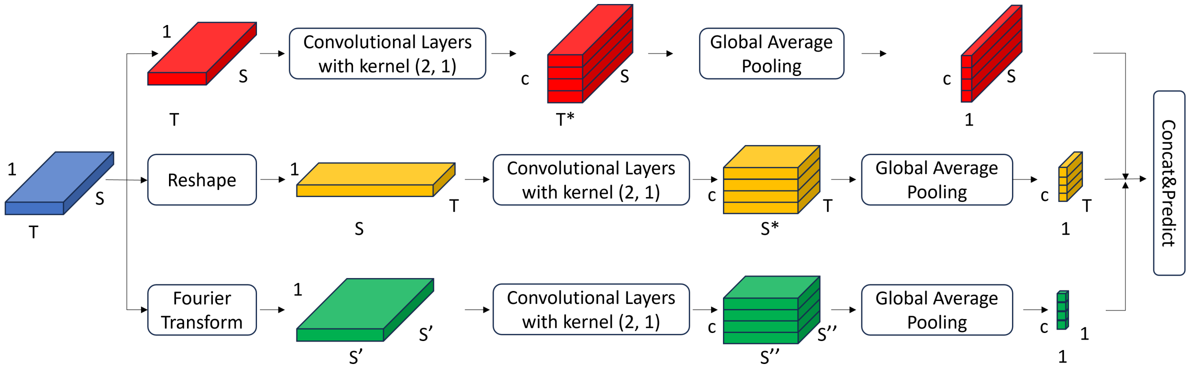


Table 4.2.: The neural network for feature extraction.

constituent components.

4.4.1 Benchmark Models

To validate the efficacy of the proposed model and elucidate the functional contributions of its components, we undertake a comparative performance analysis involving several algorithmic configurations: *(i)* Baseline Algorithm: Utilizes a conventional GA to establish a performance benchmark. *(ii)* Neural Network Algorithm (nn): Implements the three-branched neural network proposed in our model. *(iii)* No Suggestion Algorithm (noSuggestion): Operates the proposed algorithm devoid of neural network recommendations. *(iv)* No Growth Algorithm (noGrowth): Executes the proposed algorithm while omitting the individual growth component. *(v)* No Neural Network Algorithm (noNN): Deploys the proposed algorithm without integrating features generated by the specifically designed neural network. *(vi)* No Early Selection Algorithm (noEarlySel): Tests the proposed algorithm by extending the fitness calculation until all tasks in the job pool are completed. *(vii)* Proposed Algorithm: Engages the fully integrated version of the proposed algorithm as intended.

4.4.2 Benchmark Datasets

To evaluate the proposed method across diverse scenarios while ensuring methodological consistency with existing research, we employ two benchmark datasets extensively utilized in the field of WHAR. These datasets are: *(i)* HAPT Dataset (Human Activity and Posture Transition): A widely recognized dataset that provides comprehensive data suitable for testing activity recognition algorithms [133]. *(ii)* PAMAP2 Dataset (Physical Activity Monitoring for Aging People): Another pivotal dataset that includes a variety of sensor data specifically collected to enhance physical activity recognition algorithms [132].

The choice of these datasets allows for a rigorous assessment of the proposed method under varied conditions, facilitating direct comparison with other studies in the domain, and promoting the reliability and validity of our experimental results.

4.4.3 Experiment Setup

In this study, we utilized the Raspberry Pi Zero W, equipped with 512 MB of memory, as the primary computing platform. During the experimental phase, the maximum number of generations for the proposed algorithm was limited to 30. The selection and crossover probabilities were maintained at 0.1, while the mutation probability was set at 0.01. Each iteration of the algorithm involved five evaluations.

The algorithm was designed to optimize three sub-objectives: maximizing prediction accuracy and F1 score, and minimizing inference time. A strict constraint was imposed based on the available memory; if memory consumption exceeded 100 MB, the fitness value was adjusted to -1. Memory usage was assessed using the 'getsize' function from the 'os' package, following the serialization of the trained model via 'joblib'.

Given the challenge of detecting minute variations in inference time during practical applications, we introduce a latency fitness calculation, expressed as $(\text{latency}\%0.02)\lambda$, where λ represents the assigned weights.

The population was divided into two groups, each comprising 50 individuals. Each group aimed to achieve a composite objective, weighted as $[0.7, 0.2, 0.1]$ and $[0.2, 0.7, 0.1]$ respectively, aligning with the three sub-objectives. To enhance the stability of our algorithm, a 5-fold cross-validation approach was employed to compute the fitness scores.

To establish the model's generalizability, the dataset was partitioned into training and test sets using the leave-one-subject-out method. The training set was used to determine the optimal parameter settings, and the efficacy of the model was subsequently evaluated on the test set. For comparison, a conventional GA was configured with a maximum of 150 generations, maintaining the evaluation count consistent with our proposed model. All other parameters of this conventional algorithm mirrored those of the proposed method.

The training of the generator, evaluator and feature extraction networks employed the Adam optimizer [94], initialized with a learning rate of 10^{-3}. Batch training was conducted with a designated batch size of 16. The objective function of the feature extraction network was the cross-entropy loss [154]. The training protocol did not incorporate early stopping strategies or learning rate

Table 4.3.: Accuracy performance of the proposed algorithm

	baseline	nn	noSuggestion	noGrowth	noNN	noEarlyCal	proposed
HAPT	94.2	95.1	95.1	95.4	95.2	95.8	95.7
PAMAP2	85.3	80.1	73.3	85.1	83.5	85.8	85.6

adjustments. The training process was capped at a maximum of 50 epochs.

4.4.4 Result

The experimental results are summarized in Table 4.3. The data clearly indicate that the proposed model outperforms the baseline in both scenarios, thereby demonstrating the effectiveness of the proposed algorithm. Furthermore, it was observed that the synergistic approach, which integrates neural network features with ML features, enhances performance compared to the ’noNN’ and ’nn’ algorithms.

A significant decline in performance was detected in the ’noSuggestion’ algorithm, primarily due to model overfitting to the training dataset. The performance of the ’noEarlySel’ and the ’proposed’ algorithms are closely matched; however, a substantial difference was noted in their computational efficiency. The proposed algorithm completed its execution in approximately 21 hours, whereas the ’noEarlySel’ algorithm required 29 hours to achieve a similar outcome. This finding underscores the improved computational efficiency of the proposed algorithm. Additionally, the reference time for all models after optimization is less than 0.02 seconds.

4.5 Discussion

In pervasive computing, the growing demand for efficient and effective interpretable models, particularly within resource-constrained edge environments, presents significant challenges. The complexity of multimodal data streams and the requirement for low-latency, real-time processing often render traditional interpretable models inadequate for these settings. Additionally, the manual tuning typically required by expert-driven methods is not only labor-intensive but also frequently suboptimal. Although AutoML holds promise in automating and optimizing the ML pipeline, it encounters substantial difficulties when applied to embedded systems. These challenges stem from limited computational resources and the necessity of balancing multiple, often conflicting, objectives. Consequently, there is a pressing need to develop novel approaches that enhance the performance and adaptability of AutoML, with a particular focus on refining GA for improved deployment of interpretable ML models in pervasive computing environments.

This chapter presents a novel variant of GA specifically designed to optimize AutoML tasks within these constrained contexts, addressing the unique challenges they pose. The approach acknowledges the interdependence of the model parameters and draws inspiration from the Baldwin effect in natural evolution, where individual learning influences genetic evolution. By adopting this conceptual framework, the proposed method facilitates efficient optimization of individual parameters. To further enhance resource efficiency, the method incorporates early elimination of suboptimal candidates using techniques such as process pooling and early selection. These strategies significantly reduce the time required for fitness evaluations, thereby minimizing waiting times and improving overall efficiency. Furthermore, to address the need to balance multiple conflicting objectives, the method integrates these objectives into a coherent set of goals with prioritized weighting. This ensures that all relevant objectives are adequately considered and harmonized throughout the optimization process.

Through the proposed method, we not only address the technical challenges of optimizing AutoML in pervasive computing environments, but also ensure that the resulting models are interpretable and trustworthy. While this method paves the way for broader acceptance and trust in AI-driven systems, there is another point worth noting: the decisions of interpretable models are based on data features, some of which are inherently unfriendly and difficult to understand for non-experts. For example, changes in kinetic energy. In the next section, we will discuss the possibility of using interpretable models to obtain more accessible explanations.

5. Explain with Large Language Model

Since the launch of ChatGPT on 30 November 2022, Large Language Models (LLMs) have garnered substantial attention from both established technology companies and emerging startups. These organizations have invested heavily in LLMs, driving the fine-tuning and optimization of these models for specialized applications across various fields. In particular, LLMs customized for sectors such as healthcare, law, and finance have been significantly refined to enhance their specialization and accuracy by utilizing domain-specific datasets. Given the extensive foundational knowledge embedded in LLMs, their emergent causal reasoning capabilities, and the growing need for specialized knowledge to interpret applications within pervasive computing environments, it is crucial to explore the potential of LLMs in improving model explainability. This investigation is the primary focus of the research presented in this chapter.

Corresponding Publication:. Y. Huang, Z. Xue, H. Ma, and M. Beigl. Generate explanations for time-series classification by chatgpt. *Explainable Artificial Intelligence, Malta, 17th–19th June 2024*, 2024

5.1 Introduction

Expertise plays a pivotal role in interpreting black-box models within the realm of pervasive computing, a field characterized by the integration of numerous devices and sensors. These devices collectively gather multifaceted and heterogeneous data from varied environments, embodying rich contextual details such as user behaviors, environmental conditions, and interactions among devices. The complexity and diversity of these data necessitate profound expertise for effective comprehension and interpretation.

Moreover, the causal dynamics within pervasive computing is inherently more intricate than those encountered in simpler systems such as image recognition. In the latter case, the relationship between inputs and outputs tends to be more linear, for instance, correlating an image of a cat directly with the label "cat". In contrast, in pervasive computing, inputs such as sensor readings and the resultant system behaviors are often influenced by numerous interacting factors. Expert insight is crucial for elucidating these complex interactions, thereby enriching our understanding of model behaviors and enhancing the accuracy of interpretations.

Since its launch on 30 November 2022, the advent of ChatGPT has significantly propelled the prominence of LLMs. These models have attracted substantial investments from a broad spectrum of technology corporations and startups alike. LLMs undergo training on expansive datasets, which range in size from hundreds of gigabytes to several terabytes. These datasets are derived from a diverse compilation of sources, including books, articles, research papers, Wikipedia entries, social media updates, and news reports, encompassing a broad spectrum of disciplines such as science, technology, medicine, law, history, and literature. Through rigorous large-scale pre-training processes, LLMs assimilate and internalize a wealth of background knowledge and contextual nuances, thereby acquiring the capability to comprehend and generate high-quality content across various subjects.

Furthermore, the architecture of LLMs, characterized by deep neural networks, incorporates tens of billions of parameters. This intricate architecture endows LLMs with enhanced reasoning and expressive capacities essential for managing complex tasks. The comprehensive knowledge base and advanced computational abilities of LLMs position them as formidable tools for the interpretation of black-box models within the domain of pervasive computing, promising significant advancements in understanding and application.

To date, research on LLMs has predominantly centered on time series predictions, with a notable absence of studies addressing time series classification and its associated explainability. This gap highlights a critical area of potential development within the field. Consequently, this paper proposes to pioneer research on this underexplored topic by examining a novel methodology for generating explanations in time series classification tasks.

Our proposed method leverages ChatGPT to perform classifications on data samples and subsequently provides explanations for these classifications. It is essential to acknowledge that, while GPT models exhibit remarkable linguistic capabilities, their proficiency in processing numerical data is somewhat limited compared to conventional numerical methods. Therefore, our approach consciously accepts a minor compromise in accuracy in exchange for the capacity to elucidate the underlying reasoning of the model's classifications. The implementation code for this methodology is publicly accessible and can be found on GitHub[1].

5.2 Related Work

The landscape of time series classification has been enriched by several traditional methodologies aimed at enhancing explainability. Among these, the pioneering work of Senin *et al.* using Symbolic Aggregate approXimation (SAX) and vector space models stands out [148]. This technique facilitates the ranking of time series patterns according to their significance, laying the groundwork for further innovations in interpretable time series classification such as Feature Importance Ranking (FIR). While this method advances explainability, it does not necessarily achieve optimal classification performance.

Building on the principles established by the Local Interpretable Model-agnostic Explanations (LIME) framework [134], Guillemé *et al.* introduced an innovative approach known as the Local Explainer For TIme Series classifi-

---

[1] `https://github.com/lab992/Generate_explanations_for_classification_by_ChatGPT`

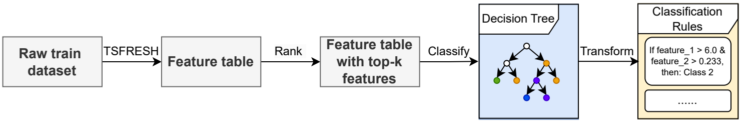


Figure 5.1.: Extract classification rules from train dataset

caTion (LEFTIST) [59]. As the first model-agnostic tool in this domain, LEFTIST is designed to provide explanations for predictions made by any time series classifier. It demonstrates that these explanations can significantly aid users in understanding classifications, particularly in straightforward scenarios. This development marks a significant step forward in making time series classification both more accessible and understandable.

In a similar vein to prior advancements, Torty *et al.* [157] have introduced the LIMESegment framework, which represents a significant enhancement over the current state-of-the-art modifications of the Local Interpretable Model-agnostic Explanations (LIME) specifically tailored for time series classification, such as those developed by Neves *et al.* [122]. LIMESegment is recognized for generating explanations that are not only more faithful, but also more robust compared to its predecessors [157].

Our research underscores the vital importance of local explanations in the realm of time series classification. Through detailed analysis of data patterns, these local explanations prove to be both dependable and insightful. Building on this foundation, we advocate for an innovative approach that integrates LLMs with local explanatory mechanisms. This integration aims to enhance the reliability of the explanations while maintaining their comprehensibility, thereby facilitating a deeper understanding of time series classification processes. This proposed synergy could potentially bridge the gap between advanced computational models and user-centric explainability.

5.3 Method

5.3.1 Components of Prompt

The methodology proposed here involves a nuanced process of transforming time series data into a narrative of movements. This narrative serves as a basis for ChatGPT to categorize the data into distinct classes. Subsequently, ChatGPT elucidates the reasoning behind its classification, providing insight into the decision-making process.

To generate a comprehensive response from ChatGPT, the construction of a well-formulated prompt is crucial. A prompt typically consists of two primary components: context and query. The context part provides essential background information and outlines the classification rules. This background details the data measurement scenario, equipping ChatGPT with a thorough understanding of the task at hand. This clarity is instrumental in enabling ChatGPT to comprehend the context and goals associated with the classification challenges it faces. The rules of classification included in the context serve as a directive, derived from the training dataset, which instructs ChatGPT on how to accurately classify the provided narrative descriptions of test data samples according to their features.

The query segment of the prompt should then pose a specific question that addresses both the correct classification and the rationale behind it. It is also essential to specify the desired format of the response to ensure that it is easily comprehensible. The instructions should be clear to ChatGPT to avoid delivering responses in a code format.

In addition, the paper will introduce a detailed methodology for crafting descriptions of both the classification rules and the test data samples. This approach ensures that the underlying mechanisms of time series classification are transparent and understandable, thereby enhancing the explainability and applicability of the model in real-world scenarios.

5.3.2 Extract Classification Rules

The incorporation of data descriptors in the methodology, rather than direct input of raw time series data into ChatGPT, provides multiple benefits. Primarily, this approach mitigates the constraints imposed by ChatGPT's limited input capacity, as raw data often surpass these limitations. Additionally, ChatGPT's processing of multi-digit numbers may be flawed, potentially leading to inaccuracies in data analysis and classification [7]. Thus, the development of an effective method to describe time series data is imperative. Utilizing classification rules derived from the training dataset emerges as an effective strategy to address these computational challenges.

The procedure for extracting classification rules is delineated in Figure 5.1. Describing data through the use of features yields a more comprehensive understanding than employing numerical values. These features provide context and enhance the interpretative depth of the analysis by elucidating the underlying characteristics of the data. Initially, the TSFRESH framework [28] is employed to generate a feature table containing hundreds of features extracted from the training dataset. Given the complexity associated with interpreting a vast array of features, those generated by TSFRESH are subsequently prioritized through a decision tree analysis, retaining only the most informative top k features. The refined feature table is then utilized to train a decision tree model. The final phase involves the extraction of classification rules from the decision tree, which are then converted into a textual format to facilitate accurate data classification by ChatGPT.

5.3.3 Generate Lookup Table

To enhance the precision and consistency of generating descriptions for classification rules and test data samples, the creation of a lookup table is essential. This table functions as a pivotal reference, elucidating the meanings of various features employed in both the classification rules and the test data. The establishment of such a table significantly improves the accuracy of the descriptions produced.

The lookup table is structured into four columns: ‘Feature Name’, ‘Meaning’, ‘Type’, and ‘Value’. Each column plays a crucial role in the interpretation process:

- **Feature Name**: This column lists the identifiers assigned to specific features, facilitating easy reference.
- **Meaning**: It provides a contextual explanation of the feature, enhancing understanding of its relevance and function within the dataset.
- **Type**: This column classifies the grammatical nature of the feature’s

meaning (e.g., noun or adjective), which is instrumental in crafting coherent and grammatically correct descriptions.

- **Value**: It quantifies the significance or magnitude of the feature, offering additional insights into its relative importance or measure.

An example of this lookup table, as utilized by the classification rules, is depicted in Figure 5.2. This structured approach not only supports accurate translation of data into descriptions, but also ensures uniformity across different datasets and analytical processes.

As illustrated in Figure 5.2, a single classification rule can include multiple features whose meanings are not immediately clear, even with documentation such as that provided by TSFRESH. For instance, the feature *number_crossing_0* is described merely as *the number of crossings*, a definition that lacks a tangible physical interpretation.

To remedy this deficiency, we have employed ChatGPT to provide more accessible explanations of these technical features. The interaction, in which ChatGPT clarifies that a high value of *number_crossing_0* indicates frequent directional changes or variations in acceleration, is documented in Figure 5.3. This interaction underscores ChatGPT's capability to render abstract data characteristics into comprehensible explanations.

The elucidations provided by ChatGPT are then meticulously selected to define the meanings of features, as represented in Figure 5.2. This selection process ensures that each feature is accompanied by a clear and contextually relevant description.

Furthermore, the actual values of these features are recorded directly in the lookup table. This data is crucial as it forms the foundation for accurately describing the features in terms of their operational significance within the dataset. This structured approach ensures that each feature is not only defined but also quantitatively assessed, enhancing the overall explainability of the classification rules.

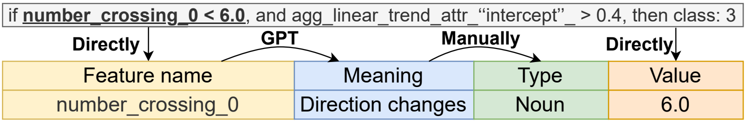


Figure 5.2.: the pipeline of generating lookup table

User: number_crossing_0 is a time series feature, what can you infer from this feature if this value is big during a measurement of accelerometer on x-axis. Give me only one most important pattern.
ChatGPT: A significant value of “number_crossing_0” in the context of accelerometer measurements on the x-axis typically indicates a pattern of rapid and frequent **changes in direction** or acceleration. This could suggest that the object or system being measured is undergoing dynamic and possibly complex movements, such as sudden stops and starts, sharp turns, or oscillations. Such patterns are often associated with activities like vehicular motion, sports activities, or machinery operations where rapid changes in movement direction or intensity occur frequently.

Figure 5.3.: The example of ask GPT to explain the given feature

5.3.4 Translate Data to Description

Upon constructing the lookup table, the process of translating the data into descriptive terms is facilitated. The columns labeled "Type" and "Value" within this table are instrumental in articulating the significance of each data feature, guiding the semantic interpretation essential for comprehending the underlying dataset.

The quantification of a feature is pivotal in determining its grammatical categorization. For instance, when the frequency of occurrence of a feature is of particular importance, it is classified grammatically as a noun. Descriptive terms such as "a lot of" or "few" are then employed based on comparative analysis; if the data feature’s value surpasses the threshold specified in the "Value" column of the lookup table, the phrase "a lot of" is utilized. Conversely, if it falls below, the term "few" is used to describe its prevalence.

Furthermore, the degree to which a feature influences or contributes to the dataset dictates its classification as an adjective. In such instances, descriptors like "big" or "slight" are used to convey the extent of the feature's impact or significance, thereby providing a nuanced understanding of its role within the broader context of the data analysis.

For classification rules, consider the rule depicted in Figure 5.1 as illustrative: "if *number_crossing_0* < 6.0 and *agg_linear_trend__attr_"intercept"*[2] > 0.4, then assign to class 3." The system utilizes the lookup table to automatically translate the values associated with these features into more intuitive descriptors. Consequently, the original rule is rendered as: "If there are few direction changes and a significant decreasing trend, then classify as class 3."

In the context of evaluating a test data sample, as shown in Figure 5.1, a *number_crossing_0* value of 9.0 translates to "A lot of direction changes," given that this value exceeds the established threshold of 6.0. The test sample is thus described using the translated meanings derived from the lookup table for each feature.

Subsequently, the comprehensive information,including an explanation of the classification rules, a description of test data samples, and the overarching objective, is sent to ChatGPT. Armed with this context, ChatGPT is tasked with classifying the data and elucidating the rationale behind its classifications, thereby ensuring a clear and informed decision-making process.

5.4 Evaluation

5.4.1 Experiment Setup

This study utilizes human activity recognition datasets, chosen for their intuitive alignment with natural human activities, enhancing the explainability of model explanations. For instance, a dataset sample exhibiting a periodic pattern can be directly linked to rhythmic activities, such as shaking. We incorporate three distinct datasets in our analysis. AllGestureWiimoteX[3], Basketball motion[4], and HMP[5]. From each dataset, we randomly selected three represen-

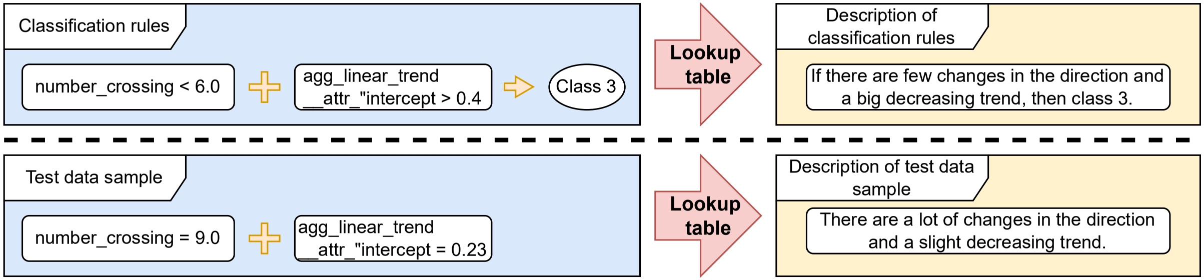
---

[2] This attribute signifies a decreasing trend.

[3] `https://www.timeseriesclassification.com/description.php?Dataset=AllGestureWiimoteX`

[4] `https://archive.ics.uci.edu/dataset/587/basketball+dataset`

[5] `https://github.com/wchill/HMP_Dataset`


Classification rules
number_crossing < 6.0
agg_linear_trend __attr_"intercept > 0.4
Class 3
Lookup table
Description of classification rules
Test data sample
number_crossing = 9.0
agg_linear_trend __attr_"intercept = 0.23
Lookup table
Description of test data sample

If there are few changes in the direction and a big decreasing trend, then class 3.

There are a lot of changes in the direction and a slight decreasing trend.

Table 5.1.: Translate data to description

tative classes to facilitate a complete evaluation.

The efficacy of our classification models is quantified using accuracy as the primary metric. This metric reflects the proportion of test samples that are correctly classified, calculated as the ratio of correctly classified instances to the total number of evaluated instances. This approach provides a clear measure of model performance across various datasets.

Our empirical assessments are conducted using three advanced models. GPT-3.5-0301, GPT-3.5-0613, and GPT-4-0613. These models were chosen to explore the impact of evolving natural language processing capabilities on classification accuracy in complex activity recognition tasks.

As a comparative baseline, we employ a decision tree classifier. Feature selection is conducted using the TSFRESH package, which identifies the most significant features from the training samples. The top three features of each class are then used to construct the feature table, which serves as input for the decision tree classifier. The classification accuracies achieved are 68.33% for AllGestureWiimoteX, 66.66% for Basketball motion, and 100% for the HMP dataset. This baseline serves to benchmark the performance enhancements offered by more advanced models.

5.4.2 Result

The results presented in Table 5.2, indicate that the accuracies of the various models are closely aligned with the baseline. Consequently, it can be deduced that our methodology demonstrates universality across selected human activity recognition datasets.

Moreover, an analysis of the model-specific accuracies reveals a clear trend: as the sophistication of the model increases, so does the performance of our method. Specifically, experiments employing GPT-4 consistently yield higher accuracy than those using GPT-3.5-turbo-0613, which, in turn, outperforms GPT-3.5-turbo-0301 across all datasets. This observation supports the hypothesis that advances in model architecture contribute positively to the efficacy of classification in human activity recognition tasks.

A representative example of the explanatory ability of the method is illustrated in Figure 5.4, which clearly shows how ChatGPT can identify and articulate the nature of a shaking movement. This example underscores the intuitive clarity of the explanations generated by our approach, thereby enhancing the explainability of the classification results.

Table 5.2.: The accuracy of experiments on 3 datasets

	Motion	Baseline	GPT-3.5-0301	GPT-3.5-0613	GPT-4
AllGesture	Shake hand	75%	66%	67%	70%
	Move to left	50%	50%	52%	48%
	Move to right	80%	79%	78%	80%
	Total	68.33%	65%	65.67%	66%
Basketball	Dribble	100%	100%	100%	100%
	Hold	100%	100%	100%	100%
	Pass	0%	0%	0%	0%
	Total	66.66%	66.66%	66.66%	66.66%
HMP	Comb hair	100%	24%	80%	92%
	Descend stair	100%	100%	100%	100%
	Lie down bed	100%	100%	100%	100%
	Total	100%	74.67%	93%	97.33%

5.5 Discussion

5.5.1 Limitation

3-Class Classification The experimental framework initially involved datasets comprising multiple categories. In order to focus specifically on evaluating ChatGPT's explanatory capabilities, we streamlined the complexity of the experiment by selecting only three classes from each dataset. This reduction was aimed at facilitating a more focused analysis of the generated explanations within a manageable experimental scope.

Manual Selection of Feature Meanings As outlined in Section 5.3.3, the selection of meanings for each feature was conducted manually. This approach was chosen to ensure precision in the interpretation of features within the context of the experiment. Nevertheless, an alternative methodology could incorporate the use of a ML model designed to automatically identify and extract relevant keywords from ChatGPT responses, which would potentially enhance the efficiency and scalability of feature interpretation.

Discussion While LLMs have shown promise in various applications, their

ChatGPT: According to the given descriptions, your data sample seems to fall into two categories - ‘shake hand’ due to the lots of changes in direction, and ‘move to left’ due to a big decreasing trend and few increasing values.
However, identification of movement is often prioritized by the most distinctive feature that clearly separates one from the others. The ‘shake hand’ gesture is most prominently characterized by a lot of changes in the direction. This aspect, along with the decreasing trend and few increasing values, may suggest that **the hand movement involved repeatedly turning the hand left and right**, while gradually dropping it down a bit, a very likely movement pattern for a shaking hand, thus falling into the ‘shake hand’ category.
On the other hand, the ‘move to left’ gesture is characterized by few changes in the direction and a strong decreasing trend. Even though these aspects are present in your data sample, the many changes in direction do not match the key characteristic of ‘move to left’, i.e., ‘few changes in direction’. Hence, it would be more accurate to categorize this data sample as shaking hand.
So, to summarize: [Class: shake hand].

Figure 5.4.: Explanation of shake hand

potential in time series classification and explainability remains underexplored. This chapter aims to bridge this gap by developing a novel methodology that leverages the capabilities of LLMs, particularly ChatGPT, to provide interpretable explanations for time series classification, thereby advancing the understanding and application of AI in pervasive computing. To this end, we introduced a novel methodology for generating interpretable explanations in time series classification tasks, particularly within the realm of human activity recognition. Our approach leverages the capabilities of LLMs to produce explanations that are not only accurate, but also align with human cognitive processes, as confirmed by evaluative studies.

The generality and effectiveness of our method have been validated across various datasets in the field of human activity recognition, consistently yielding high accuracy. This supports the robustness and adaptability of our approach. Furthermore, ongoing advancements in LLMs technologies suggest that future improvements in model performance will enhance the efficacy of our methodology. Notably, comparative analysis has demonstrated that GPT-4 outperforms its predecessor, GPT-3.5, in generating more coherent and contextually appropriate explanations.

5.5.2 Summary

The preceding three chapters have collectively addressed the challenge of enhancing the predictive performance of interpretable models in the pervasive computing domain. This challenge arises from the intricate nature of the data (**C1**) and the demanding requirements for transparency and efficiency in constrained systems (**C2**). In Chapter 3, we explore the limitations of traditional feature engineering and propose a novel automated feature engineering approach that leverages sequence information and composite transformations within a MCTS-based framework. This method demonstrated that it is possible to significantly improve the performance of interpretable models without sacrificing their transparency, particularly in complex data environments.

Chapter 4 built on this foundation by addressing the constraints of pervasive computing environments, such as limited computational resources and the need for real-time processing. We introduced an enhanced GA-based approach to optimize AutoML processes specifically for these constrained settings. This method not only achieved efficient optimization, but also ensured that the models remained interpretable and trustworthy, further extending the utility of interpretable models in practical applications.

In this chapter, we shift our focus to the explainability of time series classification models, particularly in human activity recognition tasks. Here, we harnessed the power of LLMs, such as ChatGPT, to generate explanations that align with human cognitive processes. The success of this approach across various datasets highlights its potential for broader applications in the field of interpretable ML, suggesting a promising direction for future research.

As we transition to the next phase of this research, our focus will shift towards the development of post hoc explanation techniques in pervasive computing. These techniques are designed to enhance the explainability of AI models by providing explanations that are accessible and comprehensible, even for models that are inherently complex or abstract. To achieve this objective, the goal is divided into two distinct subtasks corresponding to the second and third parts of this dissertation. In Part II, the research examines the key decision elements used by high-quality models in pervasive computing to make decisions. This analysis serves as a foundation for further advancements. In Part III, the focus will shift to developing novel post hoc explanatory methods that build on the insights gained from Part II.

Part II.

Unveiling Key Decision Elements

6. Summarize the Decision-making Elements of High-quality Models

In Part I, we advanced the application of XAI in pervasive computing by enhancing the predictive performance of explainable models. In this part, we shift our focus to elucidating the decision-making processes of black-box models. Despite the widespread adoption of deep learning models in pervasive computing, their opaque nature has significantly hindered user trust in their decisions. Existing explanation methods can highlight regions that are important for decision-making. However, these methods often lack domain-specific and systematically organized interpretive information. To develop explanation methods tailored specifically for pervasive computing, it is essential to thoroughly understand the decision-making mechanisms of high-quality models within this domain and identify the critical elements upon which these decisions rely. Such study is crucial, as it provides a foundation for the creation of novel interpretative techniques. This chapter is organized around this core.

Corresponding Publication:. Y. Huang, H. Zhao, Y. Zhou, T. Riedel, and M. Beigl. Standardizing your training process for human activity recognition models: A comprehensive review in the tunable factors. *EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2024
Y. Huang, Y. Zhou, H. Zhao, T. Riedel, and M. Beigl. A survey on wearable human activity recognition: Innovative pipeline development for enhanced research and practice. In *2024 IEEE International Joint Conference on Neural Networks (IJCNN 2024), Yokohama, 30th June-5th July 2024*, 2024

6.1 Introduction

The pervasive computing domain presents unique challenges in developing novel post hoc XAI methods, primarily due to the diversity and complexity of the data. Data in this domain are typically drawn from a range of sources, including sensor readings, user interactions, and environmental conditions, covering various types such as text, audio, video, and numerical sensor data. This variety not only introduces complexity, but also highlights the need for tailored explanations that account for the sequential and temporal dependencies inherent in the data.

Another challenge in pervasive computing data is the presence of noise, which complicates the task of explaining. Effective XAI methods must be capable of explaining model decision under the influence of noise. Moreover, these methods must demonstrate the robustness of decisions under noisy conditions to ensure that users understand how models behave with imperfect data, a common scenario in real-world applications.

The predominant explanation technique in this domain, especially in computer vision, is the use of saliency maps. These maps highlight the regions within an input, whether an image, a segment of sensor data, or a set of features, that influence a model's decision. While effective for image data due to its inherently visual nature, saliency maps face significant limitations when applied to non-visual data such as time series, which consists of sequential numerical values without spatial structure. For time series data, understanding and explaining patterns over time requires not only advanced abstraction techniques but also specialized domain knowledge. Consequently, relying solely on saliency maps for time series data is often inadequate for comprehensive model interpretation.

Given these limitations, exploring alternative methods to explain model decisions in pervasive computing has become crucial. One potential approach is to leverage expert knowledge to distill essential decision-making elements. However, this approach faces several challenges: the scarcity of domain experts, potential subjective bias, and scalability issues in diverse applications.

In light of these challenges, while expert insights are invaluable, it is not reasonable to rely exclusively on experts. In response to these limitations, we advocate for a model-centric methodology for extracting crucial decision-making elements in time series data analysis. This approach consolidates existing models utilized across the domain, capturing intricacies from data preprocessing, model training, and architectural strategies. By aggregating these data, we aim to unearth fundamental insights that influence model decisions, thereby forming a robust foundation for further explanation.

To facilitate a thorough extraction of data elements that is pivotal for model decision-making processes, more than hundred scientific articles have been systematically categorized based on the relevance of each article to these procedural steps. This organization enables a detailed analysis of existing articles. The following subsection will demonstrate the observation we have obtained after reviewing these articles.

6.2 Observation

6.2.1 Data Processing

In pervasive computing, raw sensor data frequently exhibit multiple types of noise, including electronic disturbances and external environmental interferences. Effective data processing techniques, such as data cleaning and filtering, are indispensable for addressing these disturbances.

Furthermore, data processing is instrumental in improving the quality of data for subsequent modeling phases. It enhances key features, such as variations in energy or frequency, while simultaneously suppressing undesirable attributes. This selective enhancement and suppression process significantly enhances the performance of the models. Additionally, considering the multimodal nature of pervasive computing data, processing is crucial for adjusting and aligning the data to fulfill specific model criteria, ensuring that the input data optimally supports the intended analytical tasks.

Based on these important roles of data processing, exploring the data processing process in depth and finding the elements used in it will help to develop new XAI methods. The methodology employed in data processing varies significantly based on the presentation of the processed data, allowing the categorization into three distinct types.

6.2.1.1 1D Transformation. This transformation technique generates feature vectors, in which each vector encapsulates a distinct sample. Each component of the vector is representative of a separate feature, and these features maintain independence from one another. Thakur *et al.* [163] demonstrate this process by extracting features in both the time and frequency domains, subsequently refining the selection of features through the application of a guided regularized random forest algorithm.

Fujimoto *et al.* [45] adopt a multidimensional approach to feature extraction, incorporating techniques such as a Butterworth low-pass filter, time differentiation and Fast Fourier Transform (FFT). The relevance of these features is evaluated using the Gini coefficient and the study further integrates varying degrees of Laplace noise to ensure data confidentiality and mitigate the risk of unintentional data leakage.

In the study conducted by Azmat *et al.* [13], the data is initially subjected to denoising, followed by windowing and segmentation processes. Subsequent to these preprocessing steps, feature extraction techniques are employed to identify attributes such as Parseval energy, skewness, kurtosis, and Shannon entropy, along with statistical characteristics from both time and frequency domains. To optimize the selection of features, the study utilizes the Luca metric-based fuzzy entropy (LFE) and the Lukasiewicz similarity measure (LS), achieving a 25% reduction in the feature set. Additionally, a feature optimization algorithm employing the Yeo-Johnson power transformation is utilized to enhance the efficacy of the extracted features.

All of the above methods emphasize the importance of denoising and feature selection. In addition to this, the first two methods emphasize the combination of time- and frequency-domain features.

6.2.1.2 2D Transformation. This type of transformation generates two-dimensional arrays wherein the first dimension corresponds to various features, and the second dimension represents the values attributed to each feature. While the features remain independent of each other, the values within each feature exhibit a sequential relationship, underscoring an inherent order.

Webber *et al.* [174] employ a Kalman filter to mitigate noise in the initial data sets, enhancing data integrity for further processing. Kumar *et al.* [95] deploy Welch's method to analyze the energy distribution across different signal frequencies, facilitating a more granular power spectral density analysis. Additionally, Zhao *et al.* [182] incorporate wavelet-based learnable filters to optimize sensor channel selection, thus refining data acquisition strategies.

Khtun *et al.* [89] highlight that feature vectors derived via FFT are not only resilient to noise, but also demonstrate advantageous properties such as level-of-detail representation and rotation invariance. Based on these attributes, their research advocates for the adoption of frequency domain Fourier sparse features, considering them as highly relevant for data feature selection in pervasive contexts.

Similarly to 1d conversion, 2d conversion focuses on noise reduction, combining time- and frequency-domain features, and feature selection.

6.2.1.3 3D Transformation. In computational applications, a 3D transformation produces an output in the form of a three-dimensional array. The array's first dimension, commonly identified as the 'channel', represents various features. The subsequent dimensions, the second and third, encapsulate the values corresponding to these features, often visualized as images. Notably, Gholamrezaii *et al.* [50] employ FFT to create spectrograms that serve as distinctive features. In parallel, Gholamiangonabadi *et al.* [49] investigate the utility of combining the stationary wavelet transform with empirical mode decomposition for the purpose of action recognition.

6.2.1.4 Summary of Key Data Process Trends. In summary, data analysis within the frequency domain is a prevalent methodology in data processing, facilitated primarily through FFT or various filtering techniques. Approximately one-third of the relevant studies incorporate a temporal analysis to enhance the understanding of data dynamics. Moreover, certain research endeavors focus on examining local attributes of the data, such as kurtosis and skewness, to gain deeper insights into the underlying distributions and anomalies.

6.2.2 Model Architecture

Model architectures in the domain of pervasive computing are intricately designed to extract meaningful patterns from input data, thereby enhancing predictive capabilities. The articles in this field primarily follow two significant trajectories:

- The enhancement of feature extraction capabilities through the integration of innovative architectural designs.
- The refinement of model selection and training methodologies to augment the overall accuracy and performance of these systems.

In this section, we summarize the key elements that are critical to decision-making in pervasive computing by analyzing the structure of various model architectures.

6.2.2.1 Traditional Machine Learning Approaches. Current advancements in the model architecture in pervasive computing demonstrate a dynamic evolution, with deep learning emerging as the predominant technique. Nevertheless, traditional ML techniques continue to contribute profoundly, providing essential insight and methodologies. For instance, Liu *et al.* [105] analyze human activity by breaking it down into basic, discriminative states termed 'motor units', which are analogous to phonemes in speech recognition. They employ Hidden Markov Models (HMM) for the prediction of activities. Similarly, Vu *et al.* [168] employ Uniform Manifold Approximation and Projection (UMAP) to transform high-dimensional data into 30-dimensional features, subsequently enhancing activity prediction by incorporating data from the target domain. These methodologies underscore the versatility and adaptability of traditional ML techniques within the evolving landscape of system design. Additionally, they underscore the critical importance of local features, such as 'motor units', in the decision-making process, demonstrating that these elements play an indispensable role in enhancing system performance.

The former emphasizes the importance of identifying local data patterns, while the latter stresses the fusion between different features.

6.2.2.2 Deep Learning Architectures.

The predominance of deep learning is principally attributed to its inherent ability to autonomously extract features, facilitated by the varied architectures of deep learning models. Ismail *et al.* [83] employ innovatively a genetic algorithm to fine-tune the architecture of a Convolutional Neural Network (CNN). This optimization segments a CNN into components such as conv1d, batch normalization, and LeakyReLU, encoded within a 5-bit system. The encoding specifics include the first three bits detailing conv1d parameters like the number of filters, padding, and activation function, with the subsequent bits representing batch normalization and LeakyReLU activation parameters, respectively.

Hurtado *et al.* [81] introduce an autoencoder architecture that is trained using dual loss functions: the reconstruction loss from unlabeled data and the prediction loss from labeled data. This dual training approach aims to enhance the model's efficiency in feature extraction and activity prediction. Further advancing the model architecture, Hnoohom *et al.* [65] integrate the residual model with a Squeeze-and-Excitation block, forming a ResNetSE block. This block serves as the foundation for constructing a deep CNN model, enhancing the model's capability to effectively manage feature recalibration. Haresamudram *et al.* [63] explore contrastive predictive coding in their research, aiming to augment human activity recognition models. They experiment with integrating stridden convolutions, causal convolutional aggregators, and replacing masks with direct future predictions to improve model performance. Sarkar *et al.* [140] utilize the Continuous Wavelet Transform (CWT) to transmute time series data into image formats, subsequently employing a CNN architecture for robust feature extraction. The selected features are refined using an unsupervised genetic algorithm that evaluates metrics such as mutual information, Relief-F, and m-RMR. The processed data are finally classified using the KNN algorithm, illustrating a comprehensive approach to activity prediction.

The application of normalization techniques in neural network architectures frequently results in a phenomenon termed "channel collapse," whereby numerous channels gravitate towards minimal contribution values. This condition severely limits the effective diversity of channel outputs within the network. To address this issue, Huang *et al.* [69] introduced an innovative approach known as "channel equalization." This technique rejuvenates underutilized channels by implementing whitening or decorrelation operations, thereby enhancing the overall network performance.

On a different note, the constraints imposed by the limited computational capabilities of edge devices necessitate efficient processing methods. In this context, Tang *et al.* [162] devised a novel strategy referred to as hierarchical segmentation convolution. This approach is designed to optimize the performance of CNNs without necessitating additional computational or memory resources. Initially, sensor signals are segmented into uniform-length windows via a sliding window technique. These windows are subsequently analyzed by a convolutional layer to generate a preliminary feature map. The process continues with the division of these feature maps into segments, followed by selective convolution operations. Additionally, identity mappings or cascading operations are employed. This iterative methodology of segmenting and merging ultimately leads to the aggregation of refined sub-feature maps. The enhanced feature maps are then subjected to standard convolutional layers to further refine the feature representation, culminating in a more efficient feature extraction process without overburdening computational resources.

The focus on CNN-based architectures in deep learning models for feature extraction is prominent, yet emerging research showcases the versatility of incorporating LSTM components. Ramos *et al.* [130] utilize a residual bidirectional LSTM block to underscore its efficacy in applications. Furthermore, the adoption of the Transformer architecture, renowned for its robust performance in Natural Language Processing (NLP), is now being explored for its applicability in the pervasive computing domain by Dirgova *et al.* [35].

6.2.2.3 Hybrid and Advanced Architectures. The evolution of model architectures is also witnessing a shift towards hybrid designs that amalgamate different computational strategies to capitalize on the unique advantages of each architecture, particularly for enhancing performance in specialized tasks. For instance, the integration of convolutional and recurrent layers facilitates the effective extraction of spatial and temporal features, as demonstrated in recent studies [156; 187]. In extension of this approach, Li *et al.* [102] further enhance model capabilities by incorporating residual modules alongside bidirectional LSTMs.

Additionally, the potent attention mechanisms of Transformers are being adapted into hybrid models. Shavit *et al.* [153] effectively combine CNNs with Transformers to augment the processing of long-term dependencies, which is crucial for complex sequence analysis. Similarly, multibranch techniques, which involve processing data from different channels independently prior to integration, are employed by Lu *et al.* [109] and Park *et al.* [123].

These methods emphasize the temporal characteristics of the data and the importance of multichannel feature interactions.

6.2.2.4 Ensemble Methods. Ensemble methods are increasingly recognized for enhancing predictive performance in deep learning, particularly within the pervasive computing domain. Bhattacharya *et al.* [19] propose a novel adaptation of ensemble techniques wherein the training dataset is bifurcated into two subsets. The first subset is utilized to train diverse deep learning models, each capable of capturing unique data representations. The predictions of these models, obtained using the second subset, are then aggregated and utilized as inputs to a final prediction model. This strategy leverages the distinct representations derived from various architectures, employing an additional model to optimally integrate these representations for enhanced predictive accuracy.

6.2.2.5 Summary of Key Architectural Trends. In summary, the prevalent architectures in this field typically extract local spatial information through CNNs. Subsequent integration of local and temporal features is achieved by either stacking or embedding CNNs within LSTMs. Additionally, some models transform the input data to the frequency domain, leveraging it as supplementary information to enrich the model's understanding. There are also approaches like strip CNNs that aim to capture long-range numerical dependencies. Besides, the application of the transformer structure also introduces absolute positional coding, thus emphasizing the importance of position. Collectively, these methodologies highlight a balanced emphasis on harnessing local details, frequency domain insights, and temporal dynamics, illustrating a comprehensive approach to modeling complex data. From this we can conclude that the following factors play a decisive role in decision making in high performance modeling: interdependencies among various channels, long- and short-term temporal dependencies, local region patterns, frequency patterns, and positional information.

6.3 Discussion

This study has undertaken an extensive analysis of high-quality models in the pervasive computing domain, with a particular focus on understanding the key elements that influence model decision-making. Through a meticulous examination of the data processing techniques and model architectures employed in the articles, several critical elements that significantly impact model decisions have been identified: interdependencies among various channels, long- and short-term temporal dependencies, local region patterns, frequency patterns, and positional information. These elements are consistently emphasized across various models, indicating their fundamental importance in achieving high-quality outcomes in pervasive computing.

The findings of this study offer valuable insight into the development of XAI methods tailored to the pervasive computing domain. The identified key elements provide a robust foundation for developing explanation techniques that can enhance the transparency and trustworthiness of AI models.

7. Prove Element Importance through Innovative Model Design

In the preceding section, we conducted a thorough review of the literature within the domain of pervasive computing, identifying and extracting key data elements that significantly influence the quality of model decisions. In this section, we will elucidate the importance of part of these elements by incorporating them into the design of novel models. This integration aims to substantiate the critical role these elements play in enhancing model accuracy and reliability.

Corresponding Publication:. Y. Zhou, M. Hefenbrock, Y. Huang, T. Riedel, and M. Beigl. Automatic remaining useful life estimation framework with embedded convolutional lstm as the backbone. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part IV*, pages 461–477. Springer, 2021

Y. Zhou, H. Zhao, Y. Huang, T. Riedel, M. Hefenbrock, and M. Beigl. Tinyhar: A lightweight deep learning model designed for human activity recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, pages 89–93, 2022

7.1 Introduction

We conducted an in-depth review of the literature within the domain of pervasive computing, identifying and extracting key data elements that have a significant impact on the high-quality model decisions. These elements are critical in the design and optimization of the domain-specific XAI approach.

In this chapter, we will further elaborate on the importance of these elements by integrating them into the design of our models. This integration will allow us to empirically demonstrate how these elements influence model accuracy and reliability. By systematically incorporating these data elements into the model-building process, we will not only examine their individual contributions to the predictive performance of the model but also explore the relationships and interactions among them. Our aim is to provide empirical evidence that the careful selection and utilization of these key data elements significantly enhance predictive power and demonstrate consistently strong performance across different application scenarios.

We explored accordingly and published two papers on the subject. One focused on Remaining Useful Life (RUL) prediction, and the other on human activity recognition. Estimation of RUL is a critical component in the field of predictive maintenance. To achieve accurate predictions, models with strong learning capabilities that can capture short- and long-term dependencies with high precision are essential. This does not refer to feature extraction over a long horizon through stacking CNNs, but rather to the interdependencies between local data features. To this end, we introduced a novel self-attention-based hybrid model for RUL estimation, called "Inception-Attention." This model employs an innovative feature extraction mechanism composed of a combination of three different efficient self-attention patterns. This design aims to capture temporal dependencies across varying ranges. Additionally, we incorporated a smooth regularization component into the training objective to enhance the model's ability to generate more stable and reliable RUL estimates. One purpose of this model is to demonstrate that, in time series data, local information and their interdependencies are critical to the model's decision-making process.

On the other hand, human action recognition typically requires processing long-window data from multiple sensors worn on different parts of the human body. By developing a new model called Tinyhar, which employs multimodal saliency, multimodal cooperation, and temporal information extraction, we demonstrate the importance of multimodal data interactions and long-term dependencies for model decision.

The following section demonstrates the structure of the proposed models. It is important to note that, in both of these work, I was only involved in model design and manuscript revision. The first author of the papers is Yexu Zhou. Most of the content in the following sections is copied from the corresponding paper.

7.2 Observation

7.2.1 Automatic Remaining Useful Life Estimation Framework with Embedded Convolutional LSTM as the Backbone

Formally, RUL estimation can be described as a sequence to the target problem. Given a T length sequence time series $\mathbf{X} = (\mathbf{x}_t \mid t = 1, \cdots, T)$ with $\mathbf{x}_t \in R^{n \times m}$, where n is the number of sensors and m the number of samples per cycle. Now, the aim is to predict the corresponding output

$$y_T, y_T = f\left(\mathbf{x}_t \mid t = 1, \cdots, T\right),$$

where $\mathbf{x}_t$ denotes all samples in cycle t. When the sliding window method is applied, the previous formula should be modified to $y_T = f(\mathbf{x}_t^w | t = w, \cdots, T)$, where w is the window size. The vector $\mathbf{x}_t^w$ thereby contains all the samples in the time window which is denoted as $\mathbf{x}_t^w = (\mathbf{x}_{t-w+1}, \cdots, \mathbf{x}_t)$. In our settings, the sliding step size is always set to 1. For any model, the sequence length determines how past information is used, while the window size describes the complexity of dynamic features over time. As both parameters can greatly affect the model performance, both should be considered when optimizing the model.

Inspired by the work ConvLSTM [176], we propose an extension of FCLSTM, in which a group of different 1D convolutions is embedded into the LSTM structure, which we call Embedded Convolutional LSTM (ECLSTM) .

We assume that such ECLSTM architecture is more powerful than FCLSTM in handling multivariate time series tasks.

In order to preserve the temporal information within the window, the input should be kept as a 2-dimensional tensor. This can be achieved by replacing the full connection in the FCLSTM with convolutional operation. The equations of ECLSTM are then given by

$$\begin{aligned}
i_t &= \sigma(W_i * [x_t, h_{t-1}] + b_i) \\
f_t &= \sigma(W_f * [x_t, h_{t-1}] + b_f) \\
o_t &= \sigma(W_o * [x_t, h_{t-1}] + b_o) \\
C_t &= f_t \circ C_{t-1} + i_t \circ \tanh(W_C * [x_t, h_{t-1}] + b_C) \\
h_t &= o_t \circ \tanh(C_t),
\end{aligned} \tag{7.1}$$

where $*$ indicates the convolution operator and $\circ$ the element-wise product. There are three benefits to using the convolution operator in LSTM. Firstly, the convolution parameters are only related to the defined kernel size and the number of filters, not to the size of the window. When the window size is large, the complexity of the model does not increase with it. Secondly, the hidden state H and the memory C also become 2D tensors. This means that they implicitly inherit and preserve the temporal relationship. Thirdly, the input, hidden state, and memory can even maintain a 3-dimensional shape, as this will not affect the operation of the convolution. Keeping the three-dimensional shape allows for more different convolutions.

The stacking of convolutional layers allows for hierarchical decomposition of the raw data and combinations of lower-level features. In order to get more complex features, the convolutions in (7.1) can be stacked as convolutional cells in a chain structure. If three convolutional layers are stacked in the cell, we call the ECLSTM as 3-depth-ECLSTM. Taking the input gate in ECLSTM as an example, the activation can be calculated as

$$i_t = \sigma(W_i^3 * \sigma(W_i^2 * \sigma(W_i^1 * [x_t, h_{t-1}] + b_i^1) + b_i^2) + b_i^3). \tag{7.2}$$

Other gates have the same structure but do not share the weights.

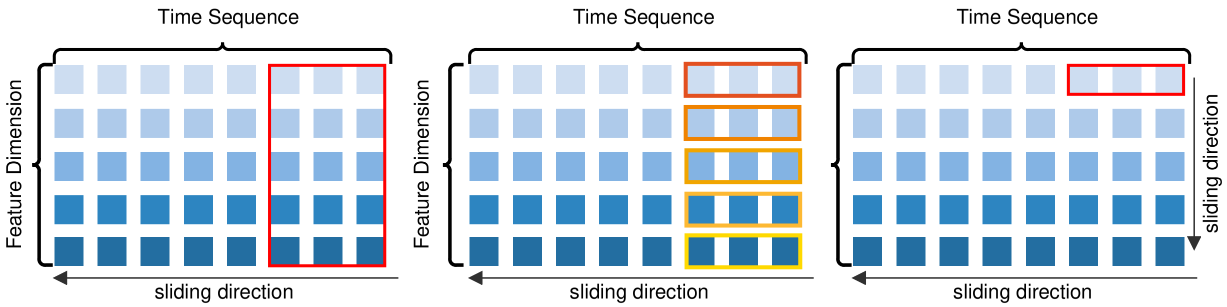
Moreover, the results of many multivariate time series analysis works like [119] indicate that different fusion strategies affect performance. Inspired by that, the convolution cell can be composed of the following three different 1-dimensional convolutions, which are shown in Figure 7.1. The first is the early fusion convolution, which is the same as conventional 1D convolution. Here, the features are extracted from all sensory information jointly. The second is the late-fusion convolution. In late fusion, the features are extracted separately from each sensor. The third is hybrid fusion convolution, where features are extracted separately from each sensor but weights are shared.

7.2.2 TinyHAR: A Lightweight Deep Learning Model Designed for HAR

TinyHAR consists of five parts. The input data of the model $\mathrm{X} \in \mathbb{R}^{T \times C \times F}$, where T denotes the temporal sliding window size, C is the number of sensor channels, and F indicates the number of filters ($F = 1$ for the raw data input, which has not been processed).

To enhance the local context, we applied a convolutional subnet to extract and fuse local initial features from the raw data. Considering the varying contribution of different modalities, each channel is processed separately through four individual convolutional layers. For each convolutional layer, ReLU nonlinearities and batch normalization [82] are used. Individual convolution means that the kernels have only a 1D structure along the temporal axis (the kernel size is 5×1). To reduce the temporal dimension, the stride in each layer is set to 2. All four convolutional layers have the same number of filters F. The output shape of this convolutional subnet is thus $\mathbb{R}^{T^* \times C \times F}$, where T^* denotes the reduced temporal length.

Work [2] successfully adopted the self-attention mechanism to learn the collaboration between sensor channels. Inspired by this, we utilized one transformer encoder block [166] to learn the interaction, which is performed across the sensor channel dimension at each time step. The transformer encoder block consists of a scaled dot-product self-attention layer and a two-layers Fully Connected (FC) feed-forward network. The scaled dot-product self-attention is used to determine relative importance for each sensor channel by considering its similarity to all the other sensor channels. Subsequently, each sensor channel utilized these relative weights to aggregate the features of all other



(a) Early Fusion 1D convolution **(b) late Fusion 1D convolution** **(c) Hybrid Fusion 1D convolution**

Table 7.1.: In the early fusion convolution, the kernel height is fixed, that is, the same as the number of features. The sliding direction of the convolution kernel is along the time axis. In the late fusion convolution, the kernel height is 1. Each feature has its own convolution kernel. The convolution kernel also has only one sliding direction, namely the time axis. In hybrid fusion convolution, the kernel height is also 1. But it has two sliding directions, one is the time axis and the other is the feature axis. Because of weight sharing, it can save many parameters. It should be noted that when the number of filters is greater than 1, the output of the early fusion convolution is 2-dimensional. The outputs of the two remaining convolutions are 3-dimensional.

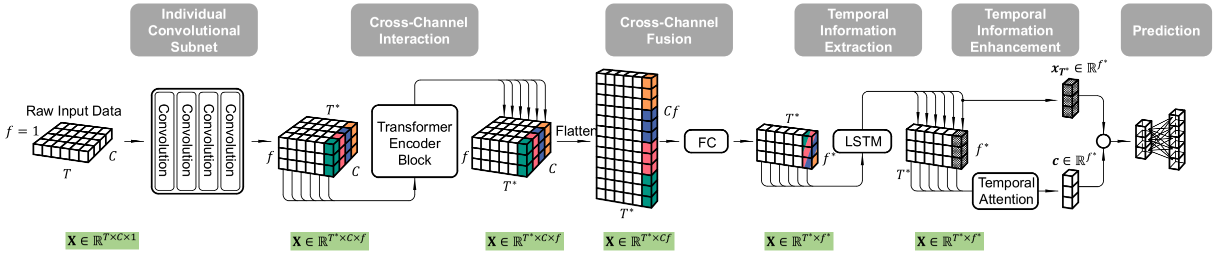

Individual Convolutional Subnet
Cross-Channel Interaction
Cross-Channel Fusion
Temporal Information Extraction
Temporal Information Enhancement
Prediction
Raw Input Data
$f = 1$
T
C
Convolution
Convolution
Convolution
Convolution
T^*
f
C
Transformer Encoder Block
Flatten
Cf
FC
f^*
LSTM
Temporal Attention
$x_{T^*} \in \mathbb{R}^{f^*}$
$c \in \mathbb{R}^{f^*}$
$\mathbf{X} \in \mathbb{R}^{T \times C \times 1}$
$\mathbf{X} \in \mathbb{R}^{T^* \times C \times f}$
$\mathbf{X} \in \mathbb{R}^{T^* \times C \times f}$
$\mathbf{X} \in \mathbb{R}^{T^* \times Cf}$
$\mathbf{X} \in \mathbb{R}^{T^* \times f^*}$
$\mathbf{X} \in \mathbb{R}^{T^* \times f^*}$

Table 7.2.: Overview of the proposed Algorithm.

sensor channels. Then, the feedforward layer was applied to each sensor channel, which further fused the aggregated feature of each sensor channel. Until now, the features of each channel have been contextualized with the underlying cross-channel interactions.

In order to fuse the learned features from all the sensor channels, we first vectorize these representations at each time step, $\mathrm{X} \in \mathbb{R}^{T^* \times C \times F}$ *to* $\mathrm{X} \in \mathbb{R}^{T^* \times CF}$. Then a FC layer is applied to weighted summation of all the features. Compared to the attention mechanism used in [111], in which the features of same sensor channel share the same weights, FC layer allows different features of same sensor channel to have different weights. Such flexibility of the FC layer leads to more sufficient feature fusion. This FC layer also works as a bottleneck layer in the proposed TinyHAR, which reduces the feature dimension to F^*. In our work, we set $F^* = 2F$.

After the features are fused across the sensor and filter dimensions, we obtain a sequence of refined feature vectors $\in \mathbb{R}^{T^* \times F^*}$ ready for sequence modeling. We then apply one LSTM layer to learn the global temporal dependencies.

Given that not all time steps equally contribute to recognition of the undergoing activities, it is crucial to learn the relevance of features at each time step in the sequence. Following the work in [111], we generate a global contextual representation $\mathrm{c} \in \mathbb{R}^{F^*}$ by taking a weighted average sum of the hidden states (features) at each time step. The weights are calculated through a temporal self-attention layer. Because the feature in the last time step $\mathrm{x}_{T^*} \in \mathbb{R}^{F^*}$ has the representation for the whole sequence, the generated global representation c is then added to x_{T^*}. Here, we introduce a trainable multiplier parameter γ to c, which allows the model has the ability to flexibly decide, whether to use or discard the generated global representation c.

7.3 Discussion

In this work, through the development of two novel models, Inception-Attention for RUL estimation and TinyHAR for HAR, we demonstrate the importance of capturing both local and long-term dependencies within time series and interdependencies among multimodal data.

For RUL estimation, our Inception-Attention model leveraged a self-attention mechanism to extract temporal dependencies across different time ranges, focusing on the critical interplay between the features of local data. This approach allowed the model to make more reliable and accurate predictions of equipment lifespan. The inclusion of smooth regularization further contributed to the model's stability, underscoring the necessity of balancing precision and consistency in predictive maintenance applications.

On the other hand, the TinyHAR model highlighted the significance of multimodal cooperation and temporal information extraction in human activity recognition tasks. By incorporating multimodal saliency detection and a combination of convolutional and transformer-based components, the model effectively utilized data from various sensors, demonstrating the importance of both individual and cross-sensor information interactions. The results of this model reaffirmed that long-term temporal dependencies are crucial to accurately recognizing complex human activities.

In both cases, the models showed how strategic design choices, such as embedding convolutional operations within LSTM and utilizing attention mechanisms, can lead to substantial improvements in accuracy and robustness. The key data elements discovered in the last section are central to high-quality modeling.

Part III.

Innovating Post hoc Explanation Techniques

8. Explain with Spatial Information

Part II analyzed the data elements typically utilized by high-quality models during the decision-making process. The key elements identified include interdependencies among various channels, long- and short-term temporal dependencies, local region patterns, frequency patterns, and positional information. Part III builds upon these elements to propose novel XAI methods aimed at addressing the challenges inherent in pervasive computing.

In pervasive computing, data are frequently collected and integrated from multiple sensors, which can exhibit complex interrelationships such as complementarity or redundancy. While traditional XAI methods can identify significant data segments that contribute to modeling decisions, they often fall short in exploring the intricate relationships between these cognitive segments. This limitation is particularly evident in time series data tasks, where critical information is dispersed across the sequence. For instance, in tasks such as recognizing the Wii action of drawing a circle, crucial data points are spread throughout the sequence. This observation raises several pertinent questions: Are specific data segments crucial for decision-making? Can alternative cognitive segments yield the same decision? Is the importance of a particular cognitive segment for decision-making consistent across different contexts? This chapter is organized around addressing these critical questions.

Corresponding publication:. Y. Huang, N. Schaal, M. Hefenbrock, Y. Zhou, T. Riedel, and M. Beigl. Mcxai: local model-agnostic explanation as two games. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2023

8.1 Introduction

Pervasive computing data is typically multimodal and collected from multiple sensors distributed across various locations. Data from different sensors or data from different locations of the same sensor within a single time window may exhibit diverse interrelationships. For example, in the context of human action recognition using sensors placed on both hands and feet, these relationships can be categorized as follows:

- **Selectivity**: In actions like brushing teeth, the data from the dominant hand is essential, while information from other sensors may be irrelevant or even introduce noise into the analysis.

- **Complementarity**: In tasks such as washing, the coordinated movement of both hands is required, indicating that data from both sensors is necessary to accurately capture the action.

- **Redundancy**: During actions like jumping, both feet provide equivalent information, making the data from either foot alone sufficient for accurate recognition.

To this end, the complex models utilized must integrate information from various channels. Consequently, the XAI methods applied to explain these models should be capable of capturing and reflecting this integration capability. However, traditional XAI methods often involve segmenting data from multiple pipelines into distinct cognitive blocks, selecting a subset of these blocks, and then assigning different levels of importance to them to represent the significance of the interpretation for a given input. While this approach can identify the factors influencing the model's decision, it tends to conflate selectivity, complementarity, and redundancy. This conflation raises several critical questions:

- If two cognitive blocks provide the same information (e.g., both feet), why is one considered important and the other unimportant?

- Is there only one valid interpretation?

- Do the same local data segment blocks hold the same importance in different explanation?
- Is one of the selected data segments (e.g., left-hand information for face-washing) essential or interchangeable?

We believe that the key to answering the aforementioned questions lies in considering the relationships between different cognitive blocks during the explanation process. To address these challenges, we present a novel post hoc explanation approach based on reinforcement learning, called Monte Carlo Tree Search for Explainable Artificial Intelligence (McXAI).

By leveraging the strengths of Monte Carlo Tree Search, McXAI can dynamically explore various combinations of cognitive blocks and their interdependencies. This enables a more nuanced understanding of how different data channels contribute to the model's decisions, ultimately leading to more accurate and interpretable AI systems. This approach involves modeling the explanation process as two games, namely the classification game and the misclassification game.

In a "classification game", the agent is assigned with finding local data segment pattern (termed local pattern) essential to support the correct decision of the model, while in a "misclassification game", the agent seeks local patterns to which the model is sensitive, i.e., local data segment whose perturbations may lead to misclassification. Agents develop their policies based on a search tree representation, which is constructed using MCTS [23]. Each game ultimately outputs a humanized representation in the form of an mct, where each node represents a set of local patterns to be examined. The input is interpreted by extracting information from the tree species: Each node in the tree describes a cognitive block. Each complete path in the tree describes an interpretation. An edge connecting two nodes describes the relationship between the interconnected nodes. Sometimes, there will be more than one node describing the same cognitive block, but the cognitive blocks represented by nodes at the same level of the tree will not duplicate each other. The descendants of a point will not contain the cognitive block represented by its ancestor. Due to the brushing of cognitive blocks, each cognitive block in the tree is important for decision making in the model. However, the same cognitive block has different importance at different locations in the tree. The importance of cognitive blocks is greater in nodes close to the root of the tree.

8.2 Related Work

In this section, we provide an overview of previous work on technologies for interpreting complex models and briefly introduce the basic Monte Carlo tree search algorithm as a preliminary.

8.2.1 Explainable Artificial Intelligence

Numerous approaches have been proposed to explain models in ways that humans can easily perceive.

Permutation Feature Importance (PFI) [137] analyzes prediction changes by randomly permuting local patterns in the instance. Class Activation Map (CAM) [184] decomposes signals propagated within its algorithm and processes them using a global average pool to provide an analysis of the prediction. Similarly, Layer-wise Relevance Propagation (LRP) [14] identifies important pixels by running a backward propagation through the neural network. All of these methods display the contribution of pixels to the prediction through heat maps.

Conversely, CLUE [8] generates explanations with the help of a variational autoencoder. EXemplar [16] explains instances using a generator. MUSE [97] produces explanations in the form of decision trees, approximating the complex model with an interpretable model and optimizing against various metrics. The Bayesian Rule Lists [177] method discretizes the feature space into partitions and defines the decision logic within each partition using IF-THEN rules. SHAP [110] uses subset examination to score feature importance.

All of these methods focus on finding evidence to support the prediction of the complex model but often overlook potential causes of erroneous predictions. Additionally, CAM requires the complex model have a global average pooling layer, and the resulting explanations can be difficult to understand.

8.2.2 Monte Carlo Tree Search

MCTS [23] is a widely-used heuristic-based reinforcement learning algorithm, particularly effective for predicting moves in board games such as chess and Go. MCTS constructs a search tree to estimate the favorability of actions in a given state, with each edge representing a move/action and each node representing a game state. To win a game, the agent performs multiple episodes, each consisting of four phases: selection, expansion, roll-out, and backpropagation.

During the selection phase, the agent selects child nodes according to a selection policy until a leaf node is reached. In the expansion phase, the agent adds one or more child nodes to this leaf node and selects one according to an expansion policy. The roll-out phase follows, where the game is played out to a terminal state (win, lose, or draw) using random moves from the selected node.

Based on the terminal state, a reward r is returned. In the backpropagation phase, every node from the new selected child up to the root node is updated with this reward. The algorithm continues until the predefined number of episodes is reached.

8.3 Method

Given a classification data set $\mathscr{D}$ with n different local patterns and c different classes, a complex model g trained with $\mathscr{D}$ is a system without any internal working knowledge. It takes an instance $x \in \mathbb{R}^n$ from $\mathscr{D}$ as input and outputs the distribution of the classes as a vector $\mathbf{g}(\mathbf{x}) \in [\mathbf{0},\mathbf{1}]^{\mathbf{c}}$, where $\mathbf{g}(\mathbf{x})[i]$ is the probability that the input instance predicted as class i and $\sum_{i=0}^{c} \mathbf{g}(\mathbf{x})[i] = 1$. The prediction of the complex model is correct if the output with the maximum value (probability), i.e. $\underset{i \in \{0,\cdots,c\}}{\operatorname{argmax}} \mathbf{g}(\mathbf{x})[i]$, equals the ground-truth class $y \in \mathbb{N}$.

The importance of local pattern is the most common explanation for classification[1] [18]. Without loss of generality, we assume that the relationship between local patterns within a certain region, e.g., the face of a dog, is key to influencing the decision. McXai determines the importance of local patterns by observing the change in prediction probability after removing the relationship

---

[1] It is different from adversarial attack, which would not change the distribution of the input instance.

among the target local patterns from the input instance x. To keep the shape of the input dimension[2], McXai masks the value of the target local patterns as a given constant τ, where τ should not give away any information about the input instance x. In the case of image data, it is set to zero or the average value of all the local patterns (pixels) in the images.

8.3.1 Tree Representation and Algorithm Framework

The explanation of a given instance $x \in \mathscr{D}$ in McXai is presented as a MCT. Each edge in the tree represents an action $a \in \{0,1\}^n$, corresponding to some of the local patterns. This action is a n-bit array (mask) where all values are zero, except for the bits corresponding to the selected local patterns, which are set to one. Each edge contains three attributes: the number of times the edge has been explored (number of visits), the expected reward for taking the action at the parent node (value) and the change in prediction probability of the target class after taking the corresponding action (performance).

The nodes in the tree represent the states of the game and are divided into three categories:

- **Start (root) node** $x_0 = x$: Represents the initial state of the game.
- **Derived nodes** x_i with $i > 0$: These are masked instances with only one parent node x_j where $i > j \geq 0$. The edge connecting these nodes corresponds to the action a applied to the parent node, i.e., $x_i = a(x_j) = x_j \odot \bar{a} + \tau \cdot a$, where $\bar{a}$ is the logical NOT of action a.
- **Terminal node** x_t: Represents the terminal state of the game.

Each path in the tree corresponds to a local pattern set, and its importance is represented by the expected value of the last edge in the path. A complete path connects the root node to a terminal node, showing the progression and influence of local pattern sets on the model prediction.

As shown in Figure 10.1, given an instance x, McXai explains the black box decision through the classification game and the misclassification game.

---

[2] Most complex models have a fix input dimension (number of local patterns).

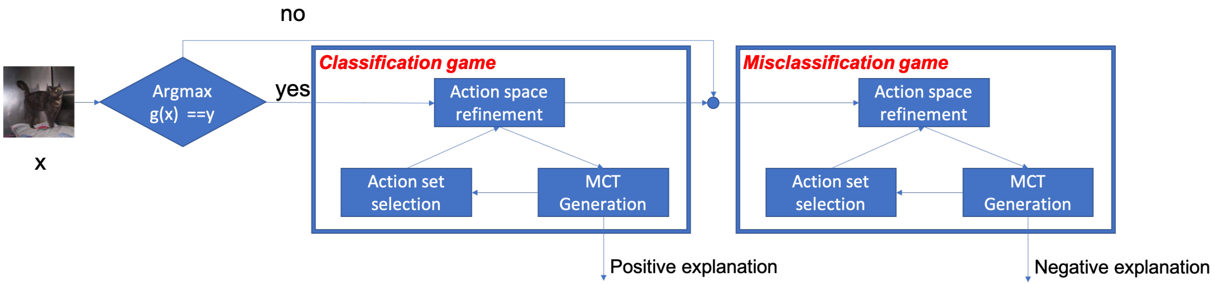


Table 8.1.: A framework of the McXai algorithm.

Depending on whether the black box correctly predicts the instance, the classification game may be skipped. Both the classification game and the misclassification game consist of the same three processes:

- **MCT Generation**: Generates the MCT according to the given action space and root node.
- **Action Set Selection**: Selects a suitable action set from the generated MCT and sends it for refinement.
- **Action Space Refinement**: Splits the actions in the given action set and sets the split actions as the new action space for the MCT generation process.

McXai enables different games to perform distinct functions by defining different root nodes and reward functions for MCT generation.

8.3.2 Monte Carlo Tree Generation

McXai applies MCTS to construct a tree representation. Here, we first introduce the pipeline for building an MCT and then detail each step of the pipeline.

McXai constructs the tree iteratively, with each iteration involving the following four phases:

- **Selection**: McXai traverses the tree from the root node according to the selection policy $\pi_s(\cdot)$, until reaching a leaf node.
- **Expansion**: A new child node is selected according to an expansion policy $\pi_e(\cdot)$ and added to the tree.
- **Roll-out**: A new action is selected randomly and applied to the current node until reaching a terminal node or the maximal depth of the tree.
- **Backpropagation**: The associated reward of the terminal node is computed according to the reward function $r(\cdot)$ and backpropagated along the current path, incrementing the ’number of visits’ and recalculating the expected reward ’value’ of all visited edges.

Given a complex model g, a root node x_0 with target y, and a terminal node x_t, similar to the general MCTS algorithm, McXai applies the Upper Confidence Bound for Trees (UCT) as a selection policy[3].

$$\pi_s(x) = \underset{a \in \mathscr{A}}{\operatorname{argmax}} \left\{ \mu_{x,a} + \lambda \cdot \sqrt{\frac{\log n(x)}{n(x,a)}} \right\} \tag{8.1}$$

where x is a node with descendants in the MCT, $\mathscr{A}$ is the action space, $\mu_{x,a}$ is the 'value' of the edge representing the action a at the node x. Additionally, $n(x) = \sum_{a \in \mathscr{A}} n(x,a)$ denotes the number of visits to node x, and $n(x,a)$ signifies the 'number of visits' to the edge. The parameter λ adjusts the trade-off between the number of visits (exploration) and the expected value (exploitation).

Unlike the selection policy $\pi_s(x)$, which directly uses the information explored to guide decisions, the expansion policy selects the action to expand with the assistance of the complex model.

$$\pi_e(x) = \begin{cases} \underset{a \in \mathscr{A}}{\operatorname{argmax}} \, (g(x)[y] - g(a(x))[y]), & \text{if } \underset{i \in \{0,\cdots,c\}}{\operatorname{argmax}} \, g(x_0)[i] = y \\ \underset{a \in \mathscr{A}}{\operatorname{argmax}} \, (g(a(x))[y] - g(x)[y]), & \text{otherwise} \end{cases} \tag{8.2}$$

where x is a leaf node to which the new node is expanded. The expansion policy $\pi_e(x)$ selects the action that brings the largest prediction probability difference after applying the action. If the complex model correctly predicts the root node x_0, it is the classification game, and $\pi_e(x)$ selects the action that causes the largest prediction probability decrease. Conversely, if the prediction of the complex model for the root node x_0 differs from the target y, it is the misclassification game and $\pi_e(x)$ selects the action that causes the largest prediction probability increase.

To conclude the roll-out process in each iteration, McXai determines whether a node x is a terminal node according to the following function:

---

[3] The feasibility of this approach is theoretically demonstrated in [29].

$$\begin{cases} g(x)[y] < 0.5, & \text{if } \underset{i \in \{0,\cdots,c\}}{\operatorname{argmax}} \; g(x_0)[i] = y \\ g(x)[y] - g(x_0)[y] > t, & \text{otherwise} \end{cases} \tag{8.3}$$

where t is the predefined constant used in the misclassification game. In the classification game, the game ends when the prediction probability of the complex model is less than 0.5. In the misclassification game, the game ends when the difference in the prediction probability between the root node x_0 and the current node x is greater than a predefined threshold t.

The reward function $r(x_t)$ of a terminal node x_t is defined as

$$r(x_t) = \left[(1-\eta) \cdot \left(1 - \frac{l(x_t)}{L}\right) + \eta \cdot q \right] \cdot \mathbb{1}_{\{l(x_t) \leq L\}} \tag{8.4}$$

with

$$q = \left(2 \cdot \mathbb{1}_{\{\underset{i \in \{0,\cdots,c\}}{\operatorname{argmax}} \; g(x_0)[i] = y\}} - 1 \right) \cdot (g(x_0)[y] - g(x_t)[y])$$

where $\mathbb{1}$ is the indicator function, $l(x_t)$ is the length of the path to the terminal node, L is the maximum allowed path length and η is a balance parameter between the path length and the prediction change. This reward function balances the contribution of the path length and the change in prediction probability to the overall reward.

The function $l(\cdot)$ returns the depth of a given node, defined as the number of actions required to reach the given state. The parameter L represents the maximal depth of the tree, serving as a limit on the tree size. Notably, an increase in the maximum depth L correlates with a higher value indicated by the number of actions. Conversely, a smaller L results in a greater reward for the same depth l, making minor step differences more significant. The variable q represents the difference in the prediction probability of the target class before and after removing the selected local patterns. The parameter $\eta \in [0, 1]$ weights the path length and the probability change. If the depth of the terminal node is less than L, the reward is a weighted sum of the depth and the probability change. However, if the depth exceeds L, indicating that the number of actions exceeds a threshold, the reward is set to zero to discourage the selection of such

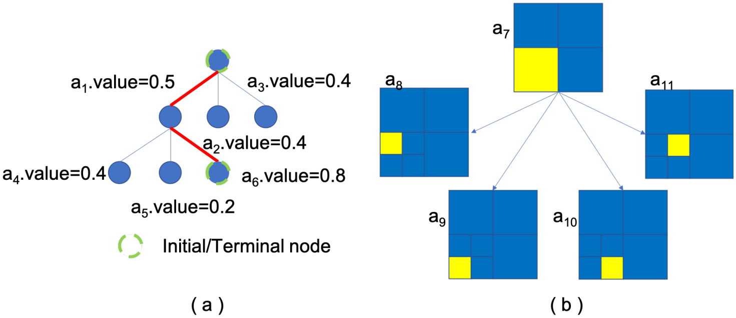


Figure 8.1.: Illustration of action selection (a) and refinement (b).

actions again.

8.3.3 Action Set Selection and Refinement

The ’Action Set Selection’ process in McXai differs from the action selection in ’MCT Generation’, which balances exploration and exploitation. Instead, it focuses on selecting a set of local patterns sensitive to prediction. Given an MCT, McXai starts at the root node and selects the edge with the largest expected value until it reaches a leaf node. If the leaf node is terminal, it returns the complete path from the root to the terminal node. If the leaf node is not terminal, it returns the entire action space. For example, in Figure 8.1, the action set $[a_1, a_6]$ is selected after this process.

The goal of the ’Action Space Refinement’ process is to refine the action set created during the ’Action Set Selection’ process. If no action set is provided, it is initialized with a single action: an n-dimensional array with each value set to one, where n is the dimension of the given instance. During refinement, McXai first identifies the region with a value of one in an action and then divides this region into four equal parts to create four new actions. This process is applied to all actions in the given action set. As illustrated in Figure 8.1, the yellow region represents the area with a value of one. For action a_7, four new actions $\{a_8, a_9, a_{10}, a_{11}\}$ are created during the refinement. If the size of the region with a value of one in any newly created action is smaller than a predefined constant k, the game ends, and the new action space will not be used to generate a new MCT. All MCTs generated throughout the algorithm serve as explanations for the given input instance at different levels of granularity.

The whole McXai algorithm is summarized in Algorithm 3.

8.4 Evaluation

In this section, we experimentally demonstrate the capabilities of our proposed approach in interpreting a complex model in three ways:

- Comparison of local pattern Importance: We compare the importance of positive local patterns identified by the classification game with those found using classical post hoc explainability methods.
- Improvement After Retraining: We measure the improvement of the complex model's performance after retraining it with the local patterns identified during the misclassification game.
- Information from Constructed MCTs: We assess the additional information provided by the constructed MCTs compared to other post-hoc explainability methods.

Unless otherwise specified, McXai's hyperparameters are set as follows: $\tau = 0$, $\eta = 0.5$, $L = |\mathscr{A}|$, $\lambda = 0.5$, and $k = 40$.

8.4.1 Classification Game: Comparing Local Patterns with Positive Impact

We hypothesize that considering individual dependencies between local patterns in an explanation enhances its quality. In our first experiment, our aim was to validate this hypothesis. We designed a task using the open source MNIST dataset and several real-world classification datasets from sklearn, including covertype, kddcup, newsgroup, and face[4]. We compared the performance of LIME [134], SHAP[5] [110], and our proposed McXai model on this

---

[4] The RCV1 dataset is excluded because it is a multi-label classification task.

[5] We selected these two well-known general post-hoc methods because the datasets used are not limited to images and some complex models do not provide gradient information.

task. To demonstrate the generality of the method, we trained different kinds of model for these datasets.

For McXai, the hyperparameter k is set to 10, and the maximal depth L of the constructed tree is set to 10 to limit the tree size, except for the newsgroup dataset, where L is set to 30 due to its significantly larger number of local patterns. A total of 50 instances were randomly selected from each dataset and used as input for the task.

The task design is inspired by the experiment in [110]. Taking the MNIST dataset as an example, the task can be described as follows: Given an instance from the MNIST dataset with the target class $y = 7$, local patterns of the instance are continuously removed according to the proposal of the XAI model until the prediction of the instance (by the complex model) changes to any other class. The post hoc explanation approach aids the task by analyzing the input instance and extracting a list of local patterns ranked by their importance.

To compare the importance of the local patterns identified by each algorithm, we measure the number of steps (NoS) required to change the predicted class according to the local pattern list of each method. Fewer steps indicate that the local patterns found by the corresponding explainability approach are more important.

The results of the experiment are summarized in Table 8.3. The proposed McXai algorithm achieves optimal results on all datasets except the newsgroup dataset, where SHAP has the best results. However, SHAP takes an average of five minutes to analyze each instance, while McXai takes an average of one minute. LIME is the fastest, but yields the worst results.

The runtime of the McXai algorithm is influenced by three factors related to the dataset:

- Number of local patterns: The more local patterns the dataset contains, the longer the run-time. For instance, analyzing an instance of the ’covertype’ dataset takes an average of 30 seconds, while analyzing an instance in the newsgroup dataset takes about one minute.

- Complexity of the Relationship Between Local Patterns and Prediction: This is measured by the number of modified local patterns needed to change the model’s prediction. The fewer the local patterns, the lower the complexity, and the shorter the runtime. For example, McXai takes an average of two minutes to analyze an instance in the dog-cat dataset, which contains 16,384 local patterns, due to its higher complexity.

- Complexity of the complex model: The more complex the model, the longer McXai takes.

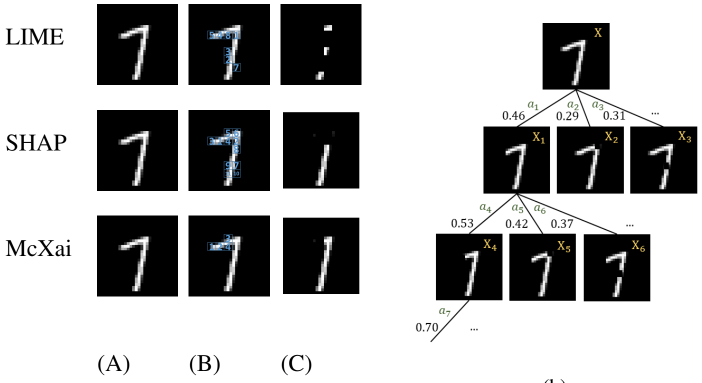
Table 8.2.: Comparing average number of steps (NoS) needed to take to change the prediction of complex model according to the suggestion of the LIME, SHAP and McXai methods.

Dataset	Type	No. local patterns	NoS-LIME	NoS-SHAP	NoS-McXai
MNIST	image	784	7.23 ± 5.65	6.23 ± 5.34	**4.82 ± 2.65**
covertype	relational	54	10.32 ± 3.36	1.72 ± 0.82	**1.59 ± 0.88**
kddcup	relational	41	**1.0 ± 0**	2.2 ± 2.0	**1.0 ± 0**
newsgroup	text	15698	46.26 ± 36.8	**6.69 ± 7.78**	8.2 ± 3.72
face	image	4096	24.94 ± 7.76	17.62 ± 6.67	**5.62 ± 0.82**

Figure 8.2a illustrates an MNIST example highlighting the local patterns identified by different XAI models. In this instance, LIME's identified local patterns are scattered across the number. However, this does not imply that the complex model bases its predictions on the overall skeleton of the number, as demonstrated by the properties found by SHAP and McXai. This observation suggests that LIME overlooks the competing relationships (e.g., one local pattern being less important when another is present) and conditional relationships (e.g., one local pattern being important only when another local pattern is present) between local patterns, leading to a misjudgment of some local patterns' importance.

Interestingly, SHAP and McXai identified almost the same local patterns, with differences only in their ranking. This difference highlights a unique attribute of McXai: it ranks the importance of already explored local patterns during its operation (as reflected by the selection policy) and prioritizes the exploration of higher-importance local patterns. The corresponding tree structure for McXai is partly shown in Figure 8.2b.

From this experiment, we conclude that the positive impact of the local patterns identified by McXai is more significant than those identified by the other two methods. This superiority is attributed to McXai's ability to consider and



(a) (b)

Figure 8.2.: (a) Ranks of local pattern importance and masked image for each method. (A) Shows the input instance. (B) Shows the explanation created by each algorithm. The local patterns colored in blue have a positive local pattern importance according to each method. (C) Shows the masked image which is no longer predicted as 7. (b) Shows the MCT created by McXai from the example of (a). The state x is the input instance. The value of each edge is written beside the corresponding edge. The path $[a_1, a_4, a_7]$ is the best path containing the actions with the highest expected value of the considered states.

rank the dependencies between local patterns effectively.

8.4.2 Misclassification Game: Testing the Improvement of Complex Model through Retraining

The misclassification game identifies local patterns that are insignificant to the target class but sensitive to other classes. In practice, this manifests as a reduction in the probability of an instance being correctly predicted by adding these local patterns to it. We hypothesize that if the misclassification game can identify such negatively impactful local patterns, then counteracting their effects should improve the performance of the complex model. To test this hypothesis, we designed the following experiment using the open-source dog-cats dataset from Kaggle:

1. Dataset Preparation: We split the dog-cats dataset into a training set $\mathscr{D}_{train}$ and a testing set $\mathscr{D}_{test}$. We train a complex model using $\mathscr{D}_{train}$ and record its accuracy.

2. Local Pattern Analysis: Each instance in the training set is analyzed using our proposed approach to identify local patterns.

3. Dataset Modification: We first remove the local patterns identified by the misclassification game from each instance in the training set to form a new dataset $\mathscr{D}_{mis}$. Then we remove the local patterns identified by the classification game and the misclassification game from each instance in the training set to form a new dataset $\mathscr{D}_{both}$.

4. Retraining and Evaluation: We first retrain the complex model using $\mathscr{D}_{train}$ and $\mathscr{D}_{mis}$, then record its accuracy. Then we retrain the complex model using $\mathscr{D}_{train}$ and $\mathscr{D}_{both}$, then record its accuracy.

5. Performance Comparison: We compare the performance of the three trained models (original, $\mathscr{D}_{mis}$, and $\mathscr{D}_{both}$) on $\mathscr{D}_{test}$.

We conducted this experiment using the following five torchvision pretrained models: *(i)* MnasNet [160] with a depth multiplier of 0.5 (mnasnet0_5); *(ii)* MnasNet with a depth multiplier of 1.0 (mnasnet1_0); *(iii)* DenseNet121 [68]; *(iv)* WideResNet [179]; *(v)* GoogleNet [64]. These models, pretrained on the ImageNet dataset, converge quickly in our experiments[6].

For each model, we trained for 20 epochs and repeated the process five times to record the mean and standard deviation of the performance. The dog-cats dataset consists of 2500 training images and 500 test images, with an equal representation of cats and dogs in both sets. We used accuracy as a performance metric.

The results are summarized in Table 8.3. In general, removing local patterns identified by the misclassification game led to improvements in accuracy or stability of the models. This demonstrates that the misclassification game can

---

[6] All models converged after two to three episodes, except for mnasnet1_0, which took an average of seven episodes. Thus, the size of the data set has minimal impact on the results

identify factors leading to incorrect predictions, and counteracting these factors improves the model's performance.

Additional insights from the experiment include: *(i)* The accuracy improvement varies by model. Models with lower original accuracy showed a more pronounced improvement. For instance, in GoogleNet, where the original accuracy was high, the improvement in accuracy was minimal (less than 1%), but the improvement in stability was significant. *(ii)* Removing local patterns identified by both the classification and misclassification games improved performance in two cases.

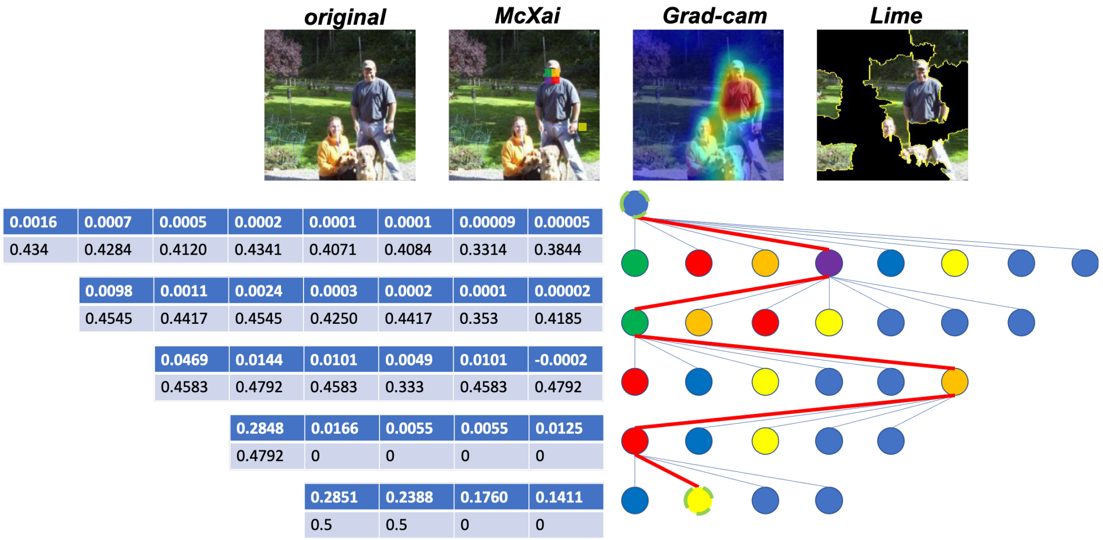
The classification game identifies local patterns that are crucial for correct predictions by the complex model. Removing these local patterns from the training set highlights the influence of other local input patterns, thereby increasing the generality and robustness of the complex model.

Table 8.3.: Comparing performance of complex model: mnasnet0_5, mnasnet1_0, DenseNet121, WideResNet and GoogleNet in these three different situations: (1) trained with training set $\mathscr{D}_{train}$ (base_score) (2) trained with training set $\mathscr{D}_{train}$ and $\mathscr{D}_{both}$ (score_both) (3) trained with training set $\mathscr{D}_{train}$ and $\mathscr{D}_{mis}$ (score_mis)

	base_score (%)	**score_both (%)**	**score_mis (%)**
mnasnet0_5	87.78 ± 3.31	87.98 ± 1.71	**89.16 ± 3.25**
mnasnet1_0	88.65 ± 3.17	**92.83 ± 1.15**	91.03 ± 3.06
DenseNet121	91.23 ± 1.46	**93.7 ± 2.63**	92.98 ± 2.51
WideResNet	83.35 ± 2.25	79.49 ± 4.42	**87.58 ± 4.66**
GoogleNet	93.97 ± 3.96	94.52 ± 1.33	**94.58 ± 1.33**

8.4.3 Extracting Explanation from MCT

In the first two experiments, we demonstrated McXai's ability to identify both positive and negative factors influencing the black box's decision-making process. In this experiment, we specifically describe the additional insight McXai provides beyond identifying local pattern impact. Figure 8.4 illustrates the explanations for a given instance from three different methods: McXai, Grad-CAM, and LIME. Notably, the complex model predicts the original image as a dog, yet each method offers a different explanation.



0.0016	0.0007	0.0005	0.0002	0.0001	0.0001	0.00009	0.00005
0.434	0.4284	0.4120	0.4341	0.4071	0.4084	0.3314	0.3844

0.0098	0.0011	0.0024	0.0003	0.0002	0.0001	0.00002
0.4545	0.4417	0.4545	0.4250	0.4417	0.353	0.4185

0.0469	0.0144	0.0101	0.0049	0.0101	-0.0002
0.4583	0.4792	0.4583	0.333	0.4583	0.4792

0.2848	0.0166	0.0055	0.0055	0.0125
0.4792	0	0	0	0

0.2851	0.2388	0.1760	0.1411
0.5	0.5	0	0

Table 8.4.: An example of the explanation of the three different methods: McXai, Grad-cam, LIME. The first row shows the original image, McXai's explanation, Grad-cam's explanation, LIME's explanation in order. The lower right part is the MCT generated by the McXai and the lower left part is the corresponding 'performance' and 'value' attributes of each edge in the MCT.

None of the methods identifies the dog as the key factor in the decision: McXai identifies the man's face as crucial. Grad-CAM emphasizes the man's clothing. LIME highlights numerous areas, including the man, woman, and plants in the environment.

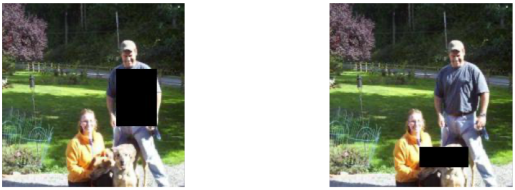
The explanations from Grad-CAM and LIME are inherently complex or difficult to interpret, especially when numerous local patterns are involved, making it hard to assess their correctness. In contrast, McXai's explanations are supported by the generated MCT.

The lower right part of Figure 8.4 displays one of the MCTs generated by McXai. Nodes of the same color, except blue, correspond to identical actions. The nodes in each layer are sorted according to the 'performance' attribute of their input edges. The lower left part of Figure 8.4 shows the 'performance' (first row) and 'value' (second row) attributes of each layer's edges in the MCT. Using the MCT, we can determine the effect of removing each local pattern from the original image on the prediction probability.

For instance, the red-marked edge in the MCT, chosen by McXai, has the greatest and shortest impact on prediction. This path is illustrated in the upper part of Figure 8.4. By adding the performance along this path, we find that removing the selected local patterns reduces the probability of predicting the image as a dog to 46%, thus confirming McXai's explanation.

Additionally, the MCT provides further valuable information: *(i)* Local pattern Combination Impact: From the first layer's performance data, we observe that individual local patterns alone have little impact on prediction; their combined effect is significant. *(ii)* High-Performance local patterns: High-performance local patterns are not always optimal. For example, the last node in the third layer has a negative performance, meaning that removing it increases the dog's prediction probability. However, its value, or expected reward, is the highest. *(iii)* Local pattern Interdependencies: The MCT reveals dependencies between local patterns. For instance, removing the green local pattern when the purple local pattern is already removed results in a performance of 0.0098.

Furthermore, we tested the effect of removing the man's clothing and the dog from the image on the model's predictions. As shown in Figure 8.3, both mod-



(cat: 1.4168e-04, dog: 9.9986e-01) (cat: 1.2461e-06, dog: 1.0000e+00)

Figure 8.3.: Two examples of complex model prediction probabilities

ifications have almost no effect on the predictions of the complex model. This reinforces that McXai effectively identifies the most impactful local patterns, providing deeper insight into the decision-making process. It can be concluded that although the explanations of all three methods differ from human perception, McXai's explanation is the most consistent with the black box.

8.5 Discussion

In this paper, we propose a novel approach called McXai to enhance the reliability of complex models by elucidating the principles behind their decisions. This method analyzes the classification decisions of a complex model by examining single local patterns or local pattern sets of an input instance in a manner that is easily comprehensible to humans, identifying factors that positively or negatively affect model predictions.

The cornerstone of this approach is the formalization of the XAI problem as two distinct games, each focusing on discovering specific properties. This simulation enables us to address the XAI problem in a manner similar to a classic game, training an agent to handle the problem. Inspired by AlphaGo [107], we adapted and applied the MCTS algorithm to our task, presenting the analysis of the prediction of an input instance in a tree structure while simultaneously training the agent. This significantly enhances explainability.

In our experiments, we compare the positive local patterns identified by various XAI approaches and test the negative local patterns identified by our proposed method across different complex models. These experiments demonstrate the capability of McXai in explaining complex model predictions. Furthermore, we found that using the identified local patterns, we can further improve the performance of complex models.

In McXAI, the model decision is elucidated using a tree structure, where the dependencies between different nodes are represented by the edges connecting them. Each distinct path within the tree corresponds to a candidate explanation of the decision. This approach enables the efficient extraction of relationships between local patterns dispersed across multidimensional data, thereby offering a more comprehensive explanation for the model's decision-making process.

Algorithm 3 McXai

```
Input: black box $g$, input $x_0$, target $y$
Parameter: $\tau$, fine-grad $k$, max-depth $L$, selection trade-off $\lambda$, reward trade-off $\eta$
Output: $o$
Let $o \leftarrow []$
Initialize action space $\mathscr{A}$ as a list that contains a single n dimensional array with value one $\{a_0\}$.
if $\underset{i \in \{0,\cdots,c\}}{\text{argmax}}\ g(x_0)[i] = y$ then
    execute process 'Action Space Refinement' to update action space $\mathscr{A}$
    while $(\forall a \in \mathscr{A})\ sum(a) > k$ do
        $o = o \cup$ execute process 'MCT Generation' to generate MCT with the given action space $\mathscr{A}$ and root node $x_0$
        execute process 'Action Set Selection' to select action set and set it as new action space $\mathscr{A}$
        execute process 'Action Space Refinement' to update action space $\mathscr{A}$
    end while
end if
apply all actions in the action space to $x_0$ to generate root node for misclassification game and set it back to $x_0$
Initialize action space $\mathscr{A}$ as a set that contains a single n dimensional bit array with value one $\{a_0\}$.
execute process 'Action Space Refinement' to update action space $\mathscr{A}$
while $(\forall a \in \mathscr{A})\ sum(a) > k$ do
    $o = o \cup$ execute process 'MCT Generation' to generate MCT with the given action space $\mathscr{A}$ and root node $x_0$
    execute process 'Action Set Selection' to select action set and set it as new action space $\mathscr{A}$
    execute process 'Action Space Refinement' to update action space $\mathscr{A}$
end while
return $o$
```

9. Explain with Temporal Information

In the previous section, we explored the interactions between local data segment patterns and their influence on the model's decisions. This section builds on that analysis by incorporating additional insights from Part II, with a particular focus on the noisy and time-dependent nature of the data.

In pervasive computing, data are often noisy due to factors such as sensor limitations and data transmission issues. Although specific values may vary across different time segments, they can still describe similar patterns, such as trends in acceleration or deceleration. Moreover, the temporal dependence of these local patterns can significantly affect the model's decisions. For example, if the variable x represents the acceleration of a car, during the acceleration process, x should initially increase and then decrease. A single time-slice feature, indicating either an increase or a decrease, is insufficient to accurately guide the model's judgement. These observations give rise to several important questions: *(i)* How can noise be effectively eliminated from a time series to identify consistent local features? *(ii)* How can dependencies between different local features be identified and leveraged to explain the model's decisions? In this chapter, we will delve into these questions.

Corresponding Publication:. Y. Huang, C. Li, H. Lu, T. Riedel, and M. Beigl. State graph based explanation approach for black-box time series model. In *World Conference on Explainable Artificial Intelligence*, pages 153–164. Springer, 2023

9.1 Introduction

With the rapid advancement of always-on network technology and micro-/nano-electromechanical systems, live sensor information has become increasingly crucial in pervasive computing. Schilit *et al.* [143] identified various types of implicit interactions with computer systems based on contextual information as early as 1994. Subsequently, this seminal work has fueled the development of diverse context-aware recommender systems, ranging from context-aware advertising to adaptive music playlists based on user behaviors [108].

Devices such as smartwatches and mobile phones, equipped with numerous sensors, are typical examples that enable these implicit interactions. They collect time series data on daily human activities and provide recommendations derived from these data. However, as the models driving these recommendations become increasingly complex, explaining their underlying logic becomes more challenging. Thus, there is a growing demand for explanatory methods tailored to sensor based time series AI models, which clarify how these models process sequential data relationships to generate context-based recommendations.

While explainability research has made substantial progress in the computer vision domain [164], the unique characteristics of sensor based time series data, such as its noisy nature and sequential structure, which are fundamental to many context-aware information retrieval models, present challenges for directly applying these advancements to time series explanations.

In recent years, several methodologies have emerged for interpreting time series models. Parvatharaju *et al.* [124] and Crabbe *et al.* [31] developed approaches that assess the significance of input data by introducing perturbations. Schlegel *et al.* [146] adapted the LIME approach to the time series domain by using six distinct segmentation methods and elucidating the target model through the training of local models with generated segments. Doddaiah *et al.* [36] extended this method to include multi-class forecasting issues. Additionally, Guidotti *et al.* [58] utilized rules created within the latent space and employed mean squared error (MSE)-based Shapelets to identify the segments that influence the model predictions.

While these methodologies provide insight into model behavior, they focus primarily on the importance of individual input segments. This approach is limited given the sequential nature of time series data, which is integral to most predictive models. For example, in predicting a head nod using data from a gravitational acceleration sensor on the head, a comprehensive assessment should include the entire sequence of movements: the initial acceleration in the direction of gravity, returning to zero, accelerating in the opposite direction, and returning to zero again. Although the value at any specific moment might show a linear correlation with the predicted outcome, it does not fully explain the prediction process.

To address the limitations of existing time series explanation methods, we propose a novel approach called State-graph Based eXplanation Artificial Intelligence (SBXAI). This method leverages Bayesian optimization to aggregate adjacent data points in a given example, thereby creating multiple, more comprehensible data units or states and reducing the effect of data noise. Additionally, it employs Directed Circular Graphs (DCG) to visualize the sequential relationships between these states, thereby elucidating the model's decision-making process.

9.2 Related Work

Theissler *et al.* [164] categorized existing time series explanation methods into three main categories: Time-point-based explanations, subsequences-based explanations, and instance-based explanations.

- **Time Points-Based Explanations**: These methods assign a weight to each time point in the input time series data, indicating the contribution of each value to the model's final decisions [70; 117; 144; 185].

- **Subsequences-Based Explanations**: These methods identify the input sub-segments most representative of the model's decisions. These sub-segments can be real-valued subsequences directly extracted from the raw time series [27; 114; 148] or discretized representations obtained through aggregate algorithms [125; 129; 178].

- **Instance-Based Explanations**: These rely on the entire time series instance to explain the model's judgments. Examples include features extracted from the entire time series instance [47; 151], the most representative examples of a particular classification made by the model [48; 161], and counter-examples that lead to changes in classification through minimal modifications [33; 87].

However, Time Points-Based Explanations and Subsequences-Based Explanations often struggle to illustrate the effect of chronological order in their explanations. Instance-Based Explanations, while useful, are based on implicit assumptions that the features or examples they identify are inherently explainable. This assumption is not always valid, as time series data, especially those with longer or more complex trends, can be difficult to understand, even for experts with domain knowledge.

9.3 Method

The preceding discussion inspired the development of an effective explanation method for time series models, emphasizing two key elements: *(i)* the ability to analyze and visually demonstrate the impact of the chronological order of input values on model predictions, and *(ii)* the avoidance of presenting excessively long time series segments that are difficult for humans to comprehend, as well as single values that provide little information. The proposed explanation method is built on these principles, as illustrated in Figure 9.1(a). The framework comprises three modules: the Segment & Clustering Module, the Perturbation Module, and the Explanation Module.

9.3.1 Segment & Clustering Module

The Segment & Clustering Module is responsible for dividing the time series data into smaller segments and categorizing them based on their similarity. The objective of this module is to summarize the time series input to be explained, referred to as Interested Data Entry x_I, into a series of clustered segments (hereinafter referred to as states) s_I that are easily interpretable by humans.

Specifically, x_I is a time series input of length m, as shown in Equation 9.1:

$$x_I = \{t_1, t_2, \ldots, t_m\} \tag{9.1}$$

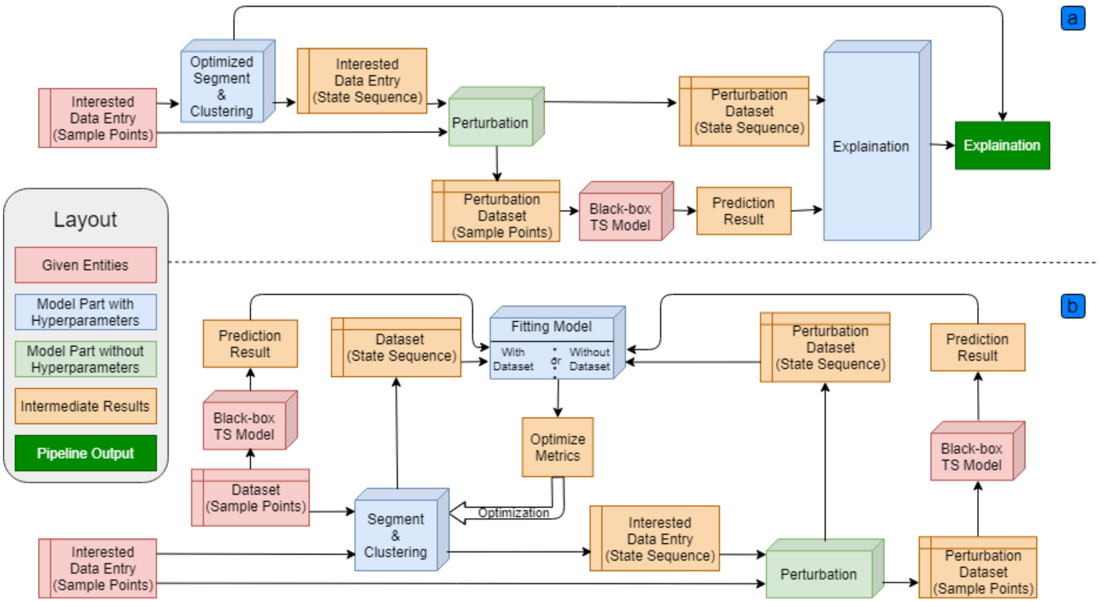


Table 9.1.: (a). Description of the Whole Model Pipeline. (b). Details of the Hyperparameter Optimization Procedure for the Segment & Clustering Module.

After processing by the Segment & Clustering Module, x_I is transformed into a sequence of states s_I with length n, where each state has the same length J, as shown in Equation 9.2:

$$s_I = \{g_1, g_2, \ldots, g_n\}, \ \ n = \lceil m/J \rceil \tag{9.2}$$

The value of each state g_i is determined by a clustering algorithm $G(e \mid K, T)$, where e is the segment to be clustered, K is the number of clusters and T represents all available segments, as described in Equation 9.3:

$$g_i = G([t_{i*J+1}, \ldots, t_{(i+1)*J}] \mid K, T) \tag{9.3}$$

The module includes two essential hyperparameters: the length of the segment J and the number of clusters K. These hyperparameters can be optimally set using Bayesian optimization. The underlying principle is that proper segmentation and clustering of the data should preserve the information within the time series. To achieve this, we train a Fitting Model $F(s)$, which uses the sequences of states as input and predicts the output of the black-box model to be explained. The performance of the Fitting Model serves as an indicator of the effectiveness of the Segment & Clustering Module (Equation 9.4).

$$\hat{K}, \hat{J} = \underset{K,J}{\arg\max} \sum_{s} \mathrm{ACC}(F(s)) \tag{9.4}$$

Here, $\hat{K}$ and $\hat{J}$ represent the optimal number of clusters and segment length, respectively, and $\mathrm{ACC}(F(s))$ denotes the accuracy of the Fitting Model. The better the Fitting Model performs, the more effective the Segment & Clustering Module is.

The Fitting Model requires sufficient data for training, which may not always be available in certain scenarios that require time series black-box explanations. To address this issue, two processes are outlined in Figure 9.1(b).

The process on the left side of the Fitting Model addresses scenarios where the original dataset used to train the black-box model is available during the explanation stage. For instance, if the model's trainer seeks to explain misclassified samples to improve model accuracy or enhance robustness against adversarial attacks, the original training data can be utilized. In this scenario, data availability is not a concern. All data entries from the given dataset are used to obtain T and to train the Fitting Model.

The process on the right side of the Fitting Model addresses scenarios where the original dataset used to train the black-box model is unavailable during the explanation stage. This is relevant when users of a black-box model have concerns about its output. In this case, x_I is processed through the Segment & Clustering Module to obtain its state sequence representation s_I.

Next, s_I is randomly shuffled and, for each shuffle, the corresponding original sample point representation is noted. This generates two perturbation datasets: the state sequence perturbation dataset and the sample point perturbation dataset. The state sequence perturbation dataset is used as the input for the Fitting Model, while the sample point perturbation dataset is fed into the black-box model to obtain the labels needed to train the Fitting Model.

9.3.2 Perturbation Module

The task of the Perturbation Module is to shuffle the s_I obtained from the Segment & Clustering Module to create a perturbation dataset. Utilizing the one-to-one correspondence between the states and the original data, the representation of the sample points of the shuffled s_I is also generated. This enables us to obtain the black-box model's predictions for the perturbation dataset. These two elements, the shuffled state sequences and their corresponding black-box model predictions, serve as inputs for the next module.

9.3.3 Explanation Module

As an output of the previous module, we conducted various chronological perturbations on the interested data entry and obtained the prediction results of the black-box model for these perturbations. By analyzing the model's responses to these perturbations, we can elucidate the behavior of the black-box model. In this module, we fit an explainable model to capture the behavior of the black-box model, enabling us to use the fitted model's explanations to interpret the black-box model.

In our study, we selected the HMM as the explainable model due to its clear and straightforward visualization. Using an instance from the ’Allgesturewiimote Y’ dataset, specifically the ’Pick-up’ class [1], and employing a LSTM black-box model, the final explanation comprises three parts.

First, the representation state generated by the Segment & Clustering Module (see Figure 9.2(c)) is discussed. Each state, representing the center of a cluster, simplifies the understanding of data trends. For instance, state 0 signifies stabilization, while states 1 and 3 indicate sinking and rising, respectively. Each state is composed of multiple single values, with its complexity defined by its length.

The second part of the explanation focuses on the importance of different features. This feature importance is determined at the state level using the existing counterfactual-based explanation method TS-MULE [146]. This importance is visually represented by the length superimposed on the state transition curve of the interested data entry, as shown in Figure 9.2(b). The explanation highlights the initial segment of the instance (state 1) as critical and assigns various importance values to it.

The final part of the explanation focuses on the significance of various state transitions, illustrated by the transitions graph of HMM shown in Figure 9.2(a). In this graph, each node represents a state, and each edge represents a transition. The edge values describe the importance of the corresponding transitions. For instance, in this example, the transition from state 3 to state 1 is assigned a value of 1, while the transition from state 1 to itself has a value of 0.8. Notably, the absence of an edge from state 3 to itself indicates a value of 0. This suggests that the classification of this example hinges on the sensor value remaining stable for an extended period following a downward trend.

Figure 9.1 visually depicts TS-MULE [146]’s explanation of the selected instance. TS-MULE clarifies the instance through the significance of the original signal segment, represented by the color overlay on the value transition curve of the interested data entry. As shown in Figure 9.1, TS-MULE attributes the classification of the given instance as a "pickup" primarily to the segment where the sensor value exhibits a decreasing trend, with the degree of importance corresponding to the degree of decrease.

---

[1] This class is characterized by the user picking up the control device from its neutral motionless position without any specific predefined manner [60].

Compared to the SBXAI method, we found the following: *(i)* Although TS-MULE assigns significance to each individual segment, the segments considered important vary widely and the manner in which these segments contribute to the prediction remains unclear. In contrast, SBXAI uses states derived from clustering techniques to characterize each segment, represented by a cluster kernel that outlines a prevalent trend among the respective clusters. For instance, the segments TS-MULE deems essential align with the state that delineates a descending trend, providing greater comprehensibility than the initial segments. *(ii)* Decisions in time series are generally based on trend values rather than the magnitude of discrete values. This is exemplified by the "pickup" process, which involves the progression from increasing to maintaining stabilization. TS-MULE emphasizes the significance of individual segments (e.g., sinking) while disregarding the sequential relationship between distinct segments (e.g., transition from sinking to maintaining stabilization). This alteration in the trend is distinctly observable within the state transition diagram in the SBXAI approach. *(iii)* TS-MULE argues that only the decreasing part of the values is relevant to the decision-making, without considering the stable part. This contradicts the class description, which indicates that the stable state is also a critical component of the classification.

9.4 Evaluation

In the previous section, we demonstrated how SBXAI explains a given instance. In this section, we empirically validate the reliability of the generated explanation, specifically how accurately the explanation provided by the proposed method aligns with the black-box model's behavior toward the prediction.

To validate, we randomly sample items from a dataset and explain them using the selected methods. We then modify each item according to its explanation. If the black-box model prediction changes after the modification, we record the modification as a success. For each dataset, we repeat this process 100 times to calculate the average attack success rate (ASR), which indicates the importance of the rules broken by the modification. We compare the performance of the following modification methods: *(i)* Replace the value

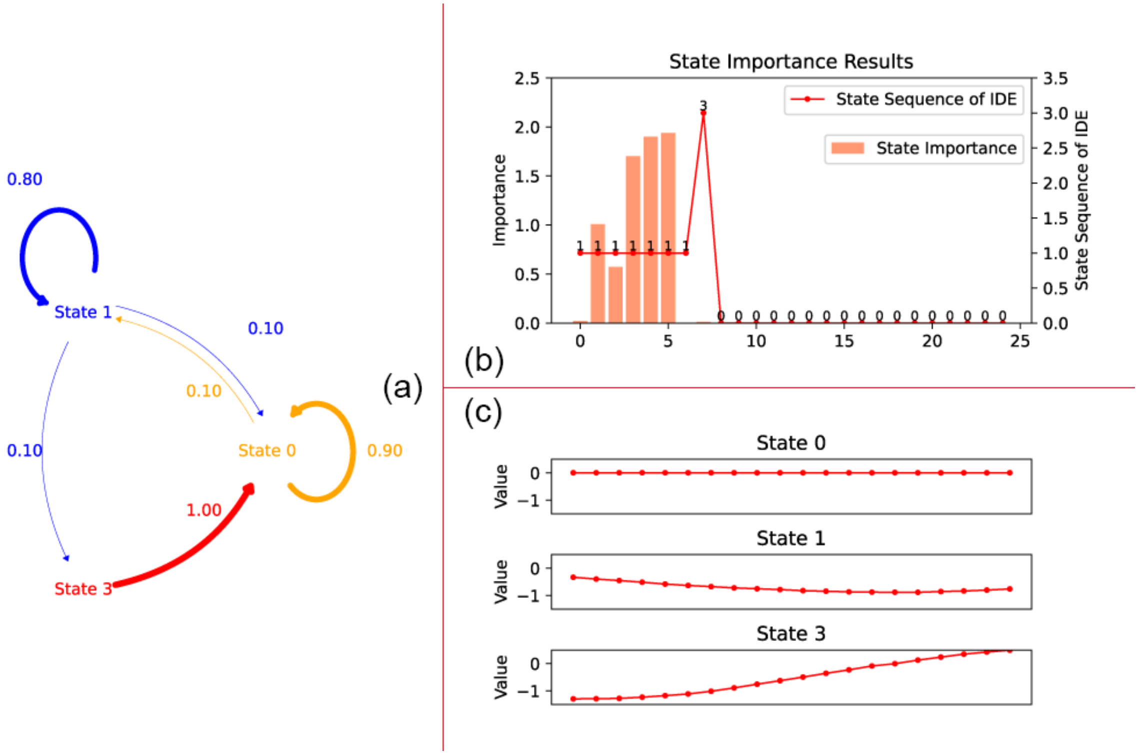


Table 9.2.: (a). Transitions graph of HMM, showing the importance of different state transitions. (b). The state sequence of the interested data entry (IDE) and the background color showing the feature importance. (c). The Correspondence between the state and the cluster core. The x-coordinates in Figures (b), (c) both represent timestamp.

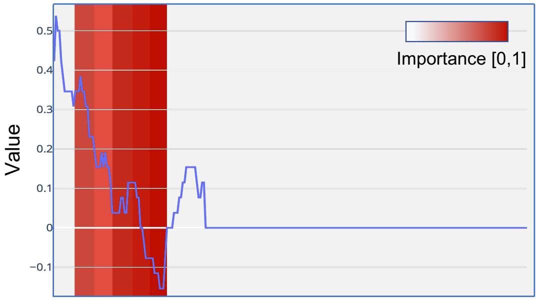


Figure 9.1.: Explanation for the example data entry given the TS-MULE, which 1). only contains the feature importance, 2). only shows the original data instead of summarizing the characteristic of the trend.

of a random position in the item with the least important state identified by the proposed method in the given item (Random-Fe). *(ii)* Exchange the positions of two random states in the given sequence (Random-Seq). *(iii)* Remove an important feature identified by TS-MULE [146] using the proposed Segment Module (TS-MULE-Fe). *(iv)* Exchange the positions of the state pair considered the most important by the proposed method (SBXAI-Seq). For example, given the state sequence [abcba], if the proposed method identifies the sequence relationship 'ab' as crucial for class identification, we transform the original sequence to [bacba] by exchanging the states.

We aim to highlight the significance of a successful attack by limiting the strength of the applied modification. During the experiment, we utilized the Hyperopt [17] package for optimization, setting its hyperparameter max_iter to 100 and using the 'tpe.suggest' optimization algorithm.

To ensure the reliability of the results, we conducted experiments on various UCR datasets [32], which feature different numbers of classes and sequence lengths. The black-box model used in these experiments is constructed

Table 9.3.: Attack Success Rate (ASR) of different modification methods.

Dataset	Number Class	Sequence Length	Random-Fe	Random-Seq	Fe	SBXAI-Seq
AllGestureWiimote	10	vary	10	7	86	**89**
Car	4	577	16	10	63	**68**
YoGA	2	426	19	12	76	**81**
ShapesAll	60	512	13	17	50	**72**
PigAirwayPressure	50	2000	3	9	18	**95**
Mallat	8	1024	10	5	**98**	88
InlineSkate	7	1882	20	17	42	**85**
CricketY	12	300	11	16	69	**83**
RefrigerationDevices	3	720	11	10	46	**64**
MixedShapesRegularTrain	5	1024	3	2	37	**65**
BirdChicken	2	512	13	28	79	**89**
WordSynonyms	25	270	12	12	57	**64**
DodgerLoopGame	2	288	0	20	42	**72**
FreezerRegularTrain	2	301	16	15	45	**96**
EthanolLevel	4	1751	11	8	96	**100**
LargeKitchenAppliances	3	720	6	8	56	**72**
FiftyWords	50	270	8	17	74	**75**
ArrowHead	3	251	16	20	74	**84**
EOGHorizontalSignal	12	1250	18	14	70	**80**
ACSF1	10	1460	0	1	17	**90**

with LSTM layers. To the best of our knowledge, no heuristic method employing sequential relations has been used to explain black-box models, so we did not compare the proposed method with other sequential-based explanatory methods. The search space and the black-box models are available in `https://github.com/HuangYiran/sbxai`.

As summarized in Table 9.3, the elements identified by the two different modification methods, TS-MULE-Fe and SBXAI-Seq, have a substantial impact on the black-box prediction. The average attack success rate for modifying the state order is 79.9%, while that for eliminating important states is 60.4%. These results suggest that the explanation generated by the proposed method is reasonable. We can conclude that the prediction of a time series depends not only on its states but also on the sequential relationship between them. For time series data, sequential changes between states have a more pronounced effect on decisions than the removal of individual states. Additionally, we find that some models are strongly influenced by the sequential relationship and are almost unaffected by the individual states, such as in the case of 'PigAirwayPressure'. This is likely because the states considered important appear multiple times within the same item. Moreover, no significant correlation was found between the attack success rate, number of classes, and sequence length.

9.5 Discussion

Most pervasive computing data are collected through sensors, which often leads to noisy data, and the temporal relationships within these data can significantly impact model decisions, which poses a huge challenge to the current XAI methods. To address these challenges, we propose SBXAI, an explanation method specifically designed to interpret complex time series models in the presence of high noise. SBXAI mitigates the effects of noise through clustering and Bayesian optimization, and visualizes the temporal relationships in the data using DCG. Empirical experiments conducted on 20 different data types demonstrate the effectiveness of this method.

Despite this achievement, there is still room for improvement in terms of experimentation and further development of algorithms. For example, all the black-box models in the experiment have the same structure. It is necessary to verify the impact of the proposed method on models with different architectures. Additionally, further improvements could be made to the algorithms by enlarging the search space, such as incorporating more segmentation and clustering methods. This could enhance the robustness and applicability of the proposed method across a wider range of scenarios.

10. Explain with other Information

In the previous two chapters, we developed novel explanation methods that focused separately on the spatial and temporal features of pervasive computing data. However, each method addressed only a single aspect of the characteristics of the data. In this chapter, we will take a more comprehensive approach by exploring the feasibility of integrating multiple factors that are critical to model decisions into a single interpretation method. This includes examining the interdependencies between various channels, both long-term and short-term temporal dependencies, local area patterns, frequency patterns, and location information.

Corresponding Publication:. Y. Huang, Y. Zhou, H. Zhao, L. Fang, T. Riedel, and M. Beigl. Extea: An evolutionary algorithm-based approach for enhancing explainability in time-series models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 429–446. Springer, 2024

10.1 Introduction

The increasing prevalence of sensor-based applications in pervasive computing domain, such as motion capture games and medical assistance systems, has heightened the reliance on time series data. However, the multimodal nature of time series data often leads to the development of complex and opaque models, creating trust issues in practical applications. Recently, several XAI methods for time series black-box models have been proposed. For example, SBXAI [73] elucidates how sequential structures in different cognitive blocks influence decision-making processes using a DCG. Here, the cognitive block refers to crucial data segments essential for model decisions. Similarly, MCXAI [74] examines the relationships among various cognitive blocks using a Monte Carlo tree structure. Another approach, TS-MULE [145], adapts LIME [134] to time series data, employing multiple distinct segmentation strategies. Although these methods advance the understanding of black-box time series models, they have limitations.

One major challenge is generating multiple explanations for a single input. State-of-the-art models often involve an ensemble decision mechanism [75], indicating that identical inputs can be governed by multiple underlying rules. These rules typically manifest through internal model mechanisms, such as the ensemble approach in a random forest model and the dropout mechanism in a neural network. Prevailing XAI methods, which focus on feature importance and Pertinent Negative counterfactuals [167], tend to combine all rules into a single explanation, leading to potential confusion and inaccuracy. Moreover, most Pertinent Positive counterfactual-based methods [167], which have not yet been tested on time series data, offer only one optimal explanation, disregarding the range of possible decision rules.

Additionally, certain time series characteristics, such as frequency information and sequential interference, are not adequately explored. The optimization of cognitive blocks in MCXAI and SBXAI is constrained, potentially hindering the discovery of high-quality cognitive blocks. Furthermore, setting hyperparameters in current methodologies, such as the number of segments in SBXAI, remains a formidable challenge.

In summary, we face a counterfactual-based XAI task that seeks multiple, distinct, and optimized explanations under multi-objective constraints related to different time series characteristics. We posit that evolutionary algorithms [39] are well-suited to tackle these challenges. Consequently, we propose ExTea, an EXplainable Artificial Intelligence method for time series models based on an Evolutionary Algorithm. In ExTea, each individual represents a potential explanation and is endowed with a self-optimization function. Our method differs from traditional evolutionary algorithms by employing a pyramidal structure for the individual pool, segmented into layers for newborns, tested, and elite individuals. This structure facilitates differentiated optimization across layers and is tailored to multi-objective tasks with clear rejection criteria. Additionally, we integrate explanatory factors into the fitness function, enhancing the explanatory power of the selected individuals.

10.2 Related Work

In the realm of time series analysis, model-agnostic explanation methods can be broadly classified into three types based on their foundational units of explanation: time-point-based, subsequence-based, and instance-based (or feature-based) explanations. Each type has distinct approaches and limitations: The first is the time-point-based explanations. SoundLime [116]: This method generates new samples by introducing minor perturbations to the original audio data and assesses the importance of each time point based on the model's predictions for these altered samples. Tsinsight [155]: Tsinsight employs an auto-encoder trained on the dataset to explain the input through reconstructed data. Salience Cam [185]: It generates a salience map based on the gradients of the model's output concerning the input data, using it to explain the decision. Although these approaches effectively determine the significance of individual time points, they fall short of exploring broader time-related features, such as frequency and trend, which require an analysis that integrates data across multiple time points.

The second type is subsequence-based explanations. TS-MULE [145]: This method assesses the importance of each cognitive block by constructing local linear models and generating cognitive blocks from sequences using methods like Symbolic Aggregate approXimation (SAX). SAX-VSM [148]: It segments time series data using SAX with overlapping windows and trains a bag-of-words model based on these segments to explain the input with generated 'words'. MCXAI [74] and SBXAI [73]: These methods provide insights into relationships between cognitive blocks, with MCXAI focusing on spatial relationships through a tree structure, and SBXAI on temporal relationships via a DCG. However, these methods do not sufficiently explore the temporal features of the data, and their explanations may merge multiple rules, complicating understanding.

The third type is the instance-based explanations. Instance-based methods extract features using statistical techniques, with explanations largely depending on the explainability of these features. This approach requires the model to rely exclusively on interpretable features for decision-making. However, the feature extraction process can lead to a loss of information, significantly limiting the model's performance.

In summary, while each of these model-agnostic explanation methods offers valuable insights in the context of time series analysis, they also have inherent limitations. Time-point-based methods may neglect broader temporal patterns, subsequence-based methods might not fully capture temporal dynamics, and instance-based methods could suffer from information loss due to feature extraction.

10.3 Method

10.3.1 Problem Definition and Individual Coding

Given a black-box model B and an input $o = [o_1, \cdots, o_l]$, where l is the length of the input signal, the objective of a local model-agnostic time series explanation method is to identify a set of masks $\mathscr{M}^o = [m_1, \cdots, m_i, \cdots]$ with $m_i \in \mathscr{M}$ and $\mathscr{M} = \langle 0, 1 \rangle^l$. These masks should highlight the most critical data points that influence the model prediction. The identified mask set $\mathscr{M}^o$ must satisfy the following conditions:

- **Prediction Consistency**:

$$\forall_{m \in \mathscr{M}^o} \; B(m(o)) = B(o), \tag{10.1}$$

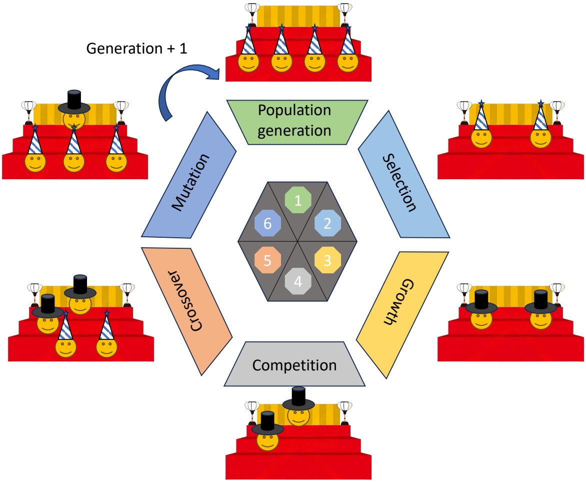


Figure 10.1.: The pipeline of the proposed ExTea algorithm.

where the function $m(o)$ involves assigning 0 to all data in the input o, except at positions marked with 1 by the mask m. The function $B(\cdot)$ yields the prediction of the black-box model.

- **Non-Subset Relation**:

$$\forall_{m_i,m_j\in\mathscr{M}^o,\ i\neq j}\ m_i \not\subseteq m_j,$$

where $m_i \subseteq m_j$ if and only if $m_i \wedge m_j = m_i$. This criterion ensures that no mask in $\mathscr{M}^o$ is completely encompassed by another, thus guaranteeing unique contributions from each individual mask.

- **Minimalism**:

$$\nexists_{m\in\mathscr{M}/\mathscr{M}^o}\left[B(m(o)) = B(o) \text{ and } \exists_{m_i\in\mathscr{M}^o}\ m \subseteq m_i\right]. \tag{10.2}$$

This condition ensures that the identified masks are the simplest possible.

By satisfying these conditions, the set of masks $\mathscr{M}^o$ provides a comprehensive, non-redundant, and minimal explanation of the critical data points influencing the black-box model's predictions.

As discussed in Sec. 10.2, time point-based explanation methods are inadequate to capture the complexity of time series data. To address this limitation, ExTea represents the mask as a list of $2n$ numerical values, where n is the number of contiguous blocks in the mask set to 1. This list must satisfy two criteria: *(i)* Each number must be a unique integer less than the length l of the input sequence, and *(ii)* the numbers must be in ascending order. Each adjacent pair $\{2i, 2i+1\}$ represents the i-th block in the mask.

Given that a numerical list and a mask can be interconverted, we denote the list as m. In ExTea, each individual is characterized by m, and $m(o)$ identifies the cognitive blocks that explain the model's decision for input o. This method compels the explanation to be composed of subsequences, thereby enhancing the optimization and exploration of individuals.

For multi-channel data, signals from different channels are concatenated into a single-dimensional signal. In this context, the length l refers to this concatenated signal, which simplifies the representation and analysis of multi-dimensional data.

10.3.2 Population Generation

The proposed algorithm employs a hierarchical structure with three distinct layers, L1, L2 and L3, to manage the population pool. Each layer plays a specific role in the selection and evolution of individuals, ensuring an efficient and organized progression of candidates through the system.

Layer L3 serves as the entry point for all newly created individuals, functioning as the initial staging ground for new candidates. Individuals in L3 are promoted to L2 after satisfying the explicit rejection condition outlined in Equation 10.1 (Prediction Consistency). This layer acts as a filter, advancing only those candidates that meet basic criteria. The transition from L2 to the elite layer L1 is based on competition, with L1 reserved for the most promising solutions, fostering the focused development of superior candidates.

New individuals are generated through a random sampling process. This begins with generating a random number to determine the number of blocks in the individual. Subsequently, a set of unique random integers, twice the number of blocks, is selected from the range $[0, l]$. These values are then organized in ascending order to form a numerical list representing the individual.

To maintain the diversity and dynamism of the population pool, the individuals of layer L3 are replenished according to its capacity s_3. This ensures a consistent influx of new candidates into the system.

10.3.3 Fitness Function Design

ExTea, adapted to the hierarchical structure of the individual pool, divides the selection process of the general evolutionary algorithm into two distinct processes: selection and competition. Each process targets different layers within the system and employs specific criteria for evaluating individuals.

Selection Process. This process focuses on the individuals in layer L3. The fitness function for selection is defined as follows:

$$f_{sel} = \mathbb{1}_{\{B(m(o))=B(o)\}} \times 2 - 1,$$

where $\mathbb{1}$ denotes the indicator function, B the black-box model, o the target of analysis, and m the mask of the individual. Only those individuals whose masked information (cognitive blocks) yields the same prediction as the original data are selected to Layer L2 for further optimization. Individuals failing this process are eliminated due to the ambiguity in evaluating their relative performance.

Competition Process. The competition process in ExTea is designed for individuals in L2 and L1. It aims to select superior individuals for promotion to L1 and demote those in L1 that fail to meet the competition standards. Evaluation in this process is based on two main criteria:

- Cognitive Block Length: The algorithm posits an inverse correlation between an individual's importance and the length of its cognitive blocks.

Shorter blocks imply higher significance and facilitate easier comprehension.

- Purity of Influencing Factors: The ideal individual should be influenced by as few factors as possible, enhancing the purity of its explanation. The impact of various elements such as time dependence, location, and frequency information on the model's decision-making process is investigated. A purer, less influenced explanation is considered superior for clarity and understanding.

In ExTea, several explorations based on basic time series characteristics are conducted, each evaluated separately to ensure a thorough understanding of the factors influencing model decisions:

- **Sequential Relationship:** This exploration assesses the impact of the sequential relationship between cognitive blocks on the model's decision-making by altering block positions and observing changes in model predictions. The scoring function f_1 is defined as the proportion of block pairs influencing the decision:

  $$f_1 = \frac{2}{n(n-1)} \sum_{i,j\in[0,\cdots,n],i\neq j} \mathbb{1}_{\left\{B(c_i^j(m(o)))\neq B(o)\right\}},$$

  where n is the number of cognitive blocks and $c_i^j(o)$ denotes the swapping operation of the i-th and j-th blocks.

- **Low-Frequency Information:** The importance of low-frequency information in cognitive blocks is evaluated by applying a Butterworth high-pass filter. The scoring function f_2 of this exploration is defined as whether the cognitive blocks after the filtering retain the original prediction:

  $$f_2 = \mathbb{1}_{\{B(b_h(m(o)))\neq B(o)\}},$$

  where $b_h(\cdot)$ represents the Butterworth high-pass filtering operation.

- **High-Frequency Information:** Similarly, a Butterworth low-pass filter is used to assess the role of high-frequency information, with the scoring function f_3 formulated as:

$$f_3 = \mathbb{1}_{\{B(b_l(m(o)))\neq B(o)\}},$$

where $b_l(\cdot)$ signifies the Butterworth low-pass filtering operation.

- **Numerical Trends:** By mirroring values within cognitive blocks, we evaluate the influence of numerical trends on model decisions. The scoring function f_4 of this exploration is defined as:

  $$f_4 = \frac{1}{n} \sum_{i\in[0,\cdots,n]} \mathbb{1}_{\{B(v_i(m(o)))\neq B(o)\}},$$

  where function $v_i(\cdot)$ indicates mirroring the data in the i-th cognitive block and n signifies the total number of cognitive blocks.

- **Blocks Relative Position:** We explore the effect of changing the positions of the blocks on the prediction of the model by shifting each block forward and backward separately. The scoring function f_5 of this exploration is defined as:

  $$f_5 = \frac{1}{2n} \sum_{j\in\{-d,d\} \text{ and } i\in[0,\cdots,n]} \mathbb{1}_{\left\{B(s_i^j(o,m))\neq B(o)\right\}},$$

  where n is the number of segments in the individual, d is a variable that signifies the distance to the neighboring blocks or the border of the time series, and the function $s_i^j(\cdot)$ means shifting the i-th block by j distance.

- **Block Position:** We explore the effect of synchronously changing all the positions of cognitive blocks both forward and backward until any block reaches the series boundary. If the prediction holds, we set the score f_6 to zero; otherwise, it is set to one.

- **Decision Intervals:** We examine the extent of numerical adjustment permissible at each point in the cognitive block without altering the model's prediction using the Reinforce method described in [72]. Due to the high time consumption, this exploration is only executed before the algorithm returns the final results.

To achieve the desired level of explanation purity, ExTea calculates the mean of the first six exploration scores. The smaller the mean, the fewer the factors affecting the explanation, resulting in a clearer and more concise explanation. Additionally, ExTea prioritizes explanations with smaller block sizes, aligning with the assumption that simpler explanations are often more effective. The competition score f_{comp} is formulated to reflect these priorities:

$$f_{comp} = -\left[\frac{\text{sum}(m)}{\text{len}(m)} + \lambda \times \frac{f_1 + f_2 + f_3 + f_4 + f_5 + f_6}{6}\right], \tag{10.3}$$

where λ denotes the balance weight, the function $\text{sum}(m)$ calculates the sum of the mask m, and the function $\text{len}(m)$ returns the length of the mask.

During each generation, individuals in Layers L1 and L2 are evaluated using Equation 10.3. The top-scoring individuals, up to the capacity s_1 of Layer L1, are then promoted to this layer. This strategy ensures that only the most refined and suitable candidates ascend to the elite layer, thereby maintaining a high standard of quality within the population pool.

After the competition process, it is crucial to perform a thorough validation of minimalism in the L1 layer, as outlined in equation 10.2, to eliminate duplicated individuals. This procedure ensures the diversity of individuals in the L1 layer, thereby guaranteeing a distinct representation at that level.

10.3.4 Growth

The growth stage, targeting individuals at layers L1 and L2, follows the Selection process and precedes the Competition process in each generational cycle. This stage addresses the challenge of superfluous information within randomly generated individuals' cognitive blocks. The primary objective is to refine these individuals, ensuring maximal succinctness by systematically eliminating non-essential information within each cognitive block.

It is important to note that identifying the minimal requisite set of explanations within the original dataset is an NP-hard problem. Even when optimization is restricted to cognitive blocks, the search space remains vast. For an individual with n blocks, each of length h, the total number of potential reductions can be approximately quantified as $2nh^2$. Due to the time-intensive nature of exhaustively exploring these possibilities, we introduced the growth function to streamline this process.

The growth function traverses the mask sequentially. When the boundary of a block is recognized, the growth kernel size u values on the boundary are set to 0 according to a given growth probability α, thus shrinking the corresponding block. The growth kernel size u signifies the unit used to narrow the blocks, while the growth rate α signifies the probability of this narrowing occurring. After the growth process, the reduced mask is validated using the black-box B. If the prediction remains consistent, the alteration is retained; otherwise, it is revoked. This process is repeated a predefined number of times in each generation.

10.3.5 Crossover and Mutation

ExTea implements a two-layer crossover mechanism, consisting of inner-layer and inter-layer crossovers. Initially, all individuals in the L1 layer undergo pairing among themselves for inner-layer crossover. Subsequently, these L1 individuals are paired with those in the L2 layer for the inter-layer crossover. Not every pair undergoes the crossover process; it occurs with a probability determined by the crossover ratio β. This crossover process employs the half-swap strategy, a common technique in evolutionary algorithms, which involves the exchange of half of the genetic material between two individuals, thereby inducing diversity within the population.

In addition to crossover, individuals in the L1 layer undergo a mutation process characterized by a mutation ratio γ. This process features a distinctive approach: the splitting of original cognitive blocks. This is crucial because the growth stage inherently only removes irrelevant information at the boundaries of each block. Consequently, when superfluous information is embedded centrally within a block, the growth process alone is insufficient for its extraction. To address this limitation, we introduce the concept of block splitting as a new form of mutation. This method enables the removal of superfluous information from any section of the block, thereby enhancing the algorithm's ability to optimize individuals effectively.

10.3.6 Explanation

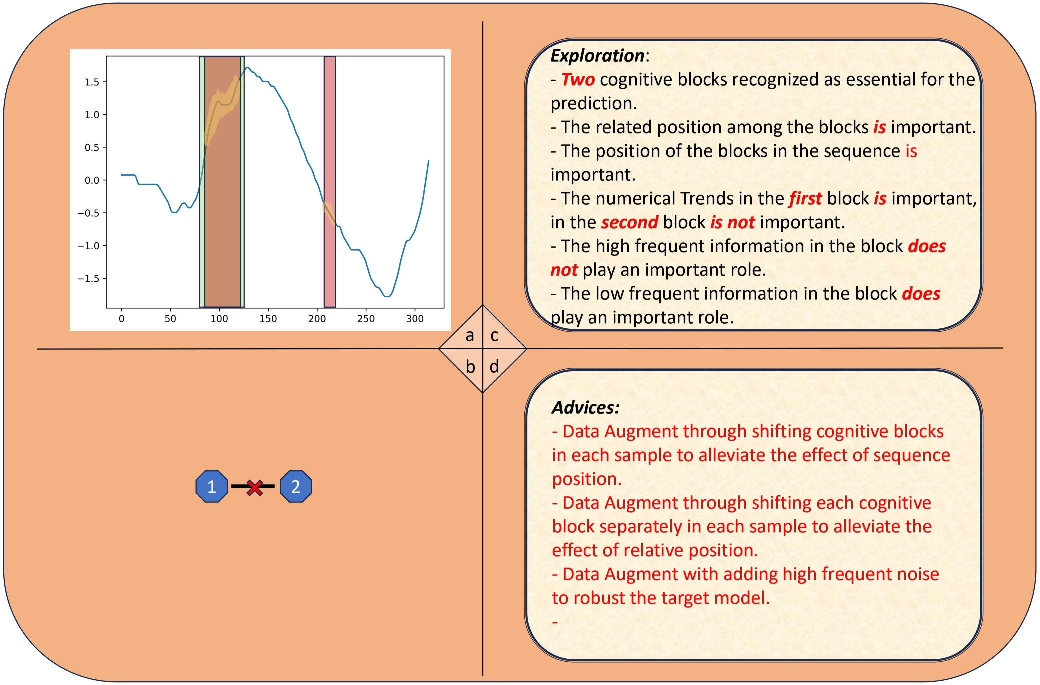
In this subsection, we provide a demonstrative example of the ExTea algorithm to address two primary questions: (i) What insights can be gleaned from the proposed method? and (ii) How can these insights be applied?

Figure 10.1 presents an exemplary explanation generated by the ExTea algorithm applied to the UWaveGestureLibraryX dataset [106]. The target model is a random forest with default parameters from the scikit-learn package [127]. The explanation comprises two images and two textual descriptions.

From Figure 10.1[a], the following insights are derived: (i) Cognitive Blocks Influencing Decision-Making: The areas highlighted in red denote cognitive blocks that significantly influence the model's decisions. (ii) Impact of the Relative Position of Cognitive Blocks: The cyan regions around the cognitive blocks represent their permissible movement range. Movement within these zones does not affect the model's decisions. Each cognitive block is independently evaluated, resulting in a unique cyan area for each element. (iii) Permissible Variability within Cognitive Blocks: Within each cognitive block, the blue line depicts the original data value, while the surrounding orange zone indicates the allowable fluctuation range.

Figure 10.1[b] examines the impact of rearranging cognitive blocks on the decision-making process. In this figure, each node denotes a cognitive block, with the number inside the node indicating its left-to-right position as shown in Figure 10.1[a]. An edge between two nodes signifies that swapping these cognitive blocks does not change the model's predictions. Conversely, a cross symbol above an edge indicates that reordering the blocks affects the decision outcome.

When visual representation of exploratory findings is impractical, we employ a rule-based methodology to generate descriptive text, as illustrated in Figure 10.1[c]. This approach combines static text (in black) with dynamic text (in red), where dynamic text varies according to the exploration results. This method also reinforces the findings initially presented visually, enhancing user comprehension. ExTea includes seven distinct exploratory analyses, each governed by a specific rule. Additionally, based on these exploratory results, we provide recommendations for improving model performance, as



Exploration:
- ***Two*** cognitive blocks recognized as essential for the prediction.
- The related position among the blocks ***is*** important.
- The position of the blocks in the sequence is important.
- The numerical Trends in the ***first*** block ***is*** important, in the ***second*** block ***is not*** important.
- The high frequent information in the block ***does not*** play an important role.
- The low frequent information in the block ***does*** play an important role.

Advices:
- Data Augment through shifting cognitive blocks in each sample to alleviate the effect of sequence position.
- Data Augment through shifting each cognitive block separately in each sample to alleviate the effect of relative position.
- Data Augment with adding high frequent noise to robust the target model.
-

Table 10.1.: An example of the ExTea explanation.

shown in Figure 10.1[d]. These suggestions are activated when the exploration results meet specific criteria. Detailed rules and algorithm code are available on `https://github.com/HuangYiran/extea`.

The original signal data shown in Figure 10.1[a] were captured by the accelerometer during a clockwise circle drawing. Based on the identified cognitive blocks, ExTea interprets this action as an increase in acceleration in the positive direction, followed by a decrease in the negative direction, closely aligned with human cognition. Notably, reversing the sequence order of the acceleration data changes the categorization from clockwise to counterclockwise circle drawing, highlighting the importance of sequential order in cognitive processing, as corroborated by the visualized results.

Contrary to our intuitive understanding, the model decision-making is influenced by variations in the acceleration trend. This discrepancy may arise from the uniformity of acceleration changes in the collected data. Ideally, frequent variations in acceleration during circle drawing should not impact the model's inference. To address this issue, Data Augmentation (DA) can be employed to generate samples with diverse acceleration patterns, thereby enhancing the model's robustness to such variations.

Furthermore, the analysis indicates that minor positional shifts between cognitive blocks can unexpectedly influence model predictions, contradicting conventional knowledge. This challenge can be mitigated through the strategic use of Data Augmentation (DA), which can further refine the model's accuracy and robustness.

10.4 Evaluation

To thoroughly evaluate our proposed methodology, we conducted two comprehensive experiments addressing two key questions: *(i)* Fidelity of the Explanation to the Original Model: This question assesses how accurately the explanations generated by our approach reflect the intrinsic mechanisms of the original model. *(ii)* Role of Algorithm Components: This question examines the specific contributions of various components within the proposed algorithm to its overall effectiveness. *(iii)* Impact of Algorithm Parameters on Performance. These experiments are designed to provide a robust analysis of both

Table 10.2.: Summary of the datasets used in the experiments.

Dataset	Type	Train size	Test size	Sequence length	Number classes
EthanolLevel	Spectro	504	500	1751	4
ECG5000	ECG	500	4500	140	5
ElectricDevices	Device	8926	7711	96	7
InsectWingBeatSound	Audio	25000	25000	256	10
EOGVerticalSignal	EOG	362	362	1250	12
UWaveGestureLibraryX	HAR	896	3582	315	8

the fidelity and functionality of our methodology.

10.4.1 Benchmark Dataset

In our experiments, we meticulously selected six datasets from diverse domains to demonstrate the adaptability and broad applicability of our proposed method:

- **Ethanol Level Dataset** [103]: Sourced from the Scotch Whisky Research Institute, this dataset is pivotal for the non-invasive detection of counterfeit spirits. It consists of spectrometer data from 1,000 spirit bottles, categorized into four alcohol levels, totaling 1,751 observations per bottle.

- **ECG5000 Dataset** [25]: Part of the BIDMC Congestive Heart Failure Database, this dataset features electrocardiography (ECG) recordings from a 48-year-old patient with severe congestive heart failure. It classifies the data into five categories, including normal rhythms and various forms of premature ventricular and ectopic beats.

- **ElectricDevices Dataset**: This dataset captures the electricity consumption patterns of 251 households. Data were recorded bi-minutely over a one-month period, with each sequence representing a full day's electricity usage.

- **InsectWingBeatSound Dataset** [24]: This dataset includes sound recordings from 5,000 individual insects, identified by species through acoustic data collected via specialized sensors.

- **EOGVerticalSignal Dataset** [42]: Utilizing the BlueGain biomedical amplifier, this dataset records electrooculography (EOG) signals, which measure the potential difference between electrodes near the eye. It includes signals from 12 participants, each representing 12 different Japanese katakana characters through eye movements.

- **UWaveGestureLibraryX Dataset** [106]: Designed for HAR tasks, this dataset compiles eight distinct gesture patterns from eight users over a month.

A comprehensive summary of these datasets is provided in Table 10.2.

10.4.2 Target Models

To rigorously evaluate our proposed methodology, we conducted experiments using four distinct models, incorporating both transparent ('white-box') and opaque ('black-box') approaches. This diverse selection was essential to assess the robustness and adaptability of our method across various computational frameworks. The models employed include: *(i)* Interpretable 'white-box' models, such as Decision Tree (DT) and Support Vector Machine (SVM), and *(ii)* 'Black-box' models, namely Random Forest (RF) and Neural Network (NN).

This combination of models allowed for a comprehensive evaluation of our approach in different contexts, ensuring a thorough analysis of its effectiveness.

10.4.3 Benchmark Algorithms

To assess the effectiveness of our proposed method, we performed a comparative analysis with three other XAI techniques: two state-of-the-art methods, MCXAI [74] and SBXAI [73], as well as the widely-used method LIME [134].

10.4.4 Experiments Design

Three distinct experiments were conducted to evaluate our proposed method. The first experiment aims to establish the fidelity of our method to the target model by measuring the precision and efficiency of various XAI methods in identifying critical features used by the target model for predictions. This experiment comprises three stages: *(i)* Model Training: The target models are trained on the training set for each dataset. *(ii)* Sampling and Interpretation: 100 samples are randomly selected from the test set, and each sample is explained using the XAI method to determine the importance of each data point. *(iii)* Reconstruction and Validation: A blank sample is incrementally populated with data points from the original sample, prioritized by their determined importance. The goal is to replicate the prediction of the target model with minimal data points, with the efficacy of an XAI method judged by the fewest data points required for accurate decision replication, indicating higher interpretative fidelity.

The second experiment, an ablation study, investigates the impact and effectiveness of the growth function in our proposed method. This study compares the performance of the method with and without the growth function, monitoring the proficiency of the most effective individual identified in each generation.

The final experiment examines the effect of the algorithm's parameters by comparing its performance under different parameter settings. This experiment also provides guidelines for the usage of the algorithm.

10.4.5 Experiment Setup

In the first two experiments and the base group in the third experiment, the number of blocks n is set to 5, and the growth kernel size u is set to 2. The growth rate is set to 0.5. In each generation, the growth process is repeated five times. All ratios α, β, γ are set to 0.1. The balance parameter λ is set to 0.1. The layer sizes are configured as follows: s_1 is set to 3, s_2 is set to 10, and s_3 is set to 50. The maximum number of generations is set to 20.

10.4.6 Evaluation

The results of the first experiment are summarized in Table 10.3, where the values represent the ratio of the number of data points required for accurate model prediction to the total sequence length. A lower ratio indicates a more

Table 10.3.: Ratio of information needed to support the model decision, the smaller the better. The bold numbers denote the smallest ratio in the corresponding groups.

Dataset	**XAI Method**	**LIME**	**MCXAI**	**SBXAI**	**Proposed**
EthanoLevel	DT	0.26	**0.02**	0.27	**0.02**
	SVC	0.49	**0.02**	0.49	**0.02**
	RF	0.21	0.04	0.40	**0.03**
	NN	0.44	**0.04**	0.22	**0.04**
ECG5000	DT	0.23	0.06	0.23	**0.01**
	SVC	0.25	0.03	0.21	**0.01**
	RF	0.26	0.05	0.27	**0.03**
	NN	0.41	0.06	0.39	**0.05**
ElectricDevices	DT	0.54	0.11	0.24	**0.10**
	SVC	0.39	0.10	0.39	**0.08**
	RF	0.50	0.27	0.29	**0.14**
	NN	0.51	**0.13**	0.45	**0.13**
InsectWingBeatSound	DT	0.50	0.06	0.51	**0.02**
	SVC	0.40	0.05	0.44	**0.02**
	RF	0.49	0.23	0.58	**0.03**
	NN	0.38	0.13	0.34	**0.02**
EOGVerticalSignal	DT	0.54	0.05	0.44	**0.02**
	SVC	0.39	0.05	0.65	**0.01**
	RF	0.50	0.06	0.45	**0.05**
	NN	0.51	**0.04**	0.57	0.06
UWaveGestureLibraryX	DT	0.43	0.06	0.46	**0.05**
	SVC	0.51	**0.15**	0.54	0.18
	RF	0.49	0.19	0.54	**0.12**
	NN	0.55	0.13	0.45	**0.10**

efficient identification of crucial data points. Our findings demonstrate that the proposed method significantly outperforms other methods across various datasets and models. This superiority highlights the efficacy of our method and suggests that it provides interpretations closely aligned with the model's inner workings, effectively serving as a localized surrogate for the original model's explanations.

The results of the first experiment are predictable, given the method's more flexible approach to identifying cognitive blocks. MCXAI is restricted by its use of a splitting method to optimize cognitive blocks, whereas SBXAI's optimization is indirect, relying on simulations of the original model. LIME has the potential for optimal results due to its focus on data points. However, its relatively simplistic local modeling approach may struggle with complex time series data, which likely accounts for its underperformance.

Figure 10.4 illustrates the impact of the growth process on the experimental results across various datasets, using the RandomForest model as the target black-box model. Inclusion of the growth process distinctly enhances both convergence speed and quality. The ExTea algorithm, which incorporates the growth process, converges more rapidly and effectively than its counterpart without it. Specifically, ExTea achieves convergence by the third generation in three datasets, the seventh generation in two datasets, and the eleventh generation in one dataset, highlighting the dataset-specific generational requirements, with most converging within ten generations. Notably, the performance of ExTea in the first generation is comparable to that of SBXAI and LIME.

Figure 10.2 illustrates the impact of various parameter settings on the performance of the algorithm. The numerical values adjacent to the straight lines represent the range of parameter values explored during the experiments, while the violin plots depict the frequency of each parameter value achieving the best performance. Our findings indicate that increasing the number of generations and L1 size generally leads to better outcomes. Specifically, a larger number of generations provides more opportunities for optimization, whereas a larger L1 size maintains a greater number of high-quality individuals within each generation. In contrast, smaller kernel sizes are advantageous, as they offer higher optimization granularity. However, increasing the number of generations and

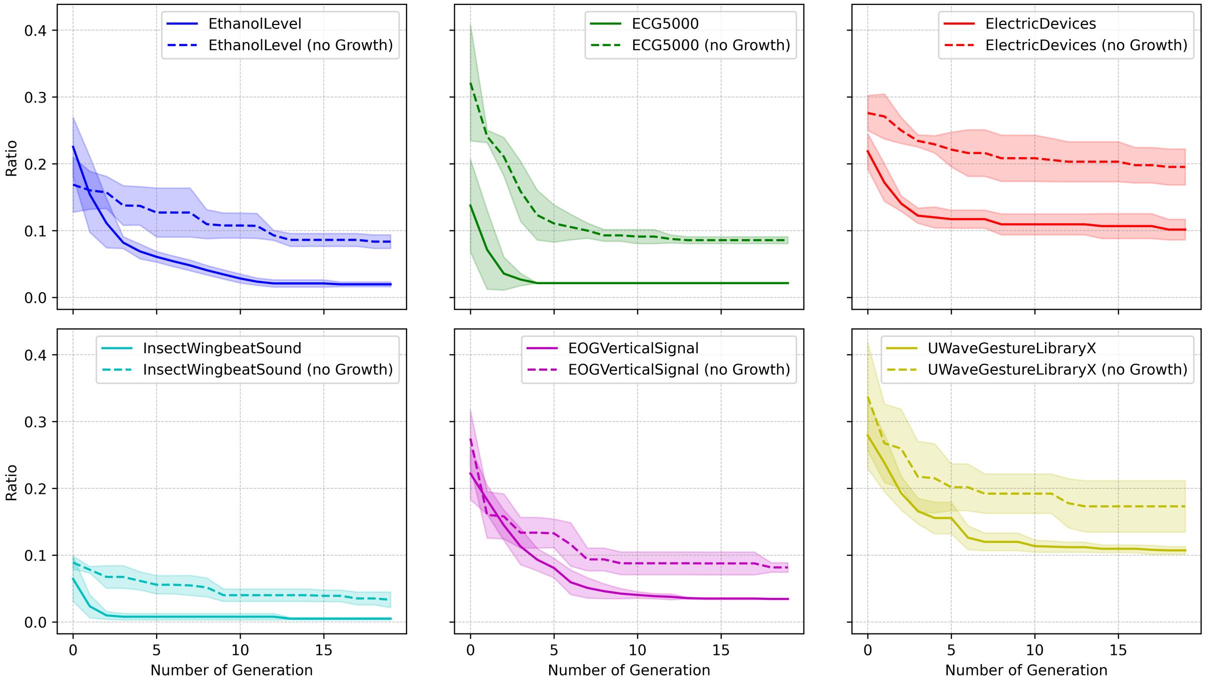


Table 10.4.: Comparison of ExTea performance with and without growth processes. The x-axis represents the number of generation (epoch) and the y-axis indicates the ratio between the final cognitive block length and the input sequence length.

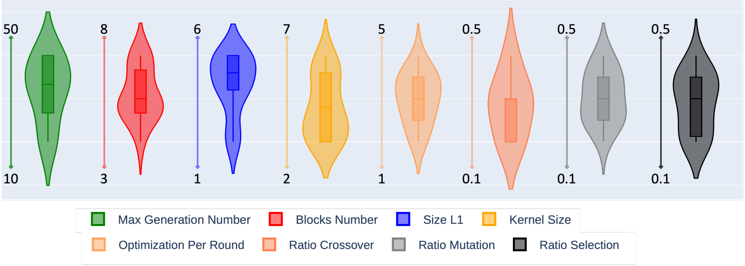


Figure 10.2.: Comparison of ExTea performance with different parameter setting. The line next to the violin describes the value range of the corresponding parameter. The violin describes the kernel density estimate of achieving the best result.

the L1 size, along with reducing the kernel size, results in a longer runtime for the algorithm. Therefore, the parameter settings should be adjusted according to the specific application context. Additionally, a higher optimization per round is not always beneficial as it may lead to the elimination of individuals that perform well initially but deteriorate in later stages, necessitating further investigation. The optimal ratio largely depends on the specific dataset, but the best performance is generally achieved within the range of 0.2 to 0.3.

The execution time of the ExTea algorithm is influenced by several factors, including algorithmic parameters, data sample shape, the complexity of the target black-box model, and hardware specifications. For instance, using the ECG5000 dataset and the RandomForest model, the average duration for one generation, without parallelization, is approximately five seconds. In contrast, MCXAI's execution time is significantly affected by its exploration of structural relationships among cognitive blocks after identification. When this step is omitted, MCXAI also averages about five seconds. Meanwhile, SBXAI requires about 0.5 seconds to evaluate a single candidate subgroup and approximately 26 seconds for 50 evaluations. This longer duration is due to the intensive parameter settings of the Bayesian optimization's objective function, particularly its default high number of cross-validations.

10.5 Discussion

This paper introduces ExTea, a novel model-agnostic algorithm designed to elucidate black-box model of time series. ExTea employs an evolutionary algorithm, treating explanations as evolving entities within an optimization process geared towards growth. The core of this method is a customized fitness function specifically designed for the intricacies of time series data, guiding the algorithm's navigation through the search space. A unique aspect of ExTea is its hierarchical pool of individuals, which stratifies them across different layers to enhance model exploration efficiency.

ExTea addresses several challenges in the domain of time series model explanation. These challenges include decomposing complex hybrid interpretations, adapting to diverse time series characteristics, navigating temporal domain features, and managing intricate parameter configurations. By tackling these issues, ExTea significantly advances the field of time series model explanation.

Empirical experiments conducted on six datasets from various domains demonstrate that ExTea offers a more effective and efficient framework for understanding the predictions of time series black-box models. This is particularly valuable in domains where explainability and transparency are crucial.

Despite its achievements, ExTea has potential for further development. Currently, its feature exploration processes operate in isolation; integrating multiple feature explorations concurrently could enhance its efficacy. Additionally, although ExTea is based on an evolutionary algorithm, its parallelism capabilities remain untapped. Optimizing parallelism alongside ExTea's parameters could provide real-time feedback. Finally, a key challenge in augmenting time series data is preserving critical decision-making information amidst random data augmentation. ExTea theoretically offers a solution, potentially allowing for targeted augmentation that preserves data integrity, an aspect that warrants further investigation for practical application.

11. Final Discussion and Future Work

In this dissertation, through research targeting three sub-goals (**Q1**, **Q2**, **Q3**), we have addressed several key challenges for advancing XAI identified in the pervasive computing domain. The methods developed and discussed across various chapters directly respond to the complexity (**C1**), deployment constraints (**C2**), data characteristics (**C3**), insufficiency of generic explanations (**C4**), and reliability issues of surrogate-based post hoc XAI methods (**C5**) as outlined in the introduction.

11.1 Discussion on Improving Interpretable Model Performance

Chatper 3, Chatper 4 and Chapter 5 target **Q1**: **How can the predictive performance of interpretable models within the pervasive computing domain be enhanced**? and make the following contributions: **Enhancement of interpretable model predictive performance through a feature engineering, constrained AutoML methodology and novel interpretable model design**.

In response to **C1**, Chapter 3 conducted on enhancing interpretable models within the pervasive computing domain. The novel automatic feature engineering framework, mCafe, demonstrates superior predictive performance compared to existing state-of-the-art methods, particularly in handling feature fusion, which is often overlooked in traditional approaches. This advancement underscores the importance of robust feature engineering in boosting the efficacy of interpretable models, particularly in complex domains like pervasive computing.

However, while mCafe represents a substantial leap forward, it is not without limitations. One potential drawback is its reliance on predefined exploratory techniques, which may not fully capture the diversity of features necessary for all pervasive computing applications. Additionally, mCafe's predictive performance may vary depending on the specific nature of the data and the problem context, potentially requiring further customization or refinement for optimal results. The dependency of the framework on the quality of the existing data also raises concerns; If the input data are noisy or incomplete, the effectiveness of the engineered features may be compromised.

To address **C1** and mitigate the effect of **C2**, Chapter 4 develops an AutoML methodology tailored for uneven search spaces and intricate parameter interdependencies. This approach not only improves the overall predictive performance and ensures that the models remain interpretable, but also limits the model to the deployment constraints, a crucial requirement in this field. Despite these strengths, this methodology may encounter challenges in highly dynamic or nonstationary environments where the trade-offs between optimization complexity and computational efficiency must be carefully managed to avoid excessive resource consumption, especially in resource-constrained pervasive computing settings.

In Chapter 5, the introduction of the LLMs proxy scheme further addresses the challenge of explaining model decisions in scenarios where expert knowledge is required for pattern explanation. By effectively combining traditional feature extraction methods with LLMs, this scheme not only achieves accurate classification but also provides explainable reasons for these decisions, bridging the gap between complex model outputs and human-understandable explanations. Nonetheless, the effectiveness of this scheme hinges on the quality and relevance of the features extracted before feeding them into the LLMs. There is also the risk that the LLMs, despite their advanced capabilities, may introduce biases or generate explanations that are too general or not sufficiently domain specific. Additionally, the dependence of the scheme on LLMs raises questions about scalability and computational cost, particularly in scenarios requiring real-time processing.

In summary, contributions in this area not only enhance the predictive performance of interpretable models but also broaden the scope of their applicability in pervasive computing, making them more reliable and effective in real-world scenarios. However, future research should focus on addressing the identified limitations, such as enhancing the adaptability of feature engineering methods, improving AutoML efficiency in dynamic environments, and refining the integration of LLMs to reduce biases and ensure domain-specific explainability.

11.2 Discussion on Extracting Model Decision Elements

Chatper 6 and Chapter 7 target the **Q2**: **Enhancement of interpretable model predictive performance through feature engineering, constrained AutoML methodology, and novel interpretable model design**? and make the following contribution: **Summarization of training processes and model architectures, identifying the key data elements utilized by high-quality models in the pervasive computing domain**.

To dig into the cause of **C3** and uncover the basis of high-quality models, Chapter 6 undergoes a systematic review of existing high-quality models and their decision-making processes within the pervasive computing domain. This has revealed essential insights into the elements that drive model success. By exploring their data processing process and analyzing the patterns emphasized by different architectures, Chapter 6 has identified the critical data elements that high-performing models leverage for decision-making.

These findings have profound implications for the advancement of post hoc XAI methods. Rather than merely refining model architectures, the emphasis of our ongoing study is on developing innovative XAI techniques that can better elucidate the decision-making processes of models. By understanding which data elements are the most influential, we can design explanation methods that provide clearer insights into model behavior, thereby enhancing transparency and trustworthiness.

However, despite these valuable insights, there are limitations to the current work. One significant limitation is the reliance on retrospective analysis of existing models, which may not fully capture the dynamic and evolving nature of data in pervasive computing environments.

Another limitation is the generalizability of the findings across different contexts within pervasive computing. The models reviewed may perform well under specific conditions but may not necessarily translate to other applications or datasets within the domain. In other words, post hoc XAI methods developed according to a single element mentioned may also face application adaptation problems.

In summary, while this research has significantly contributed to understanding the critical elements of model decision-making in pervasive computing, it also highlights areas where further work is needed.

11.3 Discussion on Post-hoc Explanation Methods

Chatper 8, Chapter 9 and Chapter 10 target the **Q3**: **How can the data elements uncovered be utilized in post hoc XAI method design?** and make the following contribution: **Integration of local data pattern, temporal dependence, noise, and other data elements discovered in Part II into explanations using novel proposed post hoc techniques**. Each chapter devises a new post hoc XAI approach based on the three challenges (**C3**, **C4**, **C5**) described above.

The exploration of post hoc explainability techniques in Part III has led to several innovative approaches that address the unique challenges posed by pervasive computing. The tree-based exploration method in Chapter 8 (MCXAI), which models the explanation process as a tree search problem, provides a novel way to interpret model decisions by highlighting spatial relationships and assigning significance to cognitive segments. This method offers a structured and comprehensive approach to understanding how models arrive at their decisions. However, the complexity of the tree search approach might lead to computational challenges, particularly with high-dimensional data, which could limit its scalability and applicability in real-time systems. Moreover, while it effectively highlights relationships between different data segments, the method may oversimplify the interactions between these segments, potentially missing more intricate patterns.

In Chapter 9 (SBXAI), the use of Markov chains to elucidate temporal relationships introduces a dynamic aspect to model explanation, allowing a deeper understanding of how sequential data influence decision-making. This approach not only improves explainability, but also aligns the explanation process more closely with the temporal nature of data in pervasive computing. Nevertheless, Markov models assume that future states depend only on the current state, which might not fully capture the complexities of long-term dependencies in some datasets. Alternative approaches, such as incorporating recurrent neural networks or attention mechanisms, could offer more nuanced explanations of temporal relationships.

In Chapter 10 (ExTea), the integration of a genetic algorithm-based technique to unify the exploration of decision-making elements across multiple dimensions, including frequency, represents a significant advancement in post-hoc explainability. This technique simplifies the explanation process while ensuring that the resulting explanations are both comprehensive and understandable. However, genetic algorithms can be computationally expensive and may require careful tuning of parameters to avoid issues like premature convergence. Furthermore, the explainability of the results might be affected by the stochastic nature of the genetic process, potentially leading to inconsistencies in the explanations in different runs.

In summary, while the proposed approaches offer significant advancements in the field of explainable AI, they also present challenges and limitations that need to be addressed. Future work should explore alternative methods and further refine these techniques to enhance their scalability, accuracy, and applicability across different domains. Additionally, a comparative analysis with existing post hoc explainability methods would provide a clearer understanding of the strengths and weaknesses of the proposed solutions, guiding future improvements.

11.4 Conclusion

This dissertation presents a comprehensive framework for enhancing the predictive performance and explainability of AI models in the pervasive computing domain. By dividing the overarching goal into three distinct sub-goals, the research systematically addresses the critical challenges associated with interpretable models, decision element elucidation, and post hoc explainability techniques.

Part I enhances the predictive capabilities of interpretable models through innovative approaches to feature engineering and model optimization. These advances ensure that interpretable models remain both effective and understandable, even when faced with complex data. Part II delves into the decision-making processes of high-quality models, uncovering the key data elements that drive success. This knowledge will aid in not only the development of future models but also in the development of targeted interpretive methods to make their modeling decisions more robust and transparent. Part III develops new methodologies for post hoc explainability, introducing techniques that provide a structured, dynamic, and comprehensive explanation of model decisions. These contributions are particularly valuable in the context of pervasive computing, where the complexity of data and the need for timely explanations are paramount.

11.5 Future Work

While this dissertation presents significant advancements in enhancing the predictive performance and explainability of AI systems within the pervasive computing domain, there remain several avenues for future research that could address the limitations identified and expand upon the findings.

- Refinement of LLMs Integration for Domain-Specific explainability: The integration of LLMs in explaining model decisions presents both opportunities and challenges. Future work should aim to refine these integrations by developing domain-specific LLMs that reduce the risk of generating overly general or biased explanations. This could involve fine-tuning LLMs on domain-specific datasets or exploring alternative architectures that are better suited to the nuances of pervasive computing data. Additionally, the scalability and computational cost of LLMs-based explanations need further investigation, particularly in real-time processing scenarios.

- Exploration of Hybrid Explainability Models: As this research has primarily focused on post-hoc explainability techniques, future work could explore the development of hybrid models that combine the strengths of interpretable models with more complex, high-performing black-box models. Such hybrid approaches could leverage the predictive power of black-box models while maintaining the explainability required for deployment in critical applications. Research in this area could also investigate the trade-offs between model complexity and explainability, aiming to find a balance that maximizes both predictive performance and transparency.

- Real-Time Monitoring and Feature Interaction Analysis: To address the limitations related to static analysis of model decision-making, future research should develop tools and methodologies for real-time monitoring of feature relevance and interaction. This could involve the use of streaming data analytics to continuously assess and adjust the importance of features as the model operates, ensuring that the decision-making process remains accurate and context-aware. Advanced interaction analysis techniques, such as the use of Shapley values or deep learning-based interaction detectors, could provide deeper insights into how features combine to influence model outcomes.

- Scalability and Computational Efficiency in Post-Hoc Explainability: The novel post-hoc explainability methods proposed in this dissertation, while innovative, may face challenges in scaling to larger datasets or more complex environments. Future work should focus on optimizing these methods for scalability, potentially by developing more efficient algorithms or by leveraging parallel processing and distributed computing techniques. Additionally, comparative studies with existing methods could help identify areas where the proposed techniques excel or where further refinement is needed.

- Integration of Explainability into AI Lifecycle: Finally, future research could explore the integration of explainability into the entire AI lifecycle, from model development to deployment. This holistic approach would ensure that explainability is not an afterthought but a core component of AI system design. Such an approach could involve developing new methodologies for embedding explainability into the model training process, as well as creating standardized metrics and evaluation frameworks to assess the quality of explanations throughout the AI lifecycle.

In summary, while this dissertation has made substantial contributions to improving the predictive performance and explainability of AI models in pervasive computing, the field remains rich with opportunities for further exploration. By addressing the limitations identified and pursuing these future research directions, it will be possible to develop even more robust, adaptive, and transparent AI systems capable of meeting the demands of increasingly complex and dynamic environments.

Bibliography

[1] Ubiquitous computing: Innovations, challenges, and future trends. Blog on EMb Global, 2023. Available online: [access link to the full article].

[2] A. Abedin, M. Ehsanpour, Q. Shi, H. Rezatofighi, and D. C. Ranasinghe. Attend and Discriminate: Beyond the State-of-the-art for Human Activity Recognition Using Wearable Sensors. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 5(1):1–22, 2021.

[3] A. Adadi and M. Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). *IEEE access*, 6:52138–52160, 2018.

[4] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[5] B. AI. Explainability i: Local post-hoc explanations. `https://www.borealisai.com/research-blogs/explainability-i-local-post-hoc-explanations/`, 2023. Accessed: 2024-08-07.

[6] W. Ali and F. Saeed. Hybrid filter and genetic algorithm-based feature selection for improving cancer classification in high-dimensional microarray data. *Processes*, 11(2):562, 2023.

[7] J. An, J. Lee, and G. Gweon. Does chatgpt comprehend the place value in numbers when solving math word problems. In *Proceedings of the Workshop" Towards the Future of AI-augmented Human Tutoring in*

Math Learning" co-located with The 24th International Conference on Artificial Intelligence in Education (AIED 2023), Tokyo, Japan, volume 3491, pages 49–58, 2023.

[8] J. Antorán, U. Bhatt, T. Adel, A. Weller, and J. M. Hernández-Lobato. Getting a clue: A method for explaining uncertainty estimates. *arXiv preprint arXiv:2006.06848*, 2020.

[9] D. W. Apley and J. Zhu. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086, 2020.

[10] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58: 82–115, 2020.

[11] R. Assaf and A. Schumann. Explainable deep neural networks for multivariate time series predictions. In *IJCAI*, pages 6488–6490. Macao, 2019.

[12] P. Auer. Finite-time analysis of the multiarmed bandit problem, 2002.

[13] U. Azmat, Y. Y. Ghadi, T. a. Shloul, S. A. Alsuhibany, A. Jalal, and J. Park. Smartphone sensor-based human locomotion surveillance system using multilayer perceptron. *Applied Sciences*, 12(5):2550, 2022.

[14] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[15] M. Barandas, D. Folgado, L. Fernandes, S. Santos, M. Abreu, P. Bota, H. Liu, T. Schultz, and H. Gamboa. Tsfel: Time series feature extraction library. *SoftwareX*, 11:100456, 2020.

[16] A. Barbalau, A. Cosma, R. T. Ionescu, and M. Popescu. A generic and model-agnostic exemplar synthetization framework for explainable ai. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 190–205. Springer, 2020.

[17] J. Bergstra, D. Yamins, D. D. Cox, et al. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. 2013.

[18] U. Bhatt, A. Xiang, S. Sharma, A. Weller, A. Taly, Y. Jia, J. Ghosh, R. Puri, J. M. Moura, and P. Eckersley. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657, 2020.

[19] D. Bhattacharya, D. Sharma, W. Kim, M. F. Ijaz, and P. K. Singh. Ensem-HAR: An Ensemble Deep Learning Model for Smartphone Sensor-Based Human Activity Recognition for Measurement of Elderly Health Monitoring. *Biosensors*, 12(6):393, 2022.

[20] L. Bibbò, R. Carotenuto, and F. Della Corte. An overview of indoor localization system for human activity recognition (har) in healthcare. *Sensors*, 22(21):8119, 2022.

[21] A. E. Blanchard, M. C. Shekar, S. Gao, J. Gounley, I. Lyngaas, J. Glaser, and D. Bhowmik. Automating genetic algorithm mutations for molecules using a masked language model. *IEEE Transactions on Evolutionary Computation*, 26(4):793–799, 2022.

[22] P. Bouchard, S. Voß, L. Heilig, and X. Shi. A case study on smart grid technologies with renewable energy for central parts of hamburg. *Sustainability*, 15(22):15834, 2023.

[23] G. Chaslot, S. Bakkes, I. Szita, and P. Spronck. Monte-carlo tree search: A new framework for game ai. *AIIDE*, 8:216–217, 2008.

[24] Y. Chen, A. Why, G. Batista, A. Mafra-Neto, and E. Keogh. Flying insect classification with inexpensive sensors. *Journal of insect behavior*, 27:657–677, 2014.

[25] Y. Chen, Y. Hao, T. Rakthanmanon, J. Zakaria, B. Hu, and E. Keogh. A general framework for never-ending learning from time series streams. *Data mining and knowledge discovery*, 29:1622–1664, 2015.

[26] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. arxiv 2014. *arXiv preprint arXiv:1406.1078*, 2020.

[27] S. Cho, W. Chang, G. Lee, and J. Choi. Interpreting internal activation patterns in deep temporal neural networks by finding prototypes. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 158–166, 2021.

[28] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package). *Neurocomputing*, 307:72–77, 2018.

[29] P.-A. Coquelin and R. Munos. Bandit algorithms for tree search. *arXiv preprint cs/0703062*, 2007.

[30] R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[31] J. Crabbé and M. Van Der Schaar. Explaining time series predictions with dynamic masks. In *International Conference on Machine Learning*, pages 2166–2177. PMLR, 2021.

[32] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019.

[33] E. Delaney, D. Greene, and M. T. Keane. Instance-based counterfactual explanations for time series classification. In *International Conference on Case-Based Reasoning*, pages 32–47. Springer, 2021.

[34] G. Demiris and B. K. Hensel. Technologies for an aging society: a systematic review of "smart home" applications. *Yearbook of medical informatics*, 17(01):33–40, 2008.

[35] I. Dirgová Luptáková, M. Kubovčík, and J. Pospíchal. Wearable sensor-based human activity recognition with transformer model. *Sensors*, 22 (5):1911, 2022.

[36] R. Doddaiah, P. Parvatharaju, E. Rundensteiner, and T. Hartvigsen. Explaining deep multi-class time series classifiers. 2023.

[37] O. Dor and Y. Reich. Strengthening learning algorithms by feature discovery. *Information Sciences*, 189:176–190, 2012.

[38] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[39] A. E. Eiben, J. E. Smith, A. Eiben, and J. Smith. What is an evolutionary algorithm? *Introduction to evolutionary computing*, pages 25–48, 2015.

[40] European Parliament. Artificial intelligence act, 2024. URL `https://en.wikipedia.org/wiki/Artificial_Intelligence_Act`. Adopted by the European Parliament in March 2024.

[41] W. Fan, E. Zhong, J. Peng, O. Verscheure, K. Zhang, J. Ren, R. Yan, and Q. Yang. Generalized and heuristic-free feature construction for improved accuracy. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 629–640. SIAM, 2010.

[42] F. Fang and T. Shinozaki. Electrooculography-based continuous eye-writing recognition system for efficient assistive communication systems. *PloS one*, 13(2):e0192684, 2018.

[43] J. Flinn, D. Narayanan, and M. Satyanarayanan. Self-tuned remote execution for pervasive computing. In *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, pages 61–66. IEEE, 2001.

[44] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[45] R. Fujimoto, Y. Nakamura, and Y. Arakawa. Differential privacy with weighted ε for privacy-preservation in human activity recognition. In *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 634–639. IEEE, 2023.

[46] R. Gaudel and M. Sebag. Feature selection as a one-player game. In *International Conference on Machine Learning*, pages 359–366, 2010.

[47] D. Gay, R. Guigourès, M. Boullé, and F. Clérot. Feature extraction over multiple representations for time series classification. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 18–34. Springer, 2013.

[48] A. H. Gee, D. Garcia-Olano, J. Ghosh, and D. Paydarfar. Explaining deep classification of time-series data with learned prototypes. In *CEUR workshop proceedings*, volume 2429, page 15. NIH Public Access, 2019.

[49] D. Gholamiangonabadi and K. Grolinger. Personalized models for human activity recognition with wearable sensors: deep neural networks and signal processing. *Applied Intelligence*, 53(5):6041–6061, 2023.

[50] M. Gholamrezaii and S. M. T. AlModarresi. A time-efficient convolutional neural network model in human activity recognition. *Multimedia Tools and Applications*, 80:19361–19376, 2021.

[51] P. Gijsbers, E. LeDell, J. Thomas, S. Poirier, B. Bischl, and J. Vanschoren. An open source automl benchmark. *arXiv preprint arXiv:1907.00909*, 2019.

[52] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

[53] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[54] B. Goodman and S. Flaxman. European union regulations on algorithmic decision-making and a "right to explanation". *AI magazine*, 38(3): 50–57, 2017.

[55] Y. Goyal, Z. Wu, J. Ernst, D. Batra, D. Parikh, and S. Lee. Counterfactual visual explanations. In *International Conference on Machine Learning*, pages 2376–2384. PMLR, 2019.

[56] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[57] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[58] R. Guidotti, A. Monreale, F. Spinnato, D. Pedreschi, and F. Giannotti. Explaining any time series classifier. In *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*, pages 167–176. IEEE, 2020.

[59] M. Guillemé, V. Masson, L. Rozé, and A. Termier. Agnostic local explanation for time series classification. In *2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI)*, pages 432–439. IEEE, 2019.

[60] J. Guna, I. Humar, and M. Pogačnik. Intuitive gesture based user identification system. In *2012 35th International Conference on Telecommunications and Signal Processing (TSP)*, pages 629–633. IEEE, 2012.

[61] P. Guo, C. Deng, L. Xu, X. Huang, and Y. Zhang. Deep multitask augmented feature learning via hierarchical graph neural network. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao,*

Spain, September 13–17, 2021, Proceedings, Part I 21, pages 538–553. Springer, 2021.

[62] A. Hannan, M. Z. Shafiq, F. Hussain, and I. M. Pires. A portable smart fitness suite for real-time exercise monitoring and posture correction. *Sensors*, 21(19):6692, 2021.

[63] H. Haresamudram, I. Essa, and T. Plötz. Investigating enhancements to contrastive predictive coding for human activity recognition. In *2023 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 232–241. IEEE, 2023.

[64] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[65] N. Hnoohom, S. Mekruksavanich, and A. Jitpattanakul. An efficient resnetse architecture for smoking activity recognition from smartwatch. *Intelligent Automation & Soft Computing*, 35(1), 2023.

[66] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

[67] J. H. Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[68] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[69] W. Huang, L. Zhang, H. Wu, F. Min, and A. Song. Channel-equalization-har: a light-weight convolutional neural network for wearable sensor based human activity recognition. *IEEE Transactions on Mobile Computing*, 2022.

[70] Y. Huang, N. Schaal, M. Hefenbrock, Y. Zhou, T. Riedel, L. Fang, and M. Beigl. Mcxai: Local model-agnostic explanation as two games. *arXiv preprint arXiv:2201.01044*, 2022.

[71] Y. Huang, Y. Zhou, M. Hefenbrock, T. Riedel, L. Fang, and M. Beigl. Automatic feature engineering through monte carlo tree search. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 581–598. Springer, 2022.

[72] Y. Huang, Y. Zhou, M. Hefenbrock, T. Riedel, L. Fang, and M. Beigl. Universal distributional decision-based black-box adversarial attack with reinforcement learning. In *International Conference on Neural Information Processing*, pages 206–215. Springer, 2022.

[73] Y. Huang, C. Li, H. Lu, T. Riedel, and M. Beigl. State graph based explanation approach for black-box time series model. In *World Conference on Explainable Artificial Intelligence*, pages 153–164. Springer, 2023.

[74] Y. Huang, N. Schaal, M. Hefenbrock, Y. Zhou, T. Riedel, and M. Beigl. Mcxai: local model-agnostic explanation as two games. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 01–08. IEEE, 2023.

[75] Y. Huang, Y. Zhou, T. Riedel, L. Fang, and M. Beigl. randomhar: Improving ensemble deep learners for human activity recognition with sensor selection and reinforcement learning. *arXiv preprint arXiv:2307.07770*, 2023.

[76] Y. Huang, Z. Xue, H. Ma, and M. Beigl. Generate explanations for time-series classification by chatgpt. *Explainable Artificial Intelligence, Malta, 17th–19th June 2024*, 2024.

[77] Y. Huang, H. Zhao, Y. Zhou, T. Riedel, and M. Beigl. Standardizing your training process for human activity recognition models: A comprehensive review in the tunable factors. *EAI International Conference*

on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2024.

[78] Y. Huang, Y. Zhou, H. Zhao, L. Fang, T. Riedel, and M. Beigl. Extea: An evolutionary algorithm-based approach for enhancing explainability in time-series models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 429–446. Springer, 2024.

[79] Y. Huang, Y. Zhou, H. Zhao, T. Riedel, and M. Beigl. Optimizing automl for tiny edge systems: A baldwin-effect inspired genetic algorithm. In *22nd IEEE International Conference on Pervasive Computing and Communicaitons (PerCom 2024)*, 2024.

[80] Y. Huang, Y. Zhou, H. Zhao, T. Riedel, and M. Beigl. A survey on wearable human activity recognition: Innovative pipeline development for enhanced research and practice. In *2024 IEEE International Joint Conference on Neural Networks (IJCNN 2024), Yokohama, 30th June-5th July 2024*, 2024.

[81] S. Hurtado, J. García-Nieto, A. Popov, and I. Navas-Delgado. Human activity recognition from sensorised patient's data in healthcare: A streaming deep learning-based approach. 2023.

[82] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[83] W. N. Ismail, H. A. Alsalamah, M. M. Hassan, and E. Mohamed. Autohar: An adaptive human activity recognition framework using an automated cnn architecture design. *Heliyon*, 2023.

[84] A. Kamilaris and F. Prenafeta-Boldu. Deep learning in agriculture: A survey, computers and electronics in agriculture. 147: 70-90, 2018.

[85] J. M. Kanter and K. Veeramachaneni. Deep feature synthesis: Towards automating data science endeavors. In *2015 IEEE international con-*

ference on data science and advanced analytics (DSAA), pages 1–10. IEEE, 2015.

[86] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[87] I. Karlsson, J. Rebane, P. Papapetrou, and A. Gionis. Locally and globally explainable time series tweaking. *Knowledge and Information Systems*, 62(5):1671–1700, 2020.

[88] G. Katz, E. C. R. Shin, and D. Song. Explorekit: Automatic feature generation and selection. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 979–984. IEEE, 2016.

[89] A. Khtun and S. G. S. Hossain. A fourier domain feature approach for human activity recognition & fall detection. In *2023 10th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 40–45. IEEE, 2023.

[90] U. Khurana, D. Turaga, H. Samulowitz, and S. Parthasrathy. Cognito: Automated feature engineering for supervised learning. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 1304–1307. IEEE, 2016.

[91] U. Khurana, H. Samulowitz, and D. Turaga. Feature engineering for predictive modeling using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[92] B. Kim, K. Patel, A. Rostamizadeh, and J. Shah. Scalable and interpretable data representation for high-dimensional, complex data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[93] B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016.

[94] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[95] D. Kumar. Human activity recognition using, colour, rms, time-domain, autocorrelation deep learning with better feature extraction.

[96] B. Kwolek and M. Kepski. Human fall detection on embedded platform using depth maps and wireless accelerometer. *Computer methods and programs in biomedicine*, 117(3):489–501, 2014.

[97] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138, 2019.

[98] H. T. Lam, J.-M. Thiebaut, M. Sinn, B. Chen, T. Mai, and O. Alkan. One button machine for automating feature engineering in relational databases. *arXiv preprint arXiv:1706.00327*, 2017.

[99] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[100] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553): 436–444, 2015.

[101] X. Li, H. Xiong, X. Li, X. Wu, X. Zhang, J. Liu, J. Bian, and D. Dou. Interpretable deep learning: Interpretation, interpretability, trustworthiness, and beyond. *Knowledge and Information Systems*, 64(12):3197–3234, 2022.

[102] Y. Li and L. Wang. Human activity recognition based on residual network and bilstm. *Sensors*, 22(2):635, 2022.

[103] J. Lines, S. Taylor, and A. Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 1041–1046. IEEE, 2016.

[104] Z. C. Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16 (3):31–57, 2018.

[105] H. Liu, Y. Hartmann, and T. Schultz. Motion units: Generalized sequence modeling of human activities for sensor-based activity recognition. In *2021 29th European signal processing conference (EUSIPCO)*, pages 1506–1510. IEEE, 2021.

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

[107] L. Longo, R. Goebel, F. Lecue, P. Kieseberg, and A. Holzinger. Explainable artificial intelligence: Concepts, applications, research challenges and visions. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 1–16. Springer, 2020.

[108] Á. Lozano Murciego, D. M. Jiménez-Bravo, A. Valera Román, J. F. De Paz Santana, and M. N. Moreno-García. Context-aware recommender systems in the music domain: A systematic literature review. *Electronics*, 10(13):1555, 2021.

[109] L. Lu, C. Zhang, K. Cao, T. Deng, and Q. Yang. A multichannel cnn-gru model for human activity recognition. *IEEE Access*, 10:66797–66810, 2022.

[110] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

[111] H. Ma, W. Li, X. Zhang, S. Gao, and S. Lu. Attnsense: Multi-level Attention Mechanism for Multimodal Human Activity Recognition. In *IJCAI*, pages 3109–3115, 2019.

[112] V. B. Magdum, S. Kumar, and M. Tarambale. Opf-aga: Optimal power flow using advance genetic algorithm. In *2022 2nd International Conference on Intelligent Technologies (CONIT)*, pages 1–7. IEEE, 2022.

[113] C. Manresa-Yee, M. F. Roig-Maimó, S. Ramis, and R. Mas-Sansó. Advances in xai: explanation interfaces in healthcare. In *Handbook of Artificial Intelligence in Healthcare: Vol 2: Practicalities and Prospects*, pages 357–369. Springer, 2021.

[114] D. Mercier, A. Dengel, and S. Ahmed. Patchx: Explaining deep models by intelligible pattern patches for time-series classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[115] S. Mirjalili and S. Mirjalili. Genetic algorithm. *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pages 43–55, 2019.

[116] S. Mishra, E. Benetos, B. L. Sturm, and S. Dixon. Reliable local explanations for machine listening. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[117] M. Munir, S. A. Siddiqui, F. Küsters, D. Mercier, A. Dengel, and S. Ahmed. Tsxplain: Demystification of dnn decisions for time-series using natural language and statistical features. In *International conference on artificial neural networks*, pages 426–439. Springer, 2019.

[118] A. Muntaner-Mas, A. Martinez-Nicolas, C. J. Lavie, S. N. Blair, R. Ross, R. Arena, and F. B. Ortega. A systematic review of fitness apps and their potential clinical and sports utility for objective and remote assessment of cardiorespiratory fitness. *Sports Medicine*, 49:587–600, 2019.

[119] S. Münzner, P. Schmidt, A. Reiss, M. Hanselmann, R. Stiefelhagen, and R. Dürichen. Cnn-based sensor fusion techniques for multimodal human activity recognition. In *Proceedings of the 2017 ACM International Symposium on Wearable Computers*, pages 158–165, 2017.

[120] F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, and D. S. Turaga. Learning feature engineering for classification. In *Ijcai*, volume 17, pages 2529–2535, 2017.

[121] M. Nauta. Explainable ai and interpretable computer vision: From oversight to insight. 2023.

[122] I. Neves, D. Folgado, S. Santos, M. Barandas, A. Campagner, L. Ronzio, F. Cabitza, and H. Gamboa. Interpretable heartbeat classification using local model-agnostic explanations on ecgs. *Computers in Biology and Medicine*, 133:104393, 2021.

[123] H. Park, N. Kim, G. H. Lee, and J. K. Choi. Multicnn-filterlstm: Resource-efficient sensor-based human activity recognition in iot applications. *Future Generation Computer Systems*, 139:196–209, 2023.

[124] P. S. Parvatharaju, R. Doddaiah, T. Hartvigsen, and E. A. Rundensteiner. Learning saliency maps to explain deep time series classifiers. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1406–1415, 2021.

[125] O. P. Patri, A. V. Panangadan, C. Chelmis, and V. K. Prasanna. Extracting discriminative features for event-based electricity disaggregation. In *2014 IEEE Conference on Technologies for Sustainability (SusTech)*, pages 232–238. IEEE, 2014.

[126] M. Pawelczyk, C. Agarwal, S. Joshi, S. Upadhyay, and H. Lakkaraju. Exploring counterfactual explanations through the lens of adversarial examples: A theoretical and empirical analysis. In *International Conference on Artificial Intelligence and Statistics*, pages 4574–4594. PMLR, 2022.

[127] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[128] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE communications surveys & tutorials*, 16(1):414–454, 2013.

[129] T. Rakthanmanon and E. Keogh. Fast shapelets: A scalable algorithm for discovering time series shapelets. In *proceedings of the 2013 SIAM International Conference on Data Mining*, pages 668–676. SIAM, 2013.

[130] R. G. Ramos, J. D. Domingo, E. Zalama, and J. Gómez-García-Bermejo. Daily human activity recognition using non-intrusive sensors. *Sensors*, 21(16):5270, 2021.

[131] K. Reinhardt. Trust and trustworthiness in ai ethics. *AI and Ethics*, 3(3): 735–744, 2023.

[132] A. Reiss and D. Stricker. Introducing A New Benchmarked Dataset for Activity Monitoring. In *2012 16th international symposium on wearable computers*, pages 108–109. IEEE, 2012.

[133] J.-L. Reyes-Ortiz, L. Oneto, A. Samà, X. Parra, and D. Anguita. Transition-Aware Human Activity Recognition Using Smartphones. *Neurocomputing*, 171:754–767, 2016.

[134] M. T. Ribeiro, S. Singh, and C. Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[135] M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[136] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[137] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[138] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. nat mach intell 1 (5): 206–215, 2019.

[139] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-Propagating Errors. *nature*, 323(6088):533–536, 1986.

[140] A. Sarkar, S. S. Hossain, and R. Sarkar. Human activity recognition from sensor data using spatial attention-aided cnn with genetic algorithm. *Neural Computing and Applications*, 35(7):5165–5191, 2023.

[141] I. Sarker, A. Colman, J. Han, P. Watters, I. H. Sarker, A. Colman, J. Han, and P. Watters. Introduction to context-aware machine learning and mobile data analytics. *Context-Aware Machine Learning and Mobile Data Analytics: Automated Rule-based Services with Intelligent Decision-Making*, pages 3–13, 2021.

[142] B. Schelling, L. G. M. Bauer, S. Behzadi, and C. Plant. Utilizing structure-rich features to improve clustering. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*, pages 91–107. Springer, 2021.

[143] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *1994 first workshop on mobile computing systems and applications*, pages 85–90. IEEE, 1994.

[144] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim. Towards a rigorous evaluation of xai methods on time series. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4197–4201. IEEE, 2019.

[145] U. Schlegel, D. L. Vo, D. A. Keim, and D. Seebacher. Ts-mule: Local interpretable model-agnostic explanations for time series forecast models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 5–14. Springer, 2021.

[146] U. Schlegel, D. L. Vo, D. A. Keim, and D. Seebacher. Ts-mule: Local interpretable model-agnostic explanations for time series forecast models. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021,*

Virtual Event, September 13-17, 2021, Proceedings, Part I, pages 5–14. Springer, 2022.

[147] J. Schmidhuber. Deep learning. *Scholarpedia*, 10(11):32832, 2015.

[148] P. Senin and S. Malinchik. Sax-vsm: Interpretable time series classification using sax and vector space model. In *2013 IEEE 13th international conference on data mining*, pages 1175–1180. IEEE, 2013.

[149] F. Serpush, M. B. Menhaj, B. Masoumi, and B. Karasfi. Wearable sensor-based human activity recognition in the smart healthcare system. *Computational intelligence and neuroscience*, 2022(1):1391906, 2022.

[150] M. Sesana, S. Cavallaro, M. Calabresi, A. Capaccioli, L. Napoletano, V. Antonello, and F. Grandi. Process and product quality optimization with explainable artificial intelligence. In *Artificial Intelligence in Manufacturing: Enabling Intelligent, Flexible and Cost-Effective Production Through AI*, pages 459–477. Springer Nature Switzerland Cham, 2024.

[151] V. Shalaeva, S. Alkhoury, J. Marinescu, C. Amblard, and G. Bisson. Multi-operator decision trees for explainable time-series classification. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 86–99. Springer, 2018.

[152] J. Sharma, M. Lal Mittal, G. Soni, and A. Keprate. Explainable artificial intelligence (xai) approaches in predictive maintenance: A review. *Recent Patents on Engineering*, 18(5):18–26, 2024.

[153] Y. Shavit and I. Klein. Boosting inertial-based human activity recognition with transformers. *IEEE Access*, 9:53540–53547, 2021.

[154] J. Shore and R. Johnson. Axiomatic Derivation of the Principle of Maximum Entropy and the Principle of Minimum Cross-Entropy. *IEEE Transactions on information theory*, 26(1):26–37, 1980.

[155] S. A. Siddiqui, D. Mercier, A. Dengel, and S. Ahmed. Tsinsight: A local-global attribution framework for interpretability in time series data. *Sensors*, 21(21):7373, 2021.

[156] R. Singla, S. Mittal, A. Jain, and D. Gupta. Convlstm for human activity recognition. In *International Conference on Innovative Computing and Communications: Proceedings of ICICC 2021, Volume 2*, pages 335–344. Springer, 2022.

[157] T. Sivill and P. Flach. Limesegment: Meaningful, realistic time series explanations. In *International Conference on Artificial Intelligence and Statistics*, pages 3418–3433. PMLR, 2022.

[158] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[159] N. A. Streitz. The disappearing computer. 2007.

[160] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019.

[161] W. Tang, L. Liu, and G. Long. Interpretable time-series classification on few-shot samples. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[162] Y. Tang, L. Zhang, F. Min, and J. He. Multiscale deep feature learning for human activity recognition using wearable sensors. *IEEE Transactions on Industrial Electronics*, 70(2):2106–2116, 2022.

[163] D. Thakur and S. Biswas. Guided regularized random forest feature selection for smartphone based human activity recognition. *Journal of Ambient Intelligence and Humanized Computing*, 14(7):9767–9779, 2023.

[164] A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti. Explainable ai for time series classification: A review, taxonomy and research directions. *IEEE Access*, 2022.

[165] S. Tonekaboni, S. Joshi, M. D. McCradden, and A. Goldenberg. What clinicians want: contextualizing explainable machine learning for clinical end use. In *Machine learning for healthcare conference*, pages 359–380. PMLR, 2019.

[166] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[167] S. Verma, V. Boonsanong, M. Hoang, K. E. Hines, J. P. Dickerson, and C. Shah. Counterfactual explanations and algorithmic recourses for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[168] Q. N. P. VU, P. Lago, and S. Inoue. Improving noise robustness of single sensor data in human activity recognition with umap and additional data. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, pages 109–111, 2022.

[169] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[170] M. Wadhwa and U. Shrivastava. Ubiquitous computing: A comprehensive review. In *International Conference on Electrical and Electronics Engineering*, pages 591–600. Springer, 2023.

[171] J. Wang, Y. Gao, and R. Li. Reinforcement learning based bilevel real-time pricing strategy for a smart grid with distributed energy resources. *Applied Soft Computing*, 155:111474, 2024.

[172] Z. Wang, Z. Yang, and T. Dong. A review of wearable technologies for elderly care that can accurately track indoor position, recognize physical activities and monitor vital signs in real time. *Sensors*, 17(2):341, 2017.

[173] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1): 91–102, 1992.

[174] M. Webber and R. F. Rojas. Human activity recognition with accelerometer and gyroscope: A data fusion approach. *IEEE Sensors Journal*, 21 (15):16979–16989, 2021.

[175] M. Weiser. The computer for the 21 st century. *Scientific american*, 265 (3):94–105, 1991.

[176] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

[177] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *International Conference on Machine Learning*, pages 3921–3930. PMLR, 2017.

[178] L. Ye and E. Keogh. Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 947–956, 2009.

[179] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[180] C. Zhang, P. Patras, and H. Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3):2224–2287, 2019.

[181] J. Zhang, J. Hao, F. Fogelman-Soulié, and Z. Wang. Automatic feature engineering by deep reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2312–2314, 2019.

[182] H. Zhao, Y. Zhou, T. Riedel, M. Hefenbrock, and M. Beigl. Improving human activity recognition models by learnable sparse wavelet layer. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, pages 84–88, 2022.

[183] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):1–55, 2014.

[184] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[185] L. Zhou, C. Ma, X. Shi, D. Zhang, W. Li, and L. Wu. Salience-cam: Visual explanations from convolutional neural networks via salience score. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2021.

[186] Y. Zhou, M. Hefenbrock, Y. Huang, T. Riedel, and M. Beigl. Automatic remaining useful life estimation framework with embedded convolutional lstm as the backbone. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part IV*, pages 461–477. Springer, 2021.

[187] Y. Zhou, H. Zhao, Y. Huang, T. Riedel, M. Hefenbrock, and M. Beigl. Tinyhar: A lightweight deep learning model designed for human activity recognition. In *Proceedings of the 2022 ACM International Symposium on Wearable Computers*, pages 89–93, 2022.