UNIVERSIDADE FEDERAL DE MINAS GERAIS

School of Engineering

Graduate Program in Electrical Engineering

Hugo Vinicius Bitencourt

Embedding Fuzzy Time Series: a framework for high-dimensional time series forecasting in IoT applications

Belo Horizonte

2025

Hugo Vinicius Bitencourt

Embedding Fuzzy Time Series: a framework for high-dimensional time series forecasting in IoT applications

Thesis presented to the Graduate Program in Electrical Engineering at the Universidade Federal de Minas Gerais, as a partial requirement for obtaining the title of Doctor in Electrical Engineering.

Supervisor: Prof. Dr. Frederico Gadelha Guimarães

Belo Horizonte

2025

Bitencourt, Hugo Vinicius.

B624e Embedding Fuzzy Time Series [recurso eletrônico] : a framework for high-dimensional time series forecasting in IoT applications / Hugo Vinicius Bitencourt. - 2025.

1 recurso online (110 f. : il., color.) : pdf.

Orientador: Frederico Gadelha Guimarães.

Tese (doutorado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Anexos: f. 107-110.

Bibliografia: f. 96-106.

1. Engenharia elétrica - Teses. 2. Internet das coisas - Teses. 3. Análise de séries temporais - Teses. 4. Previsão - Teses. I. Guimarães, Frederico Gadelha. II. Universidade Federal de Minas Gerais. Escola de Engenharia. III. Título.

CDU: 621.3(043)

Ficha catalográfica elaborada pelo bibliotecário Reginaldo César Vital dos Santos CRB/6 2165
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

ESCOLA DE ENGENHARIA

COLEGIADO DO CURSO DE GRADUAÇÃO / PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FOLHA DE APROVAÇÃO

"Embedding Fuzzy Time Series: A Framework For High-dimensional Time Series Forecasting in IoT Applications"

Hugo Vinicius Bitencourt Paula

Tese de Doutorado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Doutor em Engenharia Elétrica.

Aprovada em 31 de janeiro de 2025.

Por:

Prof. Dr. Frederico Gadelha Guimarães
DCC (UFMG) - Orientador

Prof. Dr. Guilherme de Alencar Barreto
DETI (UFC)

Prof. Dr. Ticiana Linhares Coelho da Silva
Instituto Universidade Virtual (UFC)

Prof. Dr. Danton Diego Ferreira
Dep. Automática (DAT) (UFLA)

Prof. Dr. Petrônio Cândido de Lima e Silva
(IFNMG)

Documento assinado eletronicamente por **Frederico Gadelha Guimaraes**, **Professor do Magistério Superior**, em 02/02/2025, às 21:42, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Petrônio Cândido de Lima e Silva**, **Usuário Externo**, em 03/02/2025, às 09:44, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Danton Diego Ferreira**, **Usuário Externo**, em 03/02/2025, às 11:34, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Ticiana Linhares Coelho da Silva**, **Usuário Externo**, em 10/02/2025, às 09:28, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Guilherme de Alencar Barreto**, **Usuário Externo**, em 10/02/2025, às 11:22, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3876026** e o código CRC **8D985842**.

Referência: Processo nº 23072.202021/2025-86 SEI nº 3876026

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor Prof. Dr. Frederico Gadelha Guimarães for continuous support for my Ph.D study and research, for his patience, immense knowledge and valuable feedback. His guidance helped me in all the time of my research.

I would like to offer my special thanks to Prof. Dr. Petrônio Cândido de Lima e Silva for his assistance at every stage of my research project and insightful comments.

In addition, I would like to extend my sincere thanks to my fellows Omid Orang and Patrícia Lucas for his valuable assistance. He helped me in all the time of this research and writing of this PhD Thesis.

I want to give my deepest appreciation to the undergraduate students: Luiz Augusto Facury and Matheus Cascalho dos Santos for their valuable contribution to my research. I am very grateful for their help in designing the proposed framework and testing.

I would be remiss in not mentioning my family, especially my grandmother Duce, my aunt Dadá, and my mother. I am acutely grateful for their support and love.

Finally, my thanks go to all labmates on MINDS and professors and staff of the PPGEE/UFMG.

Abstract

Na Internet das Coisas (IoT), séries temporais multivariadas (de alta dimensão) são coletadas e armazenadas continuamente a partir de diferentes fontes de dados. Essas séries temporais são não-estacionárias visto que as características estatísticas do processo de geração dos dados modificam-se ao longo do tempo tornando-se dependentes do tempo. O processo de tomada de decisão em aplicações em Internet das Coisas pode envolver vários fatores e critérios. Portanto, é de grande valia para IoT, métodos de previsão inteligentes capazes de lidar com séries temporais multivariadas não-estacionárias e múltiplas saídas. Os métodos de Fuzzy Time Series (FTS) destacam-se como algoritmos não paramétricos orientados a dados, de fácil implementação e alta acurácia. Contudo, a previsão de séries temporais usando os métodos FTS existentes pode ser inviável caso todas as variáveis sejam utilizadas no treinamento do modelo em um problema de previsão, além disso, métodos de previsão FTS frequentemente enfrentam limitações ao lidar com dados de alta dimensão e fazer previsões de múltiplas saídas. Nesse sentido, para resolver os desafios apresentados, este trabalho apresenta um novo framework para previsão de séries temporais multivariadas não-estacionárias em aplicações de IoT denominado $embEFTS$ (Embedding Fuzzy Time Series). Dessa forma, foi aplicada uma combinação de embedding (redução de dimensionalidade) com modelos FTS. A combinação dessas técnicas permitiu uma melhor representação dos dados mais relevantes para previsão das séries temporais e previsões mais acuradas. Projetado como uma solução explicável e baseada em dados, $embEFTS$ é flexível e adaptável a diversas aplicações de IoT. O $embEFTS$ foi avaliado em múltiplos conjuntos de dados reais de séries temporais de alta dimensão na área de IoT. Os resultados experimentais demonstram que nosso framework supera diversos modelos de aprendizado de máquina e aprendizado profundo, incluindo LSTM, BiLSTM, CNN-LSTM, GRU, RNN, MLP, GBM e TCN. Comparado a abordagens existentes e resultados previamente publicados na literatura, $embEFTS$ apresenta eficiência, acurrácia e parcimônia superiores. O framework proposto foi testado em várias bases de dados de IoT do mundo real. O $embEFTS$ é uma abordagem explicável e orientada a dados, que é flexível e adaptável a diferentes aplicações de IoT. Os resultados experimentais mostram que o nosso framework supera na acurácia diversos métodos de aprendizado de máquina e aprendizado profundo. Além disso, os resultados confirmam a eficiência e parcimônia do $embEFTS$ em comparação com outras abordagens e resultados publicados na literatura. Portanto, $embEFTS$ é uma solução robusta e eficaz para previsão de séries temporais não estacionárias de alta dimensão.

Palavras-chave: Séries Temporais Multivariadas; Séries Temporais Nebulosas; Redução de Dimensionalidade; Internet das Coisas; Previsão de Séries Temporais.

Abstract

In the Internet of Things (IoT), high-dimensional time series data are continuously generated from multiple sources and often exhibit intrinsic changes known as concept drifts. Additionally, IoT decision-making frequently involves multiple factors and criteria. Therefore, methods capable of handling high-dimensional non-stationary time series with multiple outputs are essential. Fuzzy Time Series (FTS) methods stand out for their data-driven, non-parametric nature, ease of implementation, and high accuracy. However, existing FTS approaches struggle with high-dimensional data and multi-output predictions. To fill these gaps, we present a new framework for forecasting high-dimensional non-stationary time series in IoT applications called *embEFTS* (Embedding Fuzzy Time Series). We apply a combination of data embedding transformation and FTS methods. Combining these techniques enables a better representation of the complex content of multivariate time series and more accurate forecasts. Designed as an explainable and data-driven solution, *embEFTS* is flexible and adaptable to various IoT applications. We evaluated *embEFTS* on multiple real-world high-dimensional IoT time series datasets. Experimental results demonstrate that our framework outperforms several machine learning and deep learning models, including LSTM, BiLSTM, CNN-LSTM, GRU, RNN, MLP, GBM, and TCN. Compared to existing approaches and previously published results, *embEFTS* showcases superior efficiency, accuracy, and parsimony. Therefore, *embEFTS* is a robust and effective solution for forecasting high-dimensional non-stationary time series. It has the potential to assist homeowners in developing more efficient energy-saving strategies and can also support decision-making in air pollution management.

Keywords: Multivariate Time Series; Fuzzy Time Series; Embedding Transformation; Internet of Things; Time Series Forecasting.

List of Figures

List of Tables

List of Algorithms

List of abbreviations and acronyms

MINDS	Machine Intelligence and Data Science
UFMG	Universidade Federal de Minas Gerais
IoT	Internet of Things
WSN	Wireless Sensor Networks
IoE	Internet of Energy
FTS	Fuzzy Time Series
NSFTS	Non-Stationary Fuzzy Time Series
emb	Embedding Transformation (Embedding Function)
$embEFTS$	Embedding Fuzzy Time Series
EFTS	Embedding-based Fuzzy Time Series
MS-EFTS	Multi-Step Embedding-based FTS
MO-ENSFTS	Multiple Output Embedding-based Non-Stationary Fuzzy Time Series
MO-WMVFTS	Multiple-Output Weighted Multivariate Fuzzy Time Series
EWMVFTS	Embedding Weighted Multivariate Fuzzy Time Series
WMVFTS	Weighted Multivariate Fuzzy Time Series
MISO	Multiple Input Single Output
MIMO	Multiple Input Multiple Output
MF	Membership function
PCA	Principal Component Analysis
KPCA	Kernel Principal Component Analysis

AE	Autoencoders
SVM	Support Vector Machine
SVR	Support Vector Regression
FIG	Fuzzy Information Granule
MLR	Multiple Linear Regression
RF	Random Forest
GBM	Gradient Boosting Machines
RMSE	Root Mean Squared Error
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLP	Multilayer Perceptron
RBMs	Restricted Boltzmann Machines
CRBM	Conditional Restricted Boltzmann Machines
FCRBM	Factored Conditional Restricted Boltzmann Machines
RNN	Recurrent Neural Networks
ANN	Artificial Neural Networks
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory Network
GRU	Gated Recurrent Unit
BLSTM	Bidirectional Long Short-Term Memory Network
DTR	Decision Tree Regression
RFR	Random Forest Regression
GBR	Gradient Boosting Regression
VMD	Variational Model Decomposition
FLR	Fuzzy Logical Relationship
FLRG	Fuzzy Logical Relationship Group

NSFS	Non-stationary fuzzy sets
LHS	Left Hand Side
RHS	Right Hand Side
RBF	Gaussian Radial Basis Function
ADF	Augmented Dickey-Fuller
KPSS	Kwiatkowski-Phillips-Schmidt-Shin
AEC	UCI Appliances energy prediction Data Set
KSH	Kaggle Smart home with Weather Information Data Set
HPC	UCI Household Power Consumption Data Set
AQB	UCI Beijing Multi-Site Air-Quality Data Data Set
AQI	UCI Air Quality Data Set
CV	Cross-Validation
NMRSE	Normalized Root Mean Squared Error
FB	Fractional Bias
HP	Hyperparameters
ACF	Autocorrelation
PACF	Partial Autocorrelation
SARIMAX	Seasonal Auto-Regressive Integrated Moving Average with eXogenous Factors
SLSTM	Stacked LSTM encoder-decoder
PWFTS	Probabilistic Weighted FTS
FIG-FTS	Fuzzy Information Granular FTS
FCMs	Fuzzy Cognitive Maps
SN	Sensor Nodes
DL	Deep Learning
VAR	Vector AutoRegression

DT	Decision Tree
KNN	K-Nearest Neighbor
M-BLSTM	Multilayer Bidirectional LSTM
TA-EDN	Temporal Attention Encoder-Decoder Network
PTC	Physical-temporal Collection
TL-BLSTM	Transferred Bi-directional LSTM
AAQP	Attention-based Air Quality Predictor
SLI-ESN	Supplementary Leaky Integrator Echo State Network
mRMR	Minimum Redundancy Maximum Relevance
pyFTS	Fuzzy Time Series Library for Python
AIC	Akaike Information Criterion
TCN	Temporal Convolutional Network
CD	Critical Distance
RFID	Radio Frequency Identification

List of symbols

Y	Crisp Multivariate Time Series
N	Size of the Multivariate Time Series
t	Time index
$y(t)$	Crisp Time Series Elements at Time T
$\mathcal{V}$	Exogenous Variables (Explanatory Variables)
$\mathcal{V}^*$	Endogenous Variables (Target Variables)
K	Embedding Dimensions
κ	Number of Fuzzy Sets
F	Fuzzy Time Series
F_{emb}	Embedding Fuzzy Time Series
SN	Number of Sensors
$f(t)$	Fuzzy Time Series Elements at Time t
$f_{emb}(t)$	Embedding Fuzzy Time Series Elements at Time t
U	Universe of Discourse of Y
$\tilde{A}$	Fuzzy Sets
A_i	Fuzzy Set
Ω	FTS model order (time delays - lags)
Φ	Nonlinear Mapping Function
$\mathcal{DS}$	Data Sets
$\mathcal{M}$	Forecasting Models
W	Data Windows

CM	Covariance Matrix
$e(\cdot)$	Encoding Function
$d(\cdot)$	Decoding Function
$\mathcal{M}_{\mathcal{FTS}}$	FTS model
Y_{emb}	Embedding Time Series
emb	Embedding Function
δ	Displacement Function
$\pi()$	Perturbation Function
ρ	Scale Factor
μ_{A_i}	Membership Function
w_e	Length of the Residuals Window
k_c	Kernel Coefficient of KPCA
$\widetilde{K}$	Centered Kernel Matrix
wg	Normalized Frequencies of Each Temporal Pattern (Weights) of SO-EWMVFTS Method
r	Fuzzy Rules
c_i	Midpoints of Each Fuzzy Set
$\hat{y}^*$	Target predict value

Contents

Chapter 1

Introduction

The main focus of this chapter is listed as follows: problem statement and motivation; objectives; contributions; and work structure.

1.1 Problem Statement and Motivation

The rise of IoT (Internet of Things) applications and the increased data storage has led to a significant increase in streaming data, often structured as time series. However, this time series can be prone to errors and noise due to sensor issues, such as aging or hardware faults. Additionally, the monitored phenomena may change over time due to seasonal or weather variations. [Ditzler et al., 2015].

These time series are characterized by intrinsic changes that modify the properties of the process of data generating (i.e. non-stationary time series), which changes its underlying probability distribution over time. A non-stationary time series is the type of time series whose statistical characteristics including mean or variance (or both) are varying over the time and the changes can take several forms known as “concept drifts”. Concept drifts may deteriorate the accuracy of the model prediction over time, which requires permanent adaptation strategies.

In IoT, data from multiple sources is continuously recorded, forming a streaming, high-dimensional time series, where each dimension represents measurements from a sensor. An IoT system with SN (number of sensors) sensors generates an SN-dimensional time series. IoT and IoE are key contributors to Big Data, as they involve connecting numerous devices to the Internet to constantly monitor and report environmental conditions.

Decision-making in IoT applications may often involve multiple factors and criteria, for instance: the house owner can see not only the overall energy consumption prediction but also the estimated energy consumption from each room and appliances; authorities may want the forecast of several air pollutants (e.g. PM2.5[1], SO_2, NO_2, PM10[2]) simultaneously.

However, the analysis of such IoT time series poses significant challenges, both statistically and numerically due to seasonal variations, irregular trends, and the presence of missing values, outliers, and uncertainties in sensor data. These factors make accurate predictions difficult due to the unpredictable and noisy of the data [Atitallah et al., 2020],[Olu-Ajayi et al., 2022], [Sepasgozar et al., 2020].

Multiple output forecasting faces challenges such as exploring inter-target dependencies, modeling complex input-output relationships, and effectively capturing dependencies among predictors [Xu et al., 2020], [Reyes and Ventura, 2019].

Multi-step-ahead problem faces uncertainty factors, such as error accumulation and insufficient information, making multi-step-ahead forecasting more challenging. Therefore, when dealing with time series data from diverse sources like IoT applications, selecting an appropriate modeling strategy for multi-step-ahead time series forecasting holds significant practical value [Chandra et al., 2021] [Gocheva-Ilieva et al., 2023a].

Consequently, there are two primary strategies for multi-step forecasting: recursive and direct. The recursive approach involves utilizing the prediction from a one-step ahead model as input for subsequent predictions. Conversely, the direct approach treats the multi-step problem as a multi-output task, creating some models equal to the number of time steps [Suradhaniwar et al., 2021] [Gocheva-Ilieva et al., 2023b]. The recursive strategy tends to accumulate errors across future horizons, making it more suitable for relatively short forecasts. This is due to the propagation of bias and variance from previous time steps into future predictions.

Finally, it is important to analyze relationships between variables to select the most valuable inputs for accurate predictions. The key challenge is identifying an optimal set of inputs that are strongly correlated with the target output.

1.2 Objectives

To overcome the challenges presented above, the main objective of this work is to propose a new framework for forecasting high-dimensional non-stationary time series called *embEFTS* (Embedding Fuzzy Time Series) that are shown in Figure 1.

Consequently, we address the following objectives:

- Investigate the potential benefits of a methodology that combines an embedding transformation and a fuzzy time series forecasting approach for dealing with high-

[1] PM2.5 concentration refers to fine particles in the air with an aerodynamic equivalent diameter of 2.5 microns or less

[2] PM10 concentration means particulate matter 10 micrometers or less in diameter

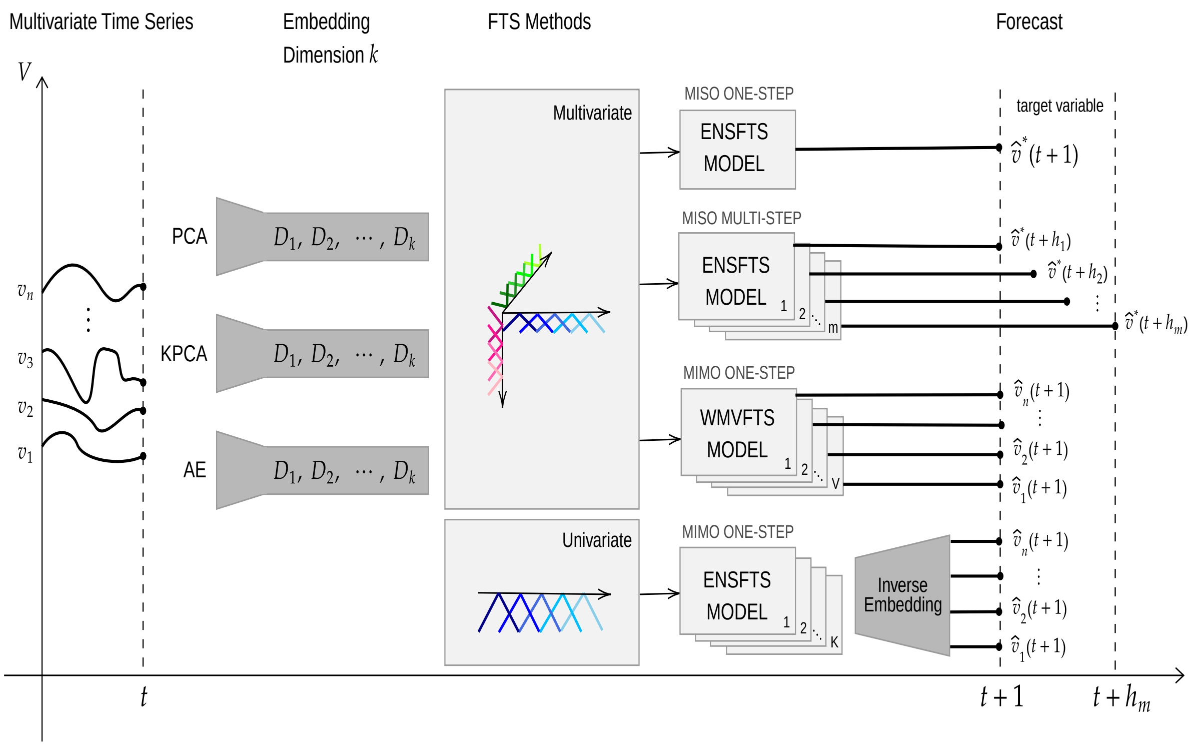

Figure 1 – Embedding Fuzzy Time Series ($embEFTS$)

dimensional non-stationary time series.

- Apply a data embedding transformation and use FTS models in a low dimensional, learned continuous representation, to provide more parsimonious and explainable models, consequently, with better possibilities of integration to real IoT applications.
- Propose forecasting methods (i.e. MISO one-step, MISO multi-step ahead, and MIMO one-step ahead) which are flexible and adaptable for many IoT applications.
- Present methods to handle high-dimensional data and multiple outputs.
- Propose methods able to provide accurate forecasting results in non-stationary environments.
- Test our proposed methods on several real-world high-dimensional IoT time-series datasets (smart buildings and smart cities applications) in order to evaluate and justify the superior accuracy of our approach with regards to the accuracy metrics.

1.3 Contributions

Guided by the aforementioned objectives, this work presented some contributions, the main ones being listed below:

1. We introduce a new framework called *embEFTS*, valuable for IoT applications in smart buildings and cities. It can help reduce electricity consumption and enhance energy-saving strategies. *embEFTS* is also useful for decision-making related to air pollution

2. A new methodology called EFTS (Embedding-based Fuzzy Time Series) [Bitencourt et al., 2023] is detailed for forecasting high-dimensional non-stationary time series. EFTS is a first-order MISO multivariate method.

3. A new multi-step forecasting method, MS-EFTS (Multi-Step Embedding-based FTS) [Bitencourt et al., 2025] is presented, extending EFTS for multi-step-ahead predictions of high-dimensional, non-stationary time series.

4. The study details our MO-ENSFTS (Multiple Output Embedding-based Non-Stationary Fuzzy Time Series) approach [Bitencourt et al., 2022], a first-order MIMO method that treats all variables as both target and explanatory variables.

5. Another MIMO forecasting methodology, MO-WMVFTS (Multiple-Output Weighted Multivariate Fuzzy Time Series) [Bitencourt et al., 2024] is introduced, improving upon EWMVFTS (Embedding Weighted Multivariate Fuzzy Time Series) [Bitencourt et al., 2023] to predict multiple outcomes. MO-WMVFTS shows higher accuracy than MO-ENSFTS [Bitencourt et al., 2022].

6. The study is among the few to address high-dimensional non-stationary time series forecasting with multiple outputs using FTS methods.

7. Experimental results indicate that the *embEFTS* framework outperforms other one-step and multi-step MISO and MIMO models in terms of accuracy, efficiency, and simplicity. Additionally, the proposed approach proves more efficient and accurate compared to existing methods in the literature.

1.4 Work Structure

To advance the objectives discussed above, the document is organized in the following chapters:

- **Chapter 2 - Literature Review**: introduce the literature review from the point of view of application problem and methodology.

- **Chapter 3 - Design of Experiments**: shows the experiments methodology, providing a description of the case of studies, experiments design, the evaluation metrics, computational experiments and reproducibility issues.

- **Chapter 4 - Embedding-based Fuzzy Time Series (EFTS)**: details our proposed and published methodology EFTS [Bitencourt et al., 2023] in order to deal with high-dimensional time series in MISO systems. This chapter also shows the experimental results of our proposed approach over several forecasting methods tested on the case of studies.

- **Chapter 5 - Multi-Step Embedding-based Fuzzy Time Series (MS-EFTS)**: introduces our multi-step forecasting methodology named Multi-Step Embedding-based FTS (MS-EFTS) [Bitencourt et al., 2025]. Additionally, we present the experimental results of MS-EFTS in comparison with various deep learning methods.

- **Chapter 6 - Multiple Output Embedding Non-Stationary FTS (MO-ENSFTS)**: describes our introduced and released approach MO-ENSFTS [Bitencourt et al., 2022] to handle high-dimensional time series and multiple output prediction (i.e. MIMO systems). Besides, we presents obtained experimental results of MO-ENSFTS against several machine learning and deep learning methods.

- **Chapter 7 - Multiple-Output Weighted Multivariate FTS (MO-WMVFTS)**: we present another MIMO forecasting methodology known as Multiple-Output Weighted Multivariate Fuzzy Time Series (MO-WMVFTS) [Bitencourt et al., 2024]. Furthermore, this chapter also highlights the performance of our proposed approach across multiple forecasting methods applied on the case of studies.

- **Chapter 8 - Conclusion and Future Works**: concludes this work, summarizing all the discussions presented. I also presents the new directions for future work based on what was conducted in this work.

Chapter 2

Literature Review

We describe the literature review from the point of view of application problem (i.e. time series prediction in Internet of Things) and and methodology (i.e. fuzzy time series and embedding transformation). In this chapter, we discuss the concepts of the Internet of Things (IoT), Time Series forecasting, Fuzzy Time Series (FTS) and embedding transformation (*emb*), which underlines the solutions proposed in this work. Furthermore, this chapter presents several current methods for energy consumption forecasting in smart buildings and air quality prediction in smart cities.

2.1 Internet of Things

We are on the verge of a new era in networking and communication that has significantly impacted our community and personal lives. This era has led to the creation of a dynamic and expansive network known as the Internet of Things (IoT). The IoT (see Figure 2) refers to a global network of interconnected devices, such as sensors, RFID (Radio Frequency Identification) tags, actuators, and mobile phones, all uniquely identifiable and connected through standard communication protocols [Union, 2005],[Gubbi et al., 2013],[Miorandi et al., 2012]. Internet of Things can impact on several aspects of everyday-life and behavior of potential users (see Figure 3), for instance:

- Sensors placed in homes and offices can enhance comfort in various ways, such as adjusting heating based on preferences and weather, changing lighting according to the time of day, preventing domestic incidents with monitoring systems, and saving energy by automatically turning off unused devices and anticipating user needs.
- Sensors can track vehicular traffic on highways, providing data like average speed and car count. They can also monitor air pollution by measuring smog levels, including carbon dioxide concentrations, and relay this information to health agencies.

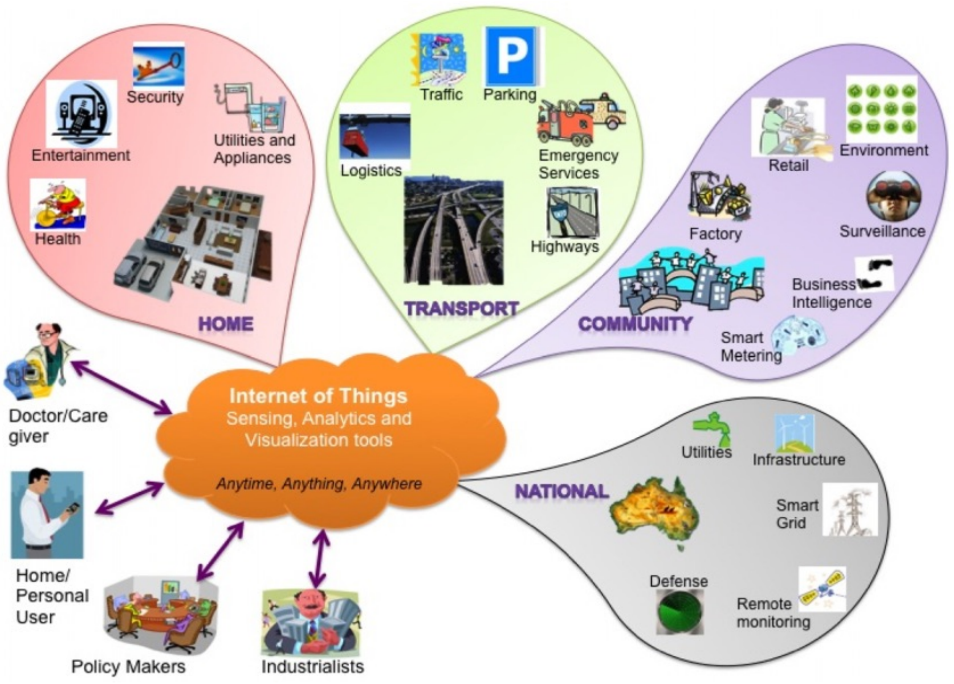

Figure 2 – Internet of Things [Gubbi et al., 2013]

- IoT can assist in predicting and managing unpredictable events like natural disasters. For instance, ambient sensors can detect the presence of hazardous chemicals.
- Wireless sensors and actuators can facilitate factory automation, inventory management, and leakage detection in liquids or gases. They are crucial for monitoring parameters like shock, noise, and temperature in remote or hard-to-reach areas, such as tanks, turbine engines, or pipelines.
- Sensors enable access to critical and hazardous areas where human presence is impractical, such as volcanic regions, ocean depths, or remote locations. They can detect anomalies, like fires, and promptly send alarms along with relevant data to support decision-making and response efforts.
- Patients can wear sensors to monitor vital signs, detect emergencies, and assist in diagnosing conditions. These sensors also gather important clinical data for rehabilitation, elderly monitoring, and support for individuals with physical impairments.

2.1.1 Internet of energy

The Internet of Energy (IoE) is a subset of the Internet of Things (IoT) that integrates sensors, actuators, smart meters, and other electrical grid components with information and communication technology (see Figure 4). IoE leverages the bidirectional flow of energy and information to provide deep insights into electricity usage, enhancing

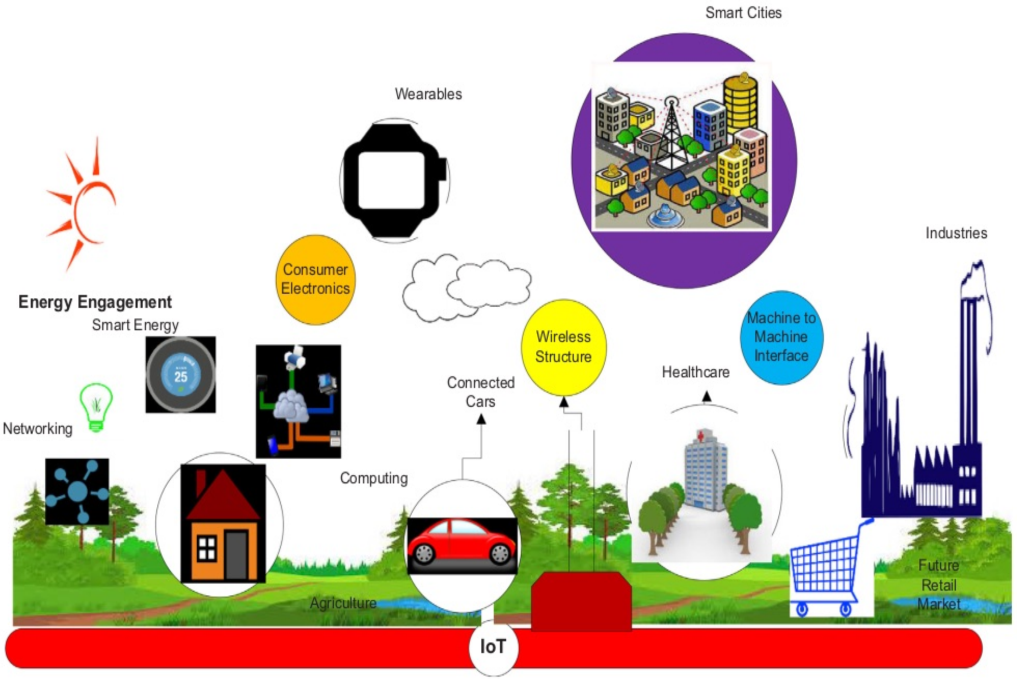

Figure 3 – Internet of Things Applications [Reka, 2018]

energy efficiency. It is crucial for implementing smart grids [Jaradat et al., 2015b], [Shahzad et al., 2020],[Shahinzadeh et al., 2019].

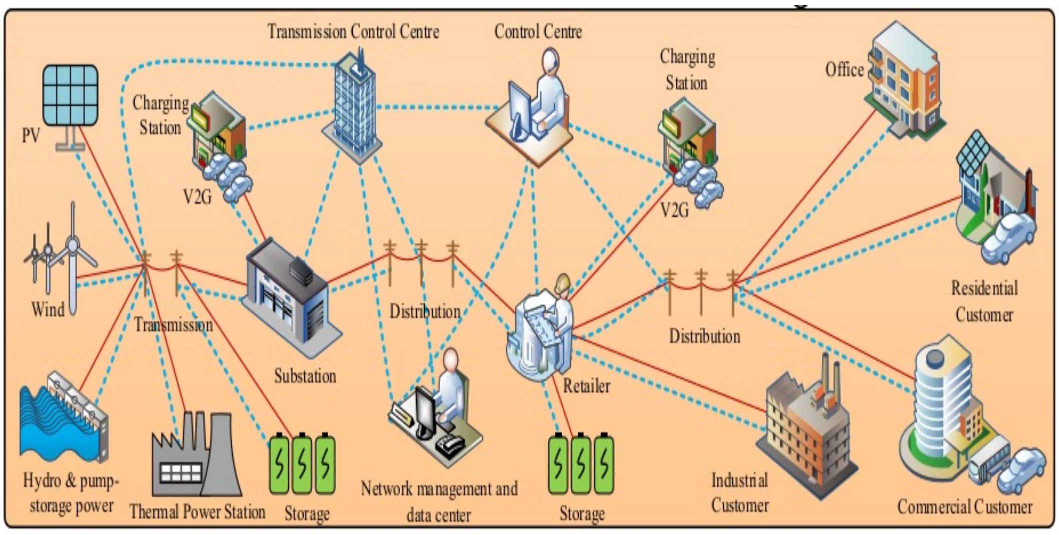

Figure 4 – Internet of Energy [Shahinzadeh et al., 2019]

Smart grids are advanced electrical power systems that modernize power distribution by digitally connecting all users, including suppliers and consumers. This integration makes power systems safer, more reliable, efficient, flexible, and sustainable. The use of Internet of Energy (IoE) technology in smart grids helps achieve these improvements [Jaradat et al., 2015a].

IoE can enhance home energy efficiency by connecting appliances to the internet, allowing homeowners to better manage utility usage and costs. Additionally, it can utilize meteorological data to predict and optimize appliance energy consumption [Mohammadi et al., 2018],[Manic et al., 2016].

2.1.2 Smart buildings and smart Cities

Smart cities and smart buildings are prominent IoT applications. Various IoT devices are deployed across modern cities to manage urban resources and improve services like healthcare, transportation, and energy. In smart buildings, sensors collect data to analyze and optimize energy consumption.

Smart buildings use IoE devices to monitor and analyze data, providing insights that optimize environments and operations. The smart home concept can also be applied to industrial, commercial, and residential buildings, adapting smart technology to various structures [Lobaccaro et al., 2016].

In smart buildings, an IoT network connects all digital devices to automate and assist users. For example, IoE devices track energy consumption and enable users to manage their electricity use through two-way communication with home appliances [Shahinzadeh et al., 2019].

Air quality is a critical concern in modern cities due to its impact on health issues like asthma, bronchitis, lung cancer, and heart problems, as well as its threat to the environment. To address this, technology is needed to measure and predict air quality. IoT is a promising solution that has significantly improved air quality monitoring and forecasting [Ameer et al., 2019].

Monitoring air pollution in modern cities enhances public health by issuing alerts when pollution levels exceed thresholds. This allows local authorities to assess the situation, predict pollution trends, and make informed decisions to address air quality issues [Ameer et al., 2019], [Zhang et al., 2021],[Bekkar et al., 2021].

2.2 Time Series Forecasting

A crisp time series Y of size N is an ordered sequence of observations $Y = (y(t_1), y(t_2), ..., y(t_N))$ where $y(t_i)$ represents the value at time period $i-th$. A multivariate time series $Y \in \mathbb{R}^{Nx\mathcal{V}}$ extends this to include multiple time-dependent variables, with the dimensionality defined by $\mathcal{V}$ where each vector $y(t) \in Y$ contains the values for all variables $\mathcal{V}_i \in \mathcal{V}$ at a given time.

Time series forecasting is a challenging problem which comprises both Multiple Input Single Output (MISO) and Multiple Input Multiple Output (MIMO) methods.

MISO methods receive a set of explanatory variables (or exogenous) $\mathcal{V}_i \in \mathcal{V}$ and only one target variable (or endogenous) $\mathcal{V}^* \in \mathcal{V}$ that is the output set, while MIMO methods use the same input set of variables as the output set, hence MIMO forecasting aims to estimate multiple values at once. Furthermore, time series forecasting methods can be divided into one-step ahead problem (i.e. forecast a time-step immediately following the latest observation) and multi-step ahead problem (i.e. predict two or more time-steps) [Xu et al., 2020] [Atitallah et al., 2020],[Olu-Ajayi et al., 2022],[Sepasgozar et al., 2020],[Syed et al., 2021],[Sajjad et al., 2020],[Ullah et al., 2020],[Parhizkar et al., 2021],[Bekkar et al., 2021],[Zhang et al., 2021],[Jin et al., 2021].

In this regard, the forecast horizon is the period for output forecasting and can be categorized into the following categories: (1) Very short-term forecast: it involves forecast outputs of less than 1 hour; (2) Short-term forecast: the forecast done for one hour, several hours, one hour, or up to seven days; (3) Medium-term forecast: it involves forecast outputs from one week to one month; (4) Longer-term forecast: it involves forecast output from one month to one year [Das et al., 2018],[Yadav and Chandel, 2014][Ahmed et al., 2020]. In general, the longer the forecast horizon, the greater is the chance of forecast error since model's error fluctuates with the change in the forecast horizon and add complexity.

2.2.1 Energy consumption forecasting

Accurately predicting energy consumption is crucial for smart buildings, as it helps reduce power usage, leading to significant energy and cost savings. Energy forecasting enables building managers to plan consumption, shift usage to off-peak times, set energy-saving targets, and optimize energy purchases [Mocanu et al., 2016].

Accordingly, energy consumption forecasting in smart buildings is a time series problem that can involve single or multiple variables. These time series often display seasonal variations, irregular trends, and may contain missing data, outliers, or uncertainties. This makes precise forecasting challenging due to unpredictable disturbances and noisy sensor data. To address these issues, various machine learning models have been developed, leveraging historical sensor data to enhance grid quality and optimize energy utilization.

Candanedo et al. [Candanedo et al., 2017] implemented and evaluated four forecast models for appliance energy consumption in a low-energy house in Belgium: Multiple Linear Regression (MLR), Support Vector Machine with Radial Kernel (SVM-radial), Random Forest (RF), and Gradient Boosting Machines (GBM). Among these, GBM performed the best, explaining 57% of the variance (R^2) and achieving an RMSE of 66.65, a MAE of 35.22, and a MAPE of 38.29% in the testing set when all variables were included.

Chammas et al. [Chammas et al., 2019] proposed a Multilayer Perceptron (MLP) with four hidden layers, each containing 512 neurons, to forecast appliance energy consumption. The MLP model explained 56% of the variance and achieved a RMSE of 66.29, a MAE of 29.55, and a MAPE of 27.96% on the testing set when using all variables. However, the dataset was small for an MLP, and neural networks are sensitive to weight initialization, prone to local minima, and exhibit slow convergence. Balancing overfitting and generalization remains a challenge for neural networks.

Mocanu et al. [Mocanu et al., 2016] developed two variants of the Restricted Boltzmann Machines (RBMs) stochastic model for forecasting residential energy consumption: the Conditional RBM (CRBM) and the Factored Conditional RBM (FCRBM). They compared these models with traditional machine learning methods like Support Vector Machines (SVM), Artificial Neural Networks (ANN), and Recurrent Neural Networks (RNN), across various time horizons and resolutions, ranging from one minute to one week. The FCRBM outperformed all other models, including ANN, RNN, SVM, and CRBM, in terms of accuracy. However, the study was limited to one-dimensional time series data.

Moldovan et al. [Moldovan and Slowik, 2021] introduced a multi-objective binary grey wolf optimization technique combined with various regression algorithms such as Random Forest, Extra Trees, Decision Trees (DT), and K-Nearest Neighbor (KNN) for energy appliance prediction.

Parhizkar et al. [Parhizkar et al., 2021] enhanced the accuracy of smart building energy forecasting by applying PCA to preprocess data and extract key features from four smart building datasets. These features were then used to improve the prediction performance of five machine learning models: linear regression, support vector regression (SVR), regression tree, random forest, and K-nearest neighbors.

Syed et al. [Syed et al., 2021] introduced a framework called HSBUFC, which integrates a hybrid stacked bi-directional and uni-directional Long Short-Term Memory Networks (LSTMs) with fully connected dense layers and a data cleaning process. This framework outperformed other hybrid models, including CNN-LSTM and ConvLSTM, in prediction accuracy.

Kim et al. [Kim and Cho, 2019] proposed a hybrid model, CNN-LSTM, for household energy consumption forecasting. CNN was used to extract variables that influence energy consumption, while the LSTM modeled the temporal patterns. This hybrid model outperformed standalone LSTM, achieving MSE of 37.38 and RMSE of 61.14. However, the authors noted that the primary drawback of the hybrid approach was the extensive trial-and-error process required to determine the optimal hyperparameters, leading to increased complexity.

Sajjad et al. [Sajjad et al., 2020] implemented a hybrid deep learning method, CNN-GRU, for energy forecasting. This approach combines CNN for extracting spatial variables with Gated Recurrent Unit (GRU) for exploiting temporal patterns through its gated structure. The results demonstrated that CNN-GRU outperformed both individual models and the CNN-LSTM hybrid model.

Similarly, Ullah et al. [Ullah et al., 2020] introduced a hybrid method integrating CNN with multilayer bidirectional LSTM (M-BLSTM) for short-term energy forecasting. Their approach was compared against BLSTM, LSTM, and CNN-LSTM models, showing superior performance across all benchmarks.

Lu et al. [Lu et al., 2021] proposed a Temporal Attention Encoder-Decoder Network (TA-EDN) hybrid method, which combines the encoder-decoder network, LSTM, and attention mechanism for 24-hour-ahead thermal load forecasting. Similarly, Antal et al. [Antal et al., 2022] introduced a model that integrates clustering with MLP classification for 24-step-ahead energy prediction, aimed at detecting energy profile classes.

Furthermore, Cascone et al. [Cascone et al., 2023] presented a convolutional LSTM model for predicting electric power consumption one week in advance, while Alobaidi et al. [Alobaidi et al., 2018] developed an ensemble ANN-based framework utilizing minute-controlled resampling techniques to predict day-ahead average household energy consumption in France.

2.2.2 Air quality forecasting

Intelligent air pollution forecasting systems play a crucial role in decision-making for both individuals and governments. Accurately predicting air quality requires a clear understanding of how pollutants change across various temporal contexts, as well as the non-linear relationships in time and space that affect forecast accuracy. As a result, air quality forecasting has become an active research area, with numerous methodologies being developed [Ameer et al., 2019][Zhang et al., 2021].

Ammer et al. [Ameer et al., 2019] implemented and compared four models Decision Tree Regression (DTR), Random Forest Regression (RFR), Multi-layer Perceptron (MLP), and Gradient Boosting Regression (GBR) to predict PM2.5 concentrations in large Chinese cities. Among these, RFR emerged as the most accurate method, achieving the lowest RMSE and shortest processing time.

Bekkar et al. [Bekkar et al., 2021] introduced a hybrid deep learning method (CNN-LSTM) to forecast hourly PM2.5 concentrations in Beijing, using data from twelve nationally-controlled air-quality monitoring sites. The method integrated historical air pollution data from the Aotizhongxin site, weather data from a nearby meteorological station, and PM2.5 concentrations from adjacent sites to predict PM2.5 concentrations at the Aotizhongxin site.

Zhang et al. [Zhang et al., 2021] proposed another hybrid deep learning model (VBM-BiLSTM) for forecasting hourly PM 2.5 concentrations in Beijing. This model combines variational model decomposition (VMD) with bi-directional LSTM. The approach was tested on air pollution and weather data from various monitoring sites individually, including Dongsi, Dingling, Gucheng, and Changping.

Jin et al. [Jin et al., 2021] proposed a data-driven deep learning method using nested LSTM to predict six air quality features (PM2.5, PM10, SO_2, NO_2, CO, and O_3) simultaneously. The approach incorporates federated learning and is among the few studies addressing multiple prediction outputs in smart cities.

Zhang at al. [Zhang et al., 2020] introduced BiAGRU, a spatial-temporal deep learning algorithm for air quality prediction. This method combines a Bidirectional GRU with an embedded attention mechanism, tested with different forecast horizons. BiAGRU outperformed several methods, including RF, KNN, LSTM, GRU, Bi-GRU, and CNN-LSTM.

Fei et al. [Fei et al., 2019] proposed an ensemble method called physical-temporal collection (PTC) for predicting PM2.5 with a forecast horizon of 24 hours. PTC integrates a cascaded LSTM (C-LSTM) for prediction and uses auxiliary information for time series correction. Additionally, XGBoost is applied to identify key temporal and weather patterns, optimizing input variables by eliminating irrelevant nodes.

Qi et al. [Qi et al., 2019] introduced a hybrid model called GC-LSTM, combining Graph CNN and LSTM to forecast the spatiotemporal variations of PM2.5 concentrations with a 24-hour prediction horizon. Additionally, Ma at al. [Ma et al., 2019] developed a multi-step-ahead method known as Transferred Bi-directional LSTM (TL-BLSTM) for PM2.5 prediction, also focusing on a 24-hour forecast. Furthermore, Gocheva-Ilieva et al. [Gocheva-Ilieva et al., 2023b] utilized RF combined with arcing (Arc-x4) to predict PM10, SO_2, and NO_2 concentrations up to 10 days in advance.

Liu et al. [Liu et al., 2019] presented the Attention-based Air Quality Predictor (AAQP) as a sequence-to-sequence model for predicting PM2.5 concentrations, with a focus on 8 and 24-hour forecasts. Xu et al. [Xu and Ren, 2019] proposed a supplementary leaky integrator echo state network (SLI-ESN) combined with minimum redundancy maximum relevance (mRMR) to forecast PM2.5 concentrations in Beijing for one, five, and ten steps ahead.

Jin et. al [Jin et al., 2021] introduced a multitask multichannel nested LSTM (MTMC-NLSTM) to predict six air quality features, including PM2.5, PM10, SO_2, NO_2, CO, and O_3. Additionally, Samal at al. [Samal et al., 2022] proposed a method known as the LSTM auto-encoder (M-LSTMA) to forecast PM2.5 and PM10 levels in Beijing. Samal at al. also introduced an approach called the multi-output temporal CNN auto-encoder (MO-TCNA) for long-term predictions of PM2.5 and PM10.

Finally, Munkhdalai et al. [Munkhdalai et al., 2019] proposed a method combining RNN with an adaptive feature selection mechanism. The approach consists of two components: the first generates context-dependent importance weights to select relevant variables, while the second component predicts the target variable. Notably, this study considers $CO(GT)$ and $NO_2(GT)$ variables from the dataset mentioned in subsection 3.1.5 along with the appliance energy consumption dataset discussed in subsection 3.1.1, as individual target variables.

2.2.3 Deep learning methods challenges

Various deep learning (DL) methods have been presented above to forecast energy consumption in smart buildings and air quality in smart cities, outperforming classical statistical methods like Vector AutoRegression (VAR). Recently, hybrid DL models have gained popularity by combining techniques to improve prediction accuracy. For example, Convolutional Neural Networks (CNNs) capture spatial variables, while Recurrent Neural Networks (RNNs), such as LSTMs, focus on temporal patterns.

However, DL models face challenges when forecasting high-dimensional non-stationary time series in IoT applications. As the number of variables grows, the presence of redundant variables can reduce training efficiency and increase the risk of falling into local minima in the error function. Moreover, hyperparameter optimization in DL models is often a trial-and-error process that requires significant effort. These methods also tend to be highly complex and offer limited explainability.

While CNNs are effective at feature extraction and LSTMs excel in handling sequential data, increasing dataset sizes and model parameters make hyperparameter tuning even more difficult, leading to higher computational costs and complexity.

2.3 Fuzzy Time Series

Fuzzy Time Series (FTS) methods have gained popularity in time series forecasting due to their simplicity, low computational cost, accuracy, and interpretability [Singh, 2016], [Bose and Mali, 2019], [Singh, 2015], [Palomero et al., 2022]. First introduced by Song and Chissom [Song and Chissom, 1993] to address ambiguity in time series data, FTS represents time series using fuzzy sets, converting numerical values into linguistic terms.

Fuzzy time series involves dividing the Universe of Discourse (U) in a time series into intervals or partitions (fuzzy sets) and analyzing their behavior by identifying patterns. The extracted rules describe how these partitions interact over time as values shift. Essentially, this process transforms the numerical time series data into a linguistic variable, where each partition serves as a linguistic term. Fuzzy sets are naturally suited for time series forecasting due to their ability to approximate functions and their intuitive, human-readable rules. Using linguistic variables enhances interpretability, making analysis accessible to both experts and non-experts.

Over time, various FTS methods have been developed, classified by their order Ω (i.e. number of lags) and time variance (i.e. whether the model changes over time) [de Lima Silva et al., 2020]. A first order FTS model requires $y(t-1)$ data to predict $\hat{y}^*(t)$, and a high order FTS model requires $y(t-\Omega), \ldots, y(t-1)$ data to predict $\hat{y}^*(t)$. Additionally, both multivariate and univariate FTS approaches have been explored in the literature [Singh, 2016], [Bose and Mali, 2019], [Singh, 2015], [Palomero et al., 2022]. Figure 5 presents the key steps of the FTS training and forecasting process.

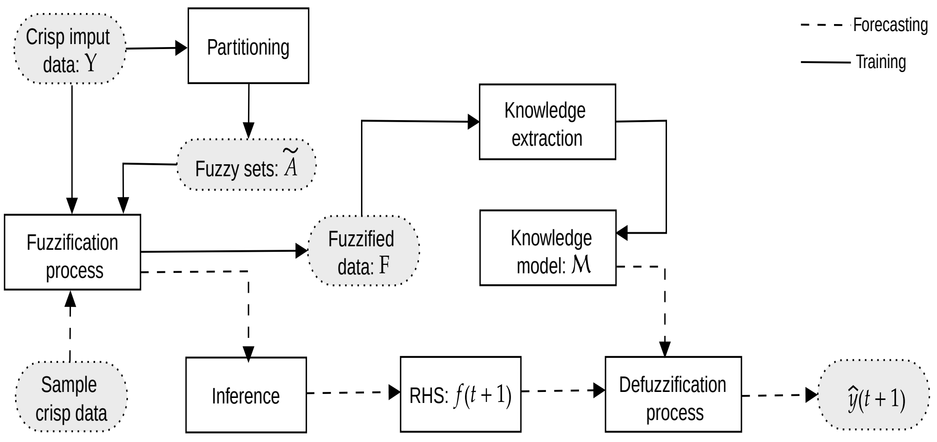

Figure 5 – General training and forecasting steps for fuzzy time series methods

2.3.1 FTS training procedure

Figure 6 shows the training procedure for a generic one-step ahead univariate FTS model (i.e. toy example). The Universe of Discourse is divided into intervals within the known range of the time series Y, where $U = [\min(Y), \max(Y)]$. For each interval, a fuzzy set $A_i \in \tilde{A}$ with its own membership function (MF) $\mu_{A_i} : \mathbb{R} \to [0,1]$ is defined, then each fuzzy set is assigned a linguistic label representing a specific range within U. For our example, the time series is divided into six partitions, such that the linguistic variable $\tilde{A} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$.

Consequently, the time series Y with its numerical values $Y(t)$ is then fuzzified into a fuzzy time series F, by mapping the values to fuzzy sets based on their maximum membership. Since the fuzzy sets in $\tilde{A}$ are overlapped, it is possible that for each $y(t) \in Y$ it belongs to more than one fuzzy set.

Temporal patterns with lag count Ω are derived from F. Each pattern represents two sequential fuzzy sets in a fuzzy time series, formatted as *Precedent* $\rightarrow$ *Consequent*. The precedent corresponds to the fuzzy set at time t, while the consequent represents the fuzzy set at time $t + 1$. Each pattern represents a fuzzy rule called Fuzzy Logical Relationship (FLR). In our example, the generated patterns are as follows: $A_1 \to A_2$, $A_2 \to A_3$, $A_3 \to A_4$, $A_4 \to A_4$, $A_4 \to A_5$, $A_5 \to A_4$, $A_4 \to A_3$, $A_3 \to A_3$.

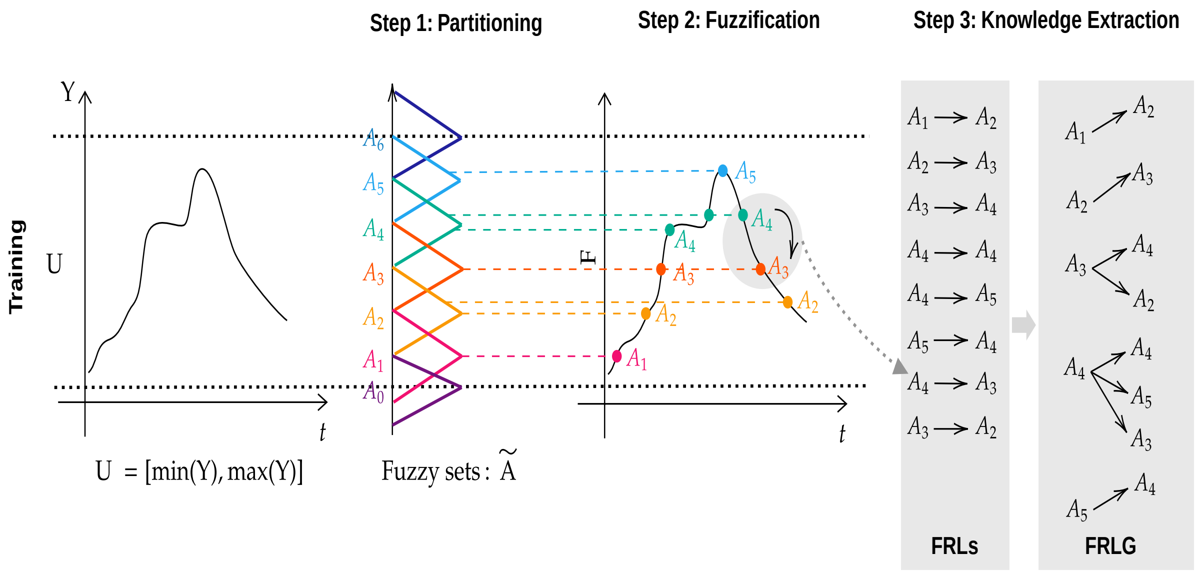

Figure 6 – FTS training procedure [Lucas et al., 2022]

Since fuzzy rules have the format *Precedent* $\to$ *Consequent*, given that the previously generated temporal patterns will be grouped by its precedents - Fuzzy Logical Relationship Group (FLRG). In the given example, the model includes a fuzzy rule for each unique precedent. The consequent of each rule is determined by combining all consequents from temporal patterns that share the same precedent. The FLRG form the rule base which is the final representation of the FTS forecasting model. For the given example, the resulting fuzzy rules are:

$A_1 \to A_2$

$A_2 \to A_3$

$A_3 \to A_4, A_2$

$A_4 \to A_4, A_5, A_3$

$A_5 \to A_4$

The fuzzy rule set represents the trained FTS model, which defines the time series behavior and enables predictions for $t + 1$. In summary, FTS is a simple and interpretable approach, easily parallelizable for big data and efficiently updatable for dynamic datasets.

2.3.2 FTS forecasting procedure

Figure 7 shows the FTS forecasting procedure. Once the FTS model is trained, it can be used to predict new values. In this sense, the crisp values $y(t - \Omega), \ldots, y(t - 1)$ are mapped to the fuzzified values $f(t-\Omega), \ldots, f(t-1)$, where $f(t) = \mu_{A_i}(y(t)), \forall A_i \in \tilde{A}$, for $t = 1, \ldots, N$. The input values are transformed into fuzzy values based on the linguistic variable, producing a corresponding fuzzy value. In the example from Figure 7, fuzzifying $y(t)$, results in the most relevant fuzzy set being A_4, so $f(t) = A_4$.

The fuzzy rules matching the given input are identified by finding the rule whose precedent equals $f(t)$. The rule's consequent then provides the fuzzy forecast for $t+1$, denoted as $f(t+1)$. In our example, for $f(t) = A_4$, the corresponding fuzzy rule is $A_4 \to A_4, A_5, A_3$. In other words, the FLRG whose precedent (Left Hand Side - LHS) is equal to the input value is selected and the candidate fuzzy sets in its consequent (Right Hand Side - RHS) are applied to estimate the predicted value.

Finally, To obtain a numerical value from $f(t+1)$, the center-of-mass method is applied. Then, the numerical forecast value $\hat{y^*}(t)$ is calculated as the mean of the centers of the fuzzy sets in $f(t+1)$.

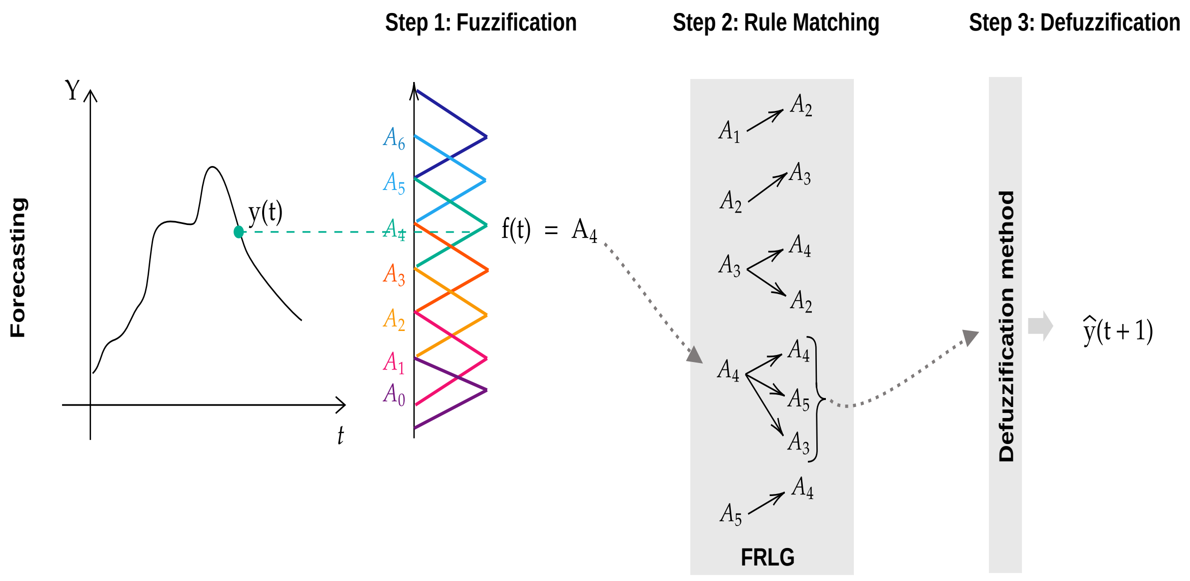

Figure 7 – FTS forecasting procedure [Lucas et al., 2022]

2.3.3 Fuzzy time series forecasting methods

As previously mentioned, various multivariate FTS models have been proposed, such as WMVFTS (Weighted Multivariate Fuzzy Time Series) [de Lima Silva et al., 2019], which allows for individual partitioning schemes for each variable, and can be distributed across network clusters for efficient training. Additionally, multivariate time series forecasting using self-organizing maps and FTS has been introduced in [dos Santos et al., 2021], along with the high-order MIMO forecasting model (FIG-FTS) [Lima e Silva et al., 2019].

Other examples of FTS methods which have been applied to time series prediction tasks, including point, interval, and probabilistic forecasting. Notable examples include Probabilistic Weighted FTS (PWFTS) [Silva et al., 2019b], Fuzzy Information Granular FTS (FIG-FTS) [Silva et al., 2019a]. An evolving multivariate FTS approach (e-MVFTS) [Severiano et al., 2021] has also been proposed for multi-step forecasting, particularly in renewable energy systems, emphasizing the method's flexibility for spatio-temporal predictions

Non-Stationary Fuzzy Sets (NSFS) [Garibaldi and Ozen, 2007] and [Garibaldi et al., 2008] dynamically adjust membership functions over time. NSFS can be used for non-stationary series forecasting but are limited to predictable changes in data variance, struggling with complex shifts like concept drift. The Non-Stationary Fuzzy Time Series (NSFTS) [de Lima Silva et al., 2020] method improves this by adapting fuzzy sets based on residual errors, though it is limited to univariate forecasting.

Fuzzy cognitive maps (FCMs) are the other category of FTS forecasting models. In [Papageorgiou and Froelich, 2012], a method was proposed for multi-step forecasting of pulmonary infection, utilizing FCM and an evolutionary learning algorithm. Froelich et al. [Froelich et al., 2012] introduced an FCM model for the long-term prediction of prostate cancer. Froelich and Salmeron [Froelich and Salmeron, 2014] suggested a multi-step FCM-based model to predict multivariate interval-valued time series.

In [Vanhoenshoven et al., 2020], a multivariate forecasting method was adopted utilizing FCM trained via the Moore-Penrose inverse. A new multi-step ahead forecasting model was presented in [Feng et al., 2021] integrating FCM, time series segmentation, and fuzzy clustering Wang et al. [Wang et al., 2022] presented a long-term time series forecasting model using an adaptive FCM model based on trend-fuzzy granulation. The method considers various forecasting horizons.

2.3.4 Fuzzy time series methods challenges

The FTS methods mentioned above show the potential for handling various forecasting tasks but often face limitations in handling high-dimensional and complex IoT datasets. A limited number of FTS methods have been developed to handle high-dimensional time series, and few multi-step FTS forecasting techniques exist in the literature. Notably, none have been applied to IoT scenarios. Moreover, FTS methods face challenges in managing high-dimensional data and making multiple output predictions.

As the dimensionality of time series increases, the accuracy of FTS methods decreases, and their complexity escalates. Each variable requires its own fuzzy sets, and the number of rules in multivariate FTS models grows exponentially with more variables, making it impractical to input all data into a single FTS model. The challenge lies in selecting an optimal set of inputs based on strong correlations with the target variable.

2.4 Embedding Transformation

Various approaches have been developed for handling high-dimensional data, with two main techniques being feature selection and feature extraction. In feature selection, a subset of the original features is chosen, while feature extraction creates new variables by mapping from the existing variables, using either linear or non-linear methods. This work concentrates on feature extraction techniques.

The objective of feature extraction is to learn a mapping function $emb : \mathbb{R}^V \rightarrow \mathbb{R}^K$ that transform V-dimensional data over N-time steps into a low dimensional space K, where $K \ll V$. Consequently, several feature extraction methods are available for different data types and needs. This work focuses on three key methods: Principal Component Analysis (PCA) [F.R.S., 1901], Kernel Principal Component Analysis (KPCA) [Kim et al., 2005] and Autoencoder (AE) [Rumelhart et al., 1986].

2.4.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a widely used feature extraction technique. It identifies the cross-correlation among variables and generates a reduced set of variables known as principal components, which are linearly uncorrelated. These principal components capture the maximum possible variance in the data while maintaining orthogonality to previously extracted components. PCA provides a compact data representation that explains a specified total percentage of the variance in the original dataset.

PCA assesses the cross-correlation among variables and identifies a reduced set of variables known as linearly uncorrelated principal components. These principal components explain the maximum possible variance in the data while maintaining orthogonality to all previously extracted variables.

PCA embedding comprises the following steps: we calculate the covariance matrix $CM \in \mathbb{R}^{V \times V}$ and extract the first K eigenvectors associated with the largest eigenvalues. Consequently, we access the matrix $Z \in \mathbb{R}^{V \times K}$ that is used to compute the embedding variables $emb(Y) : Z^\top \cdot y$ with $y \in \mathbb{R}^V$.

2.4.2 Kernel Principal Component Analysis (KPCA)

Since PCA cannot capture non-linear correlations between variables, Kernel Principal Component Analysis (KPCA) is used as an alternative to address this limitation. KPCA utilizes kernel functions (Φ) to perform non-linear feature extraction by applying a suitable non-linear mapping function. Examples of kernel functions used in KPCA include

Gaussian radial basis function (RBF), polynomial kernels, and sigmoid kernels, which enable the extraction of non-linear relationships among the features.

The Gaussian Radial Basis Function (RBF) for KPCA is implemented in three steps: (a) The similarity matrix is computed using Equation (2.1), which evaluates the similarity between two points, x_i and x_j, based on the kernel coefficient k_c.

$$k(ij) = \exp\left(-k_c \parallel x_i - x_j \parallel_2^2\right). \tag{2.1}$$

(b) Since the kernel matrix $k(ij)$ is not guaranteed to be centered, it is centralized by calculating $\widetilde{k} = k(ij) - 1_T k(ij) - k(ij)1_T + 1_T k(ij) 1_T$, where 1_T is an $T \times T$ matrix with all elements equal to $1/T$. (c) The eigenvalue problem $\widetilde{\boldsymbol{k}}\boldsymbol{\alpha}_v = emb_v T\boldsymbol{\alpha}_v$ is solved. The eigenvectors corresponding to the largest eigenvalues of the centered matrix $\widetilde{k}$ are then extracted.

2.4.3 Autoencoder (AE)

Autoencoders (AEs) are a type of artificial neural network designed to convert input data into a latent, compact representation, effectively encoding the data. First introduced by Rumelhart et al. [Rumelhart et al., 1986], autoencoders aim to reconstruct the input by minimizing reconstruction error, making them useful for unsupervised learning and transfer learning. This process involves encoding the data, then decoding it back to compare it with the original input [Mohammadi et al., 2018] [Gensler et al., 2016].

An autoencoder consists of an encoder and a decoder layers, connected through one or more hidden layers that transform high-dimensional data into a reduced K-dimensional representation. The reduced representation is achieved through bottleneck features in the hidden layers, enabling the autoencoder to capture and manage non-linear correlations between variables.

Autoencoders embedding procedure seeks to learn an encoding function $e : \mathbb{R}^V \rightarrow \mathbb{R}^K$ and decoding function $d : \mathbb{R}^K \rightarrow \mathbb{R}^V$, as follows

$$(e(\cdot), d(\cdot)) = \arg \min_{e(\cdot),d(\cdot)} ||Y - d(e(Y))||^2 \tag{2.2}$$

we utilized the encoding function e as a direct embedding function emb for the high-dimensional time series $Y \in \mathbb{R}^V$, such that

$$emb(Y) = e(Y) \tag{2.3}$$

Chapter 3

Design of Experiments

In this chapter, we describe the design of experiments. Several experiments have been conducted to assess the accuracy of the proposed framework (i.e. $embEFTS$). The accuracy is compared with several time series forecasting methods. The main focus of this chapter is listed as follows: a description of the case of studies, experiments design, the evaluation metrics, computational experiments and reproducibility issues.

3.1 Case Studies

We evaluate our proposed framework on two IoT applications: smart building and air quality monitoring. Smart cities and buildings are among the most popular IoT applications. In smart cities, a variety of IoT devices are deployed across urban areas, serving as a source of information for managing urban resources, enhancing health services, and providing energy facilities for their citizens [Ameer et al., 2019],[Zhang et al., 2021],[Bekkar et al., 2021]. Smart buildings leverage IoT devices to monitor diverse building characteristics, analyze the data, and generate insights into usage patterns and trends that can be used to optimize the building environment and operations [Lobaccaro et al., 2016].

Accordingly, five public datasets were selected to test our proposed methodology as explained in the following subsections. The input data of each dataset was refined before training the forecasting methods. Therefore, we employ data pre-processing strategies to remove outliers and missing values on datasets presented below.

3.1.1 UCI Appliances energy prediction dataset (AEC)

The first case study is "UCI Appliances energy prediction dataset" (AEC) [Dua and Graff, 2017], which includes measurements of temperature and humidity collected by a Wireless Sensor Network (WSN), weather information from a nearby Weather Station and recorded energy use of appliances and lighting fixtures. The energy appliances data was obtained by continuously measuring (every 10 minutes) a low-energy house in Belgium for 137 days (around 4.5 months). This dataset has 27 variables. Table 1 shows all variables.

Regarding to MISO forecasting, the appliances energy consumption (Wh) measured is the focus of our analysis, then it was chosen as the target variable (endogenous variable) $\mathcal{V}^*$ and the set of explanatory variables (exogenous variable) $\mathcal{V}$ is composed by the remaining variables.

Table 1 – UCI Appliances Energy Consumption dataset (AEC)

Variables	**Description**	**Units**	**Mean**	**St. dev.**
Appliances	Appliances energy consumption	Wh	97.695	102.525
lights	Light energy consumption	Wh	3.802	7.936
T1	Temperature in kitchen area	°C	21.687	1.606
RH_1	Humidity in kitchen area	%	40.260	3.979
T2	Temperature in living room area	°C	20.341	2.193
RH_2	Humidity in living room area	%	40.420	4.070
T3	Temperature in laundry room area	°C	22.268	2.006
RH_3	Humidity in laundry room area	%	39.243	3.255
T4	Temperature in office room	°C	20.855	2.043
RH_4	Humidity in office room	%	39.027	4.341
T5	Temperature in bathroom	°C	19.592	1.845
RH_5	Humidity in bathroom	%	50.949	9.022
T6	Temperature outside the building	°C	7.911	6.090
RH_6	Humidity outside the building	%	54.609	31.150
T7	Temperature in ironing room	°C	20.267	2.110
RH_7	Humidity in ironing room	%	35.388	5.114
T8	Temperature in teenager room 2	°C	22.029	1.956
RH_8	Humidity in teenager room 2	%	42.936	5.224
T9	Temperature in parents room	°C	19.486	2.015
RH_9	Humidity in parents room	%	41.552	4.151
T_out	Temperature outside	°C	7.412	5.317
Press_mm_hg	Pressure	mmHg	755.523	7.399
RH_out	Humidity outside	%	79.750	14.901
Windspeed	Wind Speed	m/s	4.040	2.451
Visibility	Visibility	km	38.331	11.795
Tdewpoint	Dew Point	°C	3.761	4.195
date	Date time stamp	y-m-d hh:mm:ss	-	-

3.1.2 Kaggle smart home with Weather Information dataset (KSH)

"Kaggle Smart home with Weather Information dataset" (KSH) [Kaggle, 2021] is selected as the second case study which is a public dataset of sensor nodes with weather features. The dataset contains the house appliances consumption in kW and weather data, ranging from January 2016 to December 2016 at a frequency of 1 minute (i.e. KSH1 or just KSH). Another scenario was also considered with time resolution equal 10 minutes (i.e. KSH10). The dataset contains 500,910 instances and 29 variables. Table 2 shows all the variables.

The total measured energy consumption (kW) is our target variable $\mathcal{V}^*$ in the MISO prediction problem, and the set of exogenous variables $\mathcal{V}$ is composed of the others variables.

Table 2 – Kaggle Smart Home with Weather Information dataset (KSH)

Variables	**Description**	**Units**	**Mean**	**St. dev.**
use	Total energy consumption	kW	0.859	1.058
gen	Total energy generated by solar resources	kW	0.076	0.128
Dishwasher	Energy consumed by dishwasher	kW	0.031	0.191
Furnace 1	Energy consumed by furnace 1	kW	0.099	0.169
Furnace 2	Energy consumed by furnace 2	kW	0.137	0.179
Home office	Energy consumed in home office	kW	0.081	0.104
Fridge	Energy consumed by fridge	kW	0.064	0.076
Wine cellar	Energy consumed by wine cellar	kW	0.042	0.058
Garage door	Energy consumed by garage door	kW	0.014	0.014
Kitchen 12	Energy consumed in kitchen 1	kW	0.003	0.022
Kitchen 14	Energy consumed in kitchen 2	kW	0.007	0.077
Kitchen 38	Energy consumed in kitchen 3	kW	0.0	0.0
Barn	Energy consumed by barn	kW	0.059	0.203
Well	Energy consumed by well	kW	0.016	0.138
Microwave	Energy consumed by microwave	kW	0.011	0.099
Living room	Energy consumed in living room	kW	0.035	0.096
Solar	Solar power generation	kW	0.076	0.128
temperature	Temperature	°C	50.742	19.114
humidity	Humidity	%	0.664	0.194
visibility	Visibility	km	9.253	1.611
apparentTemperature	Apparent Temperature	°C	48.263	22.028
pressure	Pressure	mmHg	1016.302	7.895
windSpeed	Wind Speed	m/s	6.650	3.983
windBearing	Wind Direction	°	202.357	106.520
dewPoint	Dew Point	°C	0.003	0.011
precipProbability	Precipitation Probability	%	38.694	19.088
time	Date time stamp	ID	0.056	0.166

3.1.3 UCI Household power consumption dataset (HPC)

"UCI Household Power Consumption dataset" (HPC) [Dua and Graff, 2017] contains electricity consumption data from a residential house in France. Electricity consumption was collected by continuous measurements over a four-year period from December 2006 to November 2010 with a resolution of one minute (i.e. HPC1). In this work, in some experiments we changed the time resolution of this dataset, growing it to 30 minutes (i.e. HPC30). The dataset consists of 2,075,259 instances and 12 variables, including 7 explanatory variables and 2 temporal variables. A total of 25,979 missing values were removed in pre-processing. Submetering indicate electricity consumption in the kitchen, laundry room, and for the air conditioner and electric water heater. Table 3 shows all variables.

In regard to MISO problem, the focus of our analysis is the measured global active power (kW), which we have chosen as the endogenous variable $\mathcal{V}^*$ and the set of exogenous variables $\mathcal{V}$ is composed of the past values of six variables.

Table 3 – UCI Household Power Consumption Dataset (HPC)

Variables	**Description**	**Units**	**Mean**	**St. dev.**
date	Date	dd/mm/yyyy	-	-
time	time	hh:mm:ss	-	-
global_active_power	global active power	kW	1.092	1.057
global_reactive_power	global reactive power	kW	0.124	0.113
voltage	averaged voltage	V	240.84	3.24
global_intensity	global current intensity	A	4.628	4.444
sub_metering_1	energy sub-metering No. 1.	kW	1.122	6.153
sub_metering_2	energy sub-metering No. 2	kW	1.299	5.822
sub_metering_3	energy sub-metering No. 3	kW	6.458	8.437

3.1.4 UCI Beijing multi-site air-quality data dataset (AQB)

The "UCI Beijing Multi-Site Air-Quality Data dataset" (AQB) [Dua and Graff, 2017] is the third case study. AQB is a public dataset that includes collections of hourly meteorological and air pollutants data from 12 nationally-controlled air-quality monitoring sites/stations. The meteorological data in each air quality station are matched with the nearest weather station from the China Meteorological Administration. The dataset contains 35,065 instances with multi-features in each station over a four-year period from March 1st, 2013 to February 28th, 2017. This dataset was published by Zhang et al. [Zhang et al., 2017] and all the variable are presented in Table 4.

Table 4 – UCI Beijing Multi-Site Air-Quality Data dataset (AQB)

Variables	**Description**	**Units**	**Mean**	**St. dev.**
year	Year of data	-		-
month	Month of data	-		-
day	Day of data	-		-
hour	Hour of data	-		-
PM2.5	PM2.5 concentration	ug/m^3	81.883	80.461
PM10	PM10 concentration	ug/m^3	109.135	93.707
SO2	SO2 concentration	ug/m^3	22.664	367.549
NO2	NO2 concentration	ug/m^3	59.021	37.092
CO	CO concentration	ug/m^3	1257.172	1223.586
O3	O3 concentration	ug/m^3	120.898	1803.538
TEMP	Temperature	°C	54.203	1091.904
PRES	Pressure	hPa	1043.531	5663.287
DEWP	Dew point temperature	°C	3.235	13.666
RAIN	Precipitation	mm	0.068	0.837
WSPM	Wind speed	m/s	1.718	1.205
station	Site name	-	-	-

3.1.5 UCI Air quality dataset (AQI)

Lastly, the "UCI Air Quality dataset" (AQI) [Dua and Graff, 2017], which is a public dataset containing 9358 samples of hourly averaged values from five metal oxide chemical sensors embedded in an air quality sensing device recorded from March 2004 to February 2005. These IoT devices were located in a significantly contaminated area of an Italian city. The dataset contains 12 variables and De Vito et al. [De Vito et al., 2009] published this dataset as summarized in Table 5.

Table 5 – UCI Italy Air Quality dataset (AQI)

Variables	Description	Units	Mean	St. dev.
Date	Date	DD/MM/YYYY	-	-
Time	Time	HH.MM.SS	-	-
CO(GT)	Concentration CO	mg/m^3	-34.208	77.657
PT08.S1(CO)	Sensor response	tin oxide	1048.990	329.833
NMHC(GT)	Overall NMHC concentration	$microg/m^3$	-159.090	139.789
C6H6(GT)	Benzene concentration	$microg/m^3$	1.866	41.380
PT08.S2(NMHC)	Sensor response	titania	894.595	342.333
NOx(GT)	NOx concentration	ppb	168.617	257.434
PT08.S3(NOx)	Sensor response	tungsten oxide	794.990	321.994
NO2(GT)	NO2 concentration	$microg/m^3$	58.149	126.940
PT08.S4(NO2)	Sensor response	tungsten oxide	1391.48	467.21
PT08.S5(O3)	Sensor response	indium oxide	975.072	456.938
T	Temperature	°C	9.778	43.204
RH	Relative Humidity	%	39.485	51.216
AH	Absolute Humidity	-	-6.838	38.977

3.1.6 Non-stationary analysis

In order to check which time series (multivariate time series) in the datasets are non-stationary, we used the Augmented Dickey-Fuller (ADF) [Cheung and Lai, 1995] and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) [Kwiatkowski et al., 1992] tests with a confidence level of 95%. ADF is used to determine the presence of unit root in the series. The presence of a unit root indicates that the time series is non-stationary. Additionally, the number of unit roots in the series determines how many differencing operations are needed to achieve stationarity. KPSS is used to check for stationarity of a time series around a mean or linear trend or if it is non-stationary due to the presence of a unit root.

Both the KPSS and ADF tests assess stationarity but from opposite viewpoints. The ADF test assumes non-stationarity and tests for evidence to reject this assumption by identifying unit roots. In contrast, the KPSS test assumes stationarity and checks for evidence to refute it by detecting unit roots. Their complementary nature provides a more thorough stationarity analysis. The non-stationarity analysis for each dataset is presented as follows:

- **AEC:** Both ADF and KPSS tests concluded that the *Temperature in ironing room* and *Temperature in parents room* series are non-stationary since there is a gradual concept drift (i.e. incremental mean and variance) in both time series (Figure 8), while *Appliances energy consumption* series is stationary. According to KPSS, the other series are difference stationary (i.e. one differencing is required to make the series stationary).
- **KSH:** The KPSS test concluded that *use*, *Furnace 1*, *Furnace 2*, *Home office*, *Kitchen 12*, *Kitchen 14*, *Kitchen 38*, *temperature*, *apparentTemperature*, *pressure*, *windBearing* and *dewPoint* variables are difference stationary.
- **HPC:** According to KPSS test, all variables are difference stationary.
- **AQB:** For all stations, only the *RAIN* and *TEMP* variables are stationary, then there are non-stationary times series in this dataset.
- **AQI:** According to KPPS test, the following time series are difference stationary *CO(GT)*, *NMHC(GT)*, *PT08.S2(NMHC)*, *NOx(GT)*, *PT08.S3(NOx)*, *NO2(GT)*, *PT08.S4(NO2)* and *T*. Therefore, to make the series stationary one differencing is needed.

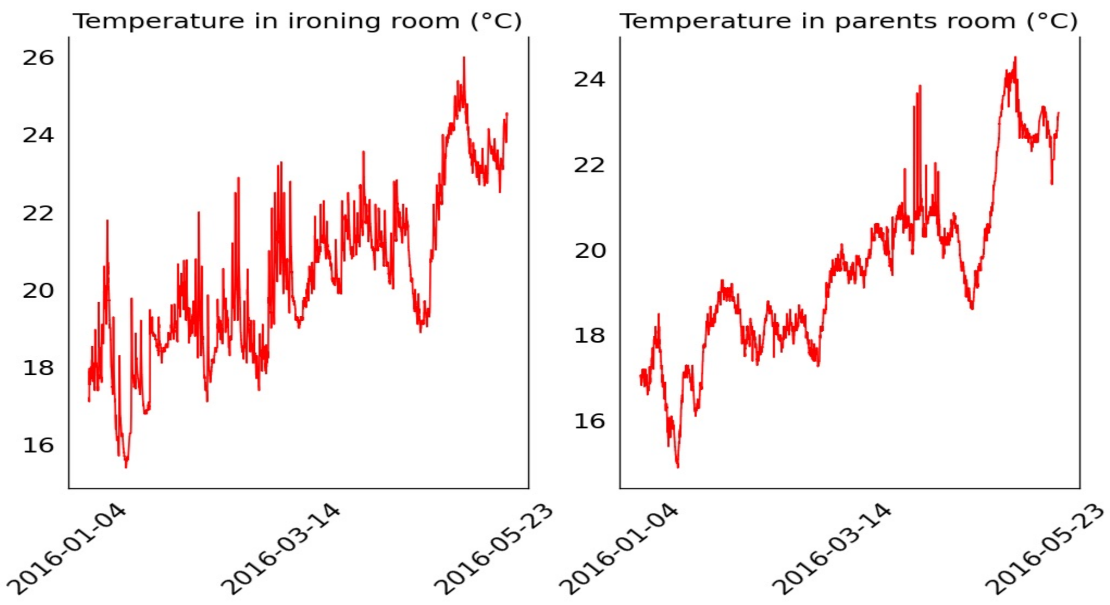

Figure 8 – Gradual concept drift in "Temperature in ironing room" and "Temperature in parents room" time series.

3.1.7 Summary

To showcase the effectiveness of our framework for high-dimensional and MIMO prediction, three scenarios were tested on the AQB dataset:

1. **AQB1**: data from only the Aotizhongxin station was used to predict all features.
2. **AQB6**: data from six stations (Aotizhongxin, Changping, Dingling, Dongs, Guanyuan, and Gucheng) were utilized to predict 66 features.
3. **AQB12**: data from 12 stations were used to forecast 132 features.

These scenarios, labeled as AQB1, AQB6, and AQB12, were referenced throughout the subsequent sections to highlight the model's capabilities across different data complexities.

In addiction, to demonstrate the high ability of our proposed framework to handle different forecast horizons, we also changed the time resolution of some of those datasets (i.e. KSH and HPC), and six scenarios are considered, which are listed in Table 6:

Table 6 – The six scenarios of tests in respect to forecast horizons

Scenario	**Time resolution**
KSH1	1 minute
KSH10	10 minutes
KSH30	30 minutes
HPC1	1 minutes
HPC10	10 minutes
HPC30	30 minutes

3.2 Experiments Methodology

In this work, we separate 75% of data for training set and 25% for testing applying sliding window cross-validation (CV) in the computational experiments. The sliding window is a re-sampling procedure based on splitting the dataset into more than one training and test subsets. The obtained results indicate the model accuracy in terms of accuracy metrics for test subset of data.

Furthermore, this method reduces error by splitting the data into smaller subsets, using each as a testing set in turn instead of a single fixed test set. This approach enhances model robustness and improves generalization to new data by ensuring evaluation on varied samples. Figure 9 shows cross-validation with sliding window.

The number of instances N (i.e. the length of the dataset or the size of the time series) of the variables $v \in \mathcal{V}$ of each dataset $ds_j \in \mathcal{DS}$ were divided into 30 data windows $w \in W$ with N_{test} instances. For each window $ds_{w,j} \in ds_j$, we train the forecasting models $m \in \mathcal{M}$ using the training set, apply the model to the test set and compute forecasting metrics over the test set. Thus, each model has 30 experiments and we evaluate the

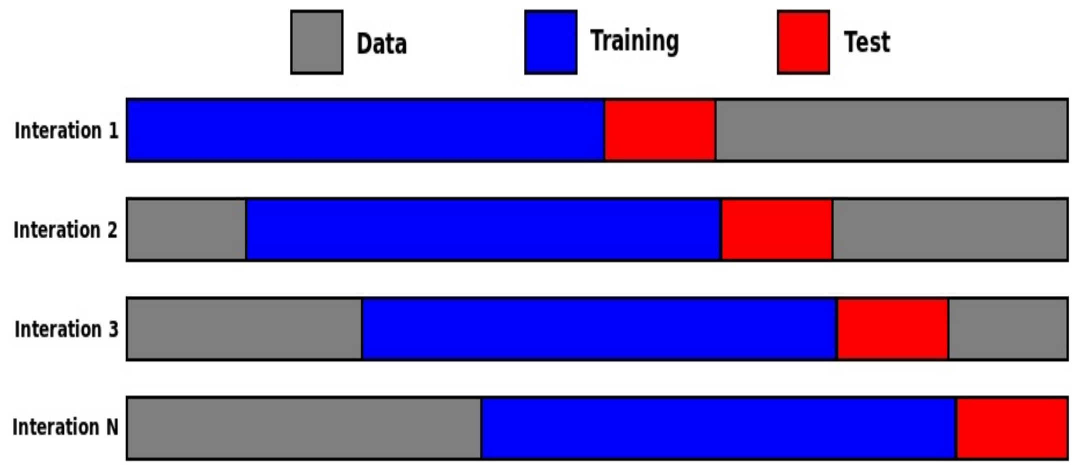

Figure 9 – Schematic of the sliding window cross-validation.

accuracy of the proposed methodology from the average and standard deviation error value measured in all windows used for forecasting in the experiments.

3.2.1 Performance metrics

In order to assess the accuracy of each model, two statistical metrics are utilized according to the following equations. Let $\hat{y}_i$ as the predicted values at time t and y_i as the observed values.

- Root Mean Squared Error (RMSE), defined by equation 3.1. It shows how accurate the forecasting model is, as it compares the difference between the predicted value and the real value (error), returning the standard deviation of this difference.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}} \tag{3.1}$$

- Mean Absolute Error (MAE), defined by equation. It demonstrates the percentage difference between predicted values.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{3.2}$$

- Normalized Root Mean Squared Error (NMRSE) by the difference between maximum and minimum, defined by equation

$$NRMSE = \frac{RMSE}{Y_{\max} - Y_{\min}} \tag{3.3}$$

- Fractional Bias (FB), a value close to -2 indicates under-prediction, while a value close to 2 designates over-prediction, and a value close to 0 stands for the optimal prediction

$$FB = 2\frac{\overline{Y} - \overline{\hat{Y}}}{\overline{Y} + \overline{\hat{Y}}} \tag{3.4}$$

3.2.2 Skill score index

We also evaluate the accuracy of our proposed methodology using the Skill Score Index. The skill score defines the difference between the forecast and the reference forecast:

$$SkillScore = 1 - \frac{M_F}{M_R} \tag{3.5}$$

where M_F refers to the value of the metric for the forecasting method and M_R is the value of the metric for the reference method. The Skill Score can be used not only to compare with a naive model, but also to compare different forecasting methods with each other Voyant et al. [2017]. For example, a skill score of 0.50 indicates a 50% improvement in accuracy over the competing model. A negative score indicates worse performance than the competitor model.

3.2.3 Parsimony and explainability

We assess the parsimony and explainability of our proposed methodology. Parsimony, based on Occam's razor, favors the simplest model that achieves reliable forecasting for non-stationary high-dimensional time series. While simpler forecasting models are preferred, a balance must be struck between simplicity and accuracy.

Minimizing forecasting model complexity is essential, as it directly affects computational cost and training time. This is particularly crucial for IoT applications, where models must run on resource-constrained edge devices. Additionally, a model's parsimony is closely linked to its explainability.

To assess the parsimony of our approach, we measure the number of fuzzy rules. While increasing fuzzy rules enhances the accuracy of FTS models, it introduces a trade-off - higher accuracy reduces parsimony and explainability, making the model more complex and harder to interpret. However, the explainability of FTS models depends on the active fuzzy rules and their length. Furthermore, we also evaluate the time complexity of our proposed methodology using the Big $\mathcal{O}$ notation.

For deep learning methods, we evaluate parsimony and explainability by counting the number of trainable parameters in the neural networks.

3.3 Computational Experiments and Reproducibility

All the experiments (including proposed methodology and baseline models) were implemented and tested with Python 3 using open source libraries including Scikit-Learn [Pedregosa et al., 2011], Keras [Chollet et al., 2015], Tensorflow [Abadi et al., 2015], PyTorch [Paszke et al., 2019], StatsModels [Seabold and Perktold, 2010] and pyFTS [Silva, 2016]. For the PyTorch models, the seed generator was set to 1 in order to optimize the hyper parameters and improve the reproducibility of the experiments.

3.3.1 A Fuzzy Time Series Library for Python (pyFTS)

To ensure transparency and reproducibility, all proposed models are available in the open-source Python library, pyFTS, developed at the Machine Intelligence and Data Science (MINDS) laboratory at the Federal University of Minas Gerais (UFMG). The models can be accessed via the pyFTS.models package, which extends conventional models as outlined in previous publications [Silva, 2016]. The architecture of the pyFTS library is shown in Figure 10

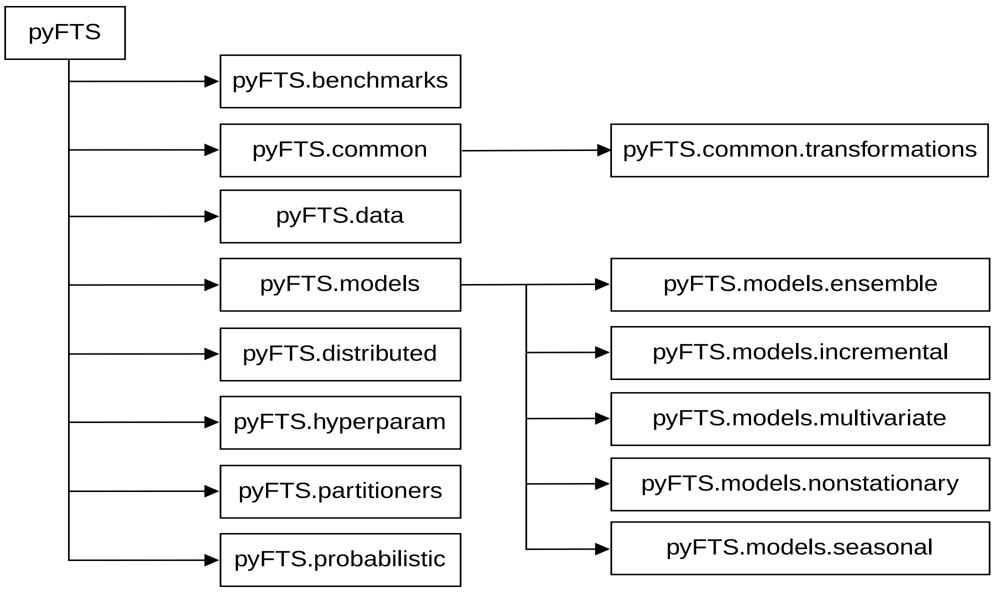

Figure 10 – Architecture of the pyFTS library packages [Lucas et al., 2022]

Chapter 4

Embedding-based Fuzzy Time Series (EFTS)

In this chapter, we describe our MISO high-dimensional non-stationary time series forecasting methodology named EFTS (Embedding-based Fuzzy Time Series) and published in [Bitencourt et al., 2023]. Accordingly, EFTS is a data-driven and explainable approach that consists of embedding, training and forecasting procedures.

4.1 Embedding Procedure

The embedding procedure generates a new variable space that more effectively represents high-dimensional time series data for forecasting task. In this approach, we focuses on two embedding methods: PCA detailed in Section 2.4.1 and Autoencoders (see Section 2.4.3).

4.2 Training Procedure

In the training procedure, a multivariate $embFTS$ model is generated to catch all the patterns in the embedding time series $Y_{emb} \in \mathbb{R}^K$ with its individual instances $y_{emb}(t) \in Y_{emb}$ for $t = 0, 1, ..., T$. Given the number of fuzzy sets κ, the training procedure, despised in Figure 11, includes the following steps:

1. **Partitioning**: we define $U_{\mathcal{V}_{emb}} = [lb, ub]$ where $lb = \min(Y_{emb}) - D_1$ and $ub = \max(Y_{emb}) + D_2$, with $D_1 = r \times |\min(Y_{emb})|$ and $D_2 = r \times |\max(Y_{emb})|$, $0 < r < 1$. Then, we extrapolate the known bounds of the variables $\mathcal{V}_{emb_i}$ as a security margin.

2. **Defining the linguistic variable**: we split $U_{\mathcal{V}_{emb}}$ in κ_i overlapping intervals U_j with midpoints c_j for $j = 0, 1, 2, ..., \kappa_i$. For each interval $U_j \in U_{\mathcal{V}_{emb}}$, we create an overlapping fuzzy set $\mathcal{A}_j^{\mathcal{V}_i}$ with the membership function $\mu_{\mathcal{A}_j}^{\mathcal{V}_i}$ and generate a variable

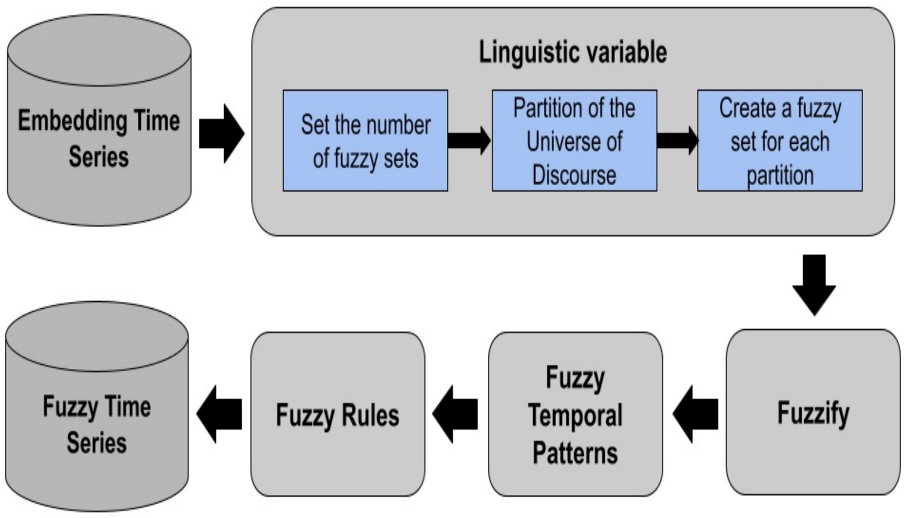
Embedding Time Series
Linguistic variable
Set the number of fuzzy sets
Partition of the Universe of Discourse
Create a fuzzy set for each partition
Fuzzify
Fuzzy Temporal Patterns
Fuzzy Rules
Fuzzy Time Series

Figure 11 – EFTS Training procedure [Bitencourt et al., 2023]

$\mathcal{V}_{emb_i}$ and a linguistic variable $\widetilde{\mathcal{V}}_{emb_i} \in \widetilde{\mathcal{V}}_{emb}$ where each fuzzy set is a linguistic variable.

3. **Fuzzification**: we change Y_{emb} into F_{emb} (embedding fuzzy time series) where each data point is an $n \times \kappa$ array with the fuzzified values with respect to the linguistic variable with the fuzzy membership defined as follows

$$f(t) \leftarrow \left\{ \mathcal{A}_j^{\mathcal{V}_i} \mid \mu_{\mathcal{A}_j}^{\mathcal{V}_i}(Y_{emb}(t)^i) > 0 \right\} \tag{4.1}$$

4. **Generate the temporal patterns**: we create temporal patterns with the format $\mathcal{A}_j^{\mathcal{V}_0}, ..., \mathcal{A}_j^{\mathcal{V}_n} \rightarrow \mathcal{A}_j^{\mathcal{V}*}$, where the precedent or LHS is $f_{emb}(t) = \mathcal{A}_j^{\mathcal{V}_i}$ and the consequent or RHS is the target variable such that $f_{emb}(t+1) = \mathcal{A}_j^{\mathcal{V}^*}$. Both LHS and RHS are related to $\mathcal{A}_j^{\mathcal{V}_i}$ with maximum membership.

5. **Generate the rule base**: finally, each pattern represents a fuzzy rule and they are grouped by their same precedents, creating a fuzzy rule r with the format $LHS \rightarrow RHS$. Each fuzzy rule indicates set of possible outcomes at time $t+1$ where a specific precedent is recognized from the past.

4.3 Forecasting Procedure

In the forecasting procedure, we discover the fuzzy rules that match a given fuzzified input and uses them to calculate a forecast value. Accordingly, we estimate $\mathcal{V}^*(t+1)$ for a given input instance and using the fuzzy rules of the *embFTS* model.

Given the Y_{emb}, the following steps are taken to forecast $\mathcal{V}^*(t+1)$.

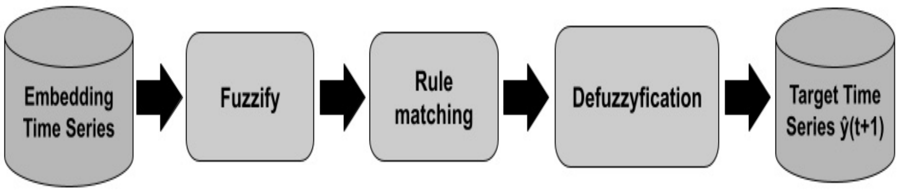

Figure 12 – EFTS Testing procedure [Bitencourt et al., 2023]

1. **Fuzzification**: for each variable, we fuzzify the embedding time series according to equation 4.1.

2. **Rule matching**: we select r fuzzy rules whether any fuzzy set of $f(t)$ is equal to LHS, then we compute the rule fuzzy membership grade using the minimum function T-norm as follows

$$\mu_q = \bigcap_{j \in \widetilde{\mathcal{V}}_i \ ; \ i \in \mathcal{V}} \mu_j i \tag{4.2}$$

3. **Rule mean points**: for each rule q, we compute the mean point mp_q of $\mathcal{V}^*$ according the following equation

$$mp_q = \sum_{j \in \widetilde{\mathcal{V}}_i^*} c_j \tag{4.3}$$

where c_j is the c parameter of the membership function from the fuzzy sets.

4. **Defuzzification**: finally, $\mathcal{V}^*(t+1)$ is obtained as the weighted sum of the rule midpoints by their membership grades μ_{A_j}, according to equation:

$$\hat{y}(t+1) = \sum_{q \in r} \mu_q \cdot mp_q \tag{4.4}$$

4.4 Embedding Weighted Multivariate Fuzzy Time Series (EWMVFTS)

To demonstrate our proposed methodology, in [Bitencourt et al., 2023] we extend the Weighted Multivariate Fuzzy Time Series (WMVFTS) [de Lima Silva et al., 2019], to enable it for high-dimensional time series. We used the embedding transformation presented in section 4.1 to reduce the dimensionality of the time series and enable efficient pattern recognition and induction of fuzzy rules. WMVFTS combined with PCA and AE are referred to as PCA-WMVFTS and AE-WMVFTS, respectively.

The training procedure is presented in Algorithm 1 and illustrated in Figure 13. In the **Generate the rule base** step, we create fuzzy rules r with the format $LHS \rightarrow wg \cdot RHS$, where $wg = |\mathcal{A}_j^{\mathcal{V}_i}|/|RHS|$ which are normalized frequencies of each temporal pattern (i.e. weights) according to equation

$$wg_i = \frac{\#\mathcal{A}_j^{\mathcal{V}^*}}{\#RHS} \quad \forall \mathcal{A}_j^{\mathcal{V}^*} \in RHS \tag{4.5}$$

where $\#\mathcal{A}$ is the number of occurrences of $\mathcal{A}_i$ on temporal patterns with the same LHS and $\#RHS$ is the total number of temporal patterns with the same precedent LHS. The other steps remain the same as described in Section 4.2.

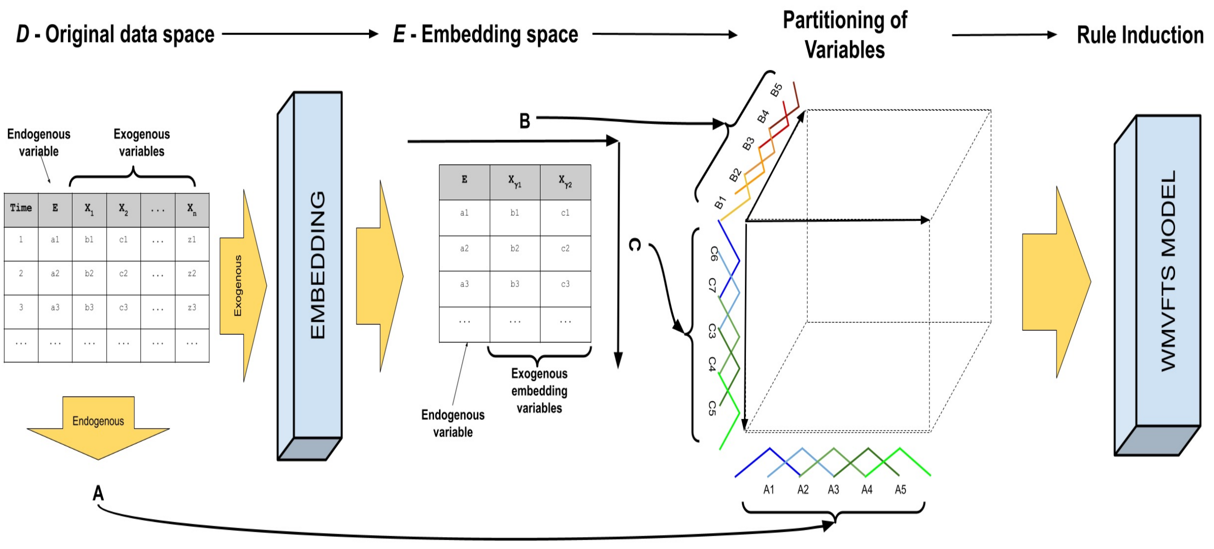

Figure 13 – EWMVFTS training for number of dimensions $k = 2$ and number of fuzzy sets $\kappa = 5$ [Bitencourt et al., 2023]

Algorithm 2 shows the forecasting procedure. In **Rule mean points** step, for each rule q, we compute the mean point mp_q of the endogenous variable $\mathcal{V}^*$ as follows

$$mp_q = \sum_{j \in \widetilde{\mathcal{V}}_i^*} wg_j \cdot c_j \tag{4.6}$$

where wg_j is the weights calculated according to 7.2. The other steps remain the same as presented in Section 4.3.

4.4.1 EWMVFTS Hyperparameters

In EWMVFTS, the most significant hyperparameters are the embedding dimensions (K) and the number of fuzzy sets (κ). Performance can be enhanced by simply increasing these hyperparameter values. In this sense, a combination of these HP was tested

Algorithm 1: EWMVFTS Training

```
input  : A multivariate time series $Y$, number of instances $N$, number of dimensions $K$,
         number of fuzzy sets $\kappa$
output : the linguistic variable $\widetilde{\mathcal{V}}$, the rule set $\mathcal{R}$
foreach $y(t) \in Y$ do
    $y_{emb}(t) \leftarrow$ Embedding Procedure($y(t)$, $K$);
    $Y_{emb} \leftarrow y_{emb}(t)$
end
for $i \leftarrow 1 \ldots k$ do
    $u \leftarrow$ Split $UoD$ in $\kappa$ overlapping intervals;
    $\mathcal{V}_i \leftarrow$ Create a variable for the dimension $i$;
    $\widetilde{\mathcal{V}} \leftarrow$ Create an empty linguistic variable for $\mathcal{V}_i$;
    for $j \leftarrow 1 \ldots N_i$ do
        $\mathcal{A}_j^{\mathcal{V}_i} \leftarrow$ Create a fuzzy set with the interval $u_j$ with membership function $\mu_{\mathcal{V}_i}$;
        $\widetilde{\mathcal{V}} \leftarrow \mathcal{A}_j^{\mathcal{V}_i}$;
    end
end
$F \leftarrow$ Create empty fuzzified data;
foreach $y_{\mathcal{E}}(t) \in Y_{\mathcal{E}}$ do
    $f(t) \leftarrow \{A_j^{\mathcal{V}_i} \mid \mu_{A_j^{\mathcal{V}_i}}(y_{\mathcal{E}}(t)^i) > 0\}$ for all fuzzy sets $A_j^{\mathcal{V}_i} \in \tilde{\mathcal{V}}\rangle$ and all variables $\mathcal{V}_i$;
    $F \leftarrow f(t)$;
end
$\mathcal{R} \leftarrow$ Create an empty set of rules;
foreach $f(t), f(t+1) \in F$ do
    foreach fuzzy set $LHS \in f(t)$ do
        $RHS \leftarrow$ all fuzzy sets $A_j^{\mathcal{V}_i}$ in $f(t+1)$ with weight $wg = |A_j^{\mathcal{V}_i}|/|RHS|$;
        $r \leftarrow$ Create rule $LHS \rightarrow RHS$;
        if $r$ does not exist in $\mathcal{R}$ then
            $\mathcal{R} \leftarrow r$;
        end
    end
end
```

empirically to catch the best values, varying these HP as following: $\kappa \in \{10, 20, 30, 40, 50\})$ and $K \in \{2, 3, 4, 5\}$. Table 7 shows the obtained mean accuracy results for 30 experiments. Our proposed methods shows similar results and reached high forecast accuracy with $K \in \{3, 4\}$ and $\kappa \in \{40, 50\}$.

4.4.1.1 Autoencoder's hyperparameters

We empirically tested seven hyperparameters: Epochs, Stack Size, Optimizer, Learning Rate, Kernel Initializer, Number of Neurons, and Activation Function. Each hyperparameter was assessed using a two-layer AutoEncoder architecture that included $l1$ regularization in both the encoder and decoder layers. Due to the random initialization of weights, results may vary slightly with each program run. To achieve a more accurate estimation for each hyperparameter set, we recorded the mean RMSE values from 10 forecast

Algorithm 2: EWMVFTS Forecasting

```
input  : A sample $y(t)$, the dimensions $K$, the linguistic variable $\widetilde{\mathcal{V}}$, the rule set $\mathcal{R}$
output : target forecasting value $\hat{y}(t+1)$
$y_{emb}(t) \leftarrow$ Embedding Procedure($y(t)$, $K$);
$f(t) \leftarrow \{A_j^{\mathcal{V}_i} \mid \mu_{A_j^{\mathcal{V}_i}}(y_{\mathcal{E}}(t)^i) > 0\}$ for all fuzzy sets $A_j^{\mathcal{V}_i} \in \tilde{\mathcal{V}}\rangle$ and all variables $\mathcal{V}_i$;
$\mathcal{R}_{fired} \leftarrow$ Create an empty set of rules;
$\mu \leftarrow$ Create the empty vector of activation of each rule;
foreach $r \in \mathcal{R}$ do
    if any fuzzy set of $f(t)$ is equal to $r_{LHS}$ then
        $\mathcal{R}_{fired} \leftarrow r$;
    end
end
foreach $r \in \mathcal{R}_{fired}$ do
    $mp_r \leftarrow \sum_{A_j^{\mathcal{V}_i} \in r_{RHS}} wg_i \cdot mp_i$ where $wg_i$ is the weight and $mp_i$ is the midpoint of the
      fuzzy set;
    $\mu_r \leftarrow \bigcap_{A_j^{\mathcal{V}_i} \in r_{LHS}} \mu_{A_j^{\mathcal{V}_i}}(y(t))$
end
$\hat{y}(t+1) = \frac{\sum_{r \in \mathcal{R}_{fired}} \mu_r \cdot mp_r}{\sum_{r \in \mathcal{R}_{fired}} \mu_r}$
```

Table 7 – PCA-WMVFTS and AE-WMVFTS accuracy with different number of K (DIM) and κ (FS) in terms of RMSE [Bitencourt et al., 2023]

		PCA-WMVFTS			AE-WMVFTS		
DIM	FS	AEC	KSH10	HPC30	AEC	KSH10	HPC30
2	10	28.957 ± 11.254	0.424 ± 0.390	0.549 ± 0.139	26.547 ± 13.601	0.439 ± 0.360	0.556 ± 0.105
2	20	17.095 ± 8.610	0.310 ± 0.310	0.511 ± 0.181	18.357 ± 22.659	0.313 ± 0.232	0.430 ± 0.093
2	30	9.859 ± 9.294	0.266 ± 0.259	0.469 ± 0.184	12.663 ± 16.707	0.289 ± 0.269	0.396 ± 0.113
2	40	8.585 ± 9.883	0.193 ± 0.168	0.410 ± 0.196	4.572 ± 3.545	0.269 ± 0.245	0.352 ± 0.134
2	50	5.457 ± 7.305	0.169 ± 0.135	0.366 ± 0.202	2.871 ± 3.021	0.261 ± 0.246	0.369 ± 0.142
3	10	16.988 ± 11.948	0.275 ± 0.238	0.517 ± 0.132	12.569 ± 9.164	0.386 ± 0.320	0.528 ± 0.099
3	20	4.117 ± 3.082	0.143 ± 0.124	0.368 ± 0.146	3.793 ± 4.638	0.289 ± 0.287	0.317 ± 0.080
3	30	2.060 ± 2.634	0.102 ± 0.126	0.276 ± 0.143	1.580 ± 2.720	0.212 ± 0.180	0.206 ± 0.073
3	40	0.996 ± 1.267	0.055 ± 0.048	0.204 ± 0.130	0.328 ± 0.461	0.169 ± 0.120	0.149 ± 0.068
3	50	0.348 ± 0.510	0.040 ± 0.064	0.162 ± 0.117	0.523 ± 1.180	0.124 ± 0.138	0.093 ± 0.069
4	10	10.451 ± 16.059	0.147 ± 0.107	0.421 ± 0.135	3.789 ± 4.669	0.358 ± 0.338	0.422 ± 0.136
4	20	1.495 ± 2.394	0.045 ± 0.034	0.236 ± 0.116	0.801 ± 1.377	0.208 ± 0.214	0.133 ± 0.077
4	30	0.415 ± 0.775	0.015 ± 0.019	0.138 ± 0.098	0.229 ± 0.774	0.119 ± 0.099	0.065 ± 0.051
4	40	0.187 ± 0.480	0.006 ± 0.009	0.084 ± 0.074	0.073 ± 0.168	0.077 ± 0.133	0.021 ± 0.029
4	50	0.083 ± 0.228	0.003 ± 0.006	0.061 ± 0.065	0.060 ± 0.175	0.041 ± 0.047	0.019 ± 0.029
5	10	6.006 ± 16.138	0.081 ± 0.049	0.371 ± 0.118	5.175 ± 20.992	0.330 ± 0.306	0.253 ± 0.106
5	20	0.587 ± 1.648	0.012 ± 0.013	0.174 ± 0.092	0.091 ± 0.218	0.153 ± 0.207	0.053 ± 0.064
5	30	0.071 ± 0.194	0.003 ± 0.004	0.089 ± 0.060	0.044 ± 0.155	0.051 ± 0.110	0.013 ± 0.025
5	40	0.042 ± 0.157	0.001 ± 0.002	0.049 ± 0.043	0.040 ± 0.150	0.016 ± 0.017	0.005 ± 0.008
5	50	0.044 ± 0.151	0.000 ± 0.001	0.028 ± 0.028	0.039 ± 0.150	0.010 ± 0.011	0.001 ± 0.005

iterations. Table 8 shows the values tested for each hyper-parameter with autoencoder and the values that obtained the best result are shown in boldface.

Table 8 – Hyperarameters tested for the AutoEncoder. The best configuration is shown in boldface [Bitencourt et al., 2023].

Hyperparameter	**Values**
Epochs	10, 50, 100, **150**
Stack Size	10, 20, 32, 64, 80, 120
Optimizer	**Adam**, SGD, RMSprop, AdaDelta, AdaGrad, AdaMax, NAdam
Learning Rate	0.001, 0.01, **0.1**, 0.2, 0.3
Kernel Initializer	leCun Uniform, **Normal**, Glorot Normal, Glorot Uniform, He Normal, He Uniform
Neurons	10, 15, 20, 30, 40, 50, **80**, 100
Activation	Linear, Softmax, Softplus, Softsign, **Tanh**, Sigmoid, Hard Sigmoid, ReLU

4.5 Results

This section presents the experimental results of our proposed models over several forecasting methods tested on five datasets. First, we provide a brief introduction of the baseline models as well as HP setting. Second, we compare the forecast results of our models with the baseline methods. Third, we compare our model's accuracy against several state-of-the-art forecasting models described in the literature review (Section 2). Fourth, we discuss the parsimony, computational cost and explainability of our methods. Finally, a discussion of the limitations and computational complexity of our proposed methods is presented.

4.5.1 Baseline models

We compared the performance of our proposed approach with the following baseline models: SARIMAX, PCA-SARIMAX, LSTM [Hochreiter and Schmidhuber, 1997], GRU [Chung et al., 2014], Temporal Convolutional Network (TCN) [Bai et al., 2018] and naive model, which is a reference technique that assumes that $y(t)$ equals $y(t-1)$.

In SARIMAX, the selection of seasonal (P, D, Q) and non-seasonal (p, d, q) components was based on the Akaike Information Criterion (AIC) using the `auto_arima` function from the pmdarima library [Smith et al., 2017–] and autocorrelation (ACF) and partial autocorrelation (PACF) plots. Besides, using the PCA algorithm, we transform M features of each dataset into $K = 3$ features and apply the SARIMAX model (henceforth called PCA-SARIMAX).

The LSTM and GRU architecture used consists of two layers and kernel regularizers, and the configuration of hyperparameters were chosen using hyperopt [Bergstra et al., 2013], which is a library designed for hyper-parameter tuning. Several combinations were tested and we used the best one as baseline. The Optuna optimization framework [Akiba et al., 2019] was employed to find the best HP values of the TCN method. A pre-defined range of parameters was tested for each dataset, and the combination that yielded the best result was chosen for the experiments.

4.5.2 EFTS versus baseline methods

In this subsection, we compare the forecast accuracy of our proposed models with the baseline methods. For all accuracy results presented in this subsection, the number of fuzzy sets κ was 50 and the number of K dimensions was 3 for our models. Table 9 presents the results of RMSE for each baseline model, as well as the accuracy metrics results for PCA-WMVFTS and AE-WMVFTS proposed models.

Table 9 – Proposed and baseline methods accuracy on the datasets in terms of RMSE(Wh) [Bitencourt et al., 2023].

	AEC	KSH1	KSH10	HPC1	HPC30
PCA-WMVFTS	**0.348 ± 0.51**	**0.024 ± 0.015**	**0.04 ± 0.064**	**0.067 ± 0.031**	0.069 ± 0.038
AE-WMVFTS	0.523 ± 1.18	0.101 ± 0.067	0.124 ± 0.138	0.095 ± 0.028	0.093 ± 0.069
PCA-SARIMAX	158.768 ± 69.073	0.587± 0.347	0.995 ± 0.751	0.258 ± 0.064	0.901 ± 0.211
SARIMAX	172.283 ± 70.574	0.663± 0.494	1.316 ± 1.816	0.26 ± 0.063	0.901 ± 0.211
NAIVE	64.749 ± 28.836	0.846 ± 0.719	0.846 ± 0.719	0.256 ± 0.065	0.899 ± 0.212
LSTM	70.544 ± 33.058	0.395 ± 0.249	0.379 ± 0.249	0.255 ± 0.061	0.260 ± 0.071
GRU	68.810 ± 31.477	0.365 ± 0.219	0.365 ± 0.226	0.255 ± 0.061	0.255 ± 0.061
RNN	68.353 ± 31.568	0.384 ± 0.268	0.383 ± 0.271	0.287 ± 0.076	0.260 ± 0.072
TCN	89.210 ± 41.911	0.410 ± 0.202	0.353 ± 0.224	0.075 ± 0.031	**0.061 ± 0.020**

Comparing the results with those obtained by baseline models, it is clear that our models outperform them on AEC, KSH1, KSH10, HPC1 datasetss, outputting a consistent and accurate prediction. However, TCN showed the smallest prediction error compared to all forecasting methods on HPC30 dataset, and TCN is superior than AE-WMVFTS on HPC1 dataset. Nevertheless, PCA-WMVFTS outperforms TCN on HPC1 dataset, presenting equally good results in all the datasets that were used.

Furthermore, we use RMSE metric to evaluate the accuracy $\epsilon[m_i, ds_j, w]$ of each prediction method. We conducted Kruskal-Wallis test [Kruskal and Wallis, 1952] with the significance level of $\alpha = 0.01$.

Given the mean $\mu_{m_i,ds_j} = \frac{\sum_{w=1}^{W} \epsilon[m_i,ds_j,w]}{W}$, Kruskal-Wallis aims to test the hypothesis where the null hypothesis (H_0) stands for the equality of means (μ_{m_i,ds_j}) of the forecasting methods, then we cannot distinguish between the methods. In contrast, alternative hypothesis (H_1) indicates that at least one of the means (μ_{m_i,ds_j}) of forecasting models is not equal.

When the H_0 is rejected, we compared the equality of each pair of means applying *post hoc* tests using the Wilcoxon test [Wilcoxon, 1945] such that

$$H_0 : \mu_{m_a,ds_j} < \mu_{m_b,ds_j}$$

$$H_1 : \mu_{m_a,ds_j} \geq \mu_{m_b,ds_j}$$

Table 10 shows the test statistics obtained and the p-values at the confidence level for the Kruskal-Wallis test. The null hypothesis H_0 has been rejected for the all datasets, therefore the model's accuracy are not statistically equivalent.

Table 10 – Kruskal-Wallis Statistical Tests [Bitencourt et al., 2023].

dataset	Statistic	p-value	Result
AEC	196.021	4.399e-38	H0 rejected
KSH1	164.462	1.863e-31	H0 rejected
KSH10	172.077	4.731e-33	H0 rejected
HPC1	168.399	2.791e-32	H0 rejected
HPC30	225.393	2.790e-44	H0 rejected

To compare all forecasting methods against each others, we employed the Wilcoxon test. Table 11 presents the summary of statistical ranking of the forecasting accuracy, which means how many times we failed to reject H_0 at confidence level. For instance, first rank belongs to PCA-WMVFTS on HPC1 dataset since we failed to reject H_0 in all cases when we tested PCA-WMVFTS against the other forecasting models, then its accuracy error is statistically smaller then other methods. The complete Wilcoxon test results are available in the supplementary materials (Section A).

Table 11 – The summary of the ranking of the forecasting methods [Bitencourt et al., 2023].

Method	AEC	KSH1	KSH10	HPC1	HPC30
PCA-WMVFTS	1	1	1	1	1
AE-WMVFTS	1	2	2	2	1
PCA-SARIMAX	8	7	8	5	8
SARIMAX	8	7	8	8	8
NAIVE	3	9	7	4	7
LSTM	5	3	4	4	4
GRU	3	3	3	4	4
RNN	3	3	3	9	4
TCN	7	3	3	1	1

PCA-WMVFTS reached the first rank in all datasets followed by AE-WMVFTS, except HPC for which TCN is statistically equivalent to our proposed methods. Thus it is also clear that our methodology outperforms the baseline forecasting methods. It can be seen from the results above that, compared to the baseline methods, our models achieve optimal prediction accuracy on all datasets, specially PCA-WMVFTS, showing that they output good results in both very high dimensional data, such as AEC and KSH, and moderate dimensional data, such as HPC, reinforcing the model's capacity to make excellent predictions for different data.

4.5.3 PCA-WMVFTS versus AE-WMVFTS

According to results presented in the Sections above, PCA-WMVFTS is slightly superior than AE-WMVFTS, but not significantly. To confirm that PCA-WMVFTS prediction error is statistically smaller then AE-WMVFTS, we test the forecast performance of both our methods over different configurations of K dimensions (i.e. $K \in \{2, 3, 4, 5\}$) and κ fuzzy sets (i.e. $\kappa \in \{10, 20, 30, 40, 50\}$). In addiction, we applied the Wilcoxon test and the hypothesis tests results are shown in Table 12.

Table 12 – Wilcoxon Statistical Tests [Bitencourt et al., 2023].

dataset	Statistic	p-value	Result
AEC	209637.0	2.65e-07	H_0 rejected
HPC30	212755.5	2.41e-08	H_0 rejected
KSH10	113873.0	1.0	Fail to Reject H_0

The hypothesis null H_0 has been failed to reject at confidence level for KSH10, hence it confirms that there is a significant difference between the two methods, in other words, PCA-WMVFTS is better than AE-WMVFTS. Nevertheless, H_0 has been rejected for the AEC and HPC30 datasets which means the our model's accuracy are statistically equivalent. Both embedding methods achieved similar performance on these datasets, reinforcing the results shown in Table 11.

It is necessary to highlight that we selected only three datasets since it takes a considerable time to train/test all the combinations of K and κ on HPC1 and KSH1 datasets. The thorough results can be accessed publicly through the following link address in supplementary materials (Annex A)

4.5.4 EFTS versus competitors methods

In this subsection, we compare the accuracy of the proposed models ($K = 3$ and $\kappa = 50$) with several forecasting models detailed in the literature review, which are the following: GBM with feature selection (the best model in [Candanedo et al., 2017]); MPL with feature selection [Chammas et al., 2019]; CNN-GRU [Sajjad et al., 2020]; HSBUFC

[Syed et al., 2021], AIS-RNN [Munkhdalai et al., 2019]; M-BDLSTM [Ullah et al., 2020]; CNN-LSTM [Kim and Cho, 2019].

It is worth noting that each one of these models used a different experimental methodology, either in terms of cross-validation methodology or train/test split. In those terms, it is counterproductive to perform a different experimental set up for each competitor model and this research used their published results as they are, to compare with our experimental results.

Figure 14 and Figure 15 shows the results of RMSE and MAE for the competitors methods and our models on AEC and HPC (HPC1) datasets, respectively.

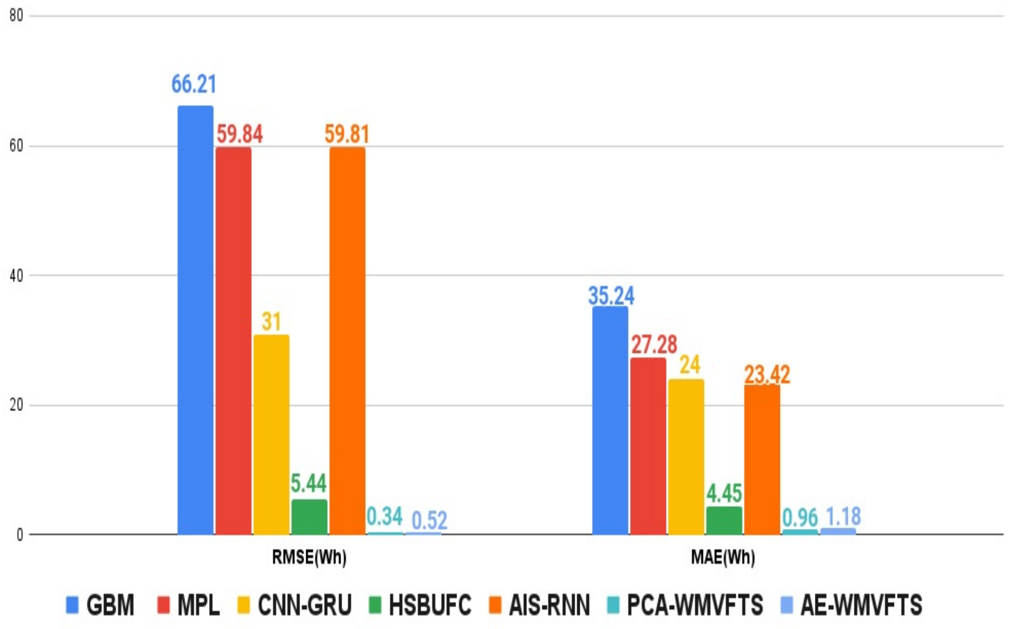

Figure 14 – Models accuracy on AEC dataset in terms of RMSE(Wh) and MAE(Wh) [Bitencourt et al., 2023]

According to Figure 14, PCA-WMVFTS is superior than all competitors models with RMSE of 0.34Wh, a MAE of 0.96Wh. AE-WMVFTS also showed a high accuracy compared to the competitors models. However, our proposed methods are inferior than HSBUFC on HPC1, but not significantly, and our forecast models outperform the other competitors methods.

Table 13 presents the skill score index of PCA-WMVFTS with AE-WMVFTS with respect to some competitors methods and the accuracy metric selected was the RMSE.

For the AEC dataset, our models significantly outperformed the competitor methods, with RMSE improvements exceeding 98% compared to GBM, MLP, CNN-GRU, and AIS-RNN. Specifically, PCA-WMVFTS and AE-WMVFTS showed a 94% and 90% improvement over HSBUFC, respectively.

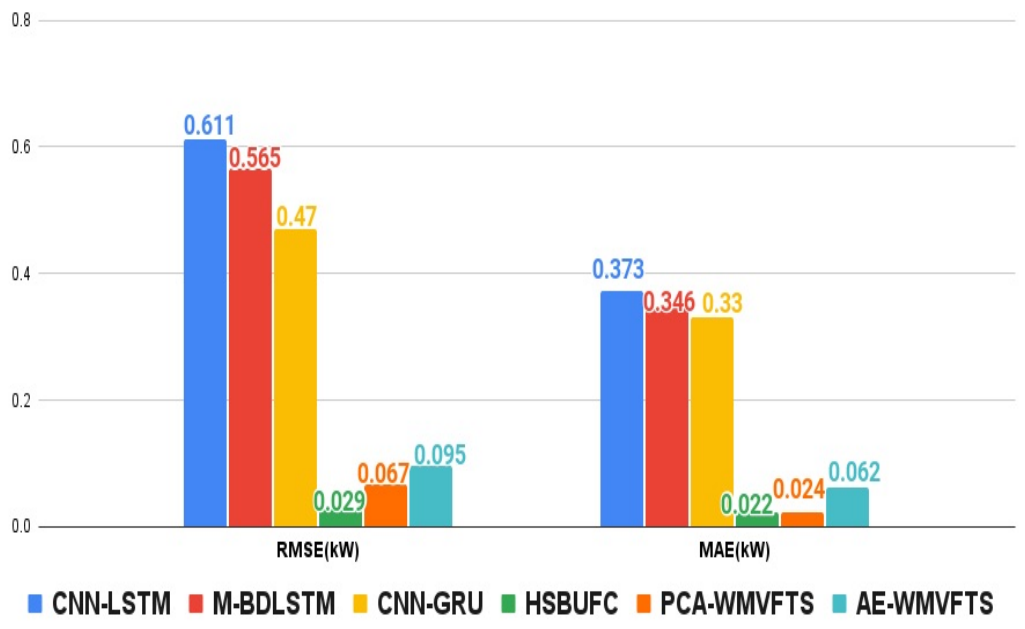

Figure 15 – Models accuracy on HPC (HPC1) dataset in terms of RMSE(Wh) and MAE(Wh) [Bitencourt et al., 2023]

Table 13 – Skill score of PCA-WMVFTS and AE-WMVFTS with respect to competitors methods on AEC and HPC1 datasets [Bitencourt et al., 2023].

dataset	Methods	PCA-WMVFTS	AE-WMVFTS
AEC	GBM	0.99	0.99
AEC	MPL	0.99	0.99
AEC	CNN-GRU	0.99	0.98
AEC	HSBUFC	0.94	0.90
AEC	AIS-RNN	0.99	0.99
HPC1	CNN-LSTM	0.89	0.84
HPC1	M-BDLSTM	0.88	0.83
HPC1	CNN-GRU	0.86	0.80
HPC1	HSBUFC	-1.31	-2.28

For the HPC dataset, our models demonstrated improvements over CNN-LSTM, M-BDLSTM, and CNN-GRU. AE-WMVFTS showed over an 80% improvement. PCA-WMVFTS outperformed CNN-GRU by 86% and surpassed CNN-LSTM and M-BDLSTM by more than 88%. However, compared to HSBUFC, PCA-WMVFTS and AE-WMVFTS exhibited a decline of around 56% and 69%, respectively.

Despite the lower performance relative to HSBUFC, our models could improve accuracy by increasing the number of embedding dimensions or fuzzy sets, potentially surpassing HSBUFC. Additionally, HSBUFC, being a deep learning model, requires significant time for training and optimization, whereas FTS models are much faster to train. Furthermore, our models are more parsimonious and interpretable than deep learning alternatives.

4.5.5 Parsimony and explainability

Parsimony is crucial as it reduces the number of parameters or rules, leading to lower computational costs and shorter training times (see Section 3.2.3). Generally, increasing the K and κ improves the accuracy of EFTS models. However, this comes with a trade-off: higher accuracy leads to reduced parsimony and explainability. Table Table 14 illustrates this trade-off by showing the mean rules generated by PCA-WMVFTS across 30 experiments.

Table 14 – The parsimony of PCA-WMVFTS with different number of embedding dimensions (K) and number of fuzzy sets (κ) [Bitencourt et al., 2023].

K	κ	AEC	KSH10	HPC30
2	10	332 ± 52	428 ± 70	293 ± 37
2	20	842 ± 117	1420 ± 259	784 ± 134
2	30	1269 ± 149	2531 ± 464	1306 ± 232
2	40	1628 ± 182	3528 ± 634	1834 ± 318
2	50	1939 ± 181	4392 ± 744	2364 ± 392
3	10	1110 ± 152	1892 ± 290	1193 ± 207
3	20	2538 ± 287	5552 ± 883	3229 ± 694
3	30	3574 ± 308	8546 ± 1365	5305 ± 1155
3	40	4351 ± 375	10652 ± 1660	7241 ± 1513
3	50	5009 ± 368	12208 ± 1773	9010 ± 1785
4	10	3138 ± 337	6133 ± 872	3829 ± 580
4	20	6552 ± 555	15007 ± 2147	9769 ± 1452
4	30	8828 ± 562	21103 ± 2922	15197 ± 2128
4	40	10431 ± 613	24949 ± 3317	19833 ± 2581
4	50	11693 ± 544	27656 ± 3280	23767 ± 2926
5	10	7948 ± 697	17722 ± 2851	9390 ± 1296
5	20	15641 ± 1059	37324 ± 5289	22260 ± 3005
5	30	20313 ± 1082	48771 ± 6226	33730 ± 4235
5	40	23369 ± 1051	55389 ± 6489	43510 ± 5125
5	50	25568 ± 917	59767 ± 6049	51707 ± 5815

Parsimony is especially important in IoT applications, where models are often deployed on edge devices with limited computational power. PCA-WMVFTS models are less parsimonious than SARIMAX but significantly more parsimonious than deep learning models like LSTM, GRU, and TCN, particularly on the KSH datasets, as demonstrated in Table 15.

Table 15 – The parsimony of PCA-WMVFTS against deep learning methods [Bitencourt et al., 2023].

Method	AEC	KSH1	KSH10	HPC1	HPC30
PCA-WMVFTS	5009	29933	12208	31401	9010
LSTM	14071	187801	132501	276751	258826
GRU	10561	140881	99401	207601	194156
RNN	35410	47041	33201	69301	64816
TCN	101401	1979924	182368	50736	81780

4.5.6 Computational Complexity

The time complexity of the proposed PCA-WMVFTS and AE-WMVFTS methods is primarily driven by the embedding and training procedures. The time complexity of the PCA embedding process is $(\min(V^3, N^3))$ [Johnstone and Lu, 2009], while the AE embedding has the same complexity as training a Multilayer Perceptron Network [Freire et al., 2022]

Training depends on κ and K. Each embedded variable has its own fuzzy sets, so the total fuzzy sets are represented as $\gamma\kappa$ as $K \times \kappa$. The fuzzification step has a potential time complexity of $\mathcal{O}(n \times \gamma\kappa)$. By using a binary search tree, the search among fuzzy sets can be reduced from $\mathcal{O}(\gamma\kappa)$, making the rule base generation step have a time complexity of $\mathcal{O}(\log \gamma\kappa)$.

For testing, the time complexity includes both the embedding and forecasting procedures. For an embedded input instance $y(t)_{\gamma_i}$, fuzzification costs $\mathcal{O}(\gamma\kappa)$ and rule matching takes $\mathcal{O}(\log \gamma\kappa)$.

In summary, the computational complexity of RNNs, LSTMs, and GRUs is significantly higher than that of MLPs [Freire et al., 2022]. EFTS methods, including PCA-WMVFTS, have much lower computational complexity due to fewer parameters, as evidenced by Table 15.

Chapter 5

Multi-Step Embedding-based Fuzzy Time Series (MS-EFTS)

In this chapter, we introduce a new multi-step-ahead forecasting methodology named Multi-Step Embedding-based FTS (MS-EFTS). This work was published to IEEE Internet of Things Journal [Bitencourt et al., 2025]. MS-EFTS builds upon our proposed Embedding-based Fuzzy Time Series (EFTS) available in [Bitencourt et al., 2023] and detailed in Chapter 4. Accordingly, MS-EFTS adapts EFTS to accommodate multi-step-ahead forecasting using a direct strategy to predict non-stationary, high-dimensional IoT time series. Similarly, embedding transformation is exploited for reducing the dimensions of time series to allow efficient pattern discovery and the induction of fuzzy rules. The forecast horizon is equal $H = \{5, 10, 15, 20, 25, 30\}$.

Through this approach, given the high-dimensional non-stationary time series $Y \in \mathbb{R}^{N \times V}$ and its instances $y(n) \in Y$ for $n = 0, 1, ..., N$. Given the target variable $\mathcal{V}^*$, and the time-steps vector (i.e forecast horizon) H where $h_i \in H$ for $i = 0, 1, ..., M$. Then we create h_m independent and parallel EFTS models. These models effectively capture all the relevant information in the time series for the subsequent multi-step-ahead forecasting. The MS-EFTS methodology is made up of learning and testing procedures, which are detailed in the next subsections.

5.0.1 Learning procedure

The MS-EFTS learning procedure, illustrated in Figure 16, involves three main steps: embedding, variable partitioning, and rule induction. It generates h_m multivariate EFTS models ($\mathcal{M}_{\mathcal{EFTS}}$), which capture all the information in the time series Y.

Given the high-dimensional time series Y, the target variable $\mathcal{V}^*$, the time-steps vector H, the number of fuzzy sets κ, and the embedding dimensions K, the following steps are performed for each h_i.

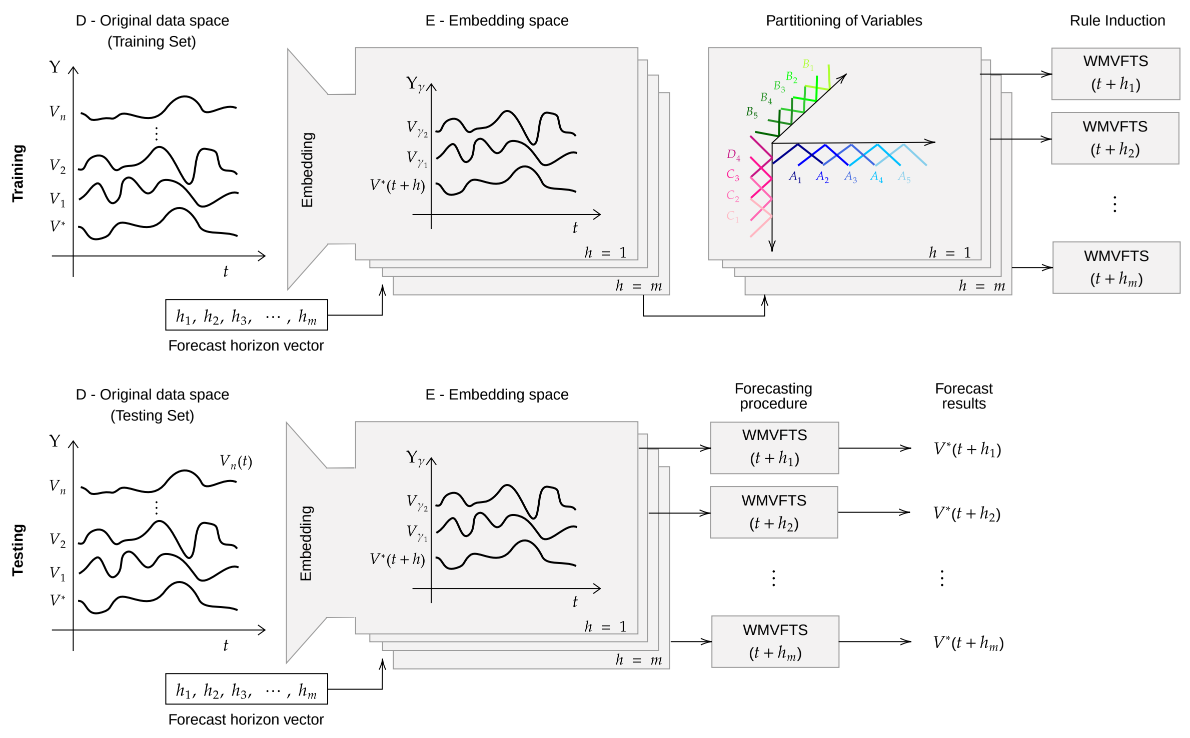

Figure 16 – MS-EFTS Learning and Testing procedures [Bitencourt et al., 2025].

5.0.1.1 Embedding

Similarly, the embedding procedure involves extracting a new variable space that more accurately captures the content of high-dimensional non-stationary time series. This process reduces dimensionality and facilitates the subsequent forecasting function. In this introduced approach, we focuses on two embedding methods: PCA detailed in Section 2.4.1 and Kernel PCA (KPCA) (see Section 2.4.2).

Subsequently, the embedding multivariate time series $Y_\gamma \in \mathbb{R}^{N \times K}$ is created. This embedding step is applied in both the learning and testing procedures.

5.0.1.2 Partitioning of variables

In this phase, the universe of discourse (UoD) is determined as $U\mathcal{V}\gamma = [lb, ub]$, where the lower bound $lb = \min(Y\gamma) - D_1$ and the upper bound $ub = \max(Y_\gamma) + D_2$. Here, $D_1 = r \times |\min(Y_\gamma)|$ and $D_2 = r \times |\max(Y_\gamma)|$, with $0 < r < 1$.

Next, $U_{\mathcal{V}_\gamma}$ is divided into κ_i overlapping intervals U_j, each with a midpoint c_j, for all $j \in 0, 1, 2, \ldots, \kappa_i$. A fuzzy set $\mathcal{A}_j^{\mathcal{V}i}$ is created for each interval, with its membership function $\mu\mathcal{A}_j{}^{\mathcal{V}i}$ defining the variable $\mathcal{V}\gamma i$ for dimension i, which in turn becomes a linguistic variable $\widetilde{\mathcal{V}}\gamma i \in \widetilde{\mathcal{V}}\gamma$.

5.0.1.3 Rule induction

At this stage, the embedding time series Y_γ is converted into a fuzzy time series F_γ. For each data point $f_\gamma(n) \in F_\gamma$, an array of size $n \times \kappa$ is created, containing the fuzzified values corresponding to the linguistic variable. The fuzzy memberships are calculated according to predetermined criteria as follows:

$$f(t) \leftarrow \left\{ \mathcal{A}_j^{\mathcal{V}_i} \mid \mu_{\mathcal{A}_j}^{\mathcal{V}_i}(Y_\gamma(t)^i) > 0 \right\} \tag{5.1}$$

Temporal patterns are created in the form of $\mathcal{A}j^{\mathcal{V}0}, ..., \mathcal{A}j^{\mathcal{V}n} \rightarrow \mathcal{A}j^{\mathcal{V}*}$, where the left-hand side (LHS) is defined as $f\gamma(t) = \mathcal{A}j^{\mathcal{V}i}$, and the right-hand side (RHS) represents the target variable $f\gamma(t+h_i) = \mathcal{A}j^{\mathcal{V}^*}$. Both sides correspond to $\mathcal{A}_j^{\mathcal{V}_i}$ with the maximum membership.

Finally, fuzzy rules r with the format $LHS \rightarrow w \cdot RHS$ are built in which the weights $w = |\mathcal{A}_j^{\mathcal{V}_i}|/|RHS|$ are defined as follows

$$w_i = \frac{\#\mathcal{A}}{\#RHS} \qquad \forall \mathcal{A}_j^{\mathcal{V}^*} \in RHS \tag{5.2}$$

where $\#\mathcal{A}$ is the number of occurrences of fuzzy rule $\mathcal{A}_i$ on temporal patterns with the same LHS and $\#RHS$ is the total number of temporal patterns with the same precedent LHS. Each fuzzy rule represents a weighted set of possibilities for what may occur at time $t+h$ (right-hand side) based on identified patterns in prior time lags (left-hand side).

5.0.2 Testing procedure

The testing procedure is illustrated in Figure 16. MS-EFTS models are applied to the test set, denoted as $Y_{test} \in \mathbb{R}^{N_{test} \times V}$. The embedding process from Section 4.1 is then used to generate the test embedding time series, $Y_{test_\gamma} \in \mathbb{R}^{N_{test} \times K}$. For each h_i, the MS-EFTS model forecasts h_i target values $(\hat{\mathcal{V}}^*(t+h_i))$ using the forecasting forecasting steps.

First, Y_{test_γ} is fuzzified for each variable $\mathcal{V}_i \in \mathcal{V}$ according to equation 7.1

Second, we choose r fuzzy rules to determine whether any fuzzy set is equal to LHS and compute the rule membership grade μ_q using the minimum function T-norm according to the equation:

$$\mu_q = \bigcap_{j \in \widetilde{\mathcal{V}}_i \ ; \ i \in \mathcal{V}} \mu_j i \tag{5.3}$$

here μ_j and μ_i represent the membership grades of the target and exploratory variables, respectively, with, μ_q being the minimum intersection of these grades.

Third, we compute the mean point mp_q of $\mathcal{V}^*$ as follows:

$$mp_q = \sum_{j \in \widetilde{\mathcal{V}}_i^*} w_j \cdot c_j \tag{5.4}$$

where w_j is the weights calculated according to 7.2.

Finally, the predicted value $\hat{y}(t + h_i)$, is calculated as a weighted sum of the rule midpoints, with the weights determined by the membership grades μ_{A_j}:

$$\hat{\dagger}^*(t + h_i) = \sum_{q \in r} \mu_q \cdot mp_q \tag{5.5}$$

In this approach, we developed two methods: PCA-MS-EFTS and KPCA-MS-EFTS, which integrate the EFTS model with PCA and KPCA, respectively.

5.0.3 MS-EFTS hyperparameter setting

As with the EFTS model, the embedding dimension (K) and the number of fuzzy sets (κ) are the most critical hyperparameters in the PCA-MS-EFTS and KPCA-MS-EFTS methods. Various combinations of these hyperparameters were tested through trial and error using the training set, with $K \in 2, 3, 4$ and $\kappa \in 30, 35, 40, 45, 50$ to find the optimal configuration. For all datasets, the best results were achieved with $K = 2$ and $\kappa = 30$.

Although increasing the values of K and κ can improve accuracy, it may also lead to an exponential increase in the number of fuzzy rules, which could compromise the model's parsimony and explainability.

5.1 Results

This section presents the experimental results of the proposed methodology compared to several deep learning multi-step forecasting methods across three datasets. Various techniques were implemented to evaluate the ability of these methods to handle different data patterns and forecast horizons.

In Section 5.1.1, the competing methods and their HP settings are explained. Section 5.1.2 compares the forecast performance of PCA-MS-EFTS and KPCA-MS-EFTS with baseline models, focusing on accuracy, parsimony, and optimal predictions. Finally, Sections 5.1.3 and 5.1.3 discuss the computational complexity and limitations of the proposed methods.

Table 16 – Optimal values of hyperparameters for the TCN, LSTM, CNN-LSTM (CLSTM), and BiLSTM methods [Bitencourt et al., 2025].

Hyperparameters	KSH30			AQB1			HPC10		
	LSTM / BiLSTM	TCN	CLSTM	LSTM / BiLSTM	TCN	CLSTM	LSTM / BiLSTM	TCN	CLSTM
Number of residual block stacks	-	2	-	-	2	-	-	2	-
Convolution filter size	-	64	64	-	64	64	-	32	32
Number of conv. layers	-	1	1	-	1	1	-	1	1
Number of conv. layer filters	-	5	5	-	2	2	-	5	5
Number of units	97	-	97	99	-	99	97	-	97
Number of LSTM layers	5	-	5	2	-	2	2	-	2
Dropout percentage	0.4	0.3	0.4	0.4	0.1	0.4	0.2	0.2	0.2
Batch normalization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

5.1.1 Deep learning baseline methods

Deep learning models, including TCN, LSTM, BiLSTM, and CNN-LSTM, were chosen as baseline methods. These models were implemented with the hyperparameters shown in Table Table 16, using the Adam optimization algorithm and the ReLU activation function. Hyperparameter optimization was performed through a random grid search.

Two experimental approaches were used: (1) utilizing $n + 1$ input values (one for each variable at time t) and (2) selecting variables with the highest correlation to the target variable. The variables chosen for each dataset are listed in Table 17. All methods produced six output values, one for each forecast horizon. The results reflect the best-observed performances, with approach (1) generally outperforming, except for CNN-LSTM in the KSH30 and AQB1 datasets.

Table 17 – Deep learning baseline methods: Selected Variables [Bitencourt et al., 2025].

Datasets	**Variables**
KSH30	Furnace 2 [kW]; Furnace 1 [kW]
AQB1	PM10; CO; NO2
HPC10	Global_intensity; Sub_metering_3; Sub_metering_2

5.1.2 MS-EFTS versus baseline forecasting methods

This section compares the forecasting accuracy of our MS-EFTS methods with baseline algorithms. The overall prediction performance is summarized by computing the mean and standard deviation of NRMSE across all time steps for each dataset, as shown in Table 18. Detailed results are available through the link in the supplementary materials (Annex B).

Our MS-EFTS methods consistently outperform the baseline methods across all datasets. Unlike the MS-EFTS models, where specific models are used for each forecast horizon, the baseline methods make direct predictions for all forecast horizons at once, which likely explains the significantly higher errors in their results.

In contrast to the MS-EFTS models, the performance of all baseline methods is significantly influenced by the number of training instances, as evident in the AQB1 dataset results. Additionally, due to their stochastic nature and numerous HP, the baseline methods show greater variance compared to the MS-EFTS models.

These results highlight the superiority of the MS-EFTS methods for multi-step time series forecasting in various IoT applications. The MS-EFTS models consistently outperform the baseline methods, demonstrating robust accuracy regardless of dataset size, while using a minimal window of just one lag to predict up to 30 steps.

Table 18 – Forecasting model's accuracy in term of mean NRMSE [Bitencourt et al., 2025].

Methods	**KSH30**	**AQB1**	**HPC10**
PCA-MS-EFTS	0.031 ± 0.02	0.028 ± 0.01	0.036 ± 0.03
KPCA-MS-EFTS	0.030 ± 0.01	0.029 ± 0.02	0.035 ± 0.03
LSTM	0.122 ± 0.05	0.917 ± 0.78	0.204 ± 0.24
BiLSTM	0.124 ± 0.05	0.655 ± 0.49	0.183 ± 0.13
TCN	0.122 ± 0.11	0.758 ± 0.68	1.418 ± 1.28
CNN-LSTM	0.180 ± 0.47	0.898 ± 1.51	0.893 ± 0.93

5.1.2.1 Statistical test

The statistical analysis, performed using the AutoRank python package [Herbold, 2020], was conducted on six populations, corresponding to the NRMSE results of the baselines and proposed methods across all datasets in the study. The analysis included 120 paired samples pertaining to the 30 windows for each of the six forecast horizons.

The statistical significance of the differences between the NRMSE means for all methods was evaluated using the non-parametric Friedman test, as normality could not be assumed for any of the samples. A significance level of 0.05 was used for the tests. The Friedman test assesses the null hypothesis that there are no significant differences between the median values of the populations. In all cases, the null hypothesis was rejected, indicating that there is a statistically significant difference between the median values of the populations.

The post-hoc Nemenyi test was applied to determine which differences are significant, which can be visualized through the critical difference diagram (Figure 17). Differences between populations are considered significant if the difference in mean ranks is greater than the critical distance (CD) of the Nemenyi test, represented in the diagram by the black bar. It is observed that, across all forecast horizons, the proposed methods show significant differences compared to the baselines, but not among themselves.

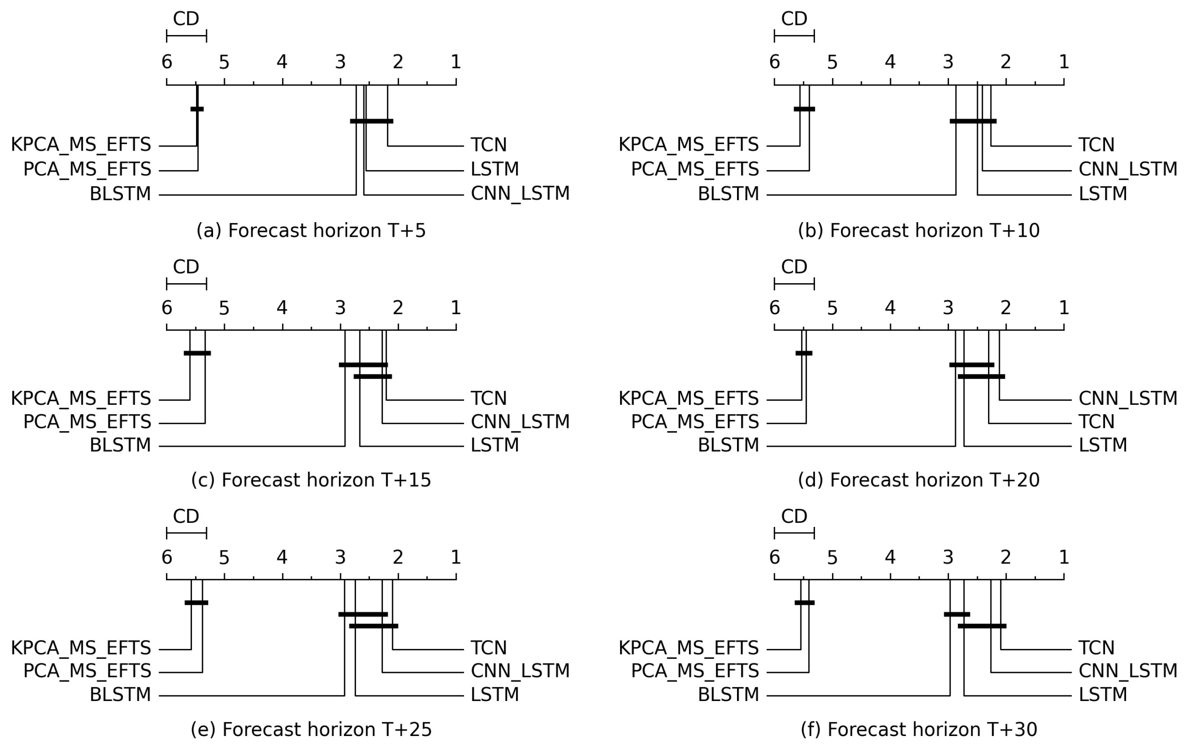

Figure 17 – Critical difference diagram of the KSH30, AQB1, and HPC10 datasets for each forecast horizon [Bitencourt et al., 2025].

5.1.2.2 Optimal prediction

Figure 18 illustrates the fractional bias (FB) index of various forecasting methods across all datasets and forecast horizons. An FB of zero indicates perfect alignment between forecasted and observed values, while positive values suggest over-prediction and negative values indicate under-prediction.

For the KSH30 and HPC10 datasets, the MS-EFTS methods exhibit FB values ranging from -0.03 to -0.09, reflecting slight under-prediction across all forecast horizons. In contrast, these methods achieve optimal prediction on the AQB1 dataset, with FB values between -0.01 and 0.01, indicating minimal systematic error in their forecasts.

In comparison, the LSTM, BiLSTM, and CNN-LSTM models tend to over-predict the target variable in the AQB1 dataset, particularly LSTM and BiLSTM, which show the highest levels of over-prediction. For the KSH30 and HPC10 datasets, these deep learning methods demonstrate moderate over-prediction, with FB values between 0.05 and 0.18. Conversely, the TCN model performs comparably to the MS-EFTS methods regarding the FB metric.

In summary, both PCA-MS-EFTS and KPCA-MS-EFTS showed comparable prediction errors and achieved the highest accuracy among the forecasting methods. These methods consistently performed well across all datasets, highlighting their effectiveness and robustness in delivering accurate predictions for various high-dimensional IoT time

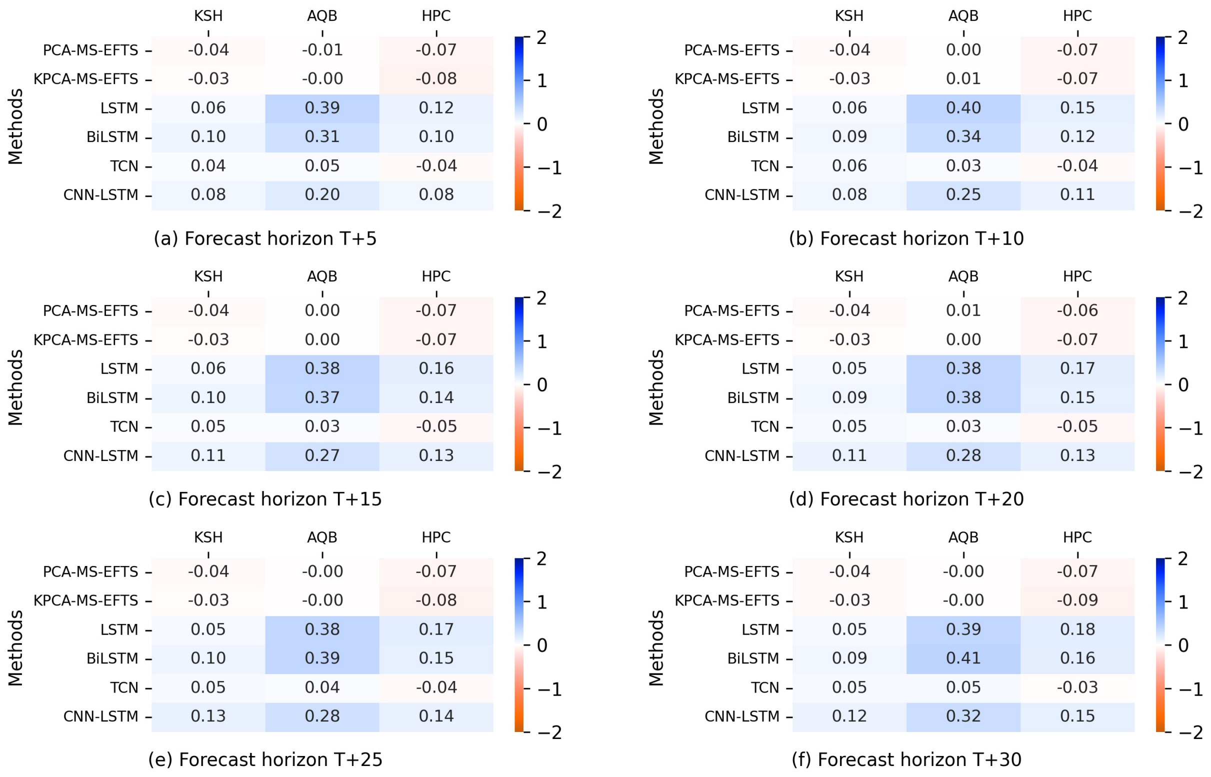

Figure 18 – Fractional bias heatmap of the KSH30, AQB1, and HPC10 datasets for each forecast horizon [Bitencourt et al., 2025].

series, as evidenced by metrics such as RMSE, NRMSE, and FB.

5.1.2.3 Parsimony and explainability

In MS-EFTS methods, complexity and explainability are influenced by HP κ and K, which increase the number of rules as they grow. However, the explainability of the models depends on the active fuzzy rules, with rule length corresponding to the number of embedding variables. Thus, despite the rise in complexity, the methods remain interpretable based on the active rules.

Table 19 highlights the parsimony of the MS-EFTS and deep learning methods by showing the mean number of rules or parameters across 30 experiments for all forecast horizons. This comparison illustrates the efficiency of each method in terms of model complexity.

Table 28 shows that MO-EFTS methods are much more parsimonious than the baseline methods. Among deep learning techniques, BiLSTM is the least parsimonious, while TCN stands out for its high parsimony. PCA-MS-EFTS is slightly more parsimonious than KPCA-MS-EFTS, but the difference is minor.

The MS-EFTS methods achieve simplicity while maintaining strong forecasting accuracy, primarily controlled by two key hyperparameters. Increasing the values of κ

Table 19 – The parsimony of MS-EFTS methods against deep learning methods [Bitencourt et al., 2025]

Methods	**KSH30**	**AQB1**	**HPC10**
PCA-MS-EFTS	22,808	17,268	15,472
KPCA-MS-EFTS	23,991	17,815	16,519
LSTM	373,068	126,132	56,370
BiLSTM	1,051,098	330,666	147,582
TCN	137,670	63,558	35,046
CNN-LSTM	141,684	139,304	60,778

and K reduces forecast error, suggesting that further improvement requires tuning these hyperparameters. In contrast, baseline deep learning models require extensive effort to optimize hyperparameters, are complex, and offer limited explainability.

5.1.3 Computational complexity of the MS-EFTS methodology

The computational complexity of the learning procedure can be analyzed by considering the computational cost of training h_m multivariate EFTS models $\mathcal{M}_{\mathcal{EFTS}}$. For each $\mathcal{M}_{\mathcal{EFTS}}$, the time complexity depends on κ and K.

Since each $\mathcal{V}_\gamma$ (i.e. embedded variable) has its own fuzzy sets, we define $\gamma\kappa$ as $K \times \kappa$. The fuzzification step has a potential cost of $\mathcal{O}(n \times \gamma\kappa)$. By using a binary search tree to sort the κ, the computational complexity for a search among them is reduced from $\mathcal{O}(\gamma\kappa)$ to $\mathcal{O}(\log \gamma\kappa)$. In addition, the embedding step costs $\mathcal{O}(\gamma)$ either $\mathcal{O}(\min(V^3, N^3))$ (PCA) or $\mathcal{O}(N^3)$ (KPCA). Therefore, the computational complexity of the learning procedure can be expressed as $\mathcal{O}(h_m \times (\mathcal{O}(\log \gamma\kappa) + \mathcal{O}(\gamma)))$

Chapter 6

Multiple Output Embedding Non-Stationary FTS (MO-ENSFTS)

In this chapter, we detail our MIMO high-dimensional non-stationary time series forecasting methodology called MO-ENSFTS (Multiple Output Embedding-based Non-Stationary Fuzzy Time Series) and published in [Bitencourt et al., 2022]. Accordingly, MO-ENSFTS is an extended form of our ENSFTS approach [Bitencourt and Guimarães, 2021]. The forecast horizon $H = \{1\}$ is one step ahead.

In this approach, the original V variables of a high-dimensional non-stationary time series are transformed into K embedded components. An independent ENSFTS model is then applied to each of the K components, creating K parallel ENSFTS models that capture the relevant information for forecasting multiple features. In this methodology, we developed two models: PCA-MO-ENSFTS and KPCA-MO-ENSFTS, which combine the ENSFTS model with PCA and KPCA, respectively. These methods involve distinct learning and testing procedures, which are detailed in the subsequent sections.

6.1 MO-ENSFTS Learning Procedure

he MO-ENSFTS learning procedure, as illustrated in Figure 19, involves four key steps. First, K ENSFTS models are generated to capture the information from the K embedding components. Next, the parameters of the fuzzy sets are adjusted based on prediction errors from the training set, followed by the creation of fuzzy rules.

In summary, the learning process consists of four main stages: embedding, training, parameter adaptation, and forecasting. It is important to note that the forecasting step is common to both the learning and testing phases of the MO-ENSFTS procedure.

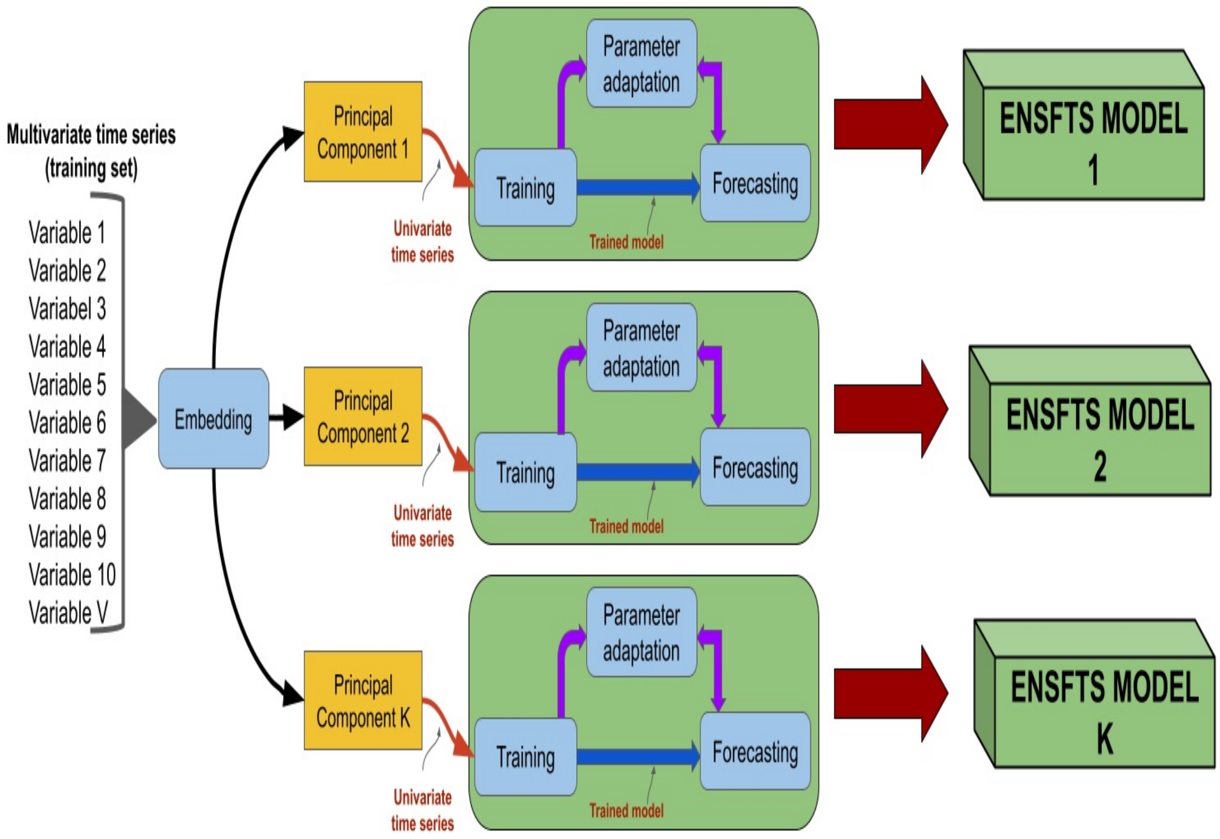

Figure 19 – MO-ENSFTS Learning Procedure [Bitencourt et al., 2022]

6.1.1 Embedding

Similar to the EFTS and MS-EFTS approaches, the embedding procedure plays a key role in extracting the principal components that best capture the essential information from high-dimensional time series, facilitating more accurate forecasting. In this approach, we focus on two embedding methods: PCA (as described in Section 2.4.1) and KPCA (detailed in Section 2.4.2). It is important to note that the embedding step is applied in both the learning and testing procedures.

6.1.2 Training

Let $Y_{emb} \in \mathbb{R}^1$ as the embedding time series where $y_{emb}(t) \in Y_{emb}$ for $t = 0, 1, ..., N$ indicates its individual instances. Also, we considered κ as the number of fuzzy sets and w_e as the length of the residuals window.

The universe of discourse is defined using $U = [lb, ub]$ in which $lb = \min(Y_{emb}) - D_1$ and $ub = \max(Y_{emb}) + D_2$, with $D_1 = r\times \mid \min(Y_{emb}) \mid$ and $D_2 = r\times \mid \max(Y_{emb}) \mid$,

$0 < r < 1$. The partitioning is oriented by the midpoints c_i of each fuzzy set A_i, as follows:

$$c_i = lb + i \times \frac{ub - lb}{\kappa - 1} \tag{6.1}$$

For each interval, a fuzzy set is defined by its triangular membership function $\mu_{A_i}(y_{emb})$

$$\mu_{A_i}(y_{emb}) = \begin{cases} 0, & \text{if } y < l \text{ or } y > u \\ \frac{y-l_i}{c_i-l_i}, & \text{if } l_i \leq y \leq c_i \\ \frac{u_i-y}{u_i-c_i}, & \text{if } c_i \leq y \leq u_i \end{cases}$$

All fuzzy sets have a perturbation function π_i

$$\pi(l, c, u, \delta, \rho) = \left\{\frac{\rho}{2} - (l + \delta), c + \delta, \frac{\rho}{2} + (u + \delta)\right\} \tag{6.2}$$

In this step, δ denotes the displacement of A_i along U, while ρ is the scaling parameter responsible for altering the coverage area of A_i, either by stretching or compressing its shape. The parameters l, c, and u represent the lower bound, midpoint, and upper bound of the triangular membership function, respectively, and they are initially set to zero.

Next, each embedded time series Y_{emb} is transformed into a FTS, denoted as F_{emb}. Temporal patterns are then extracted in the form of fuzzy rules $A_{LHS} \rightarrow A_{RHS}$, where A_{LHS} represents the precedent and A_{RHS} the consequent. Both are linked to A_i, which has the maximum membership. These patterns represent fuzzy rules and are grouped according to their precedents.

Finally, residuals are computed by applying the forecasting procedure (Section 6.1.4) to the training set. The last w_e values are predicted, and the residuals are calculated as follows:

$$E = \{e(t - w_e), e(t - (w_e - 1), ..., e(t))\} \tag{6.3}$$

where $e(t) = y_{emb}(t) - \hat{y}_{emb}(t)$ and $\hat{y}_{emb}(t)$ is the predicted value.

6.1.3 Parameter adaptation

During parameter adaptation step, the mean and variance of residuals are tracked to adjust the membership function. Let E represent the residuals, $\hat{y}(t + 1)$ the forecast value, and $y(t + 1)$ the actual value. The displacement parameter is updated based on midpoint changes, triggered when $y(t)$ falls outside the range of U, according to the following conditions:

$$\textit{if } (y(t) < lb) \ \ \textit{then } (d_l = lb - y(t)) \ \textit{else } (d_l = 0) \tag{6.4}$$

$$\textit{if } (y(t) > ub) \;\; \textit{then } (d_u = y(t) - ub) \;\; \textit{else } (d_u = 0) \tag{6.5}$$

Then, we calculate the mean $\bar{E}$ and variance σ_E of the residuals, and these values are used to update position and length parameters of the fuzzy sets. For each fuzzy set, the displacement δ_i is computed according to the following equation:

$$\delta_i = \bar{E} + \left(i\frac{r}{k+1} - d_{mp}\right) + \left(i\frac{2\sigma_E}{k-1} - \sigma_E\right) \tag{6.6}$$

where $r = d_u - d_l$ (displacement range) and $d_{mp} = r/2$ (displacement midpoint).

Finally, we measured the scaling factors ρ_i as follow: $\rho_i =\mid \delta_{i-1} - \delta_{i+1} \mid$. The new parameters values δ_i and ρ_i are used by the perturbation function.

6.1.4 Forecasting

Given Y_{emb} as the embedding univariate time series and $y_{emb}(t)$ as its instances for $t = 0, 1, ..., N$, the predicted values $\hat{y}(t+1)$ are computed as follows. First, we calculate the membership grade μ_{A_i} for each fuzzy set A_i using the membership function (MF) with parameters adjusted by π. Fuzzy sets are then selected where A_j where $\mu_{A_i} \geq 0$.

These selected fuzzy sets A_j serve as inputs for the rule base to match rules based on their precedent. The rule set is defined as $S = \{A_j \rightarrow RHS_j \mid \mu_{A_j}(y(t)) > 0\}$, where RHS_j is the consequent of the rule.

The predicted value $\hat{y}(t+1)$ is obtained as the weighted sum of the rule midpoints by their membership grades μ_{A_j}, according to equation:

$$\hat{y}(t+1) = \sum_S \mu_{A_j}(y(t)) \cdot mp(RHS_j) \tag{6.7}$$

with $mp(RHS)$ determined as follows:

$$mp(RHS) = \frac{\sum_{A_i \in RHS} c_{A_i}}{\mid RHS \mid} \tag{6.8}$$

6.2 MO-ENSFTS Testing Procedure

The testing procedure for the MO-ENSFTS model is illustrated in Figure 20. Its objective is to identify rules that correspond to a given fuzzified input and use them to forecast numerical values by applying non-stationary fuzzy sets adjusted by π. This process aims to estimate $\hat{y}(t+1)$ for the target variables $\mathcal{V}^* \in \mathbb{R}^V$, given input samples $y(t)_i \in \mathbb{R}^V$ using the non-stationary fuzzy rules from the K ENSFTS models.

Specifically, the testing set, denoted as $Y \in \mathbb{R}^{N \times V}$, which represents a high-dimensional non-stationary time series, is input into the proposed model. We then apply the embedding procedure outlined in Section 6.1.1 to generate K embedding variables.

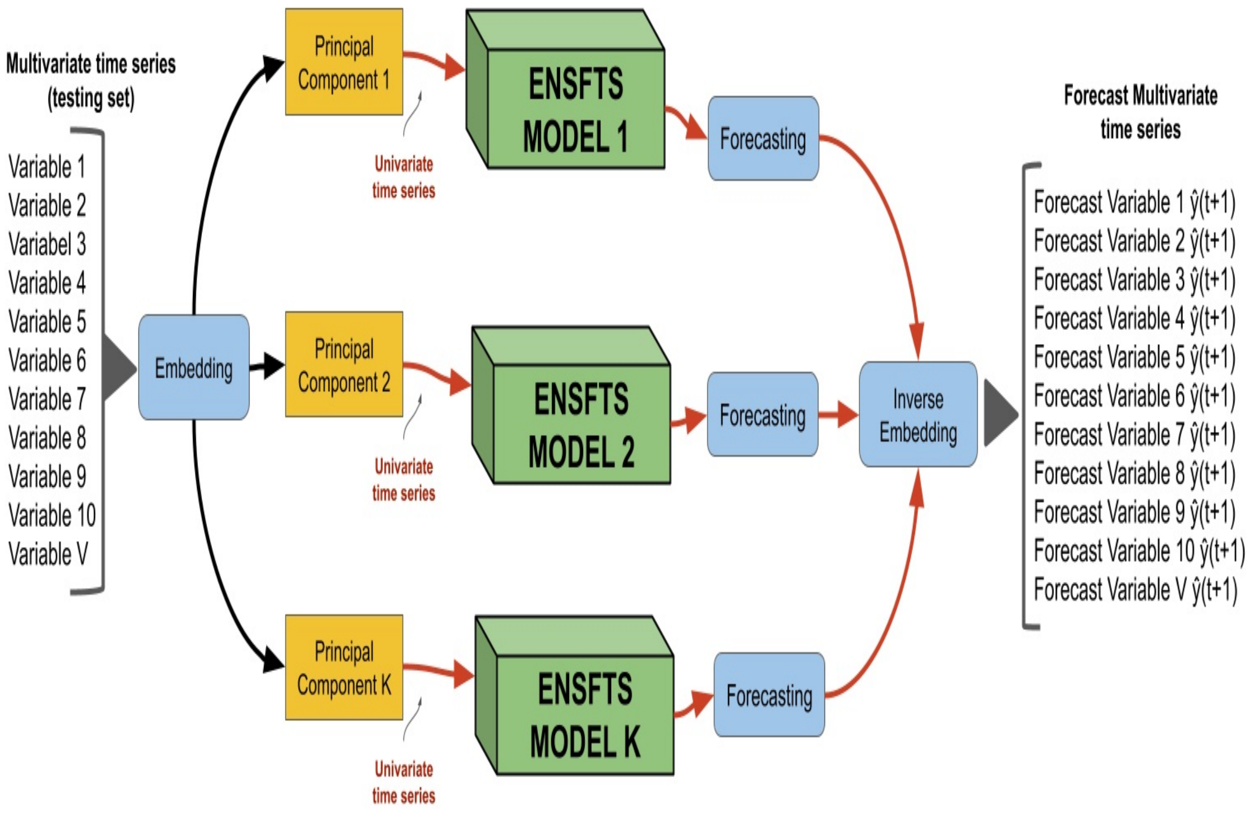
Multivariate time series (testing set)
Variable 1
Variable 2
Variabel 3
Variable 4
Variable 5
Variable 6
Variable 7
Variable 8
Variable 9
Variable 10
Variable V
Embedding
Principal Component 1
Principal Component 2
Principal Component K
Univariate time series
ENSFTS MODEL 1
ENSFTS MODEL 2
ENSFTS MODEL K
Forecasting
Inverse Embedding
Forecast Multivariate time series
Forecast Variable 1 ŷ(t+1)
Forecast Variable 2 ŷ(t+1)
Forecast Variable 3 ŷ(t+1)
Forecast Variable 4 ŷ(t+1)
Forecast Variable 5 ŷ(t+1)
Forecast Variable 6 ŷ(t+1)
Forecast Variable 7 ŷ(t+1)
Forecast Variable 8 ŷ(t+1)
Forecast Variable 9 ŷ(t+1)
Forecast Variable 10 ŷ(t+1)
Forecast Variable V ŷ(t+1)

Figure 20 – MO-ENSFTS Testing Procedure [Bitencourt et al., 2022]

For each variable, we utilize an ENSFTS learning model (as described in Section 6.1) to forecast K embedding predict values $\hat{y}_{emb}(t+1)$ using the forecasting procedure detailed in Section 6.1.4. Finally, an embedding inverse transformation is applied to revert to the original data dimensional space.

6.2.1 MO-ENSFTS Hyperparameters

Although our proposed approach can adapt to the data by adjusting the shape and scale of fuzzy sets based on the dataset, certain hyperparameters must be defined prior to the learning process. Key hyperparameters include the embedding dimension (K) the number of fuzzy sets (κ), the length of the residuals window (w_e) and the kernel coefficient of KPCA (γ). Therefore, these HPs need to be optimally tuned to achieve the best accuracy in terms of RMSE, then we empirically tested various combinations of these HPs empirically on the training set to identify the optimal configuration, adjusting each HP as follows:

κ : Both our models were tested with $\{30, 40, 50, 60, 70\}$ fuzzy sets.

K : PCA-MO-ENSFTS needs more embedding dimensions than KPCA-MO-ENSFTS because of generating higher error using inverse transformation in PCA, afterwards KPCA-MO-ENSFTS were tested using $K \in \{2, 3, 4, 5\}$ while in PCA-ENSFTS we have $K \in \{2, 3, ..., 12\}$.

w_e : For all datasets $w_e \in \{3, 4, 5\}$ except AQI that was tested with $w_e \in \{3, 5, 10, 15, 20, 25, 30\}$, then different values of w_e show little influence on the accuracy, excluding AQI.

γ : KPCA-MO-ENSFTS were executed with $\gamma \in \{0.01, 0.1, 0.5, 1\}$ and the results indicate that the model are not affected by varying these values.

Table 20 – HP values for KPCA-MO-ENSFTS model [Bitencourt et al., 2022]

HP	AEC	KSH1	AQI	AQB1	AQB6	AQB12
K	3	3	3	2	8	10
κ	50	60	60	60	65	60
w_e	3	3	25	3	3	3
γ	0.1	0.1	0.1	0.1	0.1	0.1

Tables 20 and 21 summarize the best values found. As seen in Table 20 the KPCA-MO-ENSFTS method achieves its best accuracy with a limited number of principal components. Additionally, forecasting accuracy improves with an increase in the number of partitions, with optimal performance noted at $\kappa \in \{50, 60, 65\}$ across different datasets. Table 21 indicates that higher values of K and κ correlate with increased accuracy in the PCA-MO-ENSFTS method.

A key advantage of KPCA-MO-ENSFTS is its ability to achieve optimal prediction accuracy with a lower K value compared to PCA-MO-ENSFTS. However, it is important to consider that increasing the number of fuzzy sets and reducing dimensions without limit could lead to an exponential growth in the number of fuzzy rules. In conclusion, our model's accuracy is significantly influenced by the values of K and κ, whereas their effectiveness is less affected by w_e and γ.

Table 21 – HP values for PCA-MO-ENSFTS model [Bitencourt et al., 2022]

HP	AEC	KSH1	AQI	AQB1	AQB6	AQB12
K	10	8	6	6	20	10
κ	50	60	60	70	60	60
w_e	3	3	25	3	3	3

6.3 Experimental Results

This section presents the experimental results. First, we introduce the baseline methods along with their hyperparameter settings. Next, we compare the forecast accuracy of our methods against the baseline models, highlighting their multiple prediction accuracy, mean NRMSE error, and ranking of forecasting method accuracy. Finally, we discuss the parsimony and explainability of our proposed methodology. Notably, PFTS and KFTS in all tables refer to the PCA-MO-ENSFTS and KPCA-MO-ENSFTS methods, respectively.

6.3.1 Baseline methods

The accuracy of the proposed MO-ENSFTS models was compared to several baseline models, including three types of RNNs: Vanilla RNN, LSTM, and GRU. These models were implemented following the guidelines of [LeCun et al., 2015], [Hochreiter and Schmidhuber, 1997], and [Chung et al., 2014], respectively. Optimal hyperparameters were selected empirically for each model. The best results were obtained using 1 layer, 300 epochs, a batch size of 64, a learning rate of 0.001, weight decay of 0.1, and the Adam optimizer. The hidden state size varied from 3 to 467 dimensions/units depending on the dataset.

We also implemented a Stacked LSTM encoder-decoder (SLSTM) method, a more complex version of the vanilla LSTM designed for sequence-to-sequence problems. The optimal configuration for the SLSTM was achieved with 25 epochs, a batch size of 32, a hidden state of 200, and the Adam optimizer, after testing various settings.

Additionally, we included Random Forest [Breiman, 2001] and Support Vector Regression [Drucker et al., 1996] as competitor models. Table 22 shows the optimal hyperparameters for each dataset obtained using the Randomized Search CV technique, with m-d, m-l, and m-s representing maximum depth, maximum leaf nodes, and minimum samples leaf, respectively.

Table 22 – HP values for RF and SVR models [Bitencourt et al., 2022]

Model	HP	AEC	KSH1	AQI	AQB1	AQB6	AQB12
RF	m-d	20	15	20	30	30	30
	m-l	10	20	50	30	30	30
	m-s	6	3	6	2	2	2
SVR	C	5.76	17.07	9.98	25.71	25.71	25.71
	ϵ	0.07	0.15	0.0001	0.05	0.05	0.05

6.3.2 Comparison against baselines

In this section, we compare the accuracy of our proposed models with baseline methods. Given the high number of variables in the datasets, particularly AQB6 and AQB12, presenting the complete prediction results for all datasets would be impractical. Thus, we focus on evaluating the accuracy of MIMO models on the AEC and AQB12 datasets using RMSE as the primary acurracy metric. Complete results are available publicly through the supplementary materials (Section C), where prediction accuracy can also be found using other metrics such as Mean Absolute Percentage Error (MAPE), alongside RMSE and NRMSE.

Table 23 shows the RMSE results for our PCA-MO-ENSFTS and KPCA-MO-ENSFTS models compared to baseline methods on the AEC dataset, with the best results highlighted in bold. Our proposed models consistently outperform the competitors across all variables. Specifically, the KPCA-MO-ENSFTS model delivers superior accuracy for more than 92% (24 out of 26) of the variables. However, for *Appliance* and *lights* as target variables, the PCA-MO-ENSFTS model achieves better results than other techniques.

Table 23 – Accuracy of our proposed models and baseline methods over AEC in terms of RMSE [Bitencourt et al., 2022].

	KFTS		**PFTS**		RF		GRU		RNN		LSTM		SLSTM		SVR	
Variables	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Appliances	64.76	29.06	**62.16**	**27.91**	72.92	33.65	77.57	39.05	75.02	34.27	78.07	37.21	79.55	32.48	86.81	30.96
lights	5.00	1.73	**4.79**	**1.64**	5.37	2.06	5.67	2.41	5.91	2.51	5.90	2.63	6.60	2.98	7.06	2.91
T1	**0.06**	**0.02**	0.10	0.05	0.26	0.23	0.27	0.13	0.27	0.15	0.30	0.15	0.42	0.26	0.70	0.46
RH_1	**0.42**	**0.21**	0.55	0.25	0.83	0.64	1.02	0.42	0.97	0.42	1.06	0.45	1.23	0.68	2.05	1.36
T2	**0.12**	**0.06**	0.19	0.11	0.31	0.21	0.50	0.20	0.50	0.26	0.51	0.19	0.65	0.39	1.04	0.52
RH_2	**0.28**	**0.13**	0.39	0.16	0.74	0.76	0.94	0.43	0.95	0.45	0.99	0.50	1.47	1.07	2.20	1.66
T3	**0.07**	**0.03**	0.12	0.06	0.37	0.36	0.35	0.24	0.32	0.18	0.36	0.20	0.55	0.35	0.73	0.50
RH_3	**0.20**	**0.10**	0.29	0.12	0.57	0.50	0.67	0.28	0.66	0.36	0.74	0.37	0.99	0.62	1.50	1.08
T4	**0.08**	**0.05**	0.13	0.08	0.33	0.33	0.43	0.32	0.41	0.29	0.41	0.26	0.69	0.48	0.86	0.56
RH_4	**0.17**	**0.05**	0.25	0.08	0.61	0.53	0.70	0.28	0.69	0.30	0.82	0.33	1.09	0.59	1.90	1.52
T5	**0.11**	**0.06**	0.14	0.09	0.29	0.23	0.31	0.18	0.32	0.21	0.34	0.22	0.46	0.35	0.63	0.41
RH_5	**1.90**	**1.11**	2.40	1.39	3.10	1.75	4.35	2.32	4.85	2.50	4.97	2.60	5.28	2.41	6.94	2.78
T6	**0.31**	**0.13**	0.51	0.25	0.89	0.78	1.24	0.46	1.43	0.78	1.42	0.62	1.61	0.68	2.70	1.29
RH_6	**1.42**	**0.67**	2.27	1.03	2.14	1.61	5.65	2.60	5.16	2.20	5.67	2.46	7.54	5.35	10.13	5.34
T7	**0.05**	**0.02**	0.08	0.04	0.23	0.20	0.22	0.12	0.24	0.14	0.25	0.15	0.37	0.22	0.59	0.38
RH_7	**0.22**	**0.12**	0.34	0.13	1.04	1.16	1.10	0.48	1.03	0.44	1.14	0.44	1.85	1.25	2.89	2.10
T8	**0.06**	**0.02**	0.12	0.03	0.23	0.27	0.34	0.15	0.38	0.15	0.39	0.17	0.63	0.35	0.70	0.40
RH_8	**0.28**	**0.09**	0.53	0.18	0.87	0.83	1.43	0.49	1.46	0.46	1.48	0.55	1.67	0.79	2.80	1.80
T9	**0.04**	**0.02**	0.05	0.03	0.22	0.23	0.14	0.08	0.15	0.08	0.17	0.09	0.30	0.19	0.40	0.24
RH_9	**0.21**	**0.10**	0.35	0.16	0.62	0.62	0.90	0.39	0.94	0.34	1.10	0.38	1.42	0.74	2.19	1.57
T_out	**0.22**	**0.11**	0.32	0.13	0.71	0.62	0.91	0.45	1.01	0.62	1.05	0.52	1.29	0.65	2.25	1.34
Press	**0.11**	**0.06**	0.27	0.16	1.23	1.45	1.61	1.22	1.45	1.07	1.52	1.10	2.58	1.86	3.41	3.45
RH_out	**1.06**	**0.47**	1.70	0.55	1.86	1.35	4.19	1.51	4.11	1.77	4.31	1.65	6.32	2.07	8.57	4.25
Windspeed	**0.21**	**0.05**	0.39	0.12	0.41	0.40	1.04	0.55	1.07	0.65	1.12	0.73	1.40	0.89	1.55	0.91
Visibility	**2.09**	**0.90**	3.62	1.58	2.39	1.11	6.31	3.05	7.15	3.81	7.51	3.35	6.31	3.54	7.99	4.05
Tdewpoint	**0.15**	**0.07**	0.28	0.08	0.70	0.61	0.88	0.50	0.95	0.48	1.04	0.41	1.29	0.73	1.94	1.44

Table 24 presents a comparison of the forecasting accuracy of our models against the competitors on the AQB12 dataset. Although predictions were made for all 132 variables, only the results for PM2.5, a key air quality index, are shown in this work. Once again, our models outperform the baseline models for each variable.

Table 24 – PM2.5 prediction accuracy of our proposed models and baseline methods over all stations data in terms of RMSE [Bitencourt et al., 2022].

	KFTS		**PFTS**		RF		GRU		RNN		LSTM		SLSTM		SVR	
Stations	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD	AVG	STD
Aotizhongxin	**22.19**	**11.29**	23.93	11.10	25.36	13.44	40.82	21.06	55.68	29.25	49.77	25.59	31.38	0.13	36.60	23.20
Changping	**19.49**	**7.77**	23.27	11.35	21.74	12.16	39.20	16.54	47.90	20.74	35.97	15.86	30.86	0.15	34.26	22.62
Dingling	**17.41**	**7.49**	22.26	9.84	20.57	13.02	36.18	14.77	48.05	20.08	37.00	16.22	28.41	0.16	31.32	16.76
Dongsi	**23.14**	**10.61**	26.00	12.81	25.06	12.74	46.89	24.74	61.42	32.40	41.76	20.96	31.67	0.13	38.58	21.58
Guanyuan	**20.69**	**9.05**	23.14	10.11	23.31	12.20	37.16	17.42	52.70	25.12	37.89	19.62	29.97	0.13	35.67	21.66
Gucheng	**20.69**	**8.09**	24.94	12.10	24.06	12.42	43.93	21.79	55.28	27.25	41.44	22.10	32.95	0.14	36.60	21.01
Huairou	**18.27**	**7.88**	22.85	11.76	21.08	10.96	36.95	17.12	46.17	20.77	37.62	17.34	30.00	0.15	31.35	15.61
Nongzhanguan	**22.42**	**10.62**	25.90	13.49	25.08	14.56	44.21	22.96	66.66	36.74	40.97	20.97	32.01	0.13	37.47	23.71
Shunyi	**21.95**	**10.78**	26.63	13.43	24.66	14.87	43.51	22.95	53.89	28.38	49.62	25.49	35.11	0.15	38.44	23.04
Tiantan	**21.66**	**9.40**	24.55	11.66	23.43	13.92	40.35	18.83	52.71	25.89	42.78	22.44	29.57	0.13	35.40	21.76
Wanliu	**20.20**	**8.56**	24.38	15.02	23.99	14.87	46.01	21.74	58.76	29.29	38.91	19.85	32.54	0.14	36.08	25.56
Wanshouxigong	**23.15**	**10.16**	26.47	13.04	26.76	15.68	41.80	21.71	56.58	29.93	43.02	24.48	33.43	0.13	39.55	25.06

Ranking the methods based on RMSE results is challenging due to the variability in algorithm performance for different variables. Therefore, as shown in Table 25, the mean NRMSE across all variables v_j for each dataset d_j is used to provide a more accurate comparison among the models.

Table 25 – Model's accuracy in term of mean NRMSE [Bitencourt et al., 2022].

Methods	AEC	KSH1	AQI	AQB1	AQB6	AQB12
KFTS	**0.043**	0.052	0.14	0.09	**0.098**	**0.109**
PFTS	0.063	0.116	**0.138**	0.118	0.122	0.126
RF	0.138	0.065	1.047	0.9	3.228	0.177
GRU	0.163	0.051	0.16	**0.085**	0.144	0.189
RNN	0.164	**0.046**	0.162	0.086	0.143	0.228
LSTM	0.167	0.054	0.168	0.088	0.153	0.193
SLSTM	0.273	0.43	0.284	0.417	0.507	0.171
SVR	0.394	1.123	0.341	0.839	1.077	0.275

Table 25 highlights the superior accuracy of the proposed models on the AEC, AQI, AQB6, and AQB12 datasets. However, RNN and GRU outperform the proposed models on the KSH1 and AQB1 datasets, respectively. Notably, KPCA-MO-ENSFTS generally performs better than PCA-MO-ENSFTS across most datasets.

The results demonstrate the outperformance of our proposed models, KPCA-MO-ENSFTS more specifically, to the baseline methods with high reliability and stability. To our best of knowledge, RNNs show the best accuracy for datasets with large number of instances, such as KSH1. In this case, RNN is more accurate than KPCA-MO-ENSFTS, GRU and LSTM with a very slight differences.

6.3.3 Statistical test

We assess the accuracy of each forecasting model (m_i) by calculating the mean NRMSE $\epsilon[m_i, ds_j, w]$ across all variables in each dataset (ds_j) for each window (w). A

Kruskal-Wallis test, with a significance level of $\alpha = 0.01$, was performed to evaluate differences in performance.

The test compares the mean error $\mu_{m_i,ds_j} = \frac{\sum_{w=1}^{W} \epsilon[m_i,ds_j,w]}{W}$ of the forecasting methods. The null hypothesis (H_0) assumes the equality of means (μ_{m_i,ds_j}), indicating no significant difference between the models. If H_0 is rejected, it suggests that at least one model's mean differs from the others.

In cases where H_0 is rejected, we further compare the means between each pair of forecasting models using post hoc Wilcoxon tests.

$$H_0 : \mu_{m_a,ds_j} < \mu_{m_b,ds_j}$$

$$H_1 : \mu_{m_a,ds_j} \geq \mu_{m_b,ds_j}$$

The results, presented in Table 26, include the test statistics and p-values at a defined confidence level. For the AQI dataset, we failed to reject the H_0, indicating that the methods are statistically equivalent. However, for the other five datasets, H_0 has been rejected, suggesting significant differences among the forecasting methods.

Table 26 – Kruskal-Wallis Statistical Tests [Bitencourt et al., 2022].

dataset	Statistic	p-value	Result
AEC	204.834	1.09e-40	Reject H_0
KSH1	151.692	1.79e-29	Reject H_0
AQI	7.411	3.87e-01	Fail to Reject H_0
AQB1	113.241	1.95e-21	Reject H_0
AQB6	126.861	2.84e-24	Reject H_0
AQB12	158.052	8.24e-31	Reject H_0

To further compare all methods, we utilized the Wilcoxon method. Table 27 summarizes the statistical rankings of forecasting accuracy, indicating the number of instances where we failed to reject the null hypothesis at the defined confidence level. For example, our method achieved the top rank for the AEC dataset, as we did not reject H_0 in any comparisons with other forecasting methods. Complete results from the Wilcoxon test can be found in the supplementary materials (Section C).

The KPCA-MO-ENSFTS model achieved the top rank in three datasets (AEC, AQB6, and AQB12), followed closely by PCA-MO-ENSFTS, demonstrating the superiority of our methodology in these datasets. In the KSH1 and AQB1 datasets, KPCA-MO-ENSFTS ranked second and third, respectively.

These results affirm the effectiveness of our models in handling high-dimensional datasets, such as AQB12 and AQB6, as well as the AEC dataset. Overall, the KPCA-MO-ENSFTS model demonstrated the best accuracy among the forecasting methods

Table 27 – The summary of the ranking of the forecasting methods [Bitencourt et al., 2022].

Methods	AEC	KSH1	AQB1	AQB6	AQB12
KFTS	1	2	3	1	1
PFTS	2	7	5	2	2
RF	3	2	5	2	2
GRU	4	3	1	4	5
RNN	4	1	2	4	7
LSTM	6	5	3	6	6
SLSTM	7	5	5	3	4
SVR	8	8	8	8	5

evaluated. In conclusion, our proposed methods are both effective and robust for predicting various high-dimensional time series based on accuracy metrics. It also shows the excellent prediction ability of our approach for different datasets.

6.3.4 Parsimony and explainability

As model complexity increases, so do the computational cost and training time. To address this, minimizing parsimony (i.e. the number of parameters or rules in the model) is crucial, especially in IoT applications where models are often deployed on edge devices with limited computational resources (see Section 3.2.3).

Table 28 presents the parsimony (mean number of rules/parameters across 30 experiments) of KPCA-MO-ENSFTS, PCA-MO-ENSFTS, and RNN methods. The MO-ENSFTS methods are significantly more parsimonious than deep learning models like RNNs, particularly on the KSH1 dataset. While RNNs achieved the highest accuracy on the KSH1 dataset, they required a hidden state size of 467 dimensions/units, reflecting their higher number of parameters.

Table 28 – The parsimony of our models against deep learning methods [Bitencourt et al., 2022]

dataset	**KFTS**	**PFTS**	LSTM	GRU	RNN
AEC	141	453	7132	3199	1413
KSH1	169	379	2687612	707532	244268
AQI	161	266	2623	2029	841
AQB1	113	316	67331	19163	6923
AQB6	467	899	4152	2739	1353
AQB12	423	454	2172	1761	939

FTS models, including MO-ENSFTS, are data-driven and white-box models, known for their ease of explanation and auditability—qualities that have gained importance in recent years. However, the explainability of MO-ENSFTS models is influenced by two key factors: the embedding transformation and the number of fuzzy sets. Of these, the number of fuzzy sets has the greatest impact, as increasing the number of fuzzy sets automatically raises the number of rules, which in turn makes the model more complex and harder to interpret.

Chapter 7

Multiple-Output Weighted Multivariate FTS (MO-WMVFTS)

In this chapter, we present another MIMO forecasting methodology known as Multiple-Output Weighted Multivariate Fuzzy Time Series (MO-WMVFTS). This work was published in [Bitencourt et al., 2024]. It builds upon the existing Embedding Weighted Multivariate Fuzzy Time Series (EWMVFTS) [Bitencourt et al., 2023] (see Section 4.4) to accommodate multiple prediction outcomes. The forecast horizon $H = \{1\}$ is one step ahead, and the MO-WMVFTS comprises learning and testing procedures, which are outlined next.

7.1 Learning Procedure

Consider a time series $Y \in \mathbb{R}^{N \times V}$, where each instance $y(n) \in Y$ is observed for $n = 0, 1, ..., N$. Given a specified κ and K, we construct $\mathcal{V}_n^*$ independent and parallel multivariate WMVFTS models $\mathcal{M}_{\mathcal{WMVFTS}}$. These models are designed to capture all relevant information for one-step-ahead forecasting. The learning process, as shown in Figure 21, comprises three main components: embedding, variable partitioning, and rule induction.

7.1.1 Embedding

In this method, the focus is on KPCA embedding techniques (outlined in Section 2.4.2). In this step, the embedding multivariate time series $Y_\gamma \in \mathbb{R}^{N \times K}$ is generated. Notably, this embedding step is common to both the learning and testing procedures.

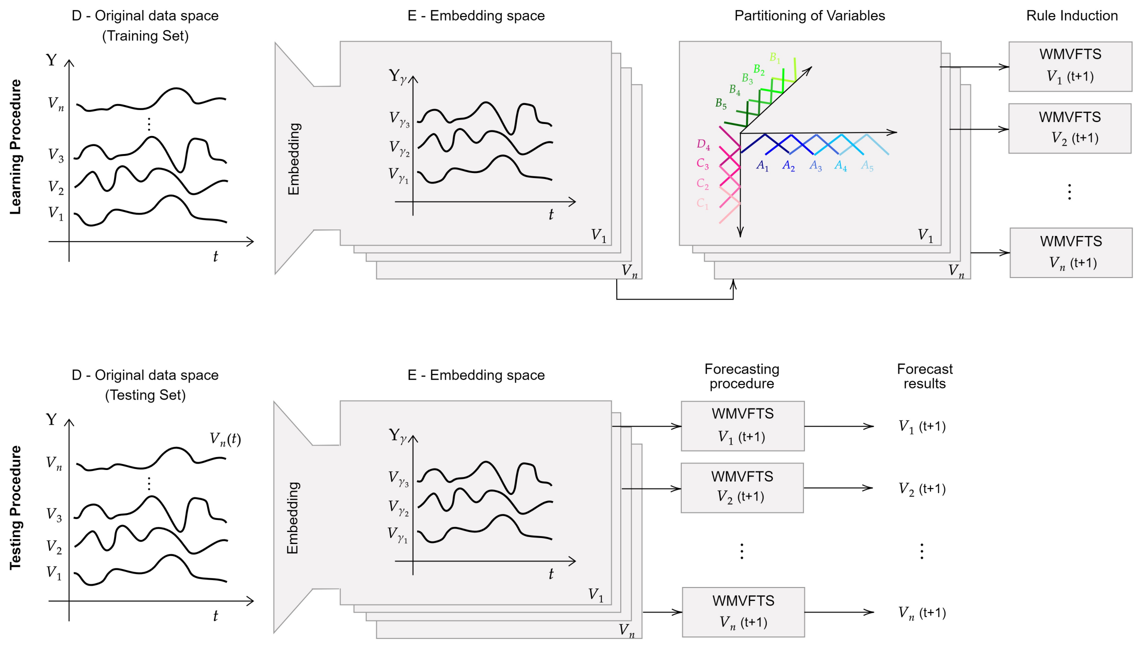

Figure 21 – MO-WMVFTS Learning and Testing procedures [Bitencourt et al., 2024].

7.1.2 Partitioning of Variables

As described as 4.4, UoD is defined as $U_{\mathcal{V}_\gamma} = [lb, ub]$ where $lb = \min(Y_\gamma) - D_1$ and $ub = \max(Y_\gamma) + D_2$, with $D_1 = r \times |\min(Y_\gamma)|$ and $D_2 = r \times |\max(Y_\gamma)|$, $0 < r < 1$. We split $U_{\mathcal{V}_\gamma}$ in κ_i overlapping intervals U_j with midpoints c_j for $j = 0, 1, 2, ..., \kappa_i$, then generating an overlapping fuzzy set $\mathcal{A}_j^{\mathcal{V}_i}$ for each interval with the membership function $\mu_{\mathcal{A}_j}^{\mathcal{V}_i}$, which results a variable $\mathcal{V}_{\gamma i}$ for the variable i and a linguistic variable $\widetilde{\mathcal{V}}_{\gamma i} \in \widetilde{\mathcal{V}}_\gamma$.

7.1.3 Rule Induction

Consequently, Y_γ is transformed into F_γ. Each data point is represented by an array, containing the fuzzified values corresponding to the linguistic variables. The fuzzy memberships for these values are predetermined based on the predefined linguistic variable structure.

$$f(t) \leftarrow \left\{ \mathcal{A}_j^{\mathcal{V}_i} \mid \mu_{\mathcal{A}_j}^{\mathcal{V}_i}(Y_\gamma(t)^i) > 0 \right\} \tag{7.1}$$

Temporal patters with the format $\mathcal{A}_j^{\mathcal{V}_0}, ..., \mathcal{A}_j^{\mathcal{V}_n} \rightarrow \mathcal{A}_j^{\mathcal{V}*}$ are created. The LHS is $f_\gamma(t) = \mathcal{A}_j^{\mathcal{V}_i}$ and the RHS is the target variable $f_\gamma(t+1) = \mathcal{A}_j^{\mathcal{V}^*}$, and both LHS and RHS are related to $\mathcal{A}_j^{\mathcal{V}_i}$ with maximum membership. Fuzzy rules r with the format $LHS \rightarrow w \cdot RHS$ are built in which the weights $w = |\mathcal{A}_j^{\mathcal{V}_i}|/|RHS|$ are defined as follows

$$w_i = \frac{\#\mathcal{A}_j^{\mathcal{V}^*}}{\#RHS} \quad \forall \mathcal{A}_j^{\mathcal{V}^*} \in RHS \tag{7.2}$$

In this context, $\#\mathcal{A}$ is the number of occurrences of $\mathcal{A}_i$ on temporal patterns with the same LHS and $\#RHS$ is the total number of temporal patterns with the same precedent LHS.

7.2 Testing Procedure

The testing set is processed through the MO-WMVFTS models. The embedding procedure is applied to generate the test embedding time series and we fuzzify each variable according to the equation 7.1. We then select r fuzzy rules where any fuzzy set matches the left-hand side (LHS) and compute the rule membership grade using the minimum function T-norm, as per the equation provided.

$$\mu_q = \bigcap_{j \in \widetilde{\mathcal{V}}_i \; ; \; i \in \mathcal{V}} \mu_j i \tag{7.3}$$

Next, we calculate the mean point mp_q of $\mathcal{V}^*$ as follows

$$mp_q = \sum_{j \in \widetilde{\mathcal{V}}_i^*} w_j \cdot c_j \tag{7.4}$$

where w_j is the weights calculated according to 7.2.

Finally, the forecast value $\hat{y}(t + h_i)$ is calculated as a weighted sum of the rule midpoints, with the weights determined by their corresponding membership grades μ_{A_j}. This method ensures that the contribution of each rule is proportional to its membership grade.

$$\hat{\mathcal{V}^*}_i(t+1) = \sum_{q \in r} \mu_q \cdot mp_q \tag{7.5}$$

The testing procedure is shown in Figure 21 and Figure 22 presents a graphic illustration of the learning and forecasting phases.

7.2.1 Hyperparameters setting

Similar to our forecasting methods discussed earlier, the most critical hyperparameters (HP) in MO-WMVFTS are the embedding dimension and the number of fuzzy sets. Through trial and error, various combinations of HP were tested to determine the best values found configuration: $K \in \{2, 3, 4, 5\}$ and $\kappa \in \{30, 35, 40, 45\}$. For all datasets,

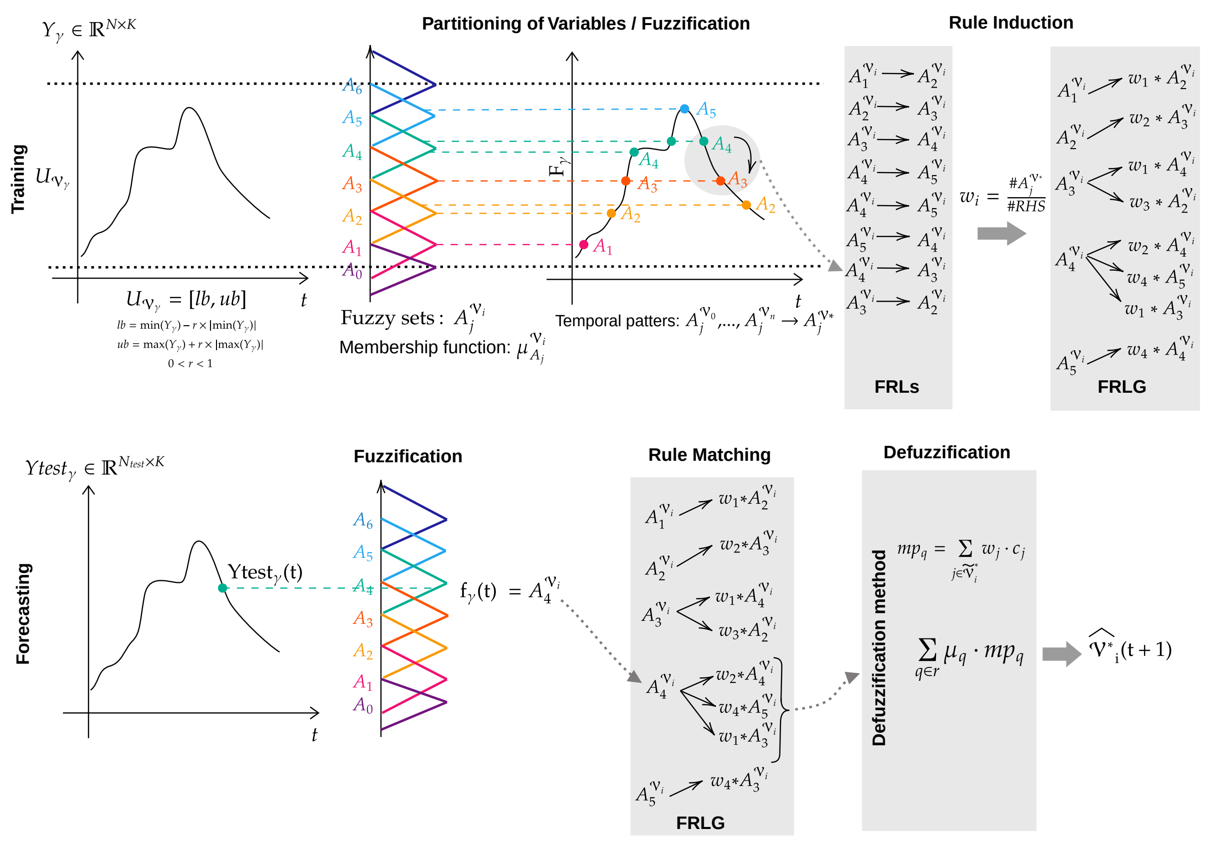
Training
Partitioning of Variables / Fuzzification
Rule Induction
$Y_\gamma \in \mathbb{R}^{N\times K}$
$U_{\mathcal{V}_\gamma} = [lb, ub]$
$lb = \min(Y_\gamma) - r \times |\min(Y_\gamma)|$
$ub = \max(Y_\gamma) + r \times |\max(Y_\gamma)|$
$0 < r < 1$
Fuzzy sets : $A_j^{\mathcal{V}_i}$
Membership function: $\mu_{A_j}^{\mathcal{V}_i}$
Temporal patters: $A_j^{\mathcal{V}_0},\dots, A_j^{\mathcal{V}_n} \rightarrow A_j^{\mathcal{V}*}$
FRLs
$w_i = \frac{\#A_j^{\mathcal{V}*}}{\#RHS}$
FRLG
Forecasting
$Ytest_\gamma \in \mathbb{R}^{N_{test}\times K}$
Fuzzification
$\mathrm{Ytest}_\gamma(t)$
$f_\gamma(t) = A_4^{\mathcal{V}_i}$
Rule Matching
Defuzzification
Defuzzification method
$mp_q = \sum_{j\in\widetilde{\mathcal{V}}_i^*} w_j \cdot c_j$
$\sum_{q\in r} \mu_q \cdot mp_q$
$\widehat{\mathcal{V}^*}_i(t+1)$

Figure 22 – Graphic illustration of the learning and forecasting procedures.

κ was configured with $\kappa = 30$ while K was settled as 2 (AEC), 3 (AQB12), 4 (AQB6, KSH10) and 5 (AQB1).

7.3 Results

In this section, we present the experimental results obtained from applying our proposed methods and compares them with various forecasting approaches tested on the datasets. We provide a brief overview of the selected baseline methods along with their hyperparameter settings. A comparative analysis of the forecast results generated by MO-WMVFTS versus these baseline methods is then performed. Finally, we assess the accuracy of our method against several existing MIMO forecasting models from the literature.

7.3.1 MO-WMFTS versus baseline methods

The introduced methods accuracy was compared to popular deep learning baseline MIMO forecasting methods like Temporal Convolutional Networks (TCN), LSTM, and

Table 29 – best values found values of hyperparameters for the TCN, LSTM, and BiLSTM methods [Bitencourt et al., 2024].

Hyperparameters	AQB1		AQB6		AEC		KSH10	
	LSTM / BiLSTM	TCN	LSTM / BiLSTM	TCN	LSTM / BiLSTM	TCN	LSTM / BiLSTM	TCN
Number of residual block stacks	-	2	-	1	-	1	-	2
Convolution filter size	-	16	-	32	-	64	-	62
Number of conv. layers	-	4	-	5	-	4	-	4
Number of conv. layer filters	-	11	-	11	-	11	-	11
Number of units	79	-	75	-	54	-	75	-
Number of LSTM layers	5	-	4	-	2	-	2	-
Dropout percentage	0.1	0.1	0.2	0.05	0.2	0.1	0.2	0.1
Batch normalization	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 30 – Model's accuracy in terms of mean NRMSE [Bitencourt et al., 2024]

Methods	**AEC**	**AQB6**	**AQB1**	**KSH10**
LSTM	2.96	2.20	2.35	6.44
BiLSTM	1.07	1.31	0.45	6.93
TCN	1.75	1.10	0.74	1.79
MO-WMVFTS	**0.022**	**0.036**	**0.074**	**0.055**

BiLSTM, all optimized using Randomized Search for hyperparameters (see Table 29). Adam optimizer and ReLU activation were used across methods based on their proven effectiveness in various problems [Schmidt et al., 2021].

Given the complexity of the datasets with numerous variables, the comparison focused on the mean NRMSE across all target variables per dataset. Table 30 highlights the results, showing that the proposed MO-MVFTS approach outperformed baseline techniques in terms of mean NMRSE. Full results, including RMSE, NRMSE, and MAPE, are provided in supplementary materials.

Deep learning methods often struggle with high-dimensional time series forecasting in MIMO problems due to challenges like redundant variables, reduced training efficiency, and complexity in tuning hyperparameters. Additionally, deep learning models tend to have limited explainability. In contrast, FTS methods offer better interpretability and can be a viable alternative. Combining embedding transformation with FTS can effectively handle high-dimensional MIMO time series forecasting by extracting relevant information and improving forecast accuracy.

7.3.2 MO-WMFTS versus competitors methods

We also compare the accuracy of MO-WMVFTS with several recently published MIMO fuzzy-based forecasting methods, including MRHFCM [Orang et al., 2024a], M-PRFCM [Orang et al., 2024b], and our other MIMO approachs published in [Bitencourt et al., 2022] and detailed in Chapter 6 (i.e. PCA-MO-ENSFTS, and KPCA-MO-ENSFTS). Table 31 presents the mean NRMSE results for all variables, providing a comparison between the competitor models and our proposed approach across various datasets.

Table 31 – MO-WMVFTS versus competitors methods in terms of mean NRMSE [Bitencourt et al., 2024]

Methods	**AEC**	**AQB6**	**KSH10**	**AQB12**
KPCA-MO-ENSFTS	0.043	0.098	0.09	0.109
PCA-MO-ENSFTS	0.063	0.122	0.126	0.126
MRHFCM	0.086	0.097	0.091	0.109
M-PRFCM	0.06	0.095	0.086	0.107
MO-WMVFTS	**0.022**	**0.036**	**0.055**	**0.022**

MO-WMVFTS demonstrates considerable superiority over all the competitor methods. The results indicate that MO-WMVFTS achieves the highest prediction accuracy across the datasets. Accordingly, Table 32 presents the skill score index of MO-WMVFTS to some competitor methods, where the selected accuracy metric was the mean NRMSE across all variables.

Table 32 – Skill score of MO-WMVFTS to competitors' methods [Bitencourt et al., 2024]

Methods	**AEC**	**AQB6**	**KSH10**	**AQB12**
KPCA-MO-ENSFTS	0.488	0.633	0.389	0.798
PCA-MO-ENSFTS	0.651	0.705	0.563	0.825
MRHFCM	0.744	0.629	0.396	0.798
M-PRFCM	0.633	0.621	0.360	0.794

MO-WMVFTS demonstrates significant performance improvements across all datasets. On the AQB12 dataset, it outperforms competitor methods by over 79%, and on AQB6, it surpasses them by more than 62%. For the AEC dataset, MO-WMVFTS beats KPCA-MO-EFTS by 48% and exceeds other models, including PCA-MO-EFTS, MRHFCM, and M-PRFCM, by over 63%. On the KSH10 dataset, MO-WMVFTS improves by 56% compared to PCA-MO-EFTS and over 36% against other methods.

In summary, the experimental results show that MO-WMVFTS outperforms some MIMO deep learning and fuzzy-based methods in accuracy, making it promising for high-dimensional time series forecasting in IoT applications. Additionally, it produces lightweight models suited for low-power IoT devices with limited computational resources.

Chapter 8

Conclusions and Future Works

This study explores the advantages of combining embedding transformations with fuzzy time series to address high-dimensional, non-stationary time series in Internet of Things (IoT) applications. High-dimensional time series pose significant challenges, as existing FTS models can become impractical when training with numerous variables. However, the flexibility, simplicity, interpretability, and accuracy of FTS methods make them a promising solution for many IoT use cases.

Accordingly, in high-dimensional time series forecasting, each variable depends on its historical values and other contemporaneous variables, making appropriate variable selection essential. Embedding techniques help create a new variable space that captures the complexity of multivariate time series, reduces collinearity, and reveals latent interactions. These algorithms extract relevant information for accurate forecasting of the target variable. Additionally, the FTS learning approach effectively manages non-stationary time series and adapts to scenarios with concept drift.

We introduced a new framework, $embEFTS$ (Embedding Fuzzy Time Series), designed to handle high-dimensional non-stationary data by combining data embedding techniques and fuzzy time series models in a reduced-dimensional space. This work brings the following novel contributions:

1. **EFTS** [Bitencourt et al., 2023] (Chapter 4): A first-order MISO methodology for forecasting high-dimensional non-stationary time series, tested in smart buildings for energy consumption prediction. PCA and autoencoder algorithms were applied to create a new feature space for more effective forecasting.

2. **MS-EFTS** [Bitencourt et al., 2025] (Chapter 5): A multi-step forecasting method that incorporates embedding transformations and weighted FTS models to create parsimonious models. It is designed for direct multi-step-ahead forecasting in IoT applications.

3. **MO-ENSFTS** [Bitencourt et al., 2022] (Chapter 6): A first-order MIMO approach where all variables act as both target and explanatory variables. It was tested on high-dimensional datasets for smart cities and smart buildings, using PCA and KPCA to identify key components for forecasting.

4. **MO-WMVFTS** [Bitencourt et al., 2024] (Chapter 7): A MIMO one-step-ahead forecasting model that integrates weighted multivariate FTS and embedding transformation, with a structure different from MO-ENSFTS, tailored for smart city and building applications.

The proposed *embEFTS* framework was evaluated against various machine learning and deep learning methods, demonstrating superior performance in handling high-dimensional non-stationary time series with 7 to 132 variables across different forecasting horizons. The results highlighted the efficiency and accuracy of *embEFTS*, particularly in IoT applications, where it outperformed other methods.

Furthermore, *embEFTS* delivers simple yet effective forecasting models that ensure reliable predictions for non-stationary high-dimensional time series, emphasizing parsimony in predictive modeling. Minimizing model complexity is crucial for reducing computational cost and training time, especially in IoT applications with resource-limited edge devices. Additionally, greater parsimony enhances model explainability, making it easier to interpret and deploy.

embEFTS is a robust and effective framework for forecasting high-dimensional time series in both MISO and MIMO scenarios, suitable for one-step and multi-step predictions. It proved especially effective for predicting smart building energy consumption and air quality in smart cities, showing potential for optimizing power usage and aiding air pollution management.

The framework is parsimonious, readable, and explainable, with accuracy primarily controlled by two hyperparameters: the number of fuzzy sets (κ) and the embedding dimension (K). Increasing these hyperparameters can enhance the model's accuracy, making *embEFTS* a versatile and valuable tool for various IoT applications.

FTS models, including *embEFTS*, are data-driven, white-box models that are easy to explain and audit, a feature that has become increasingly important. The explainability of *embEFTS* models is influenced by two factors: the embedding function and the number of fuzzy sets. The number of fuzzy sets has the greatest impact, as increasing them raises the number of rules and makes the model more complex and harder to interpret. However, the embedding function itself does not significantly hinder explainability, as it is invertible and allows fuzzy set values to be converted back into the original data space. Additionally, the explainability of FTS models depends on the active fuzzy rules and their length.

A limitation of the proposed framework is the inability to determine the most suitable embedding algorithm for a dataset in advance. Future challenges could involve analyzing the properties of the feature spaces to design a technique for selecting the optimal embedding method. Additionally, extending the *embEFTS* framework to allow users to plug in and out embedding algorithms could offer more flexibility.

Finding the optimal hyperparameters (K and κ) is essential for balancing low error and high explainability, as increasing these values raises model complexity and training time exponentially. Grid search for hyperparameter tuning is time-consuming, and the current framework does not automatically select hyperparameters based on the dataset, making this another area for future investigation.

Moreover, *embEFTS* does not inherently handle missing values and outliers, requiring pre-processing steps like imputation and outlier removal before application. These limitations suggest areas for future work, including enhancing the framework's adaptability and data-handling capabilities.

When evaluating EFTS (Chapter 4), it is also important to consider the time required to train the autoencoder, as it is a neural network that takes considerable time to optimize the reconstruction error. The number of AE parameters must also be included in the parsimony and explainability analysis, alongside the number of fuzzy rules.

MO-WMVFTS (Chapter 7) offers high precision but faces limitations such as high computational costs and extended training time, due to the individual models required for each target variable. While parallel processing can alleviate some of these issues, it is not ideal for real-time data streams or detecting adaptive data patterns. However, it performs well on non-stationary time series.

Similarly, MS-EFTS (Chapter 5) also achieves high accuracy, but its training time increases as the number of models (corresponding to time steps) rises, leading to greater computational costs. However, both learning and testing procedures can be parallelized or distributed, allowing for faster execution in high-overhead processing tasks.

We evaluated our proposed framework, *embEFTS*, in smart building and smart city IoT applications. However, *embEFTS* is highly flexible and adaptable, making it suitable for a wide range of IoT time series forecasting tasks, including healthcare, underground and underwater monitoring, logistics, and transportation systems. Beyond IoT data, embEFTS is also well-suited for general time series forecasting, including applications such as stock price prediction, sales forecasting, and industrial process outcome prediction. Furthermore, this work opens the possibility of applying *embEFTS* to unstructured and high-dimensional data problems, provided that an appropriate embedding algorithm is used. For example, it can be utilized for photovoltaic power forecasting using sky images or predicting cloud cover at solar photovoltaic plants based on satellite imagery.

8.1 Future Works

Guided by the previously mentioned conclusions, this work presents some future investigations, which the main ones being listed below:

1. **Future work 1**: Future work could involve enhancing the framework to allow flexible integration of various FTS methods (both univariate and multivariate) and embedding models, enabling users to customize their configurations. Additionally, exploring different combinations of FTS methods and embedding algorithms may offer further improvements and new insights.

2. **Future work 2**: Future work could explore the properties of variable spaces generated by embedding functions like Principal Component Analysis, Kernel Principal Component Analysis, and Autoencoders. Additionally, the integration of other embedding algorithms may lead to further enhancements in performance and adaptability.

3. **Future work 3**: Selecting optimal hyperparameters (κ and K) through grid search can improve accuracy by minimizing errors across different datasets. However, this process is time-consuming. Future research could explore developing methods that automatically determine hyperparameters based on the dataset and specific IoT application, streamlining the optimization process.

4. **Future work 4**: Data-driven methods, including our proposed methods, play a key role in detecting drifts in data streams. Future work could focus on developing drift detection techniques that enable dynamic adaptation of fuzzy sets, allowing the model to adjust to changes in high-dimensional non-stationary time series. Integrating such methods into our framework would enhance its adaptability and performance.

5. **Future work 5**: As previously mentioned, our proposed framework is adaptable to various time series forecasting tasks. Additionally, this work introduces the potential of applying *embEFTS* to unstructured data. Future research could explore its application in photovoltaic power forecasting using sky images or predicting cloud cover at solar photovoltaic plants with satellite imagery.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

R. Ahmed, V. Sreeram, Y. Mishra, and M. Arif. A review and evaluation of the state-of-the-art in pv solar power forecasting: Techniques and optimization. *RENEWABLE & SUSTAINABLE ENERGY REVIEWS*, 124, May 2020. ISSN 1364-0321. doi: 10.1016/j.rser.2020.109792.

T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

M. H. Alobaidi, F. Chebana, and M. A. Meguid. Robust ensemble learning framework for day-ahead forecasting of household based energy consumption. *Applied Energy*, 212:997–1012, 2018. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2017.12.054. URL https://www.sciencedirect.com/science/article/pii/S0306261917317695.

S. Ameer, M. A. Shah, A. Khan, H. Song, C. Maple, S. U. Islam, and M. N. Asghar. Comparative analysis of machine learning techniques for predicting air quality in smart cities. *IEEE Access*, 7:128325–128338, 2019. doi: 10.1109/ACCESS.2019.2925082.

M. Antal, L. Toderean, T. Cioara, and I. Anghel. Hybrid deep neural network model for multi-step energy prediction of prosumers. *Applied Sciences*, 12(11):5346, 2022.

S. B. Atitallah, M. Driss, W. Boulila, and H. B. Ghézala. Leveraging deep learning and iot big data analytics to support the smart cities development: Review and future directions. *Computer Science Review*, 38:100303, 2020. ISSN 1574-0137. doi: https://doi.

org/10.1016/j.cosrev.2020.100303. URL https://www.sciencedirect.com/science/article/pii/S1574013720304032.

S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

A. Bekkar, B. Hssina, S. Douzi, and K. Douzi. Air-pollution prediction in smart city, deep learning approach. *Journal of big Data*, 8(1):1–21, 2021.

J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. pages 115–123, 2013.

H. V. Bitencourt and F. G. Guimarães. High-dimensional multivariate time series forecasting in IoT applications using embedding non-stationary fuzzy time series. In *2021 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6, 2021. doi: 10.1109/LA-CCI48322.2021.9769792.

H. V. Bitencourt, O. Orang, L. A. F. de Souza, P. C. L. Silva, and F. G. Guimarães. An embedding-based non-stationary fuzzy time series method for multiple output high-dimensional multivariate time series forecasting in iot applications. *Neural Comput. Appl.*, 35(13):9407–9420, dec 2022. ISSN 0941-0643. doi: 10.1007/s00521-022-08120-5. URL https://doi.org/10.1007/s00521-022-08120-5.

H. V. Bitencourt, L. A. F. de Souza, M. C. dos Santos, R. Silva, P. C. de Lima e Silva, and F. G. Guimarães. Combining embeddings and fuzzy time series for high-dimensional time series forecasting in internet of energy applications. *Energy*, 271: 127072, 2023. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2023.127072. URL https://www.sciencedirect.com/science/article/pii/S0360544223004668.

H. V. Bitencourt, P. de Oliveira Lucas, O. Orang, P. Silva, and F. G. Guimarães. A weighted multivariate fuzzy time series method for multiple output high-dimensional time series forecasting in iot applications. In *2024 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, pages 1–6, 2024. doi: 10.1109/LA-CCI62337.2024.10814884.

H. V. Bitencourt, P. d. O. Lucas, O. Orang, P. C. L. Silva, and F. G. Guimarães. A multi-step multivariate fuzzy-based time series forecasting on internet of things data. *IEEE Internet of Things Journal*, pages 1–1, 2025. doi: 10.1109/JIOT.2025.3549715.

M. Bose and K. Mali. Designing fuzzy time series forecasting models: A survey. *International Journal of Approximate Reasoning*, 111:78–99, 2019.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

L. M. Candanedo, V. Feldheim, and D. Deramaix. Data driven prediction models of energy use of appliances in a low-energy house. *Energy and Buildings*, 140:81–97, 2017. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2017.01.083. URL https://www.sciencedirect.com/science/article/pii/S0378778816308970.

L. Cascone, S. Sadiq, S. Ullah, S. Mirjalili, H. U. R. Siddiqui, and M. Umer. Predicting household electric power consumption using multi-step time series with convolutional lstm. *Big Data Research*, 31:100360, 2023. ISSN 2214-5796. doi: https://doi.org/10.1016/j.bdr.2022.100360. URL https://www.sciencedirect.com/science/article/pii/S2214579622000545.

M. Chammas, A. Makhoul, and J. Demerjian. An efficient data model for energy prediction using wireless sensors. *Comput. Electr. Eng.*, 76:249–257, 2019. doi: 10.1016/j.compeleceng.2019.04.002. URL https://doi.org/10.1016/j.compeleceng.2019.04.002.

R. Chandra, S. Goyal, and R. Gupta. Evaluation of deep learning models for multi-step ahead time series prediction. *IEEE Access*, PP:1–1, 05 2021. doi: 10.1109/ACCESS.2021.3085085.

Y.-W. Cheung and K. S. Lai. Lag order and critical values of the augmented dickey–fuller test. *Journal of Business & Economic Statistics*, 13(3):277–280, 1995.

F. Chollet et al. Keras. https://keras.io, 2015.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

U. K. Das, K. S. Tey, M. Seyedmahmoudian, S. Mekhilef, M. Y. I. Idris, W. V. Deventer, B. Horan, and A. Stojcevski. Forecasting of photovoltaic power generation and model optimization: A review. *Renewable and Sustainable Energy Reviews*, 81:912 – 928, 2018. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2017.08.017. URL http://www.sciencedirect.com/science/article/pii/S1364032117311620.

P. C. de Lima Silva, e Patrícia de Oliveira e Lucas, and F. G. Guimarães. A distributed algorithm for scalable fuzzy time series. In *International Conference on Green, Pervasive, and Cloud Computing*, pages 42–56. Springer, 2019. doi: 10.1007/978-3-030-19223-5_4. URL https://doi.org/10.1007/978-3-030-19223-5_4.

P. C. de Lima Silva, C. A. S. Junior, M. A. Alves, R. Silva, M. Weiss-Cohen, and F. G. Guimarães. Forecasting in non-stationary environments with fuzzy time series. *Applied Soft Computing*, 97(Part B):106825, 2020. doi: 10.1016/j.asoc.2020.106825. URL https://doi.org/10.1016/j.asoc.2020.106825.

S. De Vito, M. Piga, L. Martinotto, and G. Francia. Co, no2 and nox urban pollution monitoring with on-field calibrated electronic nose by automatic bayesian regularization. *Sensors and Actuators B: Chemical*, 143:182–191, 12 2009. doi: 10.1016/j.snb.2009.08.041.

G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *IEEE Comp. Int. Mag.*, 10(4):12–25, 2015. doi: 10.1109/MCI.2015.2471196. URL https://doi.org/10.1109/MCI.2015.2471196.

M. C. dos Santos, F. G. Guimarães, and P. C. d. L. Silva. High-dimensional multivariate time series forecasting using self-organizing maps and fuzzy time series. In *2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6, 2021. doi: 10.1109/FUZZ45933.2021.9494496.

H. Drucker, C. J. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9, 1996.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

H. Fei, X. Wu, and C. Luo. A model-driven and data-driven fusion framework for accurate air quality prediction. *arXiv preprint arXiv:1912.07367*, 2019.

G. Feng, L. Zhang, J. Yang, and W. Lu. Long-term prediction of time series using fuzzy cognitive maps. *Engineering Applications of Artificial Intelligence*, 102:104274, 2021.

P. J. Freire, S. Srivallapanondh, A. Napoli, J. E. Prilepsky, and S. K. Turitsyn. Computational complexity evaluation of neural network applications in signal processing, 2022. URL https://arxiv.org/abs/2206.12191.

W. Froelich and J. L. Salmeron. Evolutionary learning of fuzzy grey cognitive maps for the forecasting of multivariate, interval-valued time series. *International Journal of Approximate Reasoning*, 55(6):1319–1335, 2014. ISSN 0888-613X. doi: https://doi.org/10.1016/j.ijar.2014.02.006. URL https://www.sciencedirect.com/science/article/pii/S0888613X14000358.

W. Froelich, E. I. Papageorgiou, M. Samarinas, and K. Skriapas. Application of evolutionary fuzzy cognitive maps to the long-term prediction of prostate cancer. *Appl. Soft Comput.*, 12:3810–3817, 2012.

K. P. F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11): 559–572, 1901. doi: 10.1080/14786440109462720.

J. M. Garibaldi and T. Ozen. Uncertain fuzzy reasoning: A case study in modelling expert decision making. *IEEE Trans. Fuzzy Syst.*, 15(1):16–30, 2007.

J. M. Garibaldi, M. Jaroszewski, and S. Musikasuwan. Nonstationary fuzzy sets. *IEEE Trans. Fuzzy Syst.*, 16(4):1072–1086, 2008.

A. Gensler, J. Henze, B. Sick, and N. Raabe. Deep learning for solar power forecasting - an approach using autoencoder and lstm neural networks. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 002858–002865, 2016. doi: 10.1109/SMC.2016.7844673.

S. Gocheva-Ilieva, A. Ivanov, and H. Kulina. Special Issue "Statistical Data Modeling and Machine Learning with Applications II". *Mathematics*, 11(12):1–4, June 2023a. URL https://ideas.repec.org/a/gam/jmathe/v11y2023i12p2775-d1174813.html.

S. Gocheva-Ilieva, A. Ivanov, H. Kulina, and M. Stoimenova-Minova. Multi-step ahead ex-ante forecasting of air pollutants using machine learning. *Mathematics*, 11(7), 2023b. ISSN 2227-7390. doi: 10.3390/math11071566. URL https://www.mdpi.com/2227-7390/11/7/1566.

J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.*, 29(7): 1645–1660, 2013. doi: 10.1016/j.future.2013.01.010. URL https://doi.org/10.1016/j.future.2013.01.010.

S. Herbold. Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, 5(48):2173, 2020. doi: 10.21105/joss.02173. URL https://doi.org/10.21105/joss.02173.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

M. Jaradat, M. Jarrah, A. Bousselham, Y. Jararweh, and M. Al-Ayyoub. The internet of energy: Smart sensor networks and big data management for smart grid. *Procedia Computer Science*, 56:592–597, 2015a. doi: 10.1016/j.procs.2015.07.250. URL https://doi.org/10.1016%2Fj.procs.2015.07.250.

M. Jaradat, M. H. A. Jarrah, A. Bousselham, Y. Jararweh, and M. Al-Ayyoub. The internet of energy: Smart sensor networks and big data management for smart grid. In *The 10th International Conference on Future Networks and Communications (FNC 2015) / The 12th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2015) / Affiliated Workshops, August 17-20, 2015, Belfort, France*, volume 56 of *Procedia Computer Science*, pages 592–597. Elsevier, 2015b. doi: 10.1016/j.procs.2015.07.250. URL https://doi.org/10.1016/j.procs.2015.07.250.

N. Jin, Y. Zeng, K. Yan, and Z. Ji. Multivariate air quality forecasting with nested long short term memory neural network. *IEEE Transactions on Industrial Informatics*, 17 (12):8514–8522, 2021.

I. M. Johnstone and A. Y. Lu. Sparse principal components analysis. *arXiv preprint arXiv:0901.4392*, 2009.

Kaggle. Smart home dataset with weather information. https://www.kaggle.com/taranvee/smart-home-dataset-with-weather-information, 2021. accessed on 28 Ago 2021.

K. I. Kim, M. O. Franz, and B. Schölkopf. Iterative kernel principal component analysis for image modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(9):1351–1366, 2005. doi: 10.1109/TPAMI.2005.181. URL https://doi.org/10.1109/TPAMI.2005.181.

T.-Y. Kim and S.-B. Cho. Predicting residential energy consumption using cnn-lstm neural networks. *Energy*, 182:72–81, 2019. ISSN 0360-5442. doi: https://doi.org/10.1016/j.energy.2019.05.230. URL https://www.sciencedirect.com/science/article/pii/S0360544219311223.

W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159–178, 1992. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(92)90104-Y. URL https://www.sciencedirect.com/science/article/pii/030440769290104Y.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

P. C. d. Lima e Silva, C. A. Severiano Jr, M. A. Alves, M. W. Cohen, and F. G. Guimarães. A new granular approach for multivariate forecasting. In *Latin American Workshop on Computational Neuroscience*, pages 41–58. Springer, 2019.

B. Liu, S. Yan, J. Li, G. Qu, Y. Li, J. Lang, and R. Gu. A sequence-to-sequence air quality predictor based on the n-step recurrent prediction. *IEEE Access*, 7:43331–43345, 2019. URL https://api.semanticscholar.org/CorpusID:115196166.

G. Lobaccaro, S. Carlucci, and E. Löfström. A review of systems and technologies for smart homes and smart grids. *Energies*, 9(5), 2016. ISSN 1996-1073. doi: 10.3390/en9050348. URL https://www.mdpi.com/1996-1073/9/5/348.

Y. Lu, Z. Tian, R. Zhou, and W. Liu. Multi-step-ahead prediction of thermal load in regional energy system using deep learning method. *Energy and Buildings*, 233:

110658, 2021. ISSN 0378-7788. doi: https://doi.org/10.1016/j.enbuild.2020.110658. URL https://www.sciencedirect.com/science/article/pii/S0378778820334447.

P. O. Lucas, O. Orang, P. C. Silva, E. M. Mendes, and F. G. Guimaraes. A tutorial on fuzzy time series forecasting models: recent advances and challenges. *Learn Nonlinear Models*, 19:29–50, 2022.

J. Ma, J. C. Cheng, C. Lin, Y. Tan, and J. Zhang. Improving air quality prediction accuracy at larger temporal resolutions using deep learning and transfer learning techniques. *Atmospheric Environment*, 214:116885, 2019.

M. Manic, K. Amarasinghe, J. J. Rodriguez-Andina, and C. Rieger. Intelligent buildings of the future: Cyberaware, deep learning powered, and human interacting. *IEEE Industrial Electronics Magazine*, 10(4):32–49, 2016. doi: 10.1109/MIE.2016.2615575.

D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2012.02.016. URL https://www.sciencedirect.com/science/article/pii/S1570870512000674.

E. Mocanu, P. H. Nguyen, M. Gibescu, and W. L. Kling. Deep learning for estimating building energy consumption. *Sustainable Energy, Grids and Networks*, 6:91–99, 2016. ISSN 2352-4677. doi: https://doi.org/10.1016/j.segan.2016.02.005. URL https://www.sciencedirect.com/science/article/pii/S2352467716000163.

M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani. Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys Tutorials*, 20(4):2923–2960, 2018. doi: 10.1109/COMST.2018.2844341.

D. Moldovan and A. Slowik. Energy consumption prediction of appliances using machine learning and multi-objective binary grey wolf optimization for feature selection. *Applied Soft Computing*, 111:107745, 2021. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2021.107745. URL https://www.sciencedirect.com/science/article/pii/S1568494621006669.

L. Munkhdalai, T. Munkhdalai, K. H. Park, T. Amarbayasgalan, E. Batbaatar, H. W. Park, and K. H. Ryu. An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series. *IEEE Access*, 7:99099–99114, 2019. doi: 10.1109/ACCESS.2019.2930069.

R. Olu-Ajayi, H. Alaka, I. Sulaimon, F. Sunmola, and S. Ajayi. Building energy consumption prediction for residential buildings using deep learning and other machine learning techniques. *Journal of Building Engineering*, 45:103406, 2022. ISSN 2352-7102. doi:

https://doi.org/10.1016/j.jobe.2021.103406. URL https://www.sciencedirect.com/science/article/pii/S235271022101264X.

O. Orang, H. V. Bitencourt, L. A. F. de Souza, P. d. O. Lucas, P. C. L. Silva, and F. G. Guimarães. Multiple-input multiple-output randomized fuzzy cognitive map method for high-dimensional time series forecasting. *IEEE Transactions on Fuzzy Systems*, pages 1–13, 2024a. doi: 10.1109/TFUZZ.2024.3379853.

O. Orang, H. V. Bitencourt, P. C. d. L. e. Silva, and F. G. Guimarães. Time series forecasting using parallel randomized fuzzy cognitive maps and reservoir computing. In F. P. García Márquez, A. Jamil, A. A. Hameed, and I. Segovia Ramírez, editors, *Emerging Trends and Applications in Artificial Intelligence*, pages 50–61, Cham, 2024b. Springer Nature Switzerland. ISBN 978-3-031-56728-5.

L. Palomero, V. García, and J. S. Sánchez. Fuzzy-based time series forecasting and modelling: A bibliometric analysis. *Applied Sciences*, 12(14), 2022. ISSN 2076-3417. doi: 10.3390/app12146894. URL https://www.mdpi.com/2076-3417/12/14/6894.

E. I. Papageorgiou and W. Froelich. Multi-step prediction of pulmonary infection with the use of evolutionary fuzzy cognitive maps. *Neurocomputing*, 92:28–35, 2012. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2011.08.034. URL https://www.sciencedirect.com/science/article/pii/S0925231212000847. Data Mining Applications and Case Study.

T. Parhizkar, E. Rafieipour, and A. Parhizkar. Evaluation and improvement of energy consumption prediction models using principal component analysis based feature reduction. *Journal of Cleaner Production*, 279:123866, 2021. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2020.123866. URL https://www.sciencedirect.com/science/article/pii/S0959652620339111.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

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.

Y. Qi, Q. Li, H. Karimian, and D. Liu. A hybrid model for spatiotemporal forecasting of pm2.5 based on graph convolutional neural network and long short-term memory. *Science of The Total Environment*, 664:1–10, 2019. ISSN 0048-9697. doi: https://doi.org/10.1016/j.scitotenv.2019.01.333. URL https://www.sciencedirect.com/science/article/pii/S0048969719303821.

S. Reka. Future effectual role of energy delivery: A comprehensive review of internet of things and smart grid. *Renewable and Sustainable Energy Reviews*, 91, 04 2018. doi: 10.1016/j.rser.2018.03.089.

O. Reyes and S. Ventura. Performing multi-target regression via a parameter sharing-based deep network. *International journal of neural systems*, 29(09):1950014, 2019.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. pages 318–362, 1986.

M. Sajjad, Z. A. Khan, A. Ullah, T. Hussain, W. Ullah, M. Y. Lee, and S. W. Baik. A novel cnn-gru-based hybrid approach for short-term residential load forecasting. *IEEE Access*, 8:143759–143768, 2020. doi: 10.1109/ACCESS.2020.3009537.

K. K. R. Samal, K. S. Babu, and S. K. Das. Multi-output spatio-temporal air pollution forecasting using neural network approach. *Applied Soft Computing*, 126:109316, 2022. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2022.109316. URL https://www.sciencedirect.com/science/article/pii/S1568494622004914.

R. M. Schmidt, F. Schneider, and P. Hennig. Descending through a crowded valley - benchmarking deep learning optimizers, 2021. URL https://arxiv.org/abs/2007.01547.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

S. Sepasgozar, R. Karimi, L. Farahzadi, F. Moezzi, S. Shirowzhan, S. M. Ebrahimzadeh, F. Hui, and L. Aye. A systematic content review of artificial intelligence and the internet of things applications in smart home. *Applied Sciences*, 10(9), 2020. ISSN 2076-3417. URL https://www.mdpi.com/2076-3417/10/9/3074.

C. A. Severiano, P. C. de Lima Silva, M. Weiss Cohen, and F. G. Guimarães. Evolving fuzzy time series for spatio-temporal forecasting in renewable energy systems. *Renewable Energy*, 171:764–783, 2021. ISSN 0960-1481. doi: https://doi.org/10.1016/j.renene.2021.02.117. URL https://www.sciencedirect.com/science/article/pii/S0960148121002962.

H. Shahinzadeh, J. Moradi, G. B. Gharehpetian, H. Nafisi, and M. Abedi. Internet of energy (ioe) in smart power systems. pages 627–636, 02 2019. doi: 10.1109/KBEI.2019.8735086.

Y. Shahzad, H. Javed, H. Farman, J. Ahmad, B. Jan, and M. Zubair. Internet of energy: Opportunities, applications, architectures and challenges in smart industries. *Comput. Electr. Eng.*, 86:106739, 2020. doi: 10.1016/j.compeleceng.2020.106739. URL https://doi.org/10.1016/j.compeleceng.2020.106739.

P. Silva, C. Severiano, M. Alves, M. Weiss, and F. Guimarães. *A New Granular Approach for Multivariate Forecasting*, pages 41–58. 11 2019a. ISBN 978-3-030-36635-3. doi: 10.1007/978-3-030-36636-0_4.

P. C. Silva, H. J. Sadaei, R. Ballini, and F. G. Guimarães. Probabilistic forecasting with fuzzy time series. *IEEE Transactions on Fuzzy Systems*, 2019b. URL http://doi.org/10.1109/TFUZZ.2019.2922152.

P. C. d. L. Silva. pyfts : Fuzzy time series for python, 03 2016.

P. Singh. A brief review of modeling approaches based on fuzzy time series. *International Journal of Machine Learning and Cybernetics*, 8, 02 2015.

P. Singh. Fuzzy time series modeling approaches: A review. *Applications of Soft Computing in Time Series Forecasting*, pages 11–39, 2016.

T. G. Smith et al. pmdarima: Arima estimators for Python, 2017–. URL http://www.alkaline-ml.com/pmdarima. [Online; accessed <today>].

Q. Song and B. S. Chissom. Fuzzy time series and its models. *Fuzzy Sets and Systems*, 54 (3):269–277, 1993. ISSN 0165-0114. doi: https://doi.org/10.1016/0165-0114(93)90372-O. URL https://www.sciencedirect.com/science/article/pii/016501149390372O.

S. Suradhaniwar, S. Kar, S. S. Durbha, and J. Adinarayana. Time series forecasting of univariate agrometeorological data: A comparative performance evaluation via one-step and multi-step ahead forecasting strategies. *Sensors (Basel, Switzerland)*, 21, 2021. URL https://api.semanticscholar.org/CorpusID:233208715.

D. Syed, H. Abu-Rub, A. Ghrayeb, and S. S. Refaat. Household-level energy forecasting in smart buildings using a novel hybrid deep learning model. *IEEE Access*, 9:33498–33511, 2021. doi: 10.1109/ACCESS.2021.3061370.

F. U. M. Ullah, A. Ullah, I. U. Haq, S. Rho, and S. W. Baik. Short-term prediction of residential power energy consumption via cnn and multi-layer bi-directional lstm networks. *IEEE Access*, 8:123369–123380, 2020. doi: 10.1109/ACCESS.2019.2963045.

I. T. Union. Itu internet report 2005: The internet of things. 2005.

F. Vanhoenshoven, G. Nápoles, W. Froelich, J. L. Salmeron, and K. Vanhoof. Pseudoinverse learning of fuzzy cognitive maps for multivariate time series forecasting. *Applied Soft Computing*, 95:106461, 2020.

C. Voyant, G. Notton, S. Kalogirou, M.-L. Nivet, C. Paoli, F. Motte, and A. Fouilloy. Machine learning methods for solar radiation forecasting: A review. *Renewable Energy*, 105:569 – 582, 2017. ISSN 0960-1481. doi: https://doi.org/10.1016/j.renene.2016.12.095. URL http://www.sciencedirect.com/science/article/pii/S0960148116311648.

Y. Wang, F. Yu, W. Homenda, W. Pedrycz, Y. Tang, A. Jastrzebska, and F. Li. The trend-fuzzy-granulation-based adaptive fuzzy cognitive map for long-term time series forecasting. *IEEE Transactions on Fuzzy Systems*, pages 1–1, 2022. doi: 10.1109/TFUZZ.2022.3169624.

F. Wilcoxon. Individual comparisons by ranking methods. biom. bull., 1, 80–83, 1945.

D. Xu, Y. Shi, I. W. Tsang, Y.-S. Ong, C. Gong, and X. Shen. Survey on multi-output learning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(7):2409–2429, 2020. doi: 10.1109/TNNLS.2019.2945133.

X. Xu and W. Ren. Prediction of air pollution concentration based on mrmr and echo state network. *Applied Sciences*, 9(9), 2019. ISSN 2076-3417. doi: 10.3390/app9091811. URL https://www.mdpi.com/2076-3417/9/9/1811.

A. Yadav and S. Chandel. Solar radiation prediction using artificial neural network techniques: A review. *Renewable and Sustainable Energy Reviews*, 33:772–781, 05 2014. doi: 10.1016/j.rser.2013.08.055.

K. Zhang, J. Thé, G. Xie, and H. Yu. Multi-step ahead forecasting of regional air quality using spatial-temporal deep neural networks: A case study of huaihai economic zone. *Journal of Cleaner Production*, 277:123231, 2020. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2020.123231. URL https://www.sciencedirect.com/science/article/pii/S0959652620332765.

S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen. Cautionary tales on air-quality improvement in beijing. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473(2205):20170457, 2017.

Z. Zhang, Y. Zeng, and K. Yan. A hybrid deep learning technology for pm 2.5 air quality forecasting. *Environmental Science and Pollution Research*, pages 1–14, 2021.

Annex A

EFTS Supplementary Materials

The following supporting information can be downloaded at: https://github.com/hugovynicius/EFTS

Annex B

MS-EFTS Supplementary Materials

The following supporting information can be downloaded at: `https://github.com/hugovynicius/MS_EFTS`

Annex C

MO-ENSFTS Supplementary Materials

The following supporting information can be downloaded at: `https://github.com/hugovynicius/MO-ENSFTS`

Annex D

MO-WMVFTS Supplementary Materials

The following supporting information can be downloaded at: https://github.com/hugovynicius/MO-WMVFTS