The Application of Complex Fuzzy Sets to Large-Scale Time-Series Forecasting

by

Sayedabbas Sobhi

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering
University of Alberta

Abstract

A number of architectures and time series forecasting algorithms use complex fuzzy sets, which are extensions of type-1 fuzzy sets. In the complex fuzzy set literature, the two most common forms of fuzzy sets are sinusoidal membership functions and complex Gaussian membership one. However, there have been no studies that combine both forms, either separately or in combination, to allow a direct comparison of their respective merits. Therefore, the first goal of this dissertation is to construct a suitable architecture to test both membership function forms, as well as their combination, on the time series forecasting task.

Previous architectures such as the Adaptive Neuro-Complex Fuzzy Inferential System (ANCFIS) have been shown to be accurate and compact forecasting algorithms. The Fast ANCFIS architecture was designed to speed up learning, and apply ANCFIS for data stream mining, by applying a fast Fourier transform to determine the parameters of sinusoidal membership functions. Randomized learning was proposed instead as an alternative strategy for speeding up learning in ANCFIS-ELM and RANCFIS. However, the learning algorithms and accuracy remained insufficient; none of the architectures were suitable for working with big data, which is an integral part of the real world today. Therefore, our second goal in this dissertation is to design a new CFS-based architecture that, while taking advantage of previous architectures, can also learn to forecast large-scale datasets efficiently and accurately.

We address both goals in this thesis by designing a new neuro-fuzzy architecture and evaluating it on multiple medium-to-large scale univariate and multivariate datasets (including a new condition-monitoring dataset for electric motors developed in the course of this dissertation work). It should be noted that one of the remarkable aspects of this thesis is that all our experiments

on large-scale datasets are based on real sensor data. The Randomized Complex Neuro-Fuzzy Inferential System (RCNFIS) is a complex fuzzy set-based model implemented in three variants; we will furthermore discuss two stages in the design of this model. Our initial model focused on modelling univariate time series problems, using sinusoidal membership functions, complex Gaussian ones, or a combination of both. Our experiments with this model indicated that the complex Gaussian fuzzy sets led to a superior forecasting model than the sinusoidal ones, or their combination, across all datasets and all variants of the architecture. We then extended this model to forecast an arbitrary number of variates. We again found that, for all variants and all datasets, the complex Gaussian fuzzy sets led to superior forecasting models than the sinusoidal ones or the combination of both types. Additionally, we have applied statistical tests to prove all these claims (Z-test and Friedman). Finally, we found that these best RCNFIS models were more accurate than our previous architectures, while also being faster to train. They were furthermore more accurate than competing results on these datasets, either from the literature or our own experiments.

Preface

This is an original work by Sayedabbas Sobhi. Chapter IV is being submitted to IEEE Access as "*Sobhi, S., M. Reshadi, and S. Dick, Condition Monitoring and Fault Detection of Small Induction Motors using Machine Learning Algorithms*." Chapter V has been submitted as "*Sobhi, S. and S. Dick, An Investigation of Complex Fuzzy Sets for Large-Scale Learning. Submitted to IEEE Transactions on Fuzzy Systems, 2022*." and Chapter VI is being prepared for submission as "*Sobhi, S. and S. Dick, Large-Scale Multivariate Time Series Forecasting using Complex Fuzzy Set. To IEEE Transactions on Systems, 2022*."

Dedication

I dedicate this thesis to my lovely wife, Sepideh, who has always been a constant source of encouragement and support during my Ph.D., and to my cute son, Armia, whose smile gave me strength to continue. It is also dedicated to my loving mother, Behjat, and my kind siblings, Zari, Ali, and Ziba, who have always been there for me and love me unconditionally. My sincere thanks go out to all of you who have supported me along the way.

Acknowledgments

It would be my pleasure to thank my supervisor, Dr. Scott Dick, for all the help and encouragement he provided during my Ph.D. Having received excellent guidance and supervision from Dr. Scott Dick, I would not have been able to complete this study without his knowledge and assistance. Furthermore, I would like to thank all the professors who served on the thesis committee.

Contents

Abstract ii

Preface iv

Dedication v

Acknowledgments vi

List of Tables xii

List of Figures xiii

List of Abbreviations and Acronyms xvi

1. Chapter I: Introduction 1

1.1. Complex Fuzzy Sets and Logic 1

1.2. Time Series Forecasting (TSF) 2

1.3. Statement of the Problem 3

1.4. Dissertation Objectives and Contributions 3

1.5. Outline of the Dissertation 4

2. Chapter II: Literature Review 6

2.1. Introduction 6

2.2. Statistical Forecasting Methods 7

2.2.1. Linear Models 8

2.2.2. Non-linear Models 12

2.3. Randomized Learning 15
2.4. Type-1 Fuzzy Sets, Logic, Relation, and Reasoning 17
2.5. Type-1 Fuzzy Inferential Systems 22
2.6. Neuro-Fuzzy Systems 24
2.6.1. Adaptive Network 24
2.6.2. ANFIS 25
2.7. Complex Fuzzy Sets 28
2.7.1. Complex Fuzzy Membership Functions 30
2.7.2. Complex Fuzzy Set Operations 33
2.7.3. Complex Fuzzy Relation 36
2.8. Complex fuzzy Logic 36
2.9. An overview of XAI 37
2.9.1. XAI and Neuro-Fuzzy Systems 40
2.10. AI Brittleness 42
2.10.1. Data/Distribution shifts and brittleness 44
2.11. Conclusion 48
3. Chapter III: Neuro-Fuzzy Systems Employing Complex Fuzzy Set and Logic 50
3.1. Introduction 50
3.2. ANCFIS 51
3.3. ANCFIS-ELM 53

3.4. RANCFIS 54
3.5. FANCFIS 55
3.6. CNFS 56
3.7. Conclusion 60
4. Chapter IV: Condition Monitoring 61
4.1. Introduction 61
4.2. Data Collection and Dataset Generation 63
4.2.1. Hardware and Instruments 64
4.2.2. Controlled Variables 65
4.2.3. Experimental Design 66
4.2.3.1. Stator Short 66
4.2.3.2. Bearing Faults 67
4.3. Methodology 68
4.3.1. Data Preprocessing 69
4.3.2. Data Preprocessing 72
4.4. Parameter Exploration Procedures 72
4.5. Results 74
4.6. Conclusion 76
5. Chapter V: Large-Scale Univariate Time Series Forecasting Using Complex Fuzzy Set 77
5.1. Introduction 77

5.2. Methodology 78
5.2.1. Proposed Models 78
5.2.2. Datasets 83
5.2.2.1. Lorenz system 83
5.2.2.2. Solar Power 84
5.2.2.3. Condition Monitoring Dataset 85
5.3. Preprocessing 87
5.4. Results 90
5.4.1. Experimental design 90
5.4.2. Experimental Results 91
5.4.3. Comparison against Existing Algorithms 93
5.4.4. Evaluating the Research Hypotheses 98
5.5. Conclusion 98
6. Chapter VI: Large-Scale Multivariate Time Series Forecasting Using Complex Fuzzy Set 99
6.1. Introduction 99
6.2. Methodology 101
6.2.1. Proposed Models 101
6.2.1.1. SISO RCNFIS 101
6.2.1.2. MISO RCNFIS 101
6.2.1.3. MIMO RCNFIS 102

6.2.2. Datasets 105
6.2.2.1. Rössler Attractor 105
6.2.2.2. Condition Monitoring Dataset 106
6.2.2.3. Gas Sensor Dataset 106
6.3. Preprocessing 107
6.4. Results 112
6.4.1. Experimental Design 112
6.4.2. Experimental Results 113
6.4.3. Out-of-Sample Results 118
6.5. Conclusion 123
7. Chapter VII: Summary, Conclusion, and Future Work 125
7.1. Summary and Conclusion 125
7.2. Future Work 127
Bibliography 129

List of Tables

Table 1: Normal Model Performance for shallow learning 74

Table 2: The final out-of-sample results for shallow learning 75

Table 3: Anomaly Detector Performance for deep learning 76

Table 4 - Obtained delay vectors parameters 90

Table 5 - Solar power dataset 91

Table 6 - Lorenz dataset 92

Table 7 - CM dataset 92

Table 8 - Compare results in different architectures in terms of RMSE 94

Table 9 - The average rank of the measures 97

Table 10 - Training time in seconds 97

Table 11 - Obtained delay vectors parameters 112

Table 12 - Explore and compare the best results of the three proposed models in different modes & methods (CM dataset – d6m7) 114

Table 13 - Explore and compare the best results of the three proposed models in different modes & methods (CM dataset – d1m3) 115

Table 14 - Explore and compare the best results of the three proposed models in different modes & methods (Rössler dataset) 116

Table 15 - Explore and compare the best results of the three proposed models in different modes & methods (Gas Sensor dataset) 117

Table 16 - Compare results in different architectures in terms of RMSE 120

Table 17 - The average rank of the measures 123

List of Figures

Figure 1 - The SLFN architecture [56] 16

Figure 2 - Triangular MF 18

Figure 3 - Trapezoidal MF 19

Figure 4 - Gaussian MF 19

Figure 5 - Fuzzy Inference System (FIS) [16] 23

Figure 6 - Examples of FISs [83-85] 24

Figure 7 - Types of adaptive networks [89] 25

Figure 8 - Basic structure of ANFIS [78] 26

Figure 9 – Codomain for a Complex Fuzzy Set [5] 29

Figure 10 - Sinusoidal CFS: a=5, b=0, c=0.5, d=0.5 31

Figure 11 - Gaussian CFS: m=5, σ=1, λ=1.5 32

Figure 12 - Rotational Invariance [23] 35

Figure 13 - ANCFIS network for univariate time series [3] 51

Figure 14 - ANCFIS-ELM network for univariate time series [214] 53

Figure 15 - RANCFIS network for univariate time series [215] 54

Figure 16 - Complex Neuro-Fuzzy System (CNFS) 57

Figure 17 - Induction motor fault locations [226] 62

Figure 18 - Types of induction motors faults 62

Figure 19 - Motor Testing Setup 64

Figure 20 - Stator Damage in Motor 3 66

Figure 21: Anomaly Detector 68

Figure 22: Phase Plot for d = 6 71

Figure 23: Proportion of False-Nearest Neighbors for different values of ratio factor (f) 71

Figure 24: A 1-dimensional convolutional layer 73

Figure 25: Normal model RMSE 75

Figure 26 - Model 1 79

Figure 27 - Banks of sinusoidal and Gaussian neurons for Model 1 80

Figure 28 - Model 2 81

Figure 29 - Model 3 82

Figure 30 - Plotted solutions of the Lorenz equations in three-dimensional phase space 83

Figure 31 - Lorenz attractor, variable X only [262] 84

Figure 32 - RSR device for measuring solar radiation [258] 85

Figure 33 - Monthly average of solar radiation (W/m2) for a 7-year period 85

Figure 34 - Time-delayed mutual information statistics for datasets 89

Figure 35 - Phase Plot for datasets and their obtained delay 89

Figure 36 - Proportion of False-Nearest Neighbors for different values of ratio factor (f) 90

Figure 37 - MIMO ANCFIS architecture for a bivariate time series problem [10] 99

Figure 38 - Three SISO Simple/Hybrid RCNFIS models in three modes for a sample variate x [278] 102

Figure 39 - Three MISO Simple/Hybrid RCNFIS models in the three modes for a sample time series with two variates x and y 103

Figure 40 - Three MIMO Simple/Hybrid RCNFIS models in the three modes for a sample time series with two variates x and y 104

Figure 41 - The Rössler attractor graph for a=b=0.2, c=5.7 106

Figure 42 - A chemical sensing system for measuring the conductivity of a sensor array exposed to different gas conditions [276] 107

Figure 43 - Data preprocessing steps in our project 108

Figure 44 - Mutual Diagrams for the datasets 110

Figure 45 - Phase Plot for the datasets based on their obtained delay..... 111

Figure 46 - Proportion of False-Nearest Neighbors for different values of ratio factor (f) 111

List of Abbreviations and Acronyms

ABC	Artificial Bee Colony
ACF	Auto Correlation Function
ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
ANCFIS	Adaptive Neuro-Complex Fuzzy Inferential System
ANCFIS-ELM	ANCFIS-Extreme Learning Machine
ANFIS	Adaptive Neuro Fuzzy Inferential System
ANN	Artificial Neural Network
AR	Auto Regressive
ARCH	Auto Regressive Conditional Heteroscedasticity
ARFIMA	Auto Regressive Fractionally Integrated Moving Average
ARIMA	Auto Regressive Integrated Moving Average
ARMA	Auto Regressive Moving Average
ARMA	Auto Regressive Moving Average
BNN	Bayesian Neural Network
CAPE	Combinatorial Algorithm for Proportional Equality
CFL	Complex Fuzzy Logic
CFR	Complex Fuzzy Relation
CFS	Complex Fuzzy Set
CFS&L	Complex Fuzzy Sets and Logic
CM	Condition Monitoring

CNFS	Complex Neuro-Fuzzy System
CNN	Convolutional Neural Network
COA	Center Of Area
CP	Consequent Parameter
DARPA	Defense Advanced Research Projects Agency
df	Degree of Freedom
DFT	Discrete Fourier Transform
DNN	Deep Neural Networks
DS	Data Science
EFA	Evolutionary Fuzzy Algorithm
EI	Explanatory Interface
ELM	Extreme Learning Machine
ERM	Empirical Risk Minimization
FALCON	Fuzzy Adaptive Learning Control Network
FANCFIS	Fast Adaptive Neuro-Complex Fuzzy Inferential System
FBSA	FCM-Based Splitting Algorithm
FCM	Fuzzy C-Mean
FFT	Fast Fourier Transform
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Controller
FNN	False-Nearest Neighbors
FPR	False Positive Rate
FS&L	Fuzzy Set and Logic

FUN	FUzzy Net
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GARCH	Generalized Autoregressive Conditional Heteroscedasticity
GRNN	General Regression Neural Network
i.i.d.	Independent and Identically Distributed
ID	In-Distribution
KBANN	Knowledge-based neural network
LIF	Leaky Integrate and Fire
LIME	Local Interpretable Model-agnostic Explanations
LMS	Least Mean Squares
LRP	Layer-wise Relevance Propagation
LSE	Least Squares Estimation
LSM	Liquid State Machine
LSTM	Long short-term memory
LV	Linguistic Variable
MA	Moving Average
MAE	Mean Absolute Error
MCB	Multiple Comparisons with the Best
MF	Membership Function
MFO	Moth-Flame Optimization Algorithm
MIMO	Multiple Input Multiple Output
MISO	Multiple Input Single Output

ML	Machine Learning
MLP	Multi Layer Perceptron
MNIST	Modified National Institute of Standards and Technology
MSE	Mean Square Error
NARX	Nonlinear Auto Regressive eXogenous
NLP	Natural Language Processing
OOD	Out-Of-Distribution
PACF	Partial Auto Correlation
PNN	Probabilistic Neural Network
PSO	Particle Swarm Optimization
RANCFIS	Randomized Adaptive Neuro-Complex Fuzzy Inferential System
RBFN	Radial Basis Function Networks
RCNFIS	Randomized Complex Neuro-Fuzzy Inferential System
REIT	Real Estate Investment Trust
RLS	Recursive Least Squares
RLSE	Recursive Least Square Estimator
RMSE	Root Mean Square Error
ROT	Rule Of Thumb
RPM	Round Per Minute
RSR	Rotating Shadow-band Radiometer
RVFLN	Random Vector Functional Link Network
SCN	Stochastic Configuration Network
SETAR	Self Exciting Threshold Auto Regressive

SISO	Single Input Single Output
SLFN	Single Hidden Layer Feed-forward Network
SMOreg	Support Vector Machine for regression
STAR	Smooth Threshold Auto Regressive
SVM	Support Vector Machine
SVR	Support Vector Regression
TAR	Threshold Auto Regressive
TCAV	Testing with Concept Activation Vectors
TPR	True Positive Rate
TSF	Time Series Forecasting
TSK	Takagi-Sugeno-Kang
UART	Universal Asynchronous Receiver-Transmitter
VAR	Vector Auto Regressive
VaR	Value at Risk
VECM	Vector Error Correction Model
VNCSA	Variable Neighborhood Chaotic Simulated Annealing
WEKA	Waikato Environment for Knowledge Analysis
WOA	Whale Optimization Algorithm
XAI	eXplainable Artificial Intelligence

1. Chapter I: Introduction

1.1. Complex Fuzzy Sets and Logic

The concept of Complex Fuzzy Sets (CFS) is an extension of the standard theory of type-1 fuzzy sets, in that their membership functions have a unit disc of the complex plane as the codomain instead of a codomain of [0,1] in type-1 fuzzy [1-5]. The related Complex Fuzzy Logic (CFL) is an infinite-valued logic whose truth values are again complex numbers from the unit disc. In a systematic review of Complex Fuzzy Sets and Logic (CFS&L) [5], Yazdanbakhsh and Dick found that there were seven main open-ended questions being followed by researchers. Amongst these, the third question asks: "What are the functional forms for complex fuzzy memberships that have been investigated?". This is a vital question, which is directly dealt with in this dissertation, and it will be discussed in more detail in subsequent chapters. However, we can briefly say that parameterized functions fall into two categories: sinusoidal CFS were proposed for the Adaptive Neuro-Complex Fuzzy Inferential System (ANCFIS) architecture [3], while Gaussian CFS were proposed for the Complex Neuro-Fuzzy System (CNFS) architecture [4]. Our study examines these two forms of fuzzy membership functions and compares the models created separately from each (and in combination) on a common architecture (something never done before). This goal has been achieved by introducing a new architecture, based on CFS&L and inspired by ANCFIS, and evaluating it on large-scale time series forecasting datasets.

1.2. Time Series Forecasting (TSF)

Time Series Forecasting (TSF) is, of course, a significant part of predictive analytics, which finds application in fields from sensor-based condition monitoring [6], to stock market analysis [7]. TSF has thus been widely studied in a variety of fields. There is, for example, an extensive literature on statistical time series forecasting; however, these models commonly require a fixed model form (e.g. ARIMA [8] models assume a linear correlation structure). In contrast, machine learning approaches can implicitly detect complex nonlinear relationships between predicted and predictor variables or between predictor variables.

Artificial Neural Networks (ANNs), in particular, have been especially successful in recent years. Among numerous design approaches, research in the past decade has shown that neuro-fuzzy systems based on Complex Fuzzy Sets (CFS) are particularly accurate and parsimonious forecasters in small-scale time series. As mentioned in the previous section, there are two architectural approaches that dominate this literature: the ANCFIS family first introduced in [3], and the CNFS family introduced in [4]. Both are based on the well-known Adaptive Neuro-Fuzzy Inferential System (ANFIS) architecture [9]. However, there has been no previous work in adapting these designs for large-scale time series forecasting.

Therefore, in this dissertation we propose a new CFS-based architecture designed for large-scale time series forecasting. We will show that, across multiple univariate and multivariate datasets, this new architecture learns faster and is more accurate than our previous CFS&L-based architectures. This includes two large-scale datasets collected at the University of Alberta (focusing on solar power forecasting, and condition monitoring of electric motors, the latter having been collected in the course of this dissertation); a medium-scale multivariate chaotic dataset (for

which we model a single or multiple variates); and a benchmark sensor-data problem, for which we again model a single or multiple variates. Furthermore, compared to other shallow learning algorithms, this algorithm is more accurate on all of the datasets.

1.3. Statement of the Problem

The CFS&L-based models discussed in previous sections have yielded accurate and parsimonious results in multiple applications, including TSF as discussed in this dissertation. This includes both univariate and multivariate [10] variants of ANCFIS, Fast ANCFIS (FANCFIS) and Randomized ANCFIS (RANCFIS). However, modeling large-scale multivariate datasets remained infeasible with the se architectures. Additionally, the design and implementation of CFS&L models are dependent on several parameters, of which the most important is what the complex membership function is selected. The design space of possible CFS functional forms has been explored empirically, but there has never been a systematic comparison of different CFS forms within a single architecture.

1.4. Dissertation Objectives and Contributions

The objective of this dissertation is to design and implement a new CFS&L-based architecture that can, despite the limitations of previous architectures of this type (mentioned in previous sections) or even those of deep learning architectures (which can accurately model large multivariate datasets, but face problems such as structural complexity and the need for expensive hardware), easily and cheaply deal with large multivariate datasets. Moreover, we will compare two types of complex membership functions, namely, sinusoidal CFS and Gaussian CFS, as well as their combination, to determine whether any of these approaches is significantly superior to the others.

A typical forecasting methodology is used to evaluate our architecture (RCNFIS) in all modes (different types of complex membership function) and methods (SISO (Single Input Single Output), MISO (Multiple Input Single Output) and MIMO (Multiple Input Multiple Output)): a one-step-ahead prediction, with all training data chronologically earlier than the test data. The performance of the univariate/multivariate RCNFIS is evaluated on a variety of chaotic and large-scale time series datasets.

1.5. Outline of the Dissertation

Following the Introduction Chapter, the remainder of this dissertation is organized into six chapters:

- Chapter II: In this chapter, a literature review is presented covering statistical forecasting methods, randomized learning, type-1 fuzzy set and logic (FS&L), ANFIS, complex fuzzy logic (CFS&L), and their related concepts. The concepts of XAI and Brittleness will also be reviewed.
- Chapter III: In this chapter, we study the literature of CFS&L-based neuro-fuzzy systems applied to time series forecasting.
- Chapter IV: The purpose of this chapter is to address an important industrial research topic: condition monitoring of small electric induction motors. Our discussion in this chapter focuses on the steps relating to obtaining, extracting, and producing the relevant dataset (which also used in our subsequent experiments in this dissertation (chapters 5 and 6) as well as designing a new anomaly detection model to deal with large-scale data.
- Chapter V: Our contributions in this chapter are first, to design and evaluate a new CFS-based neuro-fuzzy architecture (RCNFIS) for large-scale time series forecasting

applications to produce accurate and compact time-series forecasting models; and second, the first direct empirical comparison of sinusoidal vs. complex Gaussian CFS in three different modes of our architecture.

- Chapter VI: The contribution we make in this chapter is the further development of the architecture proposed in Chapter V, so that it can support large-scale multivariate time series and deliver accurate and promising results while maintaining its compact structure. For this purpose, we evaluate our SISO, MISO and MIMO designs in different modes of complex membership functions.
- Chapter VII: A summary, conclusion, and discussion of future work are presented in this chapter.

2. Chapter II: Literature Review

2.1. Introduction

Studies on Time Series Forecasting (TSF), which led to the creation of many classical algorithms and models divide into two groups: classical statistical methods and machine learning algorithms. The first models developed for TSF were based on univariate statistical models, often inspired by the auto-regression principle, such as the Auto-Regressive (AR) [11], Auto-Regressive Moving Average (ARMA) [12], and the Auto-Regressive Integrated Moving Average (ARIMA) models [13]. For many applications, however, the process data takes the form of a multivariate time series. Thus, multivariate forecasting models were developed, including the extended version of the AR model such as Vector Auto-Regressive (VAR) models [14] and the Vector Error Correction Model (VECM) [14]. A principal drawback of these methods is that these models usually require a fixed model form; for example, some assume a linear correlation structure. Nonetheless, researchers and practitioners still find the useful and interesting for certain applications, in part because they are relatively easily explained to a user.

In contrast, machine learning approaches can implicitly recognize complex nonlinear relationships between dependent and independent variables or between the independent variables themselves. A very broad range of machine learning algorithms have been applied to the time series forecasting problem; this usually requires some form of preprocessing to embed the sequential data of a time series into the tabular format expected in machine learning (e.g. [15]). Artificial neural networks in particular have received considerable attention as forecasting algorithms.

Fuzzy logic has also seen considerable use in time series forecasting (commonly in the guise of neuro-fuzzy or genetic-fuzzy hybrid systems). A great deal of progress has been made in the research and applications of type-1 fuzzy logic since Zadeh published his first paper on fuzzy set [1]. The type- 1 membership function allows the gradual evaluation of elements' membership in a set based on the two-dimensional membership function (MF). Linguistically, it provides an organized calculus for representing vague and incomplete information. Using linguistic labels specified by MFs, the linguistic information is converted into numerical values [16-18]. This approach allows type-1 fuzzy inference systems to use fuzzy if-then rules to model human expertise in specific applications. In this thesis, we discuss an extension of type-1 fuzzy set called complex fuzzy set. It was first defined in [2], which the co-domain of the membership function was the unit disc of the complex plane. In spite of earlier explorations of complex-valued memberships [19-21], Ramot, et al.'s paper on Complex Fuzzy Logic (CFL) [22] became the inspiring work on Complex Fuzzy Sets and Logic (CFS&L). A number of early theoretical results were published in [23, 24], and the first applications of CFS&L began to appear in [4, 25].

In the remainder of this chapter, we first study statistical forecasting methods. We then review the literature on randomized learning, and then we examine fuzzy sets and logic (FS&L) and complex fuzzy sets and logic (CFS&L).

2.2. Statistical Forecasting Methods

Forecasting is the use of past data to predict future events. In planning and decision making processes, prediction of future events is very critical and forecasting can help in making rational decisions [26]. In general, there are two main approaches in forecasting: qualitative methods and quantitative methods [27]. Qualitative forecasting methods involve human experts using their

experience and judgment in specific fields to make forecasts. On the other hand, quantitative forecasting methods use historical data and forecasting models to infer future behaviours from past and present ones. There are two types of quantitative forecasting methods: time-series methods and econometric methods. In this review, we just focus on time-series methods. There are also two approaches for analyzing time series: time-domain approaches and the frequency-domain approach [28]. The time-domain approach is supported by assuming a correlation between adjacent points in time in terms of the dependence of the current value on past values. On the other hand, the frequency-domain approach treats a time series as a collection of superimposed wave shapes. The most common of these is to assume a time series is a collection of superimposed sinusoids, which can be determined via the Fourier transform. Numerous other waveforms are possible; this is the topic of e.g. *wavelet* transforms (these can also mix time- and frequency-domain approaches). In this thesis, we focus our attention on time-domain approaches and frequency-domain approaches using sinusoids and the Fourier transform. We examine both linear and non-linear models in these domains [29].

2.2.1. Linear Models

Linear models predict future values x_t based on observations at p previous points of time t by developing a function f [30]:

$$x_t = f(x_{t-1}, x_{t-2}, \dots, x_{t-p}) \qquad (1)$$

In [31], the concept of stochasticity in time series was introduced. A number of time series methods have been developed since then based on this idea. In [32], the notion of autoregressive (AR) models was introduced, which expressed the series at time t as a linear regression of previous p observations [30]:

$$x_t = w_0 + \sum_{i=1}^{p} w_i x_{t-i} + \varepsilon_t \quad (2)$$

Here, ε_t is the residual error term from the AR model. By using polynomial notation, the auto-regressive model is defined as AR (p), where p refers to the order of the AR component.

The first-order AR model is denoted by AR (*1*):

$$x_t = \emptyset \varepsilon_{t-1} + \varepsilon_t \quad (3)$$

The second-order AR model is denoted by AR (*2*):

$$x_t = \emptyset_1 \varepsilon_{t-1} + \emptyset_2 \varepsilon_{t-2} + \varepsilon_t \quad (4)$$

The p^{th} order AR model is denoted by AR (p):

$$x_t = \emptyset_1 \varepsilon_{t-1} + \emptyset_2 \varepsilon_{t-2} + \cdots + \emptyset_p \varepsilon_{t-p} + \varepsilon_t \quad (5)$$

Here, $\emptyset$ is the model coefficient, ε_t is an error in time t, and p is the order of the AR model.

Another concept introduced in [32] was the moving average (MA) model. The moving average models use dependency between residual errors to forecast values in the next time period. The notation MA (q) refers to the moving average model of order q:

$$x_t = \mu - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} + \varepsilon_t \quad (6)$$

Here, μ is the mean of the series, the θ_1, ..., θ_q are the parameters of the model and the ε_t, ε_{t-1},..., ε_{t-q} are white noise error terms. The value of q is called the order of the MA model.

To evaluate whether the time series signal consists of an MA or AR component, autocorrelation (ACF) and partial autocorrelation (PACF) is used [30]. The ACF is given by:

$$ACF = \frac{cov(x_t, x_{t-h})}{var(x_t)} \quad (7)$$

The numerator in the preceding equation is covariance between the time series signal at time *t* and *t-h* where *h* is the lag in the time series signal. The PACF is also computed similarly as ACF except that correlation is computed by removing the already explained variation between intervals. This is also defined as conditional correlation.

$$PACF = \frac{cov(x_t, x_{t-2}|x_{t-1})}{\sqrt{var(x_t|x_{t-1})}.\sqrt{var(x_{t-2}|x_{t-1})}} \quad (8)$$

The AR (*p*) models tend to capture the mean reversion effect (systems that deviate from their mean behavior tend to return to the mean over time), whereas MA (*q*) models tend to capture "shock effects" in model error, which are abnormal or unpredicted events. Thus, [33] combined both AR and MA models and showed that ARMA processes can be used to model all stationary time series as long as the appropriate order was specified. An ARMA (*p*, *q*) time series forecasting model incorporates the p^{th} order AR and q^{th} order MA model, respectively. The ARMA (*p*, *q*) model is denoted as follows:

$$x_t = \mu + \emptyset_1 x_{t-1} + \emptyset_2 x_{t-2} + \cdots + \emptyset_p x_{t-p} - \theta_1 \varepsilon_{t-1} - \theta_2 \varepsilon_{t-2} - \cdots - \theta_q \varepsilon_{t-q} + \varepsilon_t \quad (9)$$

Here, $\emptyset$ and θ represent AR and MA coefficients. The μ and ε_t captures the intercept and error at time t.

There are multiple methods to select *p* and *q*. For instance, in ARMA (1,1) time series data [30], as both ACF and PACF have shown sine-wave pattern, *p* and *q* both parameters are affecting the time series signal.

Later, the use of ARIMA models and their extensions became popular. ARIMA [30], also known as the Box-Jenkins model, is a generalization of the ARMA model by including integrated components. The integrated components are useful when data has non-stationarity, and the integrated part of ARIMA helps in reducing the non-stationarity. The ARIMA applies differencing on time series one or more times to remove non-stationarity. The ARIMA (p, d, q) represent the order for AR, MA, and differencing components. The major difference between ARMA and ARIMA models is the d component, which updates the series on which forecasting model is built. The d component aims to de-trend the signal to make it stationary, after which an ARMA model can be applied to the de-trended dataset. ARIMA forecasting can be written as follows:

$$\hat{x}_t = \emptyset_1 \hat{x}_{t-1} + \emptyset_2 \hat{x}_{t-2} + \cdots + \emptyset_p \hat{x}_{t-q} + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t \qquad (10)$$

Depending on the order of p, d, and q, the model behaves differently. For example, ARIMA (1, 0, 0) is a first-order AR model. Similarly, ARIMA (0, 0, 1) is a first-order MA model.

A list of different cases of using ARIMA models are given in [29], and we briefly discuss a few in the following. In [34] an alternative to the Box-Jenkins methodology proposed that is called the ARARMA methodology which transforms time series from a large lag AR filter to a short lag filter. [35] studied annual sugarcane production of India for the period of 1950-51 to 2002-03, using an ARIMA (2, 1, 0) model to forecast for next three years. [36] studied farm price, wholesale price and pure oil price of palm oil data of Thailand for the period of 2000 to 2004 and concluded that ARIMA (2,1,0), ARIMA (1,0,1) and ARIMA (3,0,0) models are the best for the farm price, whole sale price and pure oil price respectively. [37] studied monthly data of whole sale prices of Rohu fish in west Bengal for the period of 1996 to 2005 and concluded that ARIMA (2, 1, 1) model applied to the seasonally adjusted data captures the variation in the data efficiently. [38]

studied monthly wholesale price data of coriander in the Kota market of Rajasthan for the period of April 2000 to May 2011 and compared the performance of an exponential smoothing model, ARIMA model, and ANN model and concluded that an ARIMA (1, 1, 1) model performed better than the other models. [8] used time series data from 1950- 51 to 2011-12 and developed different ARIMA models to forecast the rice yield during 2012-13 in India, observing that out of eleven ARIMA models, ARIMA (1, 1, 1) is the best fitted model in predicting efficiently the rice yield. [39] studied daily wholesale prices of pigeon pea in the markets of Amritsar and Bhatinda and all India maximum, minimum and modal prices for the period of 2012-2013 and used Autoregressive fractionally integrated moving-average (ARFIMA) model (a generalization of ARIMA where d can be a non-integer) to forecast prices for the period of January 1, 2014 to February 28, 2014. [40] studied uranium prices from 2000 to 2015, comparing ARIMA to the escalation rate model. An ARIMA (2, 1, 2) performed best.

2.2.2. Non-linear Models

A nonlinear process requires a different analytical approach than linear ones. Nonlinear time series analysis began with [41], showing that using the finite Volterra series, continuous nonlinear performance in time could be approximated [29]. Then, [42] proposed the threshold autoregressive (TAR) model and this model noted a simple departure from the linear time-series models which were popularized in [43]. After that, [44] introduced the smooth transition autoregressive (STAR) model which is a non-linear autoregressive time series model.

Suppose we rewrite Equation 5 as follows for a time series y_t [45]:

$$y_t = \mu + \emptyset_1 y_{t-1} + \emptyset_2 y_{t-2} + \cdots + \emptyset_p y_{t-p} + \sigma\varepsilon_t \qquad (11)$$

Here, $\emptyset_i$ ($i = 1, 2, \cdots, p$) are the AR coefficients, ε_t is an error in time t, and $\sigma > 0$ is the standard deviation of disturbance term. The model parameters $\emptyset = (\mu, \emptyset_1, \emptyset_2, \ldots, \emptyset_p)$ and σ are independent of time t and remain constant. To capture nonlinear dynamics, TAR models allow the model parameters to change according to the value of a threshold variable z_t:

$$y_t = X_t \emptyset^{(j)} + \sigma^{(j)} \varepsilon_t \quad if \quad r_{j-1} < z_t \leq r_j \tag{12}$$

Here, $X_t = (1, y_{t-1}, y_{t-2}, \cdots, y_{t-p})$, $j = 1, 2, \cdots, k$, and $-\infty = r_0 < r_1 < \cdots < r_k = \infty$. In essence, the $k - 1$ thresholds $(r_1, r_2, \cdots, r_{k-1})$ divide the domain of the threshold variable z_t into k different regions. In each different region, the time series y_t follows a different AR (p) model.

When the threshold variable $z_t = y_{t-d}$, with the delay parameter d being a positive integer, the dynamics of y_t is determined by its own lagged value y_{t-d} and the TAR model is called a self-exciting TAR or SETAR model. For the ease of notation, let SETAR (1) denote the one-regime linear AR model with $k = 1$, SETAR (2) denote the two-regime TAR model with $k = 2$, etc.

In [46], the Auto-Regressive Conditional Heteroscedasticity (ARCH) model was introduced. This was the first model to provide a volatility measure; it describes the changes in conditional variance as a function. The chosen form of the conditional variance function is quadratic in the past values of the time series. [47] were the first to study the effect of ARCH on forecasting. To model a time series using an ARCH process we have:

$$\varepsilon_t = \sigma_t z_t \tag{13}$$

Here, ε_t denotes the error terms. The model is split into a stochastic piece z_t and a time-dependent standard deviation σ_t characterizing the typical size of the terms. Thus, the series is given by:

$$\sigma_t^2 = \alpha_0 + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 = \alpha_0 + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 \quad (14)$$

Here, $\alpha_0 > 0 \; and \; \alpha_i \geq 0, i > 0$.

If an autoregressive moving average model (ARMA) model is assumed for the error variance, the model is a generalized autoregressive conditional heteroscedasticity (GARCH) model [48]. In that case, the GARCH (p, q) model (where p is the order of the GARCH terms σ^2 and q is the order of the ARCH terms ${\varepsilon_t}^2$, following the notation of the original paper, is given by

$$y_t = x_t' b + \varepsilon_t \quad (15)$$

$$\sigma_t^2 = w + \alpha_1 \epsilon_{t-1}^2 + \cdots + \alpha_q \epsilon_{t-q}^2 + \beta_1 \sigma_{t-1}^2 + \cdots + \beta_q \sigma_{t-q}^2 = w + \sum_{i=1}^{q} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{p} \beta_i \sigma_{t-i}^2 \quad (16)$$

[49] studied the behavior of exchange rate data using GARCH models and concluded that these models could not capture adequately the statistical properties of non-linearity present in the series. [50] showed that the ARCH and GARCH models are successful in risk management analysis of financial data and used a GARCH (1, 1) model to calculate value at risk (VaR) of three indexes (NASDAQ, Dow jones and long bonds). In a meta-study [51] examined 93 papers and found that GARCH models tend to be more parsimonious than ARCH models, but the result varied. [52] suggested that different data, sampling periods, sample frequency, forecast horizon, distribution used and the loss function can influence the result. [53] studied daily stock prices from Shanghai and Shenzhen using GARCH (1, 1) model with normal and skewed generalized error distributions and found the latter was superior. [54] studied volatility forecasting performance of real estate investment trust (REIT) using symmetric GARCH (1, 1) model with normal, student-t and skewed generalized error distributions respectively and showed the latter was best. [55]

studied three sets of monthly price data viz domestic edible oils, international edible oils and international raw cotton. GARCH models outperformed ARIMA on the first two; the third was more asymmetric, and an exponential GARCH model was found to be best.

2.3. Randomized Learning

Modern deep neural networks can have a huge number of adaptable parameters, with the largest ones (currently Google's PaLM) having over 540 *billion* parameters[1]. Such large networks require large amounts of training data to learn a vector of parameter values that are adequately generalized for the application domain. However, research has shown that, while there is only one global optimum in parameter space, in practice there are many of these *parameter vectors* that yield desirable solutions for a given problem. We can thus reframe learning as a search for a satisfactory parameter vector – with optimization only one of the possible paths to conducting this search. One possible approach is to fix the values of some parameters as randomly-chosen constants, and only optimize the others. Therefore, it may be possible to set the values of a few parameters as randomly-selected constants, and just learn the optimal values for the rest. For example, in the Single Hidden Layer Feed-forward Network (SLFN, see Figure 1), hidden-layer logistic neurons are fully connected to the input layer, and to summation neuron(s) in the output layer. The hidden-layer weights are randomly-chosen constants, while the output weights are learned via least-squares optimization [56].

[1] https://ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html

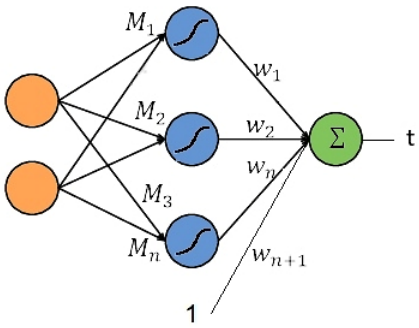


Figure 1 - The SLFN architecture [56]

The Random Vector Functional Link Network (RVFLN) [57-60] is another fast learning algorithm based on randomized learning, which is known to be a universal approximator in the probabilistic sense. The transfer function can be expressed as [60]:

$$G_L(x; a, b) = \sum_{j=1}^{L} \beta_j g(a_j^T x + b_j) \qquad (17)$$

where L is the number of hidden nodes, x is the input vector, **g** is the activation function, b_j is the bias, a_j is the input weight, and β_j is the output weight connecting the *j-th* hidden node and the output node. Given a training set $\{x_i, y_i\}$ with N samples of the target function ($i = 1, 2, \ldots, N$), and given that a_j and b_j are randomly selected and fixed, learning consists of a least squares optimization [61, 62]:

$$\min_{\beta_1 \ldots \beta_L} \sum_{i=1}^{N} (\sum_{j=1}^{L} \beta_j g(a_j^T x_i + b_j) - y_i)^2 \qquad (18)$$

which can be converted into a matrix expression, i.e. [61, 62],

$$\beta^* = arg \min_{\beta \in \mathbb{R}^L} ||H\beta - Y||_2^2 \qquad (19)$$

where H is the hidden layer output matrix [61, 62]:

$$H = \begin{pmatrix} g(a_1^T x_1 + b_1) & \dots & g(a_L^T x_1 + b_L) \\ \vdots & \ddots & \vdots \\ g(a_1^T x_N + b_1) & \dots & g(a_L^T x_N + b_L) \end{pmatrix} \quad (20)$$

A closed form solution for the output weights can be obtained by using the pseudoinverse method [63], i.e., $\beta^* = H^{\dagger}Y$. RVFLNs for robust learning were developed in [62, 64, 65]. The Stochastic Configuration Network (SCN) [66] was another shallow network using randomized learning. It has been extended to form deep SCNs [67], robust SCNs [60, 64], SCN ensembles [68], and 2D SCNs for image analytics [69].

The Liquid State Machine (LSM) architecture [70] takes a different approach to randomized learning. An LSM is made up of a large set of Leaky Integrate and Fire (LIF) neurons in a "reservoir." Each reservoir-layer neuron receives inputs from an input layer as well as other reservoir-layer neurons (making the LSM a recurrent network). The recurrent connections in the reservoir are the random elements; they approximate the idea of free-moving molecules in a liquid, as opposed to a rigid structure. The LSM can thus be viewed as a three-layer system: there is an input layer, a reservoir, and a memoryless readout circuit (also made up of LIF neurons, but without inter-layer connections) [70-76]. A number of similar algorithms have been developed (prominently including Echo-State Networks [77]); the central feature of these "reservoir computing" architectures is the randomly-connected recurrent layer.

2.4. Type-1 Fuzzy Sets, Logic, Relation, and Reasoning

"Fuzzy sets" was introduced by Zadeh as a new method to describe non-probabilistic uncertainties with the formal definition of multi-valued (fuzzy) sets in 1965 [1]. By changing the two-valued characteristic function into a multi-valued membership function, he extended traditional set theory.

The degree to which an element belongs to a fuzzy set is described by the membership function, which in general is a set of ordered pairs:

$$A = \{x, \mu_A(x) | x \in X\} \qquad \mu_A : X \rightarrow [0,1] \tag{21}$$

where $\mu_A(x)$ maps elements of X (the universe of discourse) into the interval [0,1], representing their membership in the fuzzy set A. Fuzzy logic is an infinite-valued logic that is isomorphic to fuzzy set theory, with truth values drawn from [0,1]. Parameterized membership functions, being relatively easy to compute, are quite common; some examples include Triangular, Trapezoidal, and Gaussian fuzzy sets, as follows [78]:

I. Triangular MFs:

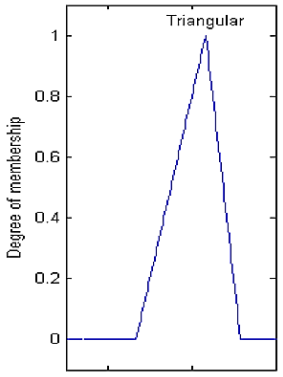


Figure 2 - Triangular MF

$$triangle(x; a, b, c) = \begin{cases} 0, & x \le a \\ (x-a)/(b-a), & a \le x \le b \\ (c-x)/(c-b), & b \le x \le c \\ 0, & c \le x \end{cases} \tag{22}$$

where $x \in X$ and the underlying triangular MF has three corners determined by the parameters {a, b, c}.

II. Trapezoidal MFs:

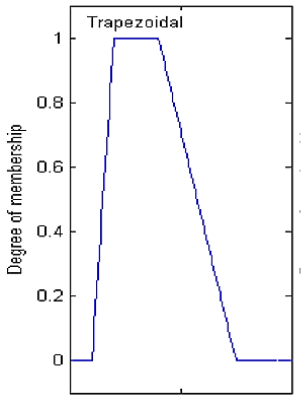


Figure 3 - Trapezoidal MF

$$trapezoidal(x;a,b,c,d)=\begin{cases}0, & x\le a\\ (x-a)/(b-a), & a\le x\le b\\ 1, & b\le x\le c\\ (d-x)/(d-c), & c\le x\le d\\ 0, & d\le x\end{cases} \quad (23)$$

where $x \in X$ and the underlying trapezoidal MF has four corners determined by the parameters {a, b, c, d}.

III. Gaussian MFs:

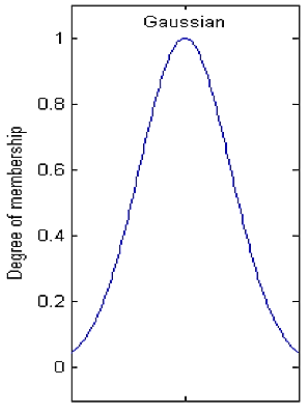


Figure 4 - Gaussian MF

$$gaussian(x;c,\sigma)=e^{\frac{-(x-c)^2}{2\sigma}} \quad (24)$$

where $x \in X$ and c and σ are the Gaussian MF center and the Gaussian width, respectively.

The fuzzy set operations of union, intersection, subsethood and complement, as well as their counterparts in the isomorphic fuzzy logic, were first defined as follows [78]:

I. Union (Disjunction): Union of two fuzzy sets *A* and *B* is a fuzzy set C that the membership *C* value of two fuzzy sets is the maximum of their membership values at any given *x*:

$$\mu_C(x) = \mu_{A\cup B}(x) = \max\left(\mu_A(x), \mu_B(x)\right), x \in X \quad (25)$$

where *max* is the classical maximum operator.

II. Intersection (Conjunction): The intersection of two fuzzy sets *A* and *B* is a fuzzy set C such that the membership *C* value at any *x* value is the minimum of the membership values of the two fuzzy sets:

$$\mu_C(x) = \mu_{A\cap B}(x) = min\left(\mu_A(x), \mu_B(x)\right), x \in X \quad (26)$$

where *min* is the classical minimum operator.

III. Subset (Containment): The set *A* is a subset of the set *B* when:

$$A \subseteq B \Rightarrow \mu_A(x) \leq \mu_B(x), x \in X \quad (27)$$

IV. Complement (Negation): The complement of fuzzy set *A*, represented by $\bar{A}$ is defined as:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x), x \in X \quad (28)$$

A fuzzy relation represents the degree to which two or more crisp sets are associated, interconnected, or interdependent. Assuming *U* and *V* are two crisp sets, then the fuzzy relation *R (U, V)* is a fuzzy subset of a Cartesian product of $U \times V$, which is stated as: $R: U \times V \rightarrow [0,1]$. The membership function of *R* is $\mu_R(x, y)$ with $x \in U$ and $y \in V$ [79]. The relationship between *x* and *y* is fully related if $R(x, y) = 1$, and if $R(x, y) = 0$, these two elements are not related at all. As a result, the values between zero and one for *R(x, y)* indicate partial association.

Approximate reasoning is a method of inference based on a set of fuzzy if-then rules, which will be combined as a fuzzy relation. A fuzzy if-then rule has the basic form:

$$\textbf{\textit{IF}}\ \text{x is A}\ \textbf{\textit{THEN}}\ \text{y is B} \tag{29}$$

where *A* and *B* are linguistic variables defined by fuzzy sets over the sets U and V respectively. A linguistic variable is variable relating linguistic values (i.e. words in natural language) to fuzzy sets that provide a mathematical semantics to the inherently vague meanings of natural language). For instance, Speed can be a linguistic variable with values *very low*, *low*, *high*, *very high*, etc., rather than numeric values such as 20, 40, 60, 80, etc.. In Formula (29) "*x is A*" is called the "antecedent" or "premise" of a rule, while "*y is B*" is called the "consequence" or "conclusion" [80]. Multiple antecedents or consequents can be joined together in a rule using logical operations. For example, we can have an if-then rule stating ***IF*** *the speed is low AND the distance is small,* ***THEN*** *the braking force should be low*).

The steps of fuzzy reasoning can be given as follows [9]:

A. Input variables are mapped to a fuzzy set representing them (very commonly a singleton fuzzy set).
B. To obtain the membership values of each linguistic label, fuzzified input variables are compared with MFs for each premise.
C. The firing strength (weight) of each rule is computed as the conjunction of the memberships for each premise..
D. The consequence of each rule is weighted by the rule's firing strength, and then all rules are combined together (commonly by a disjunction).

E. Using defined methods, such as centroid of area, the bisector of area, mean of maximum, smallest of maximum, and largest of maximum, the combined consequents are mapped to a crisp output (Defuzzification).

2.5. Type-1 Fuzzy Inferential Systems

The Fuzzy Inference System (FIS) [9, 81] is based on the concepts of fuzzy set theory, fuzzy if-then rules, and fuzzy reasoning. The system consists of four basic functional blocks, as illustrated in Figure 5.

I. **Knowledge Base:** This component itself contains two sub-components, namely the rule base and database. The rule base contains the fuzzy rules and the database includes the membership functions used in fuzzy rules. Generally, this block regulates or controls FIS decision-making.

II. **Fuzzification:** This block provides membership degrees for the fuzzy set antecedents by transforming crisp raw data inputs.

III. **Inference Engine:** This block executes the compositional rule of inference, resulting in a fuzzy output.

IV. **Defuzzification:** In this block the fuzzy sets is transformed into an explicit output (in the form of crisp inputs).

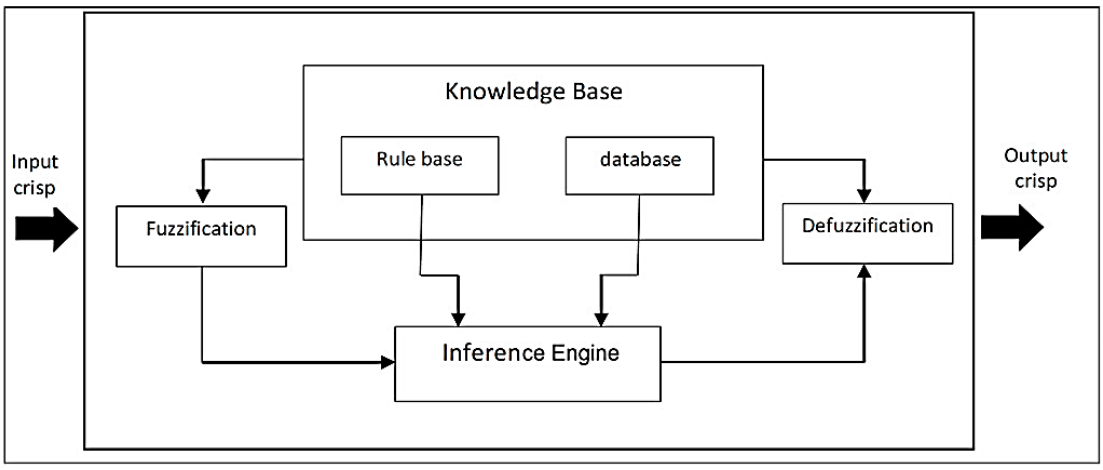


Figure 5 - Fuzzy Inference System (FIS) [16]

Mamdani and Takagi-Sugeno-Kang (TSK) are the principal designs for fuzzy inference systems [82-85]. They differ mainly in the way their fuzzy rules are applied, how they are aggregated, and how they are defuzzified. The first application of the Mamdani fuzzy inference system was the control of a steam engine and boiler combination using linguistic rules derived from human operators [83]. Several defuzzification methods are available in this model, the most famous of which is Center of Area (COA). The center of an area Z_{COA} is denoted as follows:

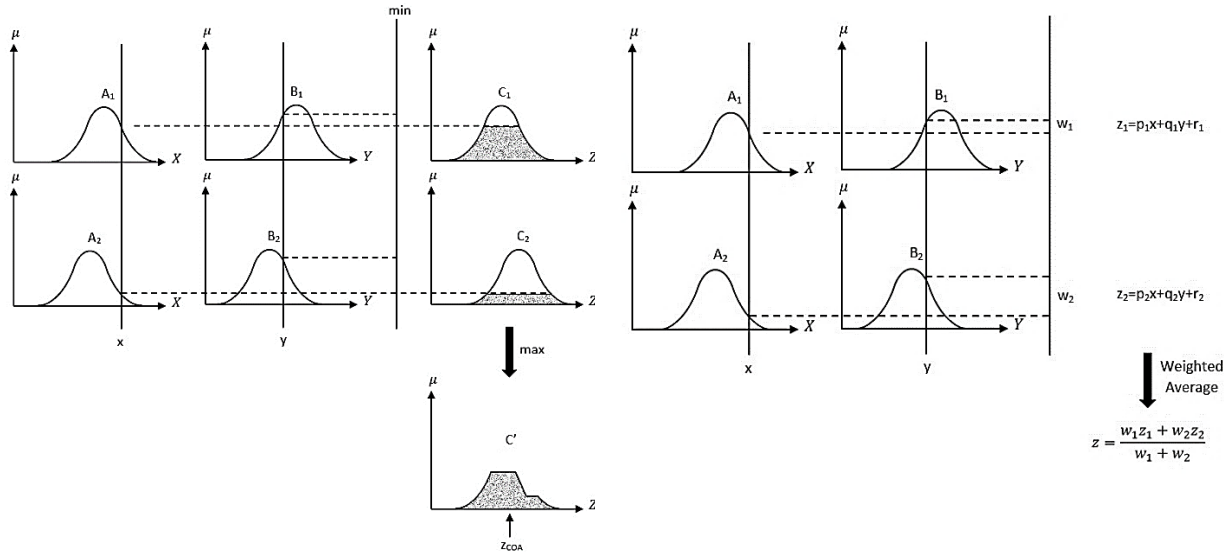
$$Z_{COA} = \frac{\int_Z \mu_A(z) z dz}{\int_Z \mu_A(z) dz} \qquad (30)$$

where $\mu_A(z)$ is the aggregated output membership function.

The Takagi-Sugeno-Kang (TSK) fuzzy inference system was first introduced by Takagi, Sugeno and Kang in 1985 [84, 85]. In TSK model, each rule has a crisp output, and the overall output is determined as a weighted average of each rule's output and represented as:

$$Z = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} z_i} \qquad (31)$$

where z_i is a polynomial function of the inputs (most commonly a linear function).



A. Mamdani FIS B. TSK FIS

Figure 6 - Examples of FISs [83-85]

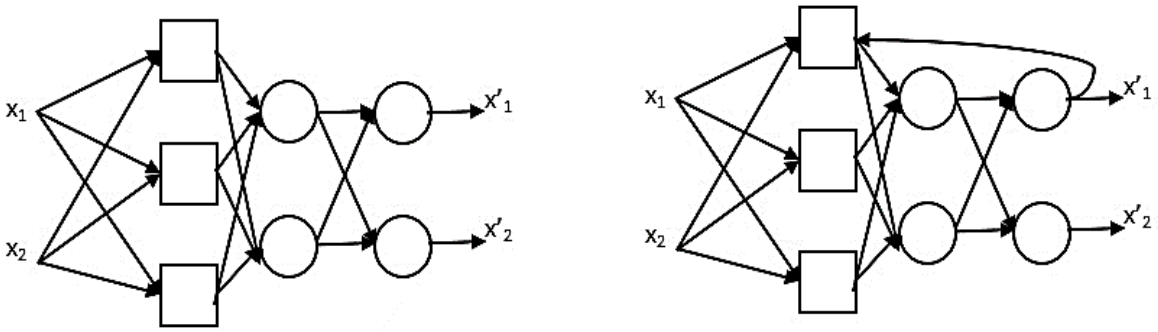
2.6. Neuro-Fuzzy Systems

Neuro-fuzzy systems are artificial intelligence approaches that combine fuzzy logic and neural networks in a hybrid system. This system has the advantage of tuning fuzzy rules using learning algorithms applied to neural networks. By taking into account rule-based fuzzy reasoning in its construction, the neural network can improve transparency [86]. Several neuro-fuzzy systems have been presented in the literature, including Fuzzy Adaptive Learning Control Network (FALCON) [87], Fuzzy Net (FUN) [88], Adaptive Neuro-Fuzzy Inference System (ANFIS) [9], etc. Amongst these, perhaps the most well-known is ANFIS. Yet ANFIS is actually a class of adaptive networks, so first we will review these networks and then study ANFIS.

2.6.1. Adaptive Network

The adaptive network consists of numerous nodes connected by directed links. A node represents a process unit, and a link represents a communication link between nodes. The transfer function of some or all of the nodes are adaptive, which means their parameters can be modified. To

minimize the system error measure, which can be the difference between the desired and actual output, a learning rule defines the way these parameters should be updated. There are two types of adaptive networks: feedforward and recurrent [89]. Adaptive networks are called feedforward if their output is spread from input to output. Alternatively, we have a recurrent adaptive network when we have feedback links which cause a circular path or loop within the graph of communication links.



A. A feedforward adaptive network B. A recurrent adaptive network

Figure 7 - Types of adaptive networks [89]

2.6.2. ANFIS

The Adaptive Neuro-Fuzzy Inference System (ANFIS) is a five-layer feedforward adaptive network which is equivalent to a TSK FIS [9]. The network combines the ability of ANNs to adapt and learn, and the linguistic nature of fuzzy logic to provide transparency. Depending on the rule, the output can be either a linear combination of input variables plus a constant term or just a constant term [78]:

$$\textbf{IF } x_1=A_1 \textbf{ and } x_2=B_1, \textbf{ THEN } y = ax_1 + bx_2 + c$$

Or

$$\textbf{IF } x_1=A_1 \textbf{ and } x_2=B_1, \textbf{ THEN } y = c$$

where {a, b, c} is a parameter set.

An outline of the basic structure of ANFIS can be seen in Figure 8, which shows two inputs and one output.

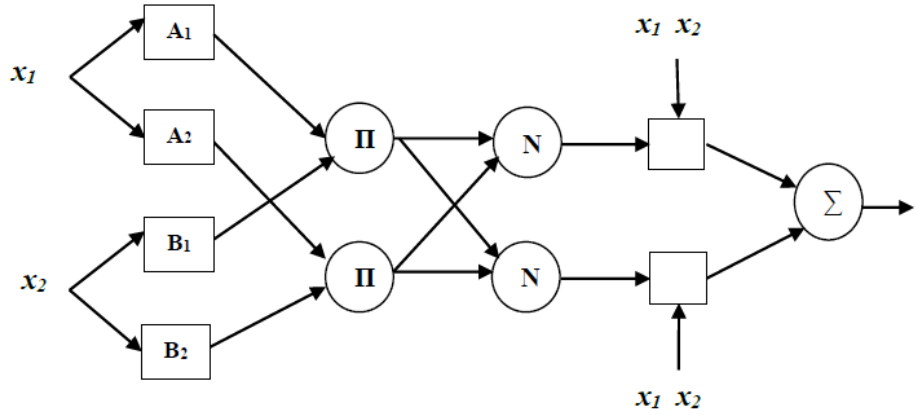


Figure 8 - Basic structure of ANFIS [78]

Each layer performs the following functions [90]:

- Layer 1 (Membership Layer): In this layer, each node represents a fuzzy membership function with adaptive parameters:

$$O_{1,i} = \mu_{A_i}(x_1)\,,\ i = 1,2 \qquad and \qquad O_{1,i} = \mu_{B_{i-2}}(x_2)\,,\ i = 3,4 \qquad (32)$$

where x_1 and x_2 are input values drawn from universes of dicourse U_1 and U_2, A_i or B_{i-2} is a linguistic label associated with a fuzzy subset of U_i, and $O_{1,i}$ is the membership value of the input in that fuzzy set.

- Layer 2 (Firing Strength Layer): In this layer, the output of each node represents the firing strength of a rule as a product of its incoming signals. The output of each node in this layer is given by:

$$O_{2,i} = w_i = \mu_{A_i}(x_1)\mu_{B_i}(x_2)\,, \qquad i = 1,2 \qquad (33)$$

- Layer 3 (Normalized Firing Strength Layer): Every node in this layer calculates a normalized firing strength, $\overline{w}_i$ as follows:

$$O_{3,i} = \overline{w}_i = \frac{w_i}{\sum_{i=1}^{n} w_i}, \quad i = 1, 2 \tag{34}$$

- Layer 4 (Consequent Layer): This layer consists of adaptive nodes that implement the linear consequent function:

$$O_{4,i} = \overline{w}_i f_i = \overline{w}_i (p_i x_1 + q_i x_2 + r_i), \quad i = 1, 2 \tag{35}$$

where $\overline{w}_i$ is the normalized firing strength given by layer 3 and p_i, q_i, r_i are consequent parameters (CPs). If the consequent is a constant value, only r_i is nonzero.

- Layer 5 (Summation Layer): The single node in this layer sums all incoming signals to compute the overall output as:

$$O_{5,i} = y = \sum_{i=1}^{n} f_i \tag{36}$$

The adaptive parameters in ANFIS are the layer 1 MF parameters, and the layer 4 CPs. In order to speed up learning, Jang [78] on a hybrid of least-squares estimation and gradient descent as the learning algorithm, as follows [78]:

a. **Forward Pass:** During the forward pass, MF parameters are held constant and node outputs are propagated up to layer-4, and the consequent parameters then are identified by least-squares estimation. Considering that the premise parameters are fixed, the output can simply be expressed as a linear combination of their values.

$$f = \overline{w}_1 f_1 + \overline{w}_2 f_2 = (\overline{w}_1 x_1) p_1 + (\overline{w}_1 x_2) q_1 + (\overline{w}_1) r_1 + (\overline{w}_2 x_1) p_2 + (\overline{w}_2 x_2) q_2 + (\overline{w}_2) r_2 \tag{37}$$

which is linear in terms of consequent parameters p_1, q_1, r_1, p_2, q_2, and r_2.

$$f = XZ \begin{cases} If\ X\ matrix\ is\ invertible\ then, & Z = X^{-1}f \\ Otherwise\ a\ pseudo-inverse\ is\ used\ to\ solve\ Z, & Z = (X^T X)^{-1} X^T f \end{cases} \quad (38)$$

b. **Backward Pass:** During the backward pass, the CPs are held constant and errors propagate backward to layer-1. MF parameters are updated in layer-1 using gradient descent. The parameters are updated according to the following rule:

$$a_{ij}(t+1) = a_{ij}(t) - \eta . \frac{E}{a_{ij}} \quad (39)$$

which η is the learning rate for a_{ij} (a parameter of the network) and can be expressed as:

$$\eta = \frac{k}{\Sigma(\frac{\delta E}{\delta a_{ij}})^2} \quad (40)$$

where k is the step size (Gradient transition length in parameter space) that can affect convergence speed. To update the parameters of the membership function, the chain rule is used to calculate partial derivatives.

$$\frac{\delta E}{\delta a_{ij}} = \frac{\delta E}{\delta f} . \frac{\delta f}{\delta f_i} . \frac{\delta f_i}{\delta w_i} . \frac{\delta w_i}{\delta \mu_{ij}} . \frac{\delta \mu_{ij}}{\delta a_{ij}} \quad (41)$$

2.7. Complex Fuzzy Sets

Complex Fuzzy Sets (CFSs) are an extension of type-1 fuzzy sets, in which the co-domain of the membership function is some subset of the complex plane, i.e. , $\mu(x) \in \mathbb{C}$ for some x drawn from a universal set U. While there were some earlier investigations of this concept (e.g. [19, 91]), the current understanding of CFS comes from Ramot et al. [2]. They defined a membership function to be $\mu(x) = r_s(x)e^{(jw_s(x))}$, where $r_s \in [0,1]$ is the magnitude and $w_s(x)$ is the phase of the complex

fuzzy set S, and $j = \sqrt{-1}$. Clearly, this means that a complex fuzzy set has a two-element membership vector for every object $x \in U$. A CFS from [2] has a co-domain as shown in Figure 9. In this graph, the complex plane Re×Im is located at right angles to the universe of discourse U and the unit disc D is projected along U, forming a cylinder. A CFS will then be a trajectory within this cylinder.

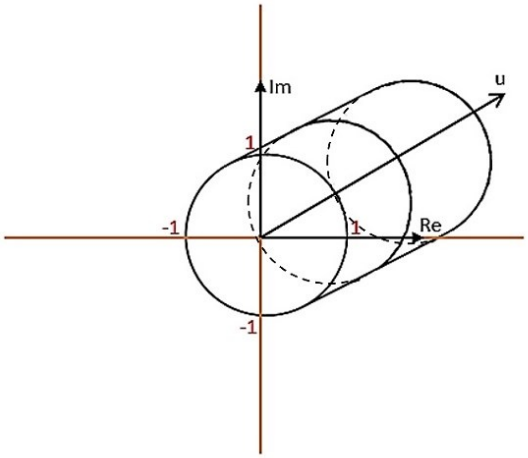


Figure 9 – Codomain for a Complex Fuzzy Set [5]

In a review of CFS [5], there are seven main open questions being pursued by researchers. Two of these (Q3: What functional forms for complex fuzzy memberships have been investigated? Q7: How are complex fuzzy sets operationalized?) are directly addressed by the current dissertation, and we discuss them in more depth below. It is also, however, worthwhile to recall the first question posed in that review: *"In what ways and for what phenomena is CFS&L a more effective framework for reasoning under uncertainty than existing fuzzy logics?"* [5]. One answer, proposed first by Ramot et al. [2], is that CFS might be a good model for periodic phenomena, as the phase dimension of the membership vector is unbounded. Dick [23] expanded on this idea, suggesting that CFS might capture *regularity*, which refers to *approximately* periodic phenomena that do not precisely repeat. This was a concept suggested by Zadeh (unpublished) as a

generalization of stationarity. Dick went on to suggest that, since time series forecasting often deals with such approximately periodic observations, CFS might be a good candidate for developing forecasting algorithms. The studies investigating this possibility form the bulk of the answer to Q7 above.

2.7.1. Complex Fuzzy Membership Functions

As pointed out in [5], type-1 fuzzy memberships are most commonly parameterized functions (e.g. Gaussian, triangular, trapezoidal, etc.) This is particularly important for learning algorithms, as those parameters can be optimized to fit a dataset. The development of CFS has likewise focused on two families of parameterized functions: sinusoidal CFS were proposed as a part of the ANCFIS architecture, while Gaussian CFS were proposed for the CNFS family. These are two very different approaches to modeling uncertainty, which we further explore below.

The sinusoidal CFS has the general form [3]:

$$r = d \cdot sin(a \cdot (\theta = g(x)) + b) + c \qquad (42)$$

where r is the magnitude of the CFS, θ is its phase, $g(): U \rightarrow [0,2\pi]$ relates the phase of the CFS membership to the object x, *sin*() is the familiar sine wave function, and a, b, c, d are adaptive parameters. To this point, the only form used for $g()$ has been the identity function, so effectively $\theta = x$. The sine function was chosen as the functional form to serve as an idealized prototype of a *regular* (approximately periodic) phenomenon; the degree to which observations match the prototype is then the membership of the phenomenon in the CFS (this is computed by convolving the sampled CFS against a window of observations in ANCFIS). Furthermore, as is well known, any periodic function can be approximated by a Fourier series, each term of which is a sine wave. Thus, a rule-base formed of sinusoidal CFS could be an effective, and compact, model for arbitrary *regular* phenomena [3, 23]. A sinusoidal CFS is depicted in the 3-dimensional plot of

Figure 10, in which we see a repeating heart-shaped trajectory in the complex plane (Re×Im) traced along the independent variable *X*.

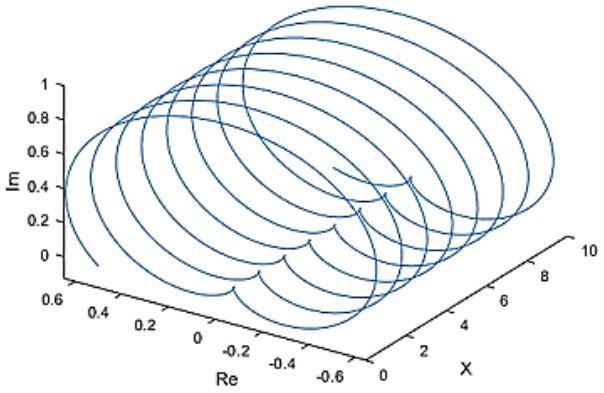


Figure 10 - Sinusoidal CFS: a=5, b=0, c=0.5, d=0.5

The complex Gaussian membership function is an extension of the Gaussian function into complex numbers, again in polar form. The general form of this membership function is [92]:

$$\mu(x; m, \sigma, \lambda) = r_\mu(x; m, \sigma) \cdot exp(j \cdot \omega_\mu(x; m, \sigma, \lambda)) \quad (43)$$

$$r_\mu(x; m, \sigma) = exp\left[-0.5 \cdot \left(\frac{x-m}{\sigma}\right)^2\right] \quad (44)$$

$$\omega_\mu(x; m, \sigma, \lambda) = -exp\left[-0.5 \cdot \left(\frac{x-m}{\sigma}\right)^2\right] \cdot \left(\frac{x-m}{\sigma^2}\right) \cdot \lambda \quad (45)$$

where $x \in U$ is the object, m is the mean of the Gaussian, σ^2 its variance, and λ modifies the phase of the CFS.

For a Gaussian-type complex fuzzy set S, the membership function $\mu_s(x; m, \sigma, \lambda)$ can also be written in Cartesian form as follows [92]:

$$\mu_s(x;m,\sigma,\lambda) = Re\big(\mu_s(x;m,\sigma,\lambda)\big) + jIm(\mu_s(x;m,\sigma,\lambda))$$
$$= r_s(x;m,\sigma)cos(\omega_s(x;m,\sigma,\lambda)) + jr_s(x;m,\sigma)sin(\omega_s(x;m,\sigma,\lambda))$$
$$= exp\left[-0.5\cdot\left(\frac{x-m}{\sigma}\right)^2\right]cos\left(-exp\left[-0.5\cdot\left(\frac{x-m}{\sigma}\right)^2\right]\cdot\left(\frac{x-m}{\sigma^2}\right)\cdot\lambda\right)$$
$$+ jexp\left[-0.5\cdot\left(\frac{x-m}{\sigma}\right)^2\right]sin\left(-exp\left[-0.5\cdot\left(\frac{x-m}{\sigma}\right)^2\right]\cdot\left(\frac{x-m}{\sigma^2}\right)\cdot\lambda\right) \quad (46)$$

where $j = \sqrt{-1}$, $r_s(x;\mathrm{m},\sigma)$ is the amplitude function of the complex membership, and $\omega_s(\mathrm{x};\mathrm{m},\sigma,\lambda): U \to [0,2\pi]$ is the phase function. A Gaussian CFS is depicted in Figure 11; unlike the sinusoid, the Gaussian CFS has a single teardrop-shaped loop centered at its mean value, and approaches a value of $0+0j$ otherwise.

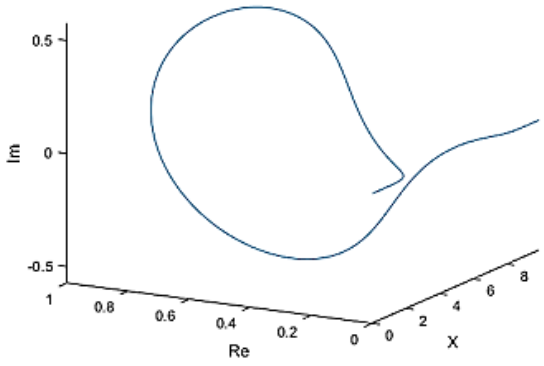


Figure 11 - Gaussian CFS: m=5, σ=1, λ=1.5

Note that the Gaussian CFS is a unimodal curve near its mean and elsewhere nearly zero – inspired our interest in combining the sinusoidal and Gaussian CFS in a single system. Very few realistic time series are purely periodic signals, and we have observed difficulties in modeling outliers with the ANCFIS architectures. A Gaussian CFS could be a better model for such rarer events. Therefore, in the experiments of this dissertation, we hypothesized that adding Gaussian

CFS neurons to a neuro-fuzzy architecture for time series forecasting will improve its accuracy compared to the same architecture using only sinusoidal CFS.

2.7.2. Complex Fuzzy Set Operations

Suppose we have two complex fuzzy sets, A and B, with membership degrees $\mu_A(x) = r_A(x)e^{(jw_A(x))}$ and $\mu_B(x) = r_B(x)e^{(jw_B(x))}$, respectively. Following is the definition of the union, intersection, and complement of these two complex fuzzy sets [2]:

- Union $A \cup B$:

$$\mu_{A\cup B}(x) = [r_A(x) \oplus r_B(x)].\, e^{jw_{A\cup B}(x)} \tag{47}$$

- Intersection $A \cap B$:

$$\mu_{A\cap B}(x) = [r_A(x) \star r_B(x)].\, e^{jw_{A\cap B}(x)} \tag{48}$$

where the $\oplus$ and $\star$ are some t-conorm and t-norm operators respectively. It remains to define $w_{A\cup B}$ and $w_{A\cap B}$. The following are several possibilities for their calculation [2]:

Sum: $w_A + w_B$ *(49)*

Difference: $w_A - w_B$ *(50)*

Max: $\max(w_A, w_B)$ *(51)*

Min: $\min(w_A, w_B)$ *(52)*

'Winner Take All': $\begin{cases} w_A: & r_A > r_B \\ w_B: & r_B > r_A \end{cases}$ *(53)*

Average: $\frac{w_A + w_B}{2}$ *(54)*

Weighted Average: $\frac{r_A w_A + r_B w_B}{r_A + r_B}$ *(55)*

- Complement $\bar{A}$:

$$\mu_{\bar{A}}(x) = r_{\bar{A}}(x).e^{jw_{\bar{A}}(x)} = (1 - r_A(x)).e^{jw_{\bar{A}}(x)} \quad (56)$$

Additionally, [1] introduced three other new operators on CFS as follows:

- Reflection:

$$Ref\big(\mu_S(x)\big) = r_S(x).e^{-jw_S(x)} \quad (57)$$

where *S* is a complex fuzzy set on U, and $\mu_S(x) = r_S(x).e^{jw_S(x)}$.

- Rotation:

$$Rot_\theta\big(\mu_S(x)\big) = r_S(x).e^{j(w_S(x)+\theta)} \quad (58)$$

where S is a complex fuzzy set on U, $\mu_S(x) = r_S(x).e^{jw_S(x)}$, and θ is the rotation (anticlockwise) of *S*.

- Directional Complex (DC) Fuzzy Complement (The combination of rotation and complement):

$$\mu_{\bar{S}^\theta}(x) = c(r_S(x)).e^{j(w_S(x)+\theta)} \quad (59)$$

where c is any complement function, c:[0,1]→[0,1].

In [23] Dick showed that considering the phase as a relative quantity in [2, 22] can be interpreted as rotational invariance, meaning that if two vectors undergo rotation by ϕ radians

about the origin, their union, intersection, or complement will be rotated by the same amount. Figure 12 illustrates this point.

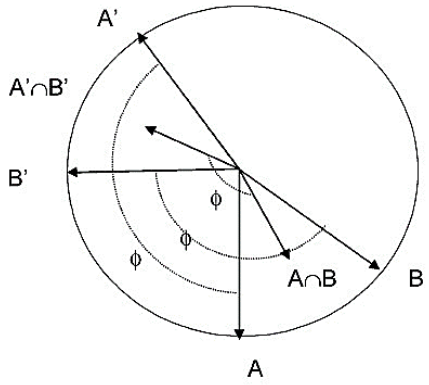


Figure 12 - Rotational Invariance [23]

The images of A and B after a rotation by ϕ radians are denoted A' and B'. Rotational invariance states that e.g. $A' \cap B'$ is the image of $A \cap B$, rotated by ϕ radians, as shown in Figure 10. Thus, the image of an operation after its arguments are rotated by a phase of ϕ is the image of the same operation on the original arguments, rotated by ϕ.

It was shown that the algebraic product and the traditional complement, $f(x) = -x$, are not rotationally invariant. Instead, Dick [23] proposed a new formulation of membership degree by considering amplitude and phase simultaneously. The algebraic product was shown to be a conjunction operator, and the existence of a dual disjunction operator was proved. (However, note that Dai [93] showed that the proposed lattice was incorrect and corrected it). He then argued that capturing the behavior of approximately periodic phenomena was a possible application for CFL, and sinusoidal functions were suggested as appropriate complex fuzzy membership functions.

Also, in [22], an aggregation operation was proposed to combine several complex fuzzy sets with unit circle codomain:

$$v: \{a|a \in C, |a| \leq 1\}^n \rightarrow \{b|b \in C, |b| \leq 1\} \quad (60)$$

$$\mu_A(x) = v\big(\mu_A(x), \mu_A(x), \dots, \mu_A(x)\big) = \sum_{i=1}^{n} w_i \mu_{A_i}(x) \quad (61)$$

where $w_i \in v: \{a|a \in C, |a| \leq 1\}$ for all *i,* and $\sum_{i=1}^{n} w_i = 1$. This operation is implemented as a weighted vector sum that considers the effects of complex fuzzy phases on the aggregation process.

2.7.3. Complex Fuzzy Relation

Complex Fuzzy Relation (CFR) represent both the degree of presence or absence of association, interaction or interconnectedness, and the phase of association, interaction or interconnectedness between the elements of two or more crisp sets [2]. The complex fuzzy relation between two discourse universes *U* and *V* is a complex fuzzy subset of *U*V*, which is characterized by a complex-valued membership function bound to an arbitrary unit disk, and defined as [2]:

$$R(U,V) = \{((x,y), \mu_R(x,y))|(x,y) \in U \times V\} \quad (62)$$

Composition of the complex fuzzy relations either defined on the same product spaces or on the different product spaces were proposed in [22].

2.8. Complex fuzzy Logic

Using the same unit-disc codomain as CFS in [2], Complex Fuzzy Logic (CFL) was first proposed in [22]. In [22], an isomorphic CFL is proposed based on the generalized Modus Ponens inference rule and a complex product as follow:

$$\mu_{A \rightarrow B}(x,y) = \mu_A(x).\mu_B(y) \quad (63)$$

where $\mu_{A \rightarrow B}(x,y)$ is the complex-valued membership function of the implication. As a result, complex fuzzy implication can be expressed quite similarly to ordinary fuzzy implication. Having

complex-valued grades, of course, makes (41) different. The complex fuzzy implication of $\mu_{A\to B}(x,y)$ for the amplitude and phase terms separately is given by:

$$r_{A\to B}(x,y) = r_A(x).r_B(y) \tag{64}$$

$$w_{A\to B}(x,y) = w_A(x) + w_B(y) \tag{65}$$

2.9. An overview of XAI

There has been recent success and advancement in artificial intelligence (AI) and machine learning (ML) algorithms, including shallow and deep artificial neural networks, that can operate better than humans in many diverse fields, but many of them perform ambiguously. This means that their decision-making process and actions are not transparent. This issue has been discussed under the title "explanations and interpretations of the decision-making process" [94-96]. For some applications of artificial intelligence, this explanation may not be necessary, and, according to some researchers, it may also make accessing the models more difficult. However, it is a very important factor in many other AI applications, including military, medical, financial, and law. It will lead to a greater sense of understanding and trust among users [97-99]. There has always been a conflict between explainability (interpretability*)* and accuracy. For example, most of the methods that have higher performance, such as deep neural networks, have less explanations, and in contrast to those that are more explainable, such as decision trees, they have less accuracy [100]. Therefore, a trade-off between the two should be sought [101].

Artificial intelligence systems that provide explanations for their behavior are called explainable artificial intelligence (XAI). In this regard, it should adhere to some general principles to make the intelligent system understandable to users; when describing its capabilities, it should

clearly describe what it has already achieved, what it is currently doing, and what it intends to do in the future [97]. Analyzing artificial intelligence's inferential processes is also a way to generate explanations. However, having an explanatory interface (EI) would be more useful than presenting a user with an interpretable artificial intelligence model. In order to provide explanations, EI must communicate with the user. Since EI is intended to enhance user trust, XAI researchers should measure how much their explanations enhance trust. Metrics such as user satisfaction, which can be gauged by assessing the explanation's clarity and usefulness, are purely based on the user's perspective [100].

Two goals are stated in [102] for science: intelligibility and instrumentality. Intelligibility means explainability, and instrumentality is expressed as performance. The difference between these two goals is similar to an issue that artificial intelligence has been involved in for decades, in that intelligibility and instrumentality are not complementary, that is, improvements in one must come at the expense of the other. This is what was previously introduced as the interpretability-accuracy trade-off [101]. Therefore, XAI is involved with two types of systems; those that emphasize intelligibility and those that focus on instrumentality. Because of this, the first type of artificial intelligence is primarily based on symbolic approaches that are fairly transparent to humans (such as expert systems, logic theory, etc.), while the non-symbolic approaches (such as statistical learning, neural networks, etc.) are less transparent and emphasize accuracy more. Among the first symbolic expert systems that extracted knowledge through IF-THEN inference rules are Heuristic-Dendral [103] and EMYCIN [104], while McCullough-Pitts [105] and DRAGON [106] can be referred to as non-symbolic expert systems.

During the 1970s, neural networks experienced their first winter after the analysis cited in [107] revealed that perceptron could only solve linearly separable problems. Nevertheless, some

work continued, such as the Neocognitron architecture [108] that led to Lecun's convolutional neural networks [109], which continue to dominate research today in signal, image, and video processing. A resurgence in interest in neural networks followed the publication of backpropagation algorithms in the late 1970s [110], followed by Hopfield's research on associative memory [111]. The popularity of neural networks led to new ideas, algorithms, and models being proposed to achieve greater performance and accuracy [112-121]. However, the black-box nature of NNs exacerbated the problem of user confidence in the late 80s and early 90s, when they were considered cutting edge in machine learning, making explanations much more difficult. As a result, XAI was given more attention during this time. An explanation can be represented in a variety of ways. For example, KBANN [115] (as a rule-based neural network architecture) trains the NN on domain datasets before initializing it with rules. It is expected that the rules extracted from this trained model will be more accurate than the original ones. Also, Feedforward networks are often represented by decision trees [122-124]. Additionally, IF-THEN rules cannot capture the effects of recurrent networks' feedback loops. Thus, recurrent networks require different representations. A visualization problem has also been considered when XAI was first developed [125]. Typically, visualization is used to explain an explanation today. Expert systems and NNs explanations were also reduced during the winter break. However, kernel-based learning, and specifically SVMs, still held interest due to their reasonable results. In contrast, SVMs are as much a black box as neural networks. Despite the deep learning revival ending the winters of neural networks and AI, the need to explain NNs has once again intensified. Although some papers claimed there was a basis for explaining SVMs [126]and other forms of machine learning at the time, such as Bayesian networks [127] and probabilistic reasoning [128].

Since NNs and especially deep neural networks (DNNs) have gained popularity since the mid-2010s, a new generation of XAIs has emerged [129-135]. For example, an explainable AI program was launched by DARPA in 2017 [136]. With advances in computing power, technology, visualizing data, and natural language processing, modern XAI is expected to provide better explanations. EI must also be designed to meet the needs of the intended beneficiaries in order to be effective [137]. Furthermore, users in the modern era expect to interact with AI for a longer periods of time. Thus, the AI must be aware not only of the user's goals, knowledge, and experiences but also of how these influence the AI [138]. There are four modern methods exist for generating in-depth explanations, including Layer-wise Relevance Propagation (LRP) [139], Local Interpretable Model-agnostic Explanations (LIME) [94], Testing with Concept Activation Vectors (TCAV) [140], and Rule Extraction with Meta-Features [141]. Each of their resources contains details about them.

2.9.1. XAI and Neuro-Fuzzy Systems

Lotfi A. Zadeh introduced fuzzy theory and its basic concepts in 1965, referring to classical mathematics' inability to deal with imprecise problems in the real world. The fuzzy sets he introduced are an extension of the classical (crisp) set to the unit interval [0, 1] [1]. Then, fuzzy logic, including union, complement, and intersection operations on fuzzy sets, was presented and then operated through a fuzzy inference system (FIS), which mapped input values to output values using fuzzy rules. FIS was remarkable in that the use of fuzzy rules (IF-THENs) made it very transparent [142]. A more interpretable network was created by combining neural networks and fuzzy logic [143, 144], and fuzzy logic has been used since the 1970s to generate explanations. Afterward, Zadeh described fuzzy systems using equations of state [145] and suggested using fuzzy sets for automatic control [146].

For XAI topics that were widely used in FIS, the introduction of Linguistic Variables (LVs) marked an important advance [147-149]; for example, a controller designed by Mamdani [150] was used in FIS to create a fuzzy logic controller (FLC) algorithm. With the help of LVs, FLC made the control rules as transparent as possible. The TSK FLC was then presented [151], which differed from Mamdani in that the outputs were always numerical functions of the inputs, not fuzzy sets. Transparency in FLC can be considered from two perspectives of XAI: on the one hand, since FLC is a linguistic rule base rather than a differential equation with several variables, it is transparent and understandable to humans since they do not require learning differential equations. On the other hand, the explanation for FLC only includes the fuzzy rule base, and neither in Mamdani nor in TSK was an EL designed for it, also there was no any attempt to evaluate the comprehensibility of FLCs.

From the perspective of computational intelligence, researchers have shown that hybrid systems including neural networks, fuzzy logic, and evolutionary computing yield better results than systems that utilize only one approach [89], and from the perspective of XAI, Hybrid systems that include fuzzy logic are more comprehensible and interpretable. Even during the winter of neural networks, hybrid systems were of interest [152-155]. Afterwards, fuzzy neural networks and neural architectures that mimic fuzzy systems (often called fuzzy neural systems) became the main approaches to studying neural fuzzy systems.

A fuzzy neural network is the combination of a neural network architecture with fuzzy logic that is tuned to improve transparency. Hayashi's fuzzy MLP [156] or Pal and Mitra's fuzzy MLP [157] are examples. While maintaining accuracy, these types of networks strived to be more transparent than traditional neural networks. Generally, fuzzy neural networks are more transparent than traditional neural networks without sacrificing much accuracy. In neural fuzzy

systems, however, more transparency is achieved by using FISs. Among all the neural fuzzy systems, ANFIS [9] is the most famous that have been proven to be equivalent to TSK fuzzy systems. There is again a trade-off between interpretability and accuracy when we compare fuzzy neural networks to neural fuzzy systems. While the former is more precise, the latter is more transparent [158]. However, it should be noted that although these architectures emphasize transparency, they do not actually design EI.

Evolutionary fuzzy algorithms (EFAs) are a combination of meta-heuristics and fuzzy algorithms. Like fuzzy neural systems, these algorithms aim to create systems that model datasets more accurately and interpretably. Among the first examples in this field are genetic fuzzy systems [159-161]. Multi-objective genetic fuzzy systems can also optimize FIS accuracy and interpretability simultaneously [101]. A FIS's design and optimization can be done in an EFA. EFA elements can all be learned. Depending on the dataset, an EFA can learn the entire FIS inductively or fine-tune an initial one. EFA algorithms provide explanations based on fuzzy rules, as discussed earlier, but an EI is not designed by them. Like EFAs, deep neuro-fuzzy systems (a combination of fuzzy logic and deep neural networks) also try to improve transparency for DNNs. Fuzzy DNNs include the following examples: Fuzzy Deep MLP [162], Fuzzy Restricted Boltzmann Machine [163], Fuzzified Echo State Networks [164], Fuzzified Stacked Autoencoders [165], Deep Fuzzy MLP [166], Fuzzified ResNet [167], Deep fuzzy Decision Tree [168], and a combination of ANFIS and LSTM [169]. The architectures of deep neuro-fuzzy systems also do not typically include a dedicated EI.

2.10. AI Brittleness

The reality that artificial intelligence (AI) is not as capable as it is thought to be has raised concerns among the public, although it has been very successful in a number of domains recently. Despite

the fact that AI is used in many everyday applications, including shopping and home automation, its use in safety-critical systems such as transportation and medicine is of great concern, since the wrong application of AI can lead to very dire consequences. For example, problems with the car's computer vision can be cited as contributing factors in many fatal Tesla crashes [170, 171]and the death of a pedestrian in an Uber self-driving car crash [172].

While many companies have promised full AI driving for years, many have backed off in an effort to moderate public and investor expectations [173, 174]. Negative sentiment about AI applications is on the rise due to public opposition to AI and privacy, as well as concerns about AI embedded in social media that may manipulate people. According to some experts, this backlash may lead to another artificial intelligence winter, which will reduce trust in legitimate AI developments and reduce funding [175]. Given this potential outcome, it is important to take a step back and analyze exactly why AI is struggling to achieve safety-critical systems and how the roadmap to success must change to achieve positive outcomes.

The use of computer vision in safety-critical topics such as transportation and healthcare is based on algorithms that use machine learning to "understand" the world and make decisions. For example, the use of deep learning algorithms in driverless cars to identify pedestrians [176] or in healthcare, to detect tumors in grainy lung images [177]. Despite significant advances in computer vision and deep learning algorithms over the past decade, such approaches to developing real-world perceptual models have been brittle. Brittleness occurs when an algorithm cannot generalize to conditions beyond its assumptions or adapt to new ones. As an example, many natural language processing (NLP) algorithms are brittle when they can understand the speech of a native English speaker but cannot understand it if spoken by someone with a foreign accent [178]. While this brittleness may be frustrating for a person attempting to navigate through an automated telephone

system, in a safety-critical system that relies on any kind of machine learning to perceive the environment, it can be fatal.

Therefore, when a machine learning algorithm does not have the ability to generalize its perception in the face of uncertainty, perceptual brittleness arises. For example, an inability of driverless car computer vision to cope with weather changes [179] is one example of brittleness. Other examples like when the curbside is partially covered by something like snow and their edges are no longer detectable to the system [180] or not recognizing traffic signs correctly when they are partially covered with something like leaves [181]. Research typically addresses such brittleness by training models with more images and using techniques such as augmentation [182]. Computer vision based on deep learning is still a relatively new field of research, so new problems are being discovered. In a recent study, researchers discovered that neural networks don't capture accurate images of depth [183], which has significant implications for safety. Therefore, a field of study called adversarial machine learning [184] has emerged, which examines how deep learning algorithms can be tricked or defeated (see [185, 186]).

2.10.1. Data/Distribution shifts and brittleness

The term data shift refers to the shift in the distribution of data [187]. A Machine Learning model attempts to uncover the relationship between the input and the target variable. After creating a model on this data, one might feed new data of the same distribution and expect similar results. However, this is rarely the case in real-world situations. Changes in consumer behaviour, technological breakthroughs, political, socioeconomic, and other unpredictable factors can dramatically impact either a) the input dataset, b) the target variable, or c) the underlying patterns and relationships between input and output. They all lead to the same thing in Data Science: model

performance degradation [187]. In order to better understand data shifts, let's give them a more formal definition:

I. **Covariate shift:** it is the change of distributions in one or more of the independent variables (input features) [188]. In mathematical terms, covariate shift is termed the situation where $P_{trn}(Y|X)=P_{tst}(Y|X)$ but $P_{trn}(X) \neq P_{tst}(X)$. Covariate shift may happen due to a changing environment that affects the input variables but not the target variable. To deal with it, various solutions have been provided, such as one provided by [189], which uses node-based Bayesian neural networks (BNNs). In this method, node-based BNNs are proposed that infer epistemic uncertainty by multiplying hidden nodes with latent random variables. As a result, they construct BNNs that perform well under covariate shifts caused by input corruption by interpreting these latent noise variables as implicit representations of simple and domain-agnostic data perturbations during training. Their study observed that implicit corruption diversity depends on the entropy of the latent variables, which led them to propose a straightforward method for increasing entropy during the training of these variables. After evaluating the method on out-of-distribution image classification benchmarks, they showed improved uncertainty estimation for node-based BNNs with covariate shifts caused by perturbations of inputs. Additionally, the method provides robustness against noisy training labels.

II. **Prior probability shift**: it can be thought of as the exact opposite of covariate shift; it is the case that input feature distributions remain the same but the distribution of the target variable changes [190]. In mathematical terms, prior probability shift is termed the situation where $P_{trn}(X|Y)=P_{tst}(X|Y)$ but $P_{trn}(Y) \neq P_{tst}(Y)$. A prior probability shift can occur in cases where despite the input variables remain the same, our target variable changes. To

deal with this type of shift, there are several solutions, one of the most recent of which is [191] designed an algorithm, called CAPE (Combinatorial Algorithm for Proportional Equality), that ensures fair classification under such shifts. The algorithm has two phases: training and prediction. CAPE takes as input a training dataset D and a vector $\Theta = (\theta_1, \ldots, \theta_k) \in [0, 1]^k$. CAPE trains an ensemble of classifiers, with the desired prediction prevalence of each classifier being one of the $\theta \in \Theta$ values. Moreover, CAPE is separately trained for each group $z \in [G]$, since it is hypothesized that the relationship between the non-sensitive features X and the outcome variable Y may differ across groups. Thus, each group is best served by training classifiers on datasets obtained from the corresponding group. The results have shown that CAPE guarantees a high degree of fairness in its predictions.

III. **Concept drift:** it happens where the relations between the input and output variables change. So we are not anymore only focusing on X variables or only the Y variable but on the relations between them. In mathematical terms, a concept drift is termed the situation where $P_{trn}(Y|X) \neq P_{tst}(Y|X)$. A concept drift may happen in situations where the data is trully temporal and thus depend heavily on time. To deal with this type of shift, there are several solutions, one of the most recent of which is [192] which proposed two techniques, an Error Rate Based Concept Drift Detection and Data Distribution Based Concept Drift Detection and studied their impact.

Other categories of distribution shifts are also provided. According to [193], it can be divided into two types: domain generalization and subpopulation shift. It has been shown that either of these shifts can significantly degrade model performance in many real-world scenarios. In domain generalization, the training and test distributions comprise data from related but distinct

domains, such as patients from different hospitals [194], images taken by different cameras [195], bioassays from different cell types [196], or satellite images from different countries and time periods [197]. Subpopulation shift considers test distributions that are subpopulations of training distributions, with the goal of doing well even in worst-case subpopulations; for example, we might search for models that perform well across all demographic subpopulations, including minority populations [198].

While these types of distribution shifts are commonly observed in real-world deployments, they are underrepresented in ML datasets today [199]. As most of these datasets were designed for the standard i.i.d. (Independent and Identically Distributed) setting, with the same distribution as the training and test sets, previous research has focused on retrofitting them with distribution shifts that are cleanly characterized but do not always occur in real-world deployments. As an example, many papers in the last few years have examined datasets with shifts caused by synthetic transformations, such as changing the colour of MNIST digits [200], or by generalizing from cartoons to photos [201]. For systematic studies, datasets such as these are important testbeds; however, to develop and evaluate methods for real-world shifts, we need datasets that capture real shifts.

ML models face a wide array of distribution shifts in the wild, which are represent by WILDS [193], a curated benchmark of 10 datasets with evaluation metrics. WILDS datasets span many important applications: animal species categorization [202], tumour identification [203], bioassay prediction [204, 205], genetic perturbation classification [206], wheat head detection [207], text toxicity classification [208], land use classification [209], poverty mapping [210], sentiment analysis [211], and code completion [212, 213]. There have been natural shifts in the distribution of these datasets as a result of a variety of cameras, hospitals, molecular scaffolds,

experiments, demographics, countries, time periods, users, and codebases. For WILDS to succeed, domain experts often need to deal with distribution shifts in order to collect data. Using the following criteria, they identified, selected, and adapted datasets for WILDS: 1) Distribution shifts with performance drops 2) Real-world relevance 3) Potential leverage (to generalize models to arbitrary distribution shifts, benchmarks must be non-trivial but also feasible to solve).

Each WILDS dataset is associated with a type of domain shift: domain generalization, subpopulation shift, or a hybrid of both. In each setting, it can be seen the overall data distribution as a mixture of D domains D = {1, . . . ,D}. Each domain d ∈ D corresponds to a fixed data distribution P_d over (x, y, d), where x is the input, y is the prediction target, and all points sampled from P_d have domain d. In order for a dataset to be included in WILDS, its train/test split must cause significant performance drops in standard models. The researchers determined this for each dataset by training standard models using empirical risk minimization (ERM), i.e. minimizing the average training loss, and then comparing their out-of-distribution (OOD) performance with their in-distribution (ID) performance [193]. The results obtained for each data set show that the performance of OOD was consistently and significantly lower than that of ID. Overall, the obtained results demonstrated that the real-world distribution shifts reflected in the WILDS datasets meaningfully degrade standard model performance.

2.11. Conclusion

Throughout this chapter, we reviewed the types of algorithms and methods available for dealing with time series forecasts (TSF), which can be broadly categorized into two categories: statistical models and machine learning-based models. Having reviewed the statistical models in the first part, in the second part we examined one of the most important theories and concepts of machine learning, fuzzy set and logic (FS&L), followed by one of its main extensions, complex fuzzy set

and logic (CFS&L). The following chapters have been devoted to discussing various models based on this, as well as the proposed algorithms and methods for this purpose.

Moreover, based on what was discussed and reviewed about XAI, the effect of different architectures and approaches on XAI and how they affect the perception of explainability was investigated. This process was followed by reviewing basic architectures to symbolic and non-symbolic approaches and then computational intelligence until we reached the role of fuzzy sets and fuzzy logic, followed by fuzzy neural networks and neural fuzzy systems and their impact in explaining artificial intelligence. Then, we examined the combination of other machine learning algorithms such as evolutionary algorithms and deep neural networks with fuzzy logic and concluded that the artificial intelligence systems based on fuzzy sets and logic act very transparently and this advantage is more evident for hybrid neuro-fuzzy systems as well as fuzzy-evolutionary systems. Finally, the necessity of designing dedicated explanatory interfaces (EIs) was emphasized.

Finally, it mentioned that among the many safety-critical applications that are being transformed by artificial intelligence, machine learning is among those that will offer new forms of collaboration between humans and computers that were previously unimaginable. Even though AI does have limitations, especially in safety-critical systems. To ensure that AI-enabled systems can operate effectively within their intended operational domains, companies need to develop clear criteria and testing protocols before buying or approving them. Data/distribution shift as a possible phenomenon in machine learning models can significantly degrade model performance in many real-world scenarios and it has different types whose definition and some strategies to deal with them were referred to above.

3. Chapter III: Neuro-Fuzzy Systems Employing Complex Fuzzy Set and Logic

3.1. Introduction

As described in the previous chapter, one of the notable extensions of Type-1 fuzzy set and logic (FS&L) is the complex fuzzy set and logic (CFS&L) introduced in [2, 22], on which various architectures have been designed and implemented. They have shown promising results in time series forecasting (TSF). The adaptive Neuro-Complex Fuzzy Inferential System (ANCFIS) [3] was the first neuro-fuzzy architecture to combine CFS and rule inference for TSF. Like its predecessor (ANFIS [9]), ANCFIS combines least-mean-squares (LMS) optimization in the forward pass with gradient descent in the backward pass, followed by a derivative-free final optimization. However, this hybrid algorithm is relatively slow, which prevents its wider use. Therefore, randomized learning was explored in CFS&L-based architectures called ANCFIS-ELM [214] and RANCFIS [215].

In ANCFIS-ELM and RANCFIS, the training algorithm is based on randomized learning, which was discussed in the previous chapter. In both architectures, the layer-1 MF parameters are randomly selected. As these are the only parameters updated on the backwards pass, this stage of learning can be entirely eliminated, resulting in a very large speedup. The consequent parameters are learned in the forward pass as usual; ANCFIS-ELM rules use a singleton output, while RANCFIS uses a linear function. Another architecture in the ANCFIS family is FANCFIS [216], which uses a Fourier transform to directly estimate the parameters of the layer-1 CFSs; this also results in the complete elimination of the backwards pass, but at the cost of executing a Fast Fourier Transform (FFT) on the training dataset.

3.2. ANCFIS

The ANCFIS architecture [3] is a six-layer model that works based on CFS&L [2, 22, 23]. The ANCFIS architecture is derived from the ANFIS architecture [9], but there are some fundamental differences between them. Firstly, in order to work with periodic time series, ANCFIS uses a sinusoidal Membership Function (MF) as suggested by Dick [23] and it is as follows:

$$r(\theta) = dsin(a(\theta = x) + b) + c \qquad (66)$$

Where $r(\theta)$ is amplitude, θ is the phase of a CFS, x is an object of the Universal set X, the coefficients of a, b, c, d shows the MF parameters. Also, it should be taken into account that the following two conditions must be considered in order to keep the complex fuzzy membership degree within the unit disc of the complex plane:

$$0 \leq d + c \leq 1 \quad , \quad 1 \geq c \geq d \geq 0 \qquad (67)$$

The second difference is an additional layer in ANCFIS to implement rule interference, inspired by Ramot in [22]. Also, we need to know that the signals in this layer are complex-valued. The last but not the least, the gradient descent phase employs a derivative-free optimization step. An ANCFIS network for univariate time series with two rules is depicted in Figure 14.

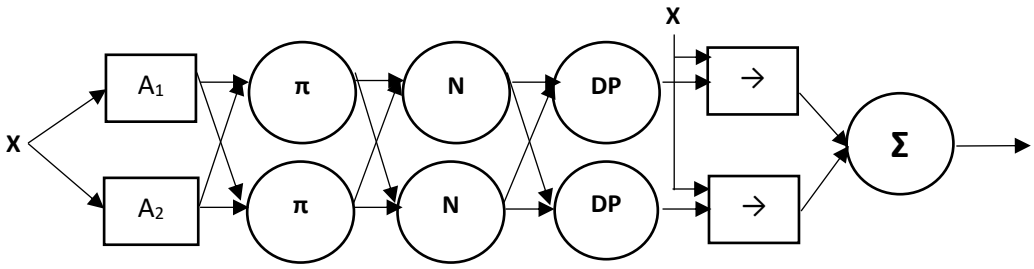


Figure 13 - ANCFIS network for univariate time series [3]

Theneuron transfer functions in each layer are as follows [3]:

- Layer 1: The parameters $\{a_i, b_i, c_i, d_i\}$ in Formula (42) are set in this layer for $i = 1,2 ,\dots ,$ *n_CMF*, where *n_CMF* is the number of complex membership functions. Then, the membership of an input vector in a CFS is determined by complex convolution.

$$conv = \sum_{k=1}^{2n-1} h(k) = \sum_{k=1}^{2n-1} \sum_{j=max(1,k+1-n)}^{min(k,n)} f(j)g(k+1-j) \qquad (68)$$

where $f(.)$ is a data point in the n-element input vector, $g(.)$ is sampled membership function, *n* is the length of input vector, and *k* is the element index of the complex samples. Then, the grades are passed to the Elliot function to ensure that they are restricted to the unit disk, without changing the phase of the convolution sum.

$$O_{1,i} = \frac{conv}{1+|conv|} \qquad (69)$$

- Layer 2: Each node in this layer represents a complex fuzzy rule. Thus, the node output is the algebraic product of the complex inputs (called the firing strength).

$$O_{2,j} = \prod_i O_{1,j} \quad , \quad i = 1, 2, \dots, |O_1| \qquad (70)$$

where $|O_1| = m^n$ (*m* is the number of membership function and *n* is the number of input vectors).

- Layer 3: The output of each node in this layer is the i^{th} rule's normalized firing strength.

$$O_{3,i} = \overline{W_i} = \frac{w_i}{\sum_{j=1}^{|O_2|} |w_j|} \quad , \quad i = 1,2, \dots, |O_2| \qquad (71)$$

where $|O_2|$ is the number of rules.

- Layer 4: Each node in this layer represents interference between a rule and the other rules being fired. It is computed from the dot product of the output of the *i*-th node in layer 3 and the complex sum of all other outputs of the layer 3.

$$O_{4,i} = w_i^{DP} = \overline{w_i} \cdot \sum_{i=1}^{|O_3|} \overline{w_i} \ , \ i = 1,2, \dots, |O_3| \qquad (72)$$

where $|O_3|$ is the number of nodes in layer 3.

- Layer 5: The output of each node in layer 5 is a linear combination of input vectors, given by:

$$O_{5,i} = w_i^{DP}\left[\sum_{j=1}^{n} p_{i,j} x_j + r_i\right] \qquad (73)$$

- Layer 6: This layer computes the sum of all incoming signals.

$$O_{6,i} = \sum_{i=1+(j-1)*N}^{j*N} O_{5,i} \qquad (74)$$

where j is the number of outputs and N is the number of rules.

3.3. ANCFIS-ELM

ANCFIS-ELM [214] is a five-layer network based on ANCFIS [3] that uses a randomized learning algorithm; it means the sinusoidal MF parameters are selected randomly, thus this will eliminate the backward pass entirely. Both ANCFIS-ELM and ANCFIS are similar in layers 1 to 4, but since the ANCFIS-ELM is based on the zero-order Takagi-Sugeno model (constant consequents), it does not have ANCFIS's layer 5, and the outputs of layer 4 are directly connected to a weighted sum in the output layer.

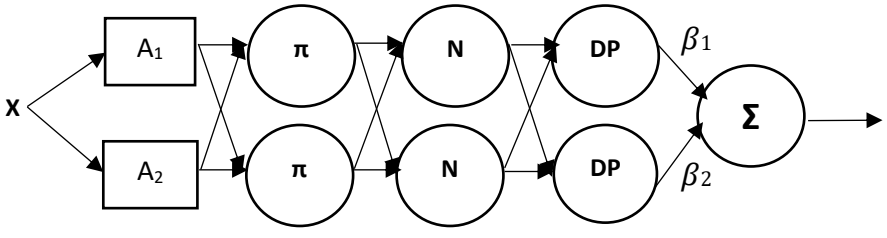


Figure 14 - ANCFIS-ELM network for univariate time series [214]

ANCFIS-ELM was inspired by both the Extreme Learning Machine (ELM) algorithm [217] and the SLFN architecture [56]. Thus, it uses randomized learning, where the input weights are selected from uniform random values in [-1, 1] and the output are determined by minimizing the following error [56, 218]:

$$\varepsilon^2 = \sum_{i=1}^{N}(y_i - \sum_{j=0}^{k} w_j f_{ij})^2 \qquad (75)$$

where N is the number of data points, y_i is target, k is number of hidden neurons in the hidden layer, w_j is the output weight and f_{ij} is the activation value of the *j-th* hidden neuron on the *i-th* data point.

3.4. RANCFIS

RANCFIS architecture [215] is also a six-layer model using the same neuron transfer functions as ANCFIS [3], but with an additional step for adaptive weights between layers 5 and 6. As with ANCFIS-ELM, the sinusoidal membership function parameters {a, b, c, d} are randomly selected from a uniform distribution and keep constant during the whole training and testing. This eliminates the backward pass in training completely.

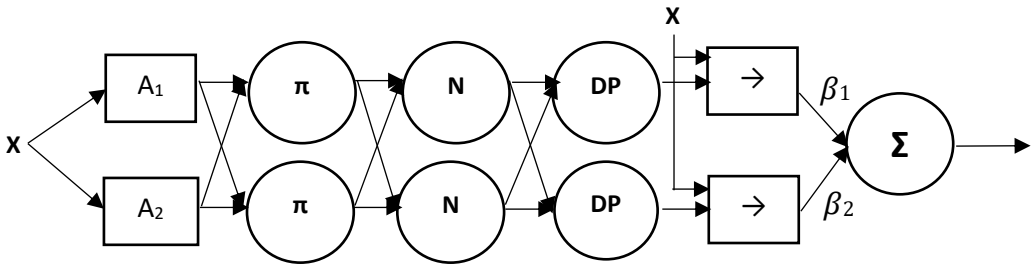


Figure 15 - RANCFIS network for univariate time series [215]

Each connection between layer 5 and 6 is weighted by a coefficient β_i and the i-th input to a layer 6 neuron is:

$$I_{6,i} = w_i^{DP}\left[\sum_{k=1}^{j}\sum_{l=1}^{n} p_{i,kl}x_{kl} + r_i\right].\beta_i = w_i^{DP}\left[\sum_{k=1}^{j}\sum_{l=1}^{n} \gamma_{i,kl}x_{kl} + \eta_i\right] \quad (76)$$

where $\gamma_{i,}=p_{i,kl}.\beta_i$ and $\eta_i=r_i.\beta_i$ The output of the Layer 6 neuron is the sum of these weighted inputs:

$$O_{6,j} = \sum_{i=1+(j-1)*N}^{j*N} I_{6,i} \quad (77)$$

where j is the number of network outputs and N is the number of rules. The output can be reformulated as follow:

$$T = H^{\dagger}\beta \quad (78)$$

where † is the Moore-Penrose generalized inverse, and

$$\beta^T = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} & \eta_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{j1} & r_{j2} & \cdots & r_j & \eta_j \end{bmatrix}, \quad (79)$$

where β is estimated by Recursive Least-Squares (RLS) algorithm [112].

3.5. FANCFIS

The Fast Adaptive Neuro-Complex Fuzzy Inference System (FANCFIS) [216] consists of a six-layer feed forward neural network. While FANCFIS and RANCFIS both overcome ANCFIS's slow learning speed, both remain highly accurate time series forecasting algorithms. The FANCFIS achieves this without the use of random learning, in contrast to RANCFIS. By taking an FFT (Fast Fourier Transform) of a portion of the time series, sinusoidal membership functions are derived directly. In general, it would expect that a FANCFIS network, which achieves the same accuracy on the time series, will be smaller than a RANCFIS network due to the membership functions being tailored specifically for the dataset. A two-step process is used in FANCFIS learning, which includes an initialization and incremental learning phases. The initialization step determines membership function parameters as well as delay embedding. Afterwards, the

FANCFIS network's output weights are calculated. Recursive least squares is used for incremental learning step in order to update the output weights as each new observation is received [216].

By using a Fourier transform, the chronologically first segment is used to induce the CFS membership function parameters in the initialization step. For a time series with *N* data points *A(k), k = 1, 2, ..., N,* the discrete Fourier transform (DFT) is calculated as follows [219]:

$$F_A(n) = \sum_{k=0}^{N-1} \frac{A(k)}{N} e^{\frac{-i2\pi nk}{N}} \qquad (80)$$

where $F_A(n)$ is complex-valued; the real and imaginary components are the amplitude of a sine and a cosine wave, respectively, at frequency *n*. The inverse DFT reconstructs a time series as [219]:

$$A(k) = \sum_{n=0}^{N-1} F_A(n) e^{\frac{i2\pi nk}{N}} = \sum_{n=0}^{N-1} F(n)_{real} \cos\left(\frac{2\pi nk}{N}\right) - \sum_{n=0}^{N-1} F(n)_{image} \sin\left(\frac{2\pi nk}{N}\right) \qquad (81)$$

Also, the membership functions are determined as:

$$r(\theta) = F(n_i)_{real} \cos(\theta) - F(n_i)_{image} \sin(\theta) + DC\,, \qquad i = 1,2,\dots,N_{mf} \qquad (82)$$

where N_{mf} is the number of CFS to be used, *r* and $\theta = \frac{2\pi nk}{N}$ $(k = 1, 2, \dots, N)$ are the amplitude and phase of the complex membership grade, respectively, $x \in X$ is an element of the universe of discourse *X*, n_i is the frequency with the i-th highest power $F(n_i)_{real}$ and $F(n_i)_{image}$ are the real and imaginary parts of the Fourier transform in the frequency of n_i, *N* is the length of the subsequence, and *DC* is the zero frequency component.

3.6. CNFS

In [4], an adaptive complex neuro-fuzzy system (CNFS), based again on ANFIS, is proposed for function approximation. As with ANFIS and ANCFIS, a hybrid learning method is used to train

this architecture. This algorithm uses particle swarm optimization (PSO) algorithm and the recursive least square estimator (RLSE) algorithm. CNFS is a six-layer network; a CNFS with one inupt and two rules is depicted in Figure 13. The neuron transfer functions in each layer are:

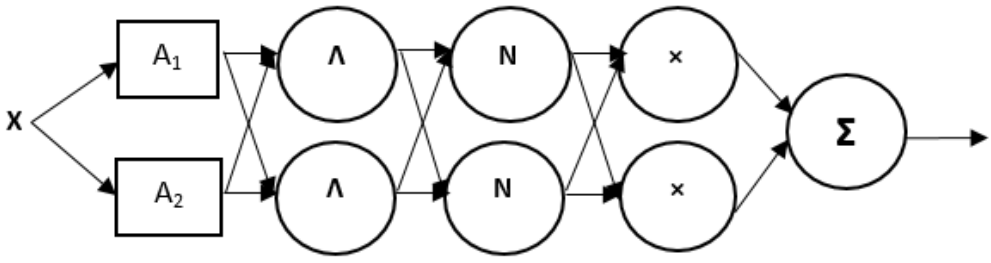


Figure 16 - Complex Neuro-Fuzzy System (CNFS)

- Layer 0 (Input Layer): This layer receives the input vector at time t (Formula (83)) and passes it directly to the next layer.

$$H(t) = [h_1(t), h_2(t), \dots, h_M(t)]^T \qquad (83)$$

where *h* is the base variable for the CFS.

- Layer 1 (Fuzzy-Set Layer): In this layer membership degrees are calculated using CFSs.

- Layer 2 (Firing-Strengths Layer): For calculating the firing strength of the *i-th* rule, we have:

$$\beta^i(t) = \Lambda_{j=1}^{M} r_j^i \left(h_j(t)\right) exp(j\omega_{A_1^i \cap \dots \cap A_M^i}) \qquad (84)$$

where Λ is t-norm operator, r_j^i is the amplitude of complex membership degree for the *j-th* fuzzy set of the *i-th* rule, and ω_s is the phase function.

- Layer 3 (Firing Strength Normalization Layer): The normalized firing strength for the *i-th* rule is given by:

$$\lambda^i(t) = \frac{\beta^i(t)}{\sum_{i=1}^{K} \beta^i(t)} \quad (85)$$

- Layer 4 (Consequents Normalization Layer): The normalized consequent of the *i-th* rule is given by:

$$\xi^i(t) = \lambda^i(t) \times Z^i(t) = \lambda^i(t) \times \left(a_0^i + \sum_{j=1}^{M} a_j^i h_j(t)\right) \quad (86)$$

where a_j^i $(j = 0,1, \dots, M)$ are the consequent parameters.

- Layer 5 (Output Layer): Given the input taken from the previous layer, the CNFS output is calculated as follows:

$$\xi(t) = \sum_{i=1}^{K} \xi^i(t) \quad (87)$$

The output of CNFS is complex-valued:

$$\xi(t) = \xi_{Re}(t) + j\xi_{Im}(t) = |\xi(t)| \times cos\left(\omega_\xi\right) + j|\xi(t)| \times sin\left(\omega_\xi\right) \quad (88)$$

where $\xi_{Re}(t)$ is the real part and $\xi_{Im}(t)$ is the imaginary part of the CNFS output. The CNFS proposed in [4] was modified in [220] by using the product operator instead of min operator in layer 2 for the t-norm calculation of the firing strength. [220] furthermore used a Gaussian-type complex fuzzy set:

$$cGaussian(x, m, \sigma) = Re(cGaussian(x, m, \sigma)) + jIm(cGaussian(x, m, \sigma)) \quad (89)$$

where, *Re (.)* and *Im (.)* are real and imaginary parts of the membership grade which are defined as:

$$Re(cGaussian(x, m, \sigma)) = exp[-0.5(\frac{x-m}{\sigma})^2] \quad (90)$$

$$Im(cGaussian(x, m, \sigma)) = exp[-0.5(\frac{x-m}{\sigma})^2] \times (\frac{x-m}{\sigma^2}) \quad (91)$$

where $\{m, \sigma\}$ are mean and spread of the Gaussian function, and *x* is the input.

Another development of this architecture in [221] was of the hybridization of the artificial bee colony (ABC) algorithm and RLSE as the ABC-RLSE learning method for training the CNFS. The ABC algorithm is an optimization method which simulates the nectar-searching behavior by bees. Also, in [222], a self-organizing learning method is used to set up the structure of the model, which has two phases: the structure and parameter learning phases. For structure learning, the FCM-Based Splitting Algorithm (FBSA) is used to determine the initial structure of knowledge base for the proposed CNFS. For parameter learning the PSO–RLSE method is used. The FBSA algorithm was first proposed in [223], which is a clustering algorithm based on the fuzzy c-mean (FCM) method and a validity index for the clustering results. With the FBSA, the cluster with the worst score can be identified at each step of the algorithm and split into two new clusters. To determine the worst cluster, a score function *S(i)* is defined as follows:

$$S(i) = \frac{\sum_{k=1}^{n} u_{ki}}{number\ of\ data\ in\ cluster\ i} \quad (92)$$

where *S(i)* is the score of the i^{th} cluster and $U=u_{ij}$ is the set of membership degrees by the FCM (*i=1,2,. . .,n* and *j=1,2,. . .,c*).

3.7. Conclusion

In this chapter, we have discussed a variety of CFS&L-based architectures. All of these architectures have improved the structure and algorithms used in previous ones to improve the accuracy of results and/or learning speed. While these efforts are very significant and valuable, these architectures still have problems, one of the most important of which is dealing with big data. In the following chapters, we address this limitation.

4. Chapter IV: Condition Monitoring

In this chapter, we address an important industrial-research topics, condition monitoring of small electric induction motors. We discuss the basic design of these motors, and the major types of faults they suffer. We then discuss our procedures for obtaining, extracting, and producing the relevant dataset for our subsequent experiments (chapters 5 and 6), which are presented in this dissertation. Then, we will develop a condition-monitoring solution for these kind of motors, based on anomaly detection in time series [6].

4.1. Introduction

Induction motors are a hugely important class of electrical machines, shouldering much of the task of transforming energy into useful work in modern industry [224]. As such, maintenance of these often-critical machines is of paramount importance. The most common faults that occur in induction motors are bearing faults, stator-shortfaults, and cracked rotor bars [225]. Bearings faults can be caused by incorrect lubrication, mechanical stresses, incorrect assembling, misalignment, etc. Stator faults can be consequences of overheating, contamination, project errors, etc. Rotor faults are usually caused by broken bars or end rings, rotor misalignment and imbalance [226]. Figure 17 depicts these fault locations on an induction motor [227]. Bearing failures account for about 40% of faults in induction motors [226] while stator, rotor and other faults comprise roughly 38%, 10% and 12%, of the faults, respectively (see Figure 18) [228]. Over 40 years of research has shown that these fault classes can be detected in motor sensor data before motor failure; there has thus been great interest in using this fact to reduce motor failures.

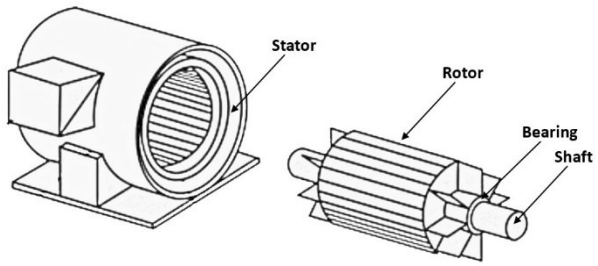


Figure 17 - Induction motor fault locations [226]

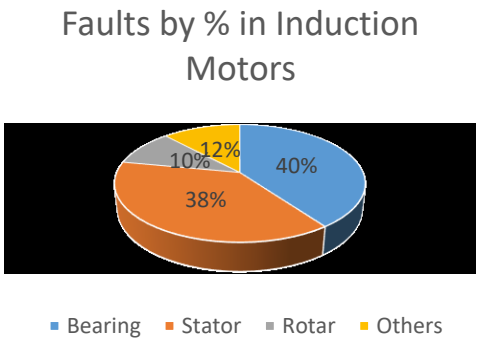


Figure 18 - Types of induction motors faults

Condition Monitoring (CM) is the use of non-destructive testing or sensed data to detect changes within a monitored system. While earlier approaches could only flag the occurrence of a change, more modern approaches can identify a causative fault, its location, and the damage done so far [229]. These are largely inferential sensing approaches (the estimation of a complex, time-varying system state based on reading from simpler sensors [230-232]); some examples include [233-236]. Some adaptive systems can even attempt self-repair based on that information [229]. CM today is a well-known, economical approach to maintaining large and expensive systems, allowing for cheaper preventative maintenance before a fault worsens or causes failure [237, 238].

Effective sensing modalities for CM of electric motors include axial electromagnetic flux monitoring, current and voltage monitoring, thermal / infrared sensors, vibration sensors, acoustic sensors, and chemical analysis of motor oil [239]. Much recent work has focused on stator current analysis, particularly current harmonics; this is a non-invasive, in-operation modality, and characteristic frequency responses for various faults have been derived from machine physics [240]. Machine learning (ML) approaches for fault detection and diagnosis of complex machines have been explored in [224, 241]; deep learning in particular has been applied to fault diagnosis in e.g. [242-245]. However, small electrical motors (rated at 10 HP or less) have not normally been

monitored as closely as they are easily replaced and not directly process-critical. However, a large industrial site may contain hundreds of these motors, and there is an increasing recognition that their failure nonetheless impacts the plant's operation as they often operate important subcomponents of, or protection systems for, high-value systems. These can include operating cooling / lubrication / HVAC equipment that protects those larger systems. Clearly, any CM solution for these motors must be inexpensive, as the individual motors have a low replacement cost.

4.2. Data Collection and Dataset Generation

To the best of our knowledge, there are no publicly-available CM datasets for groups of electric motors. There are furthermore no available datasets for CM of small electric motors that 1) include multiple failure modes across multiple instances of a common motor type; 2) simultaneously record current, voltage and temperature; and 3) do so at sampling rates up to 10 kHz. The first would allow us to compare CM solutions for both fault detection and fault identification in a constant motor design; the second and third point offer the opportunity to study both high-rate features such as current harmonics as well as low-rate ones such as power profiles. Thus, our first task was to design an experimental apparatus and procedures to collect such a dataset.

An Edmonton-area motor manufacturer donated a set of 20 identical three-phase motors, each rated at 1 HP. A mount was manufactured to hold a motor with the rotor horizontal, and couple it to a dynamometer. Electrical and thermal sensors, along with data loggers, were then attached. Our basic experimental design is to run the motors under no-load and full-load conditions in an undamaged state. Then, after all the undamaged data has been collected, each motor will be damaged in a particular manner, and the tests repeated (clearly, they would terminate if the motor failed catastrophically; this, however, did not occur).

4.2.1. Hardware and Instruments

Our experimental apparatus is depicted in Figure 19.

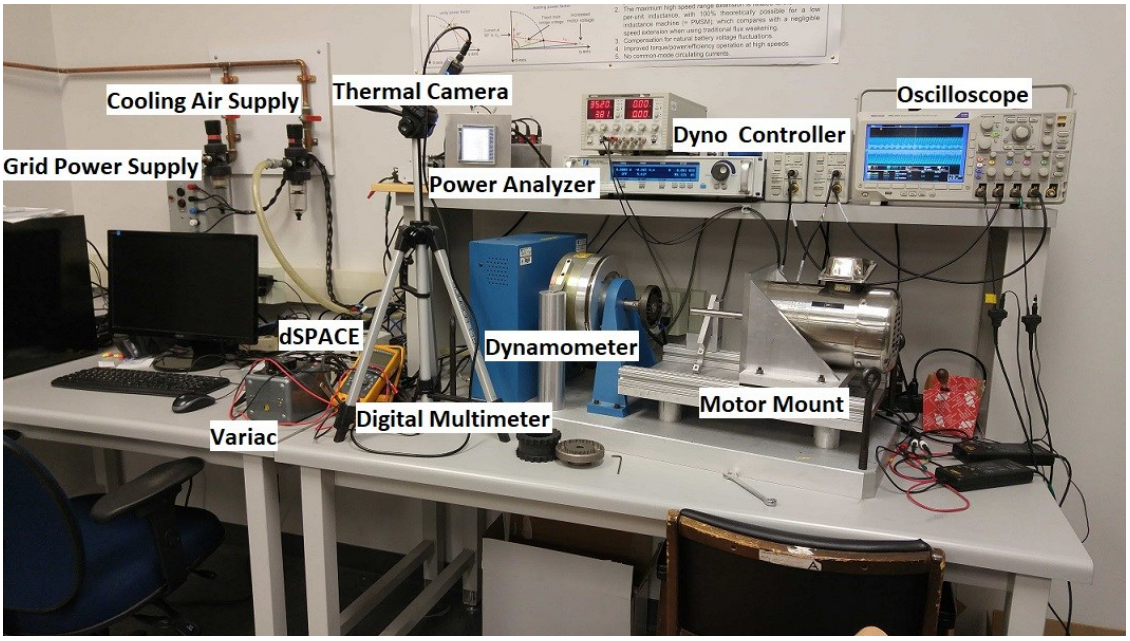


Figure 19 - Motor Testing Setup

The motors used in our experiments were stainless steel 3-phase 1 HP motors with 6205 sized drive and fan bearings. They are rated at both 208V and 480V line-to-line, but for this application, we ran them at 208V. This gave a maximum current of 3.8 A at full load. The full load rated speed is 1145 RPM. The current transformers that fed into the power analyzer are a 10:1 ratio. Since our motor had a full load current of only 3.8 A, the wire for each phase was wrapped through the current transformer 4 times to give it a high enough amplitude that the transformer would output a readable signal to the power analyzer.

A digital multimeter measured the current for the line to line locked rotor tests. For all other tests, we used the dSPACE tool. In this project 6 of the 8 analog-to-digital converters (ADCs) were used (3 phase currents and 3 line to line voltages) as well as one UART (Universal Asynchronous Receiver-Transmitter) input for the dynamometer. The dynamometer was sampled at 2 Hz, and the ADC channels were sampled at 10 kHz. The dynamometer controller was set to create a

maximum (full load) torque of 5.624 N·m, and was used to measure the shaft's rotational speed in RPM when the motor was coupled to the dynamometer. For the no-load uncoupled tests, the tachometer was used to record this shaft speed.

The oscilloscope used for this project is capable of sampling at 100MHz, but was used to sample at 500 KHz for every test except the inrush test. For the inrush test the oscilloscope was set to sample at 12.5MHz. The thermal camera used for this project was connected via an Ethernet switch to a desktop where the video feed was saved. Video of the entire experimental sequence was saved for each test, including the 2.5 hour warming period for each "hot" test (i.e. sensed data is collected once the motor has reached steady state heat). Finally, a custom mount was built out of aluminum to hold the dynamometer, rails for the mount to slide on, two mounts to bolt the motors onto and an adjustable rotor lock so that locked rotor test could be run.

4.2.2. Controlled Variables

The controlled variables in our experiments were motor temperature, applied load, motor alignment, rotor locking, dynamometer coupling, and motor damage. Motor temperature was either cold (the motor had been stopped and allowed to cool to room temperature) or hot (motor had run under full load for 2.5 hours, reaching steady-state temperature; this length of time was empirically determined using the thermal imager). The applied load was either full (dynamometer set to 5.624 N·m) or no load. Motor alignment is either centered (default condition) or twisted (the drive end bearings were under uneven lateral pressure). Rotor lock is a binary variable indicating whether the rotor lock we designed is applied, in which case the rotor cannot rotate. Dynamometer coupling is also binary, indicating whether or not the motor is coupled to the dynamometer. Finally, each motor will either be undamaged, or will have been damaged in one way; thus, for each motor, there are two possible values of this variable.

4.2.3. Experimental Design

We now discuss the experiments run on the apparatus above. Our basic design is a complete factorial experiment for each motor (considering that the motor will only have one type of damage). We begin with trial runs of healthy motors, refining our observation procedures and troubleshooting the data collection systems. Once those were complete, we executed the no-damage ("healthy") portion of the experimental design. We then damage the motors, and execute the "damage" portion of the design. In the next subsection, we detail the types of faults we induce, our rationale for selecting them, and the specific manner in which the damage is caused.

4.2.3.1. Stator Short

Within the 38% of failures caused by stator faults, the overwhelming majority of them are caused by shorts in the stator [228]. In order to test if we were able to detect a stator short, motor 3 was opened, and a Dermal was used to strip away some of the insulation between two wires in a winding and the two wires were soldered together. The solder can be seen outline by the red box in Figure 20. The motor was then reassembled and run through the same full tests as would be done in a "hot" test. However, it was also subject to a number of short tests that were run to gather data on the change in impedance from the short caused.

Figure 20 - Stator Damage in Motor 3

4.2.3.2. Bearing Faults

It is unknown exactly why the majority of bearings fail in industry settings, but experience indicates that the majority of bearing faults in industry are caused by the following:

1. Foreign materials entering the bearing and causing increased wear and pitting on the balls, cage, and inner or outer tracks.
2. Under greasing, or not putting any grease in the bearing, which could cause the bearing to overheat. This could cause the bearings and tracks to deform or warp if they are too hot under load or are going through multiple heating and cooling phases.
3. Over greasing the bearing, causing the seal to be broken, which subsequently causes all of the grease to leak out. This would lead to the same problem as under greasing the bearing.
4. Unbalanced loads on the motor can cause one side of the bearing to wear at a much greater rate that another side, which can lead to static eccentricity.
5. Simple friction wear caused by running the motor over many working hours.

To simulate these faults multiple bearings were put through different types of damage:

1. To simulate foreign materials entering the bearing and causing increased wear and pitting multiple silica carbide beads were packed into the bearing. The bearing was then run on a lathe with the outer ring held stationary at 150 rpm for 40 minutes. As a side note, after the bearing was run through a full test with the 2.5 hour warming period it was found that the cage holding the bearing balls in place was broken. This cage was not broken before the test was run. The carbide beads were not extensively cleaned out before running the motor for all of its tests, nor was the motor re-greased after the damage was caused.
2. To simulated overheating caused by lack of grease the bearing inner track was heated to a cherry red glow with an acetylene torch. This caused all of the grease to be burned of, and

left the bearing slightly warped which caused irregular damage on the inner and outer tracks as well as the ball bearings. The bearing very clearly ran rough after the damage was caused and no grease was placed back in the bearing. Running the bearing through the 2.5 hour warming period did not seem to cause additional apparent damage.

3. The last damage type that was created was a 3 mm hole in the outer track. While this is unlikely to appear in industry, it was done to compare to our results to other research paper's results as they focused primarily on single point bearing faults such as a drilled hole in the inner or outer track. The reason they focused on this type of damage is because the damage is supposed to appear at specific frequencies as opposed to general noise increases that random damage causes. Additional bearings were damaged by hitting a bearing's inner track with a hammer, once and multiple times, and creating a score on the outer track.

4.3. Methodology

As discussed, our CM design is an anomaly detector, as in Figure 21. It has two components: a model of normal behavior that predicts the current observation from previous ones, and the actual detector that identifies when the actual and predicted observations differ [246, 247].

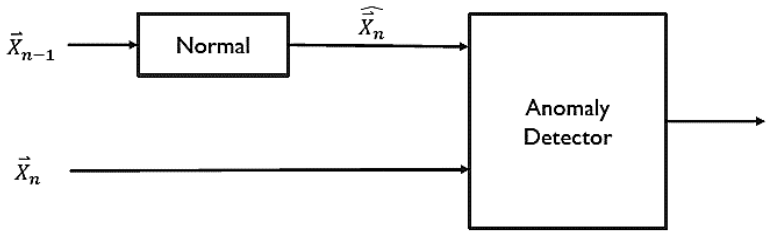


Figure 21: Anomaly Detector

Our normal model is a one-step-ahead time-series forecasting algorithm, applied to the current and voltage time series across all three phases (giving six variates total). The essential idea is to

estimate the current observation vector $\overrightarrow{x_n}$ given previous values $\overrightarrow{x_{n-1}}$. As this model is supposed to predict the next value with the assumption that our system is in the healthy state, it is trained on healthy data only.

The anomaly detector is responsible for comparing the 6 values predicted by the normal model with the actual observed values for the next time-step. Similar in structure to a classifier, the models designed for this part take 12 inputs and have only one binary output, indicating the occurrence of a failure. We may, also, use 18 inputs by adding the differences between the predicted and the actual values.

We have explored both shallow learning and deep learning approaches for designing our anomaly detector. Both were applied to a dataset constructed from multiple experimental runs in the laboratory setup above, performed on three separate motors (in order to guard against over-specialization of the models to one motor's particular behavior). Healthy runs are grouped together first followed by the runs after the motors were individually damaged. The data recorded for each motor consists of current and voltage for each of the three motor phases, measured at each sample instant. We use the method of delay embedding [247] to convert this multivariate time series into a form suitable for machine learning, and then conduct parameter exploration studies for each of the normal model / anomaly detector combinations we have tested.

4.3.1. Data Preprocessing

We begin by normalizing each variate within the data using the positive linear transform

$$X_n = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (93)$$

where x_{min} is the minimal and x_{max} the maximal observed value for the variate X, while x is the current value. We then chronologically order and (non-randomly) split the data into training and out-of-sample test partitions, assigning 70% of the dataset to the training partition, and the

remaining 30% for testing. This is the *chronologically ordered single-split* design, commonly used in time series forecasting experiments.

After splitting the data, they need to be converted from a multivariate time series into a suitable form for general (table-oriented) machine learning algorithms. One common approach is the delay embedding [15]. The current observation in a time series, and multiple prior observations, are concatenated into a vector. This vector can be shown to be isomorphic to the (unrecoverable) state vector of the system the time series was observed from at that time instant. The form of a delay vector in a univariate time series is [248]:

$$\overrightarrow{S_n} = (S_{n-(m-1)d}, S_{n-(m-2)d}, \dots, S_n) \qquad (94)$$

where S_i is the *i*-th element of the time series.

The embedding is repeated for every S_i in the time series (save those at the beginning for which insufficient former examples are available). For a multivariate time series, the embedding process is repeated for each variate, and the delay vectors for each are concatenated together. Per Formula (94), creating a delay vector comes down to finding two important parameters: the delay (d) and the dimensionality (m). Finding d allows us to get an optimal level of autocorrelation within each delay vector (by taking each successive value, every second, every third, or so on) [215]. d is usually determined heuristically by taking the first minimum of the time-delayed mutual information [247]. We then consider the shape of the selected delay using the phase diagram; for condition monitoring of electric motors, the general shape of Figure 6 is desired. m is the dimensionality of the delay vector, and is mathematically critical; Takens' result above only holds if m is large enough (at least twice the actual dimensionality of the original state space). The value of m can be determined heuristically by using the false-nearest neighborhood algorithm [249, 250]. In this algorithm, we systematically survey the data points and their neighbors in dimension d = 1,

then d = 2, and so on. As the state-space attractor is unfolded, some points originally close together will abruptly separate; these are termed “false nearest neighbors.” The algorithm works by setting a threshold for how large of a jump constitutes a false neighbor, and the heuristic is to find a value *m* for which the fraction of false neighbors plateaus across many values of this threshold [251].

The TISEAN package [248] implements these concepts and other elements of nonlinear time series analysis as a collection of numerical routines; the outputs are then mean to be plotted by some graphing tool, and interpreted by an analyst. Figure 22 presents the phase plot and Figure 23 the false-nearest neighbors plot for our dataset.

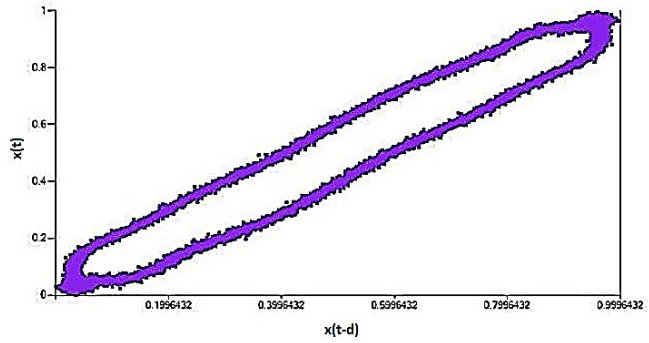


Figure 22: Phase Plot for d = 6

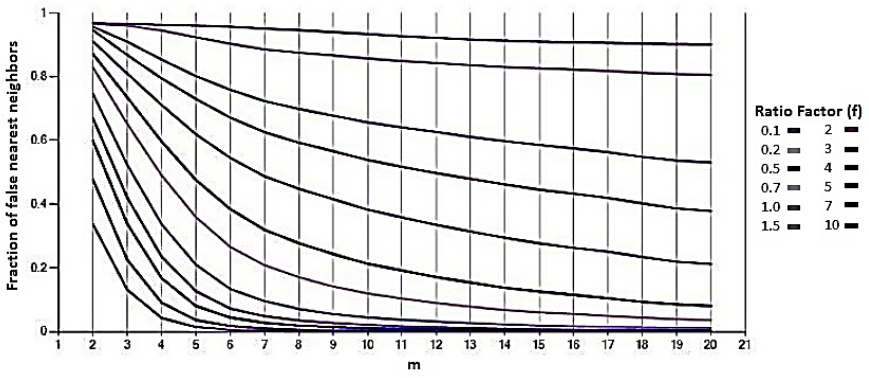


Figure 23: Proportion of False-Nearest Neighbors for different values of ratio factor (f)

4.3.2. Data Preprocessing

The data obtained from each test consist of six columns (currents and voltages; Ia, Ib, Ic, Va, Vb, Vc) and 130,000 records. Each of these files was normalized and converted to delay vectors (note that doing so has the effect of removing data skews between different motors). We then combined the tests for three of our motors (all of which were assigned to the damaged-bearings treatment) together, to test whether our CM algorithm is effective on a population of motors rather than just one. This dataset contains 109 features after delay embedding (including one class label) and about 1,800,000 records.

4.4. Parameter Exploration Procedures

There are of course many shallow and deep algorithms that can be employed in our normal model and anomaly detector. We focus our search on well-known approaches that have worked well in many other domains. We test both shallow and deep learning approaches for the normal model, choosing ones that perform well in time series forecasting. For shallow learning, we have chosen to explore Radial Basis Function Networks (RBFNs) implemented in WEKA, while we have chosen 1-dimensional CNNs in Tensorflow as a deep learner. The RBFN contains one hidden layer, whose neurons implement the Gaussian transfer function:

$$\varphi(r) = exp\left(\frac{r^2}{2\sigma^2}\right) \quad (95)$$

Where r is the Euclidean distance from a feature vector to the centre point of the Gaussian function, and σ is the standard deviation of the Gaussian.

In the WEKA implementation [252], the hidden layer neuron centers and widths are determined via k-means clustering. The output layer takes a weighted sum of the hidden layer outputs, with the weights determined via least-squares estimation, and passes the result through a

logistic function. We loop over our variable parameters with the number of neurons changing from 1 to 500 and the standard deviation selecting from the following set $\{10^i\}$, i=(-5,…,-1).

A Convolutional Neural Network (CNN) is a class of deep neural networks, usually applied to visual tasks. CNNs can capture the spatial and temporal dependencies within an input domain through the application of local convolution windows (Kernels). In the input domain these take the form of convolution masks, implemented as groups of neurons that share a weight vector (and thus implement the same transfer function). A thorough explanation is found in [253]. In a 1-dimensional convolutional layer a 1-dimensional kernel is used, as illustrated in Figure 24.

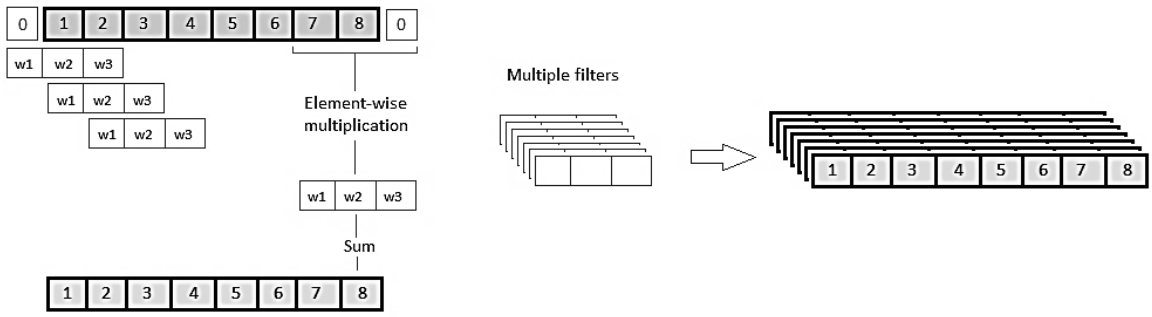


Figure 24: A 1-dimensional convolutional layer

In order to determine the hyper parameters for our models, we need a validation dataset to prevent overfitting on the final out-of-sample test set. For the RBFN, due to the limitations of the WEKA package, our parameter explorations will also be single-split designs. After the chronologically ordered split in Section 3.1, we will further partition 30% of the training data as the validation dataset, and the test set will be set aside for the final evaluation of the whole model. The 1-dimensional CNN is trained on an NVIDIA Titan Xp GPU, and so we use a tenfold cross-validation design (again. only within the training set) for parameter exploration.

For the anomaly detector, we have chosen to examine only shallow architectures, as there will only be twelve inputs for the n-th observation vector (one predicted and one actual value of each component). We have selected multilayer perceptron [112] (implemented in WEKA [254]), decision trees (Breiman's CART [255]) and random forests [256] implemented in Scikit-learn [257]. All hyper parameters of the models are explored. For the multilayer perceptron these include the number of hidden layers, number of neurons in each layer, activation functions, and initialization methods. For the decision tree and random forest models, the hyper parameters include the minimum number of samples required at a leaf node and the maximum depth of the tree [257].

4.5. Results

We first examine present our results for shallow learning in Table 1; the performance for the normal models is given first. We then present the out-of-sample results for our anomaly detector in Table 2. As CM is essentially an alarm, we present the detector's performance in terms of the True Positive Rate (TPR; also called sensitivity) and the False Positive Rate (FPR; also called the false alarm rate). TPR is the fraction of actual Positive (damaged) examples identified as such. FPR is the fraction of actual Negative (healthy) examples identified as Positive.

Table 1: Normal Model Performance for shallow learning

Number of Clusters	Standard Deviation	Train	Test
		RMSE	RMSE
100	10^{-5}	0.0323	0.0766

Table 2: The final out-of-sample results for shallow learning

MODEL	TPR	FPR	Accuracy
MLP	0.947	0.366	0.791
RBF	0.918	0.085	0.912
Decision Tree	0.967	0.207	0.881
Random Forest	0.993	0.264	0.864

For deep learning, the normal model we designed is a 1-dimensional convolutional neural network [253] with 17 dimensions for each of the 6 inputs, and a vector of 6 elements as the output, each belonging to one of the time-series. Figure 25 illustrates the training of the network using the Adam optimizer, in terms of Root Mean Square Error. Plainly, the CNN test error is roughly an order of magnitude less than the RBFN.

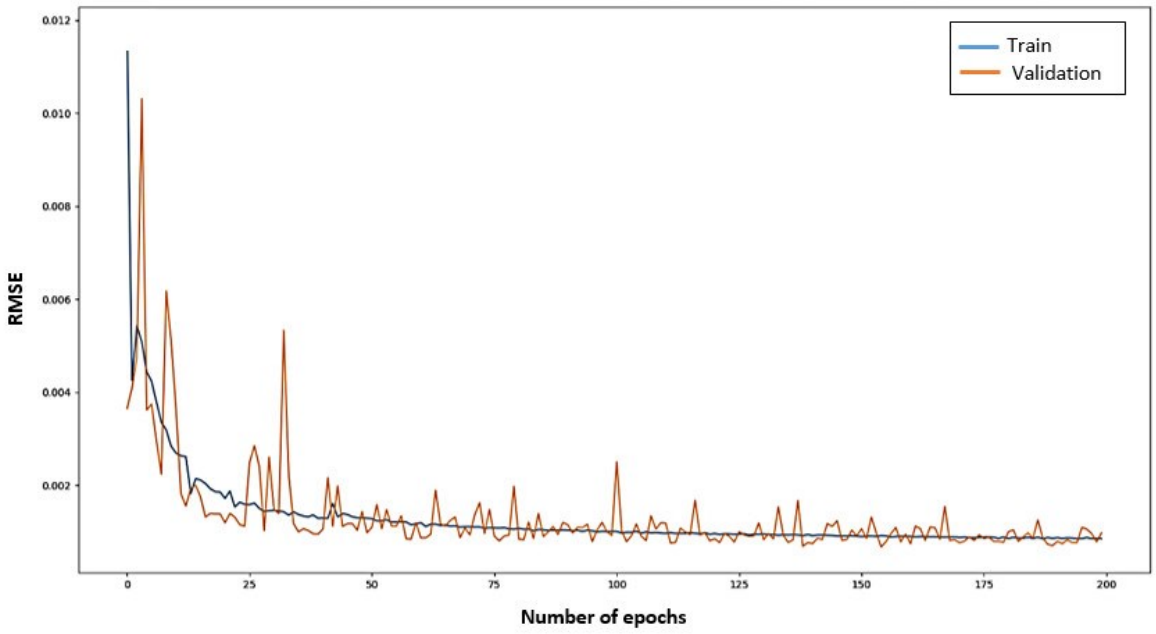


Figure 25: Normal model RMSE

Table 3 presents the out-of-sample results for our anomaly detector with deep learning, in terms of accuracy, TPR, and FPR. According to the results, random forest (with 100 estimators) with the original 12 inputs outperformed all the rest (TPR = 0.948, FPR=0.033). Examining Table 2, both decision trees and random forests produce models with slightly better TPR; however the FPR (false

alarm rate) is much higher. As false alarm rates are critical considerations in a monitoring application, we believe the deep model is superior.

Table 3: Anomaly Detector Performance for deep learning

MODEL	TPR	FPR	ACCURACY
Fully Connected Neural Nets – 12 Inputs	0.732	0.166	0.824
Fully Connected Neural Nets – 18 Inputs	0.795	0.129	0.861
Decision Tree – 12 Inputs	0.944	0.037	0.956
Decision Tree – 18 Inputs	0.748	0.162	0.811
Random Forest – 12 Inputs	0.948	0.033	0.968
Random Forest – 18 Inputs	0.841	0.101	0.895

4.6. Conclusion

In this chapter, we have discussed the failure of small induction motors and have described the various experiments we performed to collect data about them, which ultimately led to the acquisition of relevant datasets for use in the models and algorithms presented in this dissertation. Also, we have designed a condition monitoring solution for multiple small electric motors, as might be found in an industrial site. An anomaly-detection architecture is employed, with the normal model being a time-series forecasting algorithm. Both shallow and deep neural networks are investigated, with the deep network having a much lower false alarm rate (at the cost of a slightly higher false-negative rate).

In future work, we intend to apply this basic architecture (deep learning for a normal model, shallow learning for the anomaly detector) to other monitoring problems and domains. We hypothesize that the much smaller feature space for the anomaly detector means that deep learning will not be effective for this component, whereas the deep normal model will tend to be more accurate than the shallow one; and hence, a hybrid deep/shallow model is the most effective approach for anomaly detection in sensed data. Early results from pipeline leak detection experiments tend to support this contention.

5. Chapter V: Large-Scale Univariate Time Series Forecasting Using Complex Fuzzy Set

5.1. Introduction

Complex fuzzy sets (CFSs) are an extension of type-1 fuzzy sets with complex-valued membership functions. Over the last 20 years, time-series forecasting has emerged as the most important application of complex fuzzy sets, with neuro-fuzzy systems employing them shown to be accurate and compact forecasting models. In the complex fuzzy sets literature [3, 4, 214], two dominant approaches to designing forecasters can be observed: sinusoidal membership functions are used in the ANCFIS family of architectures, while complex-valued Gaussian memberships are employed in the CNFS family. To date, however, there has never been a systematic investigation that compares the performance of these two membership types (or their combination) within a common architecture.

In this chapter, we design and evaluate a new CFS-based neuro-fuzzy architecture for moderate-to-large-scale time series forecasting applications. A major focus of our work is to compare the sinusoidal and complex Gaussian CFS to determine if one or the other (or their combination) is more effective in the forecasting task. We thus hypothesize (H_1) *that adding both sinusoidal and Gaussian CFS neurons to a neuro-fuzzy architecture for time series forecasting will improve its accuracy compared to the same architecture using only sinusoidal CFS*. We have designed three variants of our architecture to explore the design space for our system, and we test each variant with exclusively sinusoidal CFS, exclusively Gaussian CFS, and a combination of both. The architectures furthermore make use of randomized learning to minimize training times. On two univariate time series (a large solar power dataset collected in [258], and a moderately-

sized realization of the Lorenz system of equations), we found that the purely Gaussian CFS were consistently superior, for all variants, compared to either the sinusoidal CFS or the combination of both. Therefore, we hypothesize (H_2) *that a CFS-based neuro-fuzzy architecture employing randomized learning will produce accurate and compact models for forecasting large-scale time series, with substantially faster training times.*

Our contributions in this chapter are firstly, the design of three new neuro-fuzzy systems that leverage CFS to produce accurate and compact time-series forecasting models; and second, the first direct empirical comparison of sinusoidal vs. complex Gaussian CFS in these three architectures.

5.2. Methodology

5.2.1. Proposed Models

Evaluating (H_1) plainly requires comparing sinusoidal and Gaussian CFS across multiple architectures, while (H_2) calls for randomized learning to be used. Our approach is thus to design three neural network models (all based on ANCFIS with a reduced number of layers and using randomized learning), and evaluate each of the three when using purely sinusoidal CFS, purely Gaussian CFS, or a combination of the two.

Model 1

One finding from [3] is that, in univariate time series, ANCFIS Layer 2 nodes make no changes to the output of Layer 1. Thus, in order to increase speed, this layer and the subsequent layer 3 were removed. We furthermore removed the linear consequent function in Layer 5, so the 6-layer ANCFIS architecture was converted to a 3-layer architecture, as shown in Figure 26. Plainly, we can easily substitute either sinusoidal or Gaussian CFS in this model. In order to examine the

combination of the two, we create a bank of sinusoidal CFS neurons, and a separate bank of Gaussian CFS neurons, both in Layer 1 (see Figure 27).

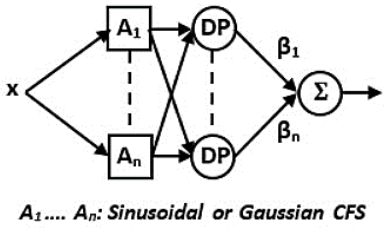


Figure 26 - Model 1

The node activation function in each layer of model 1 are as follows:

- Layer 1: The parameters $\{a_i, b_i, c_i, d_i\}$ in Formula (42) and $\{m_i, \sigma_i^2, \lambda_i\}$ in Formula (43) are drawn from a uniform distribution and set in this layer for i = 1, 2, …, *n_CMF*, where *n_CMF* is the number of complex membership functions. The values of these parameters must obey the following conditions [3, 92]:

$$0 \le d + c \le 1, \quad 1 \ge c \ge d \ge 0, \quad m, \sigma^2 \in [0,1], \quad \lambda \in [0, \pi] \qquad (96)$$

Then, the membership of an input vector in a CFS is determined by complex convolution [3]:

$$conv = \sum_{k=1}^{2n-1} h(k) = \sum_{k=1}^{2n-1} \sum_{j=max(1,k+1-n)}^{min(k,n)} f(j) g(k+1-j) \qquad (97)$$

where $f(.)$ is a data point in the n-element input vector, and $g(.)$ is the uniformly sampled membership function (also n elements in length).

- Layer 2: The output of the i-th node of this layer is inner (dot) product of the output of the i-th node of layer 1 versus the complex sum of all layer-1 outputs [3]:

$$O_{2,i} = O_{1,i} \cdot \sum_{j=1}^{|O_1|} O_{1,j} \ , \ i = 1,2, \dots, |O_1| \qquad (98)$$

where the complex-valued output of node i in layer 1 is denoted $O_{1,i}$ and $|O_1|$ is the number of nodes in layer 1.

- Layer 3: At the *j*-th node of layer 3, the outputs of the layer 2 nodes are weighted by the coefficients β_j and then summed:

$$O_{3,j} = \sum_{i=1}^{n} O_{2,i} \cdot \beta_{j,i} \qquad (99)$$

where $n = |O_2|$. This output can be reformulated as:

$$T = H^{\dagger}\beta \qquad (100)$$

where † is the Moore-Penrose generalized inverse, and

$$\begin{aligned} T &= [O_{3,1} \quad O_{3,2} \quad \cdots \quad O_{3,j}]\,, \\ H &= [O_{2,1} \quad O_{2,2} \quad \cdots \quad O_{2,n}]\,, \\ \beta^T &= [\beta_1 \quad \beta_2 \quad \cdots \quad \beta_m] \end{aligned} \qquad (101)$$

where β is estimated by the Recursive Least-Squares (RLS) algorithm [112], and m is the number of layer-3 neurons.

The randomized learning approach we use, as in ANCFIS-ELM [214], is to randomly choose the membership function parameters. For the sinusoidal CFS, this means *a*, *b*, *c*, *d* are chosen at random (subject to constraints to ensure the membership value stays within the unit disc). Likewise, the parameters $\{m, \sigma, \lambda\}$ of the Gaussian CFS are randomly chosen. Thus, the backward pass of learning is entirely eliminated. The CFS memberships pass through the rule interference step (inner product), and then the weights of the final layer are obtained by the RLS algorithm.

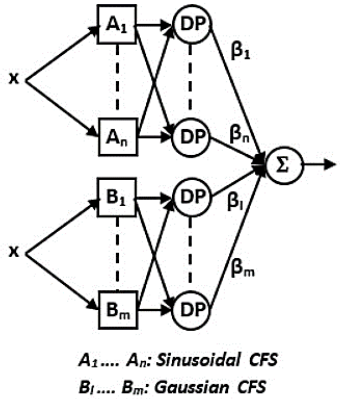


Figure 27 - Banks of sinusoidal and Gaussian neurons for Model 1

Model 2

In our second model, we test eliminating the rule interference layer, replacing it with the conjunction operation from ANCFIS Layer 2 (algebraic product). In all other respects, Model 2 is the same as Model 1. Model 2 is depicted in Figure 28.

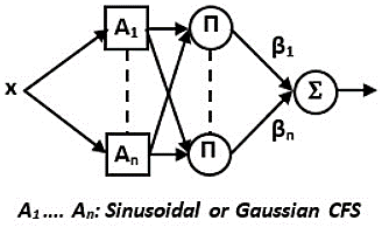


Figure 28 - Model 2

The node activation function in each layer of model 1 are as follows:

- Layer 1: This layer is identical to layer 1 of model 1.
- Layer 2: The output of each node of this layer is the algebraic product of the output of the nodes of layer 1, which are considered as the input of that node. Thus, the i-th output of this layer is calculated as follows:

$$O_{2,i} = \pi_i^{O_{1,i}}, \ i = 1,2,\dots,|O_1| \qquad (102)$$

  where the complex-valued output of node i in layer 1 is denoted $O_{1,i}$, $|O_1|$ is the number of nodes in layer 1 and π is algebraic product of the input vectors.

- Layer 3: This layer is identical to model 1.

Model 3

Finally, our third model adds the consequent layer back onto Model 2, to test how the additional degrees of freedom affect the network's performance. Since RLS was used to determine the output weights of Models 1 and 2, we do not expect to incur a substantial time penalty with this layer.

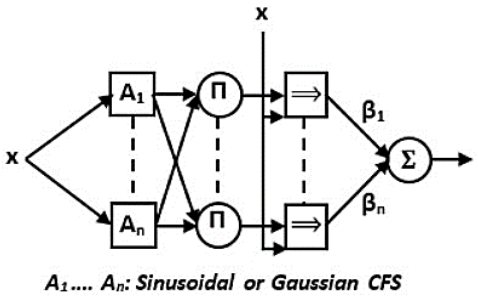


Figure 29 - Model 3

In model 3, layers 1 and 2 are identical to the corresponding layers in model 2. In layer 3, the output of the *i-th* neuron is:

$$O_{3,i} = w_i^{\pi}\left[\sum_{j=1}^{n} p_{i,j}x_j + r_i\right] \quad (103)$$

where $w_i^{\boldsymbol{\pi}}$ is the output of layer 2 and $\vec{p}$ and r are the rule consequent parameters, identified in the forwarding pass using a linear least squares estimator [259, 260], as in ANFIS [9]. Thus, the output of the last layer (layer 4) of our third proposed model can be described mathematically as follows:

$$X_{out} = O_4 = \sum_{j=1}^{n} O_{3,j}.\beta_j \quad (104)$$

and we have:

$$T = H\beta, H = \begin{bmatrix} w_1^{DP}x_{11} & w_1^{DP}x_{21} & \cdots & w_1^{DP}x_{j1} \\ \vdots & \vdots & \ddots & \vdots \\ w_1^{DP}x_{1n} & w_1^{DP}x_{2n} & \cdots & w_1^{DP}x_{jn} \end{bmatrix},\ \beta^T = \begin{bmatrix} \gamma_{11} & \gamma_{21} & \cdots & \gamma_{j1} & \eta_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma_{1n} & \gamma_{2n} & \cdots & \gamma_{jn} & \eta_n \end{bmatrix} \quad (105)$$

where $\gamma_{ij} = p_{i,j}.\beta_{\boldsymbol{i}}$, $\eta_i = r_{\boldsymbol{i}}.\beta_i$.

An important point to consider is that the final output of our models is complex-valued but our datasets are real-valued. While the CNFS architectures treat complex outputs as a real-valued 2-tuple, our philosophy in developing ANCFIS was always that a complex number should be treated as an atomic value. We thus treat the modulus of the complex output as the network's predicted output; if $z=x+iy$, then the modulus of z is $|z| := \sqrt{x^2+y^2}$, and we derive the RCNFIS network error accordingly.

5.2.2. Datasets

In order to evaluate the efficiency of the proposed algorithm, we run our prediction experiments on two univariate datasets, which are described below.

5.2.2.1. Lorenz system

The chaotic Lorenz system is a system of ordinary differential equations first studied by Edward Lorenz [261]. The Lorentz attractor is a set of chaotic solutions of the Lorentz system, which consists of the following three differential equations:

$$\frac{dx}{dt} = \sigma(y - x) \ , \ \frac{dy}{dt} = x(\rho - z) - y \ , \quad \frac{dz}{dt} = xy - \beta z \qquad (106)$$

For certain values of the positive constants $\sigma, \rho,$ and β the Lorenz system displays chaotic behavior; a common example is given by $\sigma = 10,\ \rho = 28,\ \beta = 8/3$. To perform our experiments in this work, which required univariate chaos datasets, we take the values obtained for the *x* variable using these standard parameter values in the full system. In this way, we obtained a time series with 16384 observations, which we consider to be a moderately-sized dataset. The Lorenz attractor is depicted in Figure 31 [262].

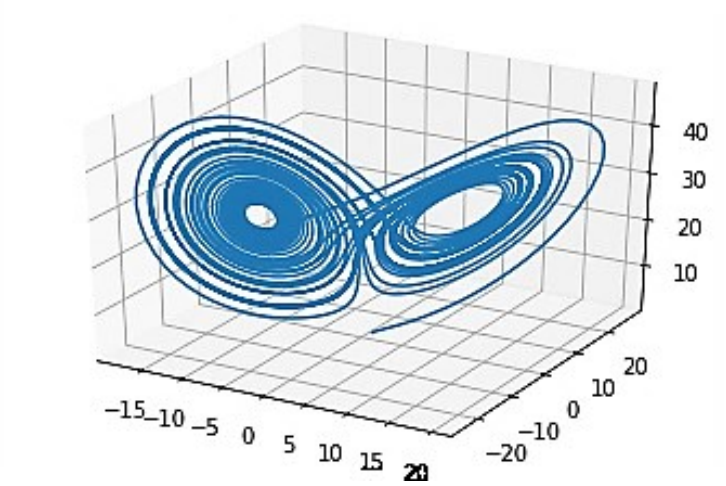


Figure 30 - Plotted solutions of the Lorenz equations in three-dimensional phase space

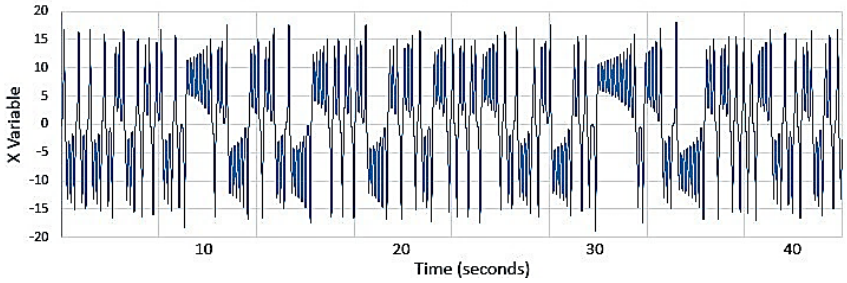


Figure 31 - Lorenz attractor, variable X only [262]

5.2.2.2. Solar Power

This dataset is another univariate time series developed from the recorded air temperature (°C) and total solar radiation (W/m^2) every minute from 2008-2014 at the U.S. Department of Energy's Lowry Range Solar Station (2,212,520 data points). To measure the air temperature, a thermometer was used placed 5 feet above the ground and in a radiant shield, and solar radiation was measured by a LICOR LI-200 Pyrometer placed 7 feet above the ground on a Rotating Shadowband Radiometer (RSR) (Figure 32) [258]. The results of these two measured variables were later converted to an estimated power output for a model solar photovoltaic cell [263], resulting in a new time series of solar power at one-minute intervals during the 7 year observation period. Figure 33 illustrates the monthly average solar radiation (W/m^2) for the study period. In [264], a subset of 20000 records (two continuous weeks of data) from the dataset was used to perform their experiments, but in this work, we use all 2.2 million observations. Our results are thus not directly comparable to the results in [216, 264].

Figure 32 - RSR device for measuring solar radiation [258]

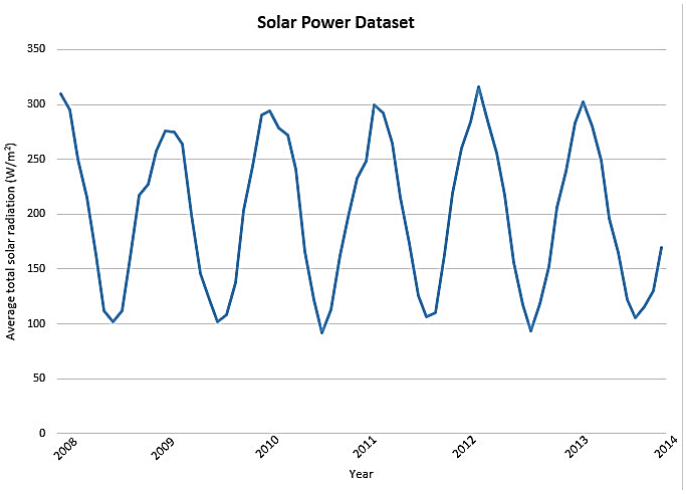


Figure 33 - Monthly average of solar radiation (W/m2) for a 7-year period

5.2.2.3. Condition Monitoring Dataset

As discussed in detail in Chapter 4, Condition Monitoring (CM) is using non-destructive testing or sensory data to detect changes in a monitored system. Whereas preceding approaches might possibly signal the incidence of a change, some cutting-edge tactics can determine and pinpoint why an error occurred, where it is, and therefore the damage done so far [229]. Examples of these methods can be found in [233-236]. Some adaptive systems may even restore themselves primarily

based totally on that information [229]. Nowadays, CM is a well-known and cost-effective technique to maintaining massive and pricey systems that permits to maintain the forecast earlier than a fault escalates or breaks down [237, 238].

A major application of CM is for electrical (induction) motors, as discussed in chapter 4, because these types of motors have vital and significant uses in various industries and are one of the main arteries of industrial activities, thus any disruption or interruption in their performance causes huge and sometimes irreparable damage. The main faults of induction motors are related to bearing, stator, and rotor failures, which make up 40%, 38% and 10% of errors, respectively, and the other 12% is related to other failures [226, 228]. Most of the CM approaches used for these motors are about the large and expensive ones, and in smaller ones, they have received less attention because of their low replacement cost and not directly process-critical. However, a massive industrial site could contain many these motors, and their breakdown affects the performance of the entire site, and it is in this case that the need to monitor them is strongly felt.

In Chapter 4, a study was performed on small induction motors and data were collected under different test conditions and multiple failure modes. To collect such a dataset, a collection of twenty identical three-phase motors (1 HP each) equipped with a dynamometer, electrical and thermal sensors, and data loggers was installed. The experiments had been designed so that the motors would first run underneath no-load and full-load in a non-damaging state and then, after all of the undamaged records were collected, each motor would be damaged in a specific way and damaged data were conjointly collected. Moreover, 6 converters out of 8 analog to digital converters (ADC) (3 phase currents and 3 line-to-line voltages) as well as a UART (Universal Asynchronous Receiver-Transmitter) input were used for the dynamometer. In [6], more details

are provided about the design of the experiments, other tools used in the experiments, and the types of faults that occurred, and the specific damage they cause.

Overall, the data collected from every experiment consisted of six columns (currents and voltages; I_a, I_b, I_c, V_a, V_b, V_c) and 130,000 records. Once normalization and obtaining the delay vectors, the experiments of the 3 motors (all dedicated to the treatment of damaged bearings) have been combined to affirm whether or not the CM algorithm proposed in [6] was effective on a populace of motors. Finally, the dataset contained 6 attributes and roughly 1.3 million records.

To carry out the experiments for univariate time series in this chapter, we consider only one variate (in this case, V_a) and apply the relevant operation, but in the next chapter (6th), we will take into account all variables that we will discuss in the related section.

5.3. Preprocessing

We first normalize our datasets to [0, 1] using the min-max Formula:

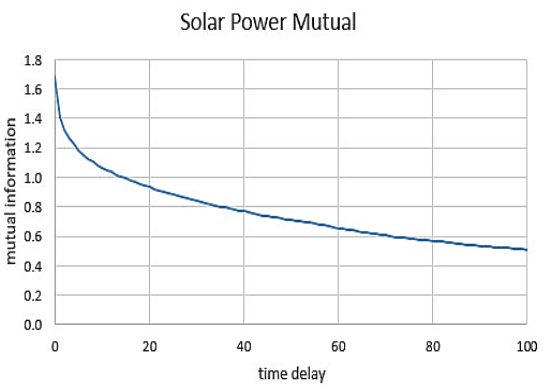
$$X_n = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{107}$$

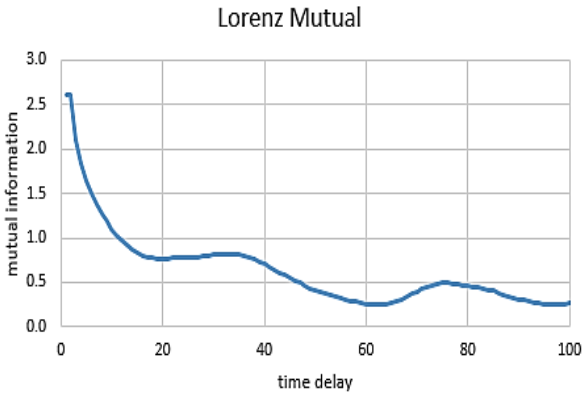
where X is the original value, X_{min} is the minimum value of X, X_{max} is the maximum value of X, and X_n is the normalized value. We then perform a chronologically ordered single split of our data, assigning the earliest 70% of the data to the training set and reserving the latest 30% as the testing set. We then need to embed the data in a tabular format for the Gaussian CFS experiments. Fortunately, previous research has determined [265] that the feature vectors from a delay embedding, *if treated as sampled windows of the time series*, are congruent to our input representation in ANCFIS. We thus use this as the common input representation for all models and all CFS.

The form of a delay vector in a univariate time series is [248]:

$$S_n = (S_{n-(m-1)d}, S_{n-(m-2)d}, \dots, S_n) \quad (108)$$

Where S_n is the *n-th* element of the time series. As it can be seen in Formula (108), two main parameters, namely the delay (d) and the dimensionality (m), must be determined for generating a delay vector. According to [247], the delay d is usually chosen as the one that gives the first local minimum of the time-delayed mutual information. This is the mutual information between the original sequence and its image after a delay [215]. m can be estimated by the false-nearest neighborhood algorithm [250]. Using the functions in the TISEAN package [266], we determined the best values for d and m in each dataset (note that inspection of the phase plot is required to confirm the initial value of d from the mutual information statistic, while manual inspection of the false-nearest neighbors plot is also required). Figure 34 presents the time-delayed mutual information for each dataset, with the phase plots in Figure 35. Figure 36 present the false-nearest neighbor plot. Detailed discussions of how to interpret these diagrams are provided in [247].





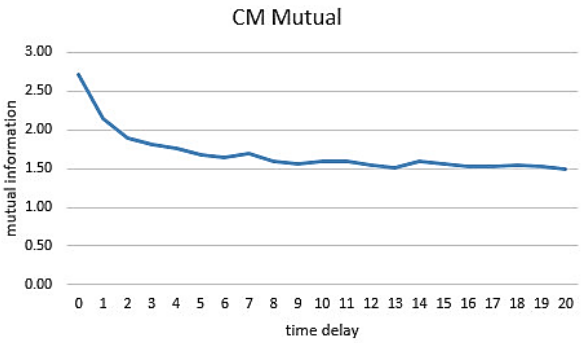


Figure 34 - Time-delayed mutual information statistics for datasets

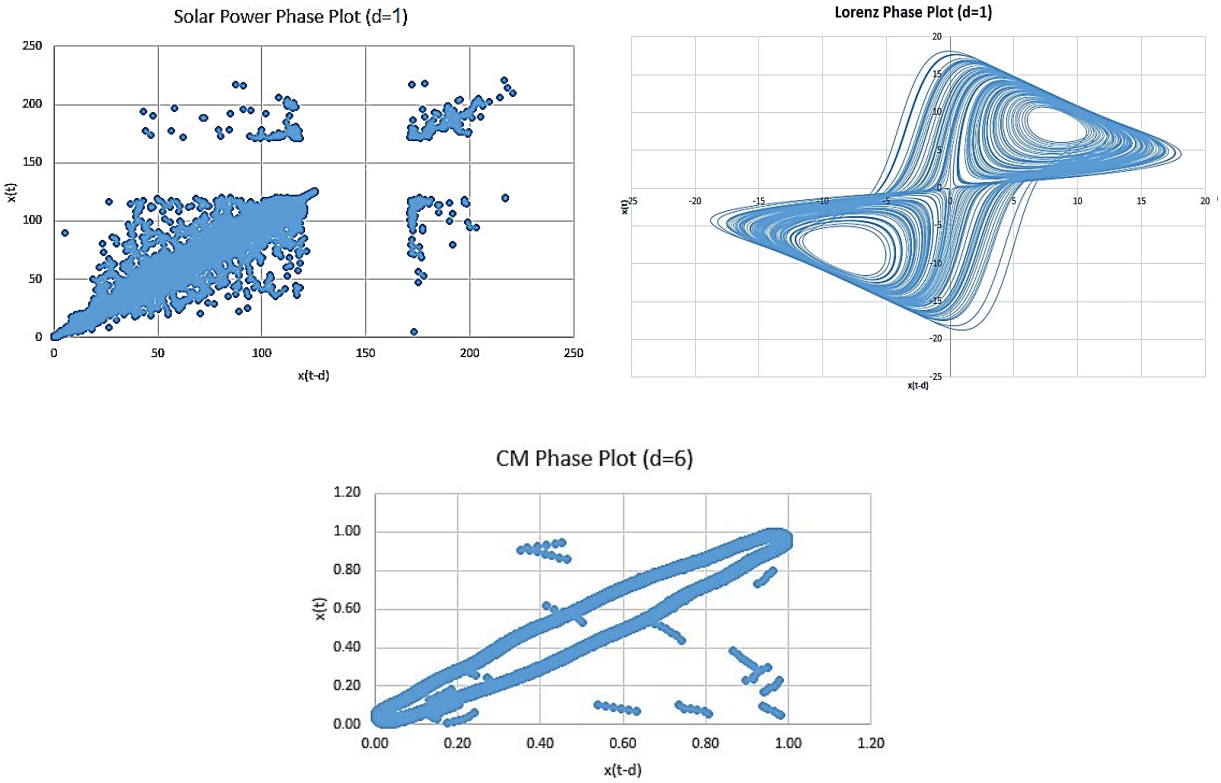


Figure 35 - Phase Plot for datasets and their obtained delay

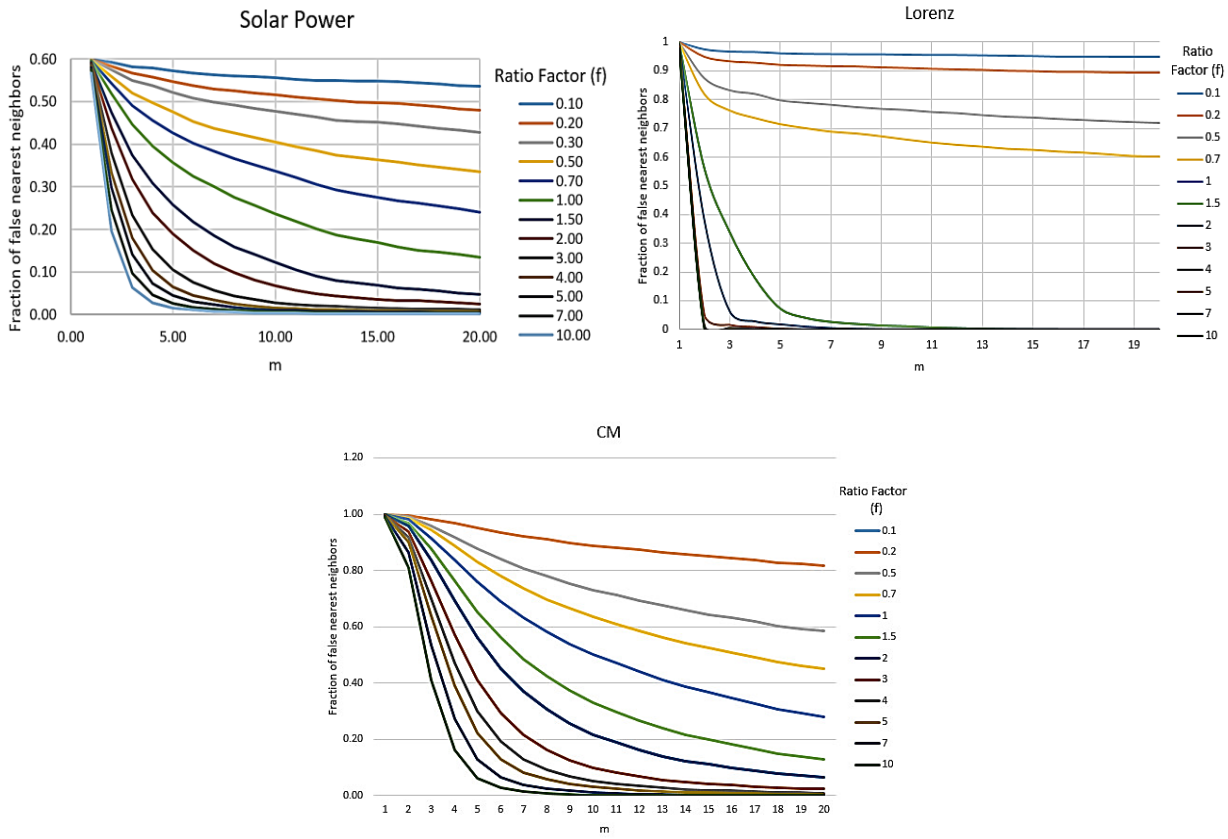


Figure 36 - Proportion of False-Nearest Neighbors for different values of ratio factor (f)

5.4. Results

5.4.1. Experimental design

In Table 4, we present the delay vector parameters obtained for each of the datasets.

Table 4 - Obtained delay vectors parameters

Dataset	Delay Vector Parameters	
	Delay (d)	Dimension (m)
Lorenz	1	3
Solar Power	1	3
CM	1	3

In order to evaluate our proposed model and compare it with previous different approaches, we use the Root Mean Square Error (RMSE), computed over the out-of-sample test points, and then compare the result with other models:

$$RMSE = \sqrt{\frac{1}{K}\sum_{i=1}^{K} MSE_i} \quad , \quad MSE_i = \frac{\sum_{j=1}^{n}(y_j - \hat{y}_1)^2}{n} \tag{109}$$

Where K is the number of variates in the time series, MSE_i is mean squared one step-ahead of i-th variate, y_j is predicted value, and $\hat{y}_1$ is desired value.

For each dataset, we varied the number of neurons in Layer 2 from 4 to 500, and tested values of the forgetting factor λ in the RLS algorithm. We furthermore compared sinusoidal, complex Gaussian, and mixed CFS forms. For each parameter combination we ran 1000 repetitions of the training & testing experiments, to determine the mean and standard of the network RMSE. All models and experiments have been implemented in the Visual Studio C++ environment.

5.4.2. Experimental Results

Table 5 - Solar power dataset

d1 m3	Unit	λ	RMSE	SD
Model 1 (Sin)	4	0.5	8.49E-02	5.33E-02
Model 1 (Gauss.)	20	0.5	1.25E-02	2.28E-03
Model 1 (Mixed)	50	0.5	1.36E-02	1.69E-03
Model 2 (Sin)	4	0.1	3.65E-01	5.33E-01
Model 2 (Gauss.)	300	0.5	1.02E-03	3.91E-03
Model 2 (Mixed)	4	0.5	2.16E-02	5.62E-01
Model 3 (Sin)	4	0.5	3.06E-01	5.52E-01
Model 3 (Gauss.)	20	0.5	1.27E-02	5.05E-03
Model 3 (Mixed)	4	0.5	2.88E-02	1.14E+00

Tables 5 to 7 show a comparison between the results of exploring the parameters for the different modes of our proposed models for each datasets. The best parameterization of each model for each form of CFS (as determined by the mean RMSE over 1000 repetitions) is recorded in these tables.

Table 6 - Lorenz dataset

d1 m3	Unit	λ	RMSE	SD
Model 1 (Sin)	50	0.01	2.09E-02	1.05E-03
Model 1 (Gauss.)	500	0.5	3.42E-05	3.60E-06
Model 1 (Mixed)	50	0.5	5.98E-03	5.17E-03
Model 2 (Sin)	20	0.1	2.12E-02	9.39E-03
Model 2 (Gauss.)	20	0.5	1.11E-03	5.84E-03
Model 2 (Mixed)	50	0.5	2.57E-03	3.14E-02
Model 3 (Sin)	50	0.01	2.09E-02	1.06E-02
Model 3 (Gauss.)	50	0.5	6.71E-04	5.75E-03
Model 3 (Mixed)	50	0.5	2.01E-03	7.76E-02

Table 7 - CM dataset

d1 m3	Unit	λ	RMSE	SD
Model 1 (Sin)	4	0.01	1.49E-01	9.19E-02
Model 1 (Gauss.)	50	0.5	2.57E-03	4.75E-04
Model 1 (Mixed)	20	0.5	8.80E-02	1.03E-01
Model 2 (Sin)	4	0.1	6.10E-01	9.88E-02
Model 2 (Gauss.)	50	0.5	1.42E-03	8.78E-04
Model 2 (Mixed)	50	0.5	4.53E-03	3.48E-01
Model 3 (Sin)	4	0.01	6.67E-01	2.27E+00
Model 3 (Gauss.)	50	0.5	1.73E-03	1.16E-03
Model 3 (Mixed)	50	4	3.39E-01	1.27E+00

The very clear result of Tables 5 to 7 is that, for all three models on both datasets, the exclusive use of a Gaussian CFS is superior to either exclusively sinusoidal CFS, or to the combination of both types.

5.4.3. Comparison against Existing Algorithms

Table 8 compares the average out-of-sample error for both datasets (across 1000 repetitions) against the existing literature, and new experiments on well-known shallow learning algorithms (note that the results for the Solar Power dataset on ANCFIS and FANCFIS refer to the 20,000 observation subset; it is infeasible to train those algorithms for the full 2.2 million observations). The best RCNFIS model for Solar power is Model 2 with 300 hidden neurons, and for the Lorenz dataset it was Model 1 with 500 hidden neurons. We selected and parameterized the Radial Basis Function Network (RBFN, implemented in WEKA [254], Multi-Layer Perceptron (MLP, implemented in WEKA and Scikit-learn [257]) and the Sequential Minimum Optimization for Regression variant of support vector machines (SMOreg, implemented in WEKA) on both datasets. Separate parameter explorations were performed for each dataset in all three algorithms following the same experimental design as RCNFIS, and the out-of-sample result for the best parameterization is presented in Table 8. For both datasets, the average RMSE of RCNFIS was superior to all of the experimental contrasts.

Table 8 - Compare results in different architectures in terms of RMSE

Architecture	**Dataset/RMSE**		
	Solar Power	Lorenz	CM
RCNFIS	Model 2 (Gauss.) **1.02E-03**	Model 1 (Gauss.) **3.42E-05**	Model 2 (Gauss.) **1.42E-03**
ANCFIS [265]	3.11	-	-
FANCFIS [216]	4.90	-	-
Adaptive Neuro FIS [89]	-	1.43E-01	-
Pseudo Gaussian RBF [267]	-	9.40E-02	-
ARMA [268]	-	8.76E-02	-
Elman-NARX [269]	-	7.14E-05	-
Scaled Hybrid Model [269]	-	3.46E-05	-
RBF	6.93E-02	9.22E-02	4.97E-02
MLP	2.20E-03	1.20E-03	2.40E-03
SMOreg	3.41E-02	2.40E-03	4.90E-02

We apply a *Z*-test (specifically the one-sample location test) to determine whether the differences observed in Table 8 are significant [270, 271]. The null hypothesis H_0 is that differences are insignificant, while the alternative is that our model is significantly more accurate (this is a one-sided test). We compute the *Z*-statistic using the 1000 repetitions of our models as the sample population, and then determine the *p*-value associated with the observed accuracy of the best alternative. The *Z*-statistic is given by:

$$Z = \frac{\bar{X} - \mu_0}{s/\sqrt{n}} \qquad (110)$$

where $\bar{X}$ is the sample mean, μ_0 is constant being compared, s is the sample standard deviation, and n is the sample size. While we are using the sample variance instead of the population variance, our sample size of n = 1000 observations should mean that this is a very close approximation of

the correct *Z*-score. Plainly, $\bar{X}$ and *s* are determined from our 1000 experiments, while μ_0 is the RMSE of the best alternative from Table 8.

For the Solar Power dataset we have:

H_0: $\mu_0 \geq 0.0022$

H_a: $\mu_0 < 0.0022$ (This is a left-sided (one tailed) test)

The MLP has the lowest RMSE of all competing methods for the Solar dataset in Table 8 at 0.0022. We set our significance level α = 0.01. We have *n* = 1000, $\bar{X} = 0.00102$, *s* = 0.003905. From Formula (110) we have:

$$Z_c = \frac{0.00102 - 0.0022}{\frac{0.003905}{\sqrt{1000}}} = -9.55$$

Using the Z-table, we find the corresponding *p*-value to be ≅ 0.00003; this is much smaller than α, and hence we reject H_0. Thus, we find that RCNFIS is superior to the other models in Table 8 for the Solar dataset.

For the Lorenz dataset, the next-lowest RMSE is obtained by the Scaled Hybrid Model [269], at 3.42E-05. For the *Z*-test we thus have:

H_0: $\mu_0 \geq 0.0000346$

H_a: $\mu_0 < 0.0000346$ (This is a left-sided (one tailed) test)

n = 1000, $\bar{X} = 0.0000342$, *s* = 0.0000036, α = 0.01, and so from Formula (110) we have:

$$Z_c = \frac{0.0000342 - 0.0000346}{\frac{0.0000036}{\sqrt{1000}}} = -3.51$$

Using the Z-table, we find the corresponding *p*-value to be 0.00022. This is much less than 0.01, and so H_0 is rejected. Thus, we find that RCNFIS is superior to the other models in Table 8 for the Lorenz dataset.

Finally, for the CM dataset, we have:

H_0: $\mu_0 \geq 0.0024$

H_a: $\mu_0 < 0.0024$ (This is a left-sided (one tailed) test)

n = 1000, $\bar{X} = 0.00142$, s = 0.00088, α = 0.01, and so from Formula (110) we have:

$$Z_c = \frac{0.00142 - 0.0024}{\frac{0.00088}{\sqrt{1000}}} = -35.216$$

Using the Z-table, we find the corresponding p-value which is much less than 0.01, and so H_0 is rejected. Thus, we find that RCNFIS is superior to the other models in Table 8 for the Lorenz dataset.

Furthermore, in order to determine which CFS improves accuracy statistically significantly (H_1), we applied the Friedman test. This test is calculated as follows [93]:

$$S = \frac{12n}{k(k+1)} \sum_{j=1}^{k} (\bar{R}_j - \frac{k+1}{2})^2 \qquad (111)$$

where k for our case is the number of different CFS types applied on n measures (three different models applied on three datasets in a full-factorial design), and $\bar{R}_j$ is the average rank of the j-th CFS type applied for the n different measures. The null hypothesis H_0 is that the average rank of the methods are the same; rejecting H_0 implies that at least one of the average ranks are different. We implement the test in Python, using the friedmanchisquare() function from SciPy. We have k = 3, n = 9, and we choose a significance of α = 0.05. The calculated p-value from the data presented in Tables 2-4 is equal to 0.0001 < α, and so H_0 is rejected.

Having rejected H_0, we use the MCB (Multiple Comparisons with the Best) method to determine which methods are superior to the others [272]. In this method the null hypothesis of no difference is rejected if $|\bar{R}_u - \bar{R}_v| \geq r_{\alpha,k,n}$ where $\bar{R}_u$ is the average rank of the u-th method

and $r_{\alpha,k,n} \approx q_{\alpha}\sqrt{\frac{k(k+1)}{12n}}$, where q_{α} is the α percentile point for the range of k independent standard normal variables. We obtain this from the one-tailed Dunnett's table at the given values of α, *k* and *df* (degree of freedom). For α = 0.05, k=3, and df=2 we obtained $q_{\alpha} = 3.354$ giving $r_{\alpha,k,n} = 1.806$.

We present the differences in average ranks for our three CFS types in Table 9. Gaussian fuzzy sets were clearly the superior alternative to sinusoidal ones; however, the differences between either sinusoids or Gaussians and the mixed-CFS design were not significant. What we can say, however, is that using mixed CFS was not superior to just using Gaussian CFS.

Table 9 - The average rank of the measures

	$\overline{\text{Sin}}$	$\overline{\text{Gauss}}$	$\overline{\text{Mixed}}$
$\overline{\text{Sin}}$	0	2	1
$\overline{\text{Gauss}}$		0	1
$\overline{\text{Mixed}}$			0

We also compare the learning time of RCNFIS against RBF, MLP, and SMOreg in Table 10, which presents the time needed to train each model (determined by calling the clock() function or equivalent at the beginning and end of training). Plainly, our proposed models are faster than these alternatives.

Table 10 - Training time in seconds

Dataset	Time spent (seconds)			
	RCNFIS	RBF	MLP	SMOreg
Solar Power	Model 2 (Gauss.) **232.5**	249.55	399.12	860.14
Lorenz	Model 1 (Gauss.) **35.1**	37.25	74.51	95.19
CM	Model 2 (Gauss.) **137.6**	147.62	235.56	498.38

5.4.4. Evaluating the Research Hypotheses

We now consider our final evaluation of (H_1) and (H_2). The results of Tables 5 to 7 quite clearly do not support (H_1); indeed, they indicate that Gaussian CFS, on their own, produce superior prediction models. By contrast, (H_2) is clearly supported by the results of Tables 8 and 10. RCNFIS is significantly more accurate than the existing algorithms tested on this dataset, while also being faster to train. A very accurate RCNFIS forecaster took just under four minutes to train on a time series of two million observations.

5.5. Conclusion

In this chapter we have sought to design large-scale learning algorithms for time series. Based on prior results, we designed our algorithms to employ complex fuzzy sets, as they have been shown to result in accurate and compact models. We furthermore undertook the first direct comparison of the sinusoidal and complex Gaussian CFS membership's functions; across three variants of our large-scale learning architecture. We found that using Gaussian CFS led to superior accuracy compared to either the sinusoidal CFS or a mixture of the two across two moderate-to-large-scale datasets. We also found that the CFS architectures were more accurate and faster to train than the existing literature and three additional shallow learning algorithms.

In future work we will replicate this study for large-scale multivariate datasets, which are common problems in the Big Data world. If our findings concerning CFS in this manuscript are confirmed, then we will also seek to understand under what circumstances the sinusoidal CFS remain useful models of uncertainty; we do know from [3] that they are able to model finite mixtures of sinusoidal functions with zero error.

6. Chapter VI: Large-Scale Multivariate Time Series Forecasting Using Complex Fuzzy Set

6.1. Introduction

In Chapter 3, we reviewed the various models and algorithms that were based on CFS&L to deal with TSF. Despite being designed to handle multivariate problems, all of the mentioned models were evaluated on univariate time series problems. In [10] and [273], dealing with multivariate time series problems using CFS&L were considered for the first time. In [273], three different alternative designs for ANCFIS [3] were studied to create a system to handle multivariate time series prediction, namely the use of SISO, MISO and MIMO methods in the ANCFIS architecture; given that we have also used these three methods in designing the models of this study, in the methodology section we will explain and how to use them.

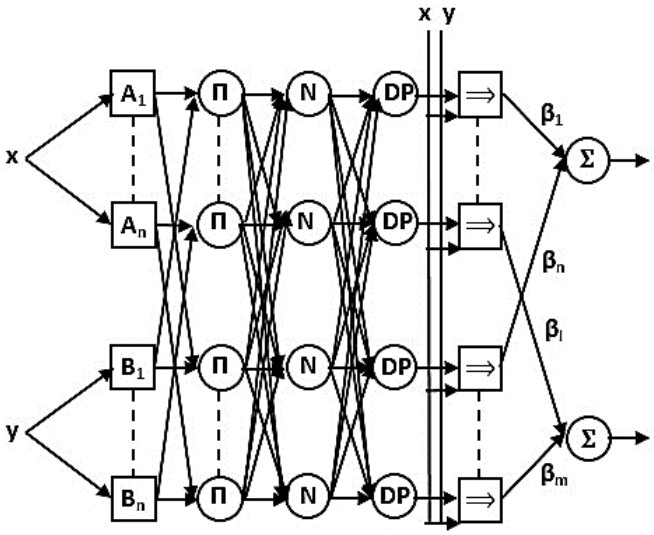


Figure 37 - MIMO ANCFIS architecture for a bivariate time series problem [10]

In [10], in addition to using these three methods, other approaches are considered to apply further changes in ANCFIS model (including some changes to the design of the layers and

operators used, as well as a few modifications to the VNCSA algorithm [3]) specifically to the MIMO ANCFIS model, which has led to better results. Figure 37 shows a diagram of the MIMO ANCFIS architecture for a bivariate time series problem. However, the methods and models presented in [10] and [273] only applied to small datasets and were problematic for large-scale datasets.

In this chapter, we extend the RCNFIS models for multivariate TSF on chaotic and large-scale datasets. As described in chapter V, three proposed models for RCNFIS are presented. In this chapter, we extend the same three models for multivariate time series and test each variant with exclusively sinusoidal CFS, exclusively Gaussian CFS, and a combination of both, in order to determine which modes (or their combination) is more effective at forecasting. In addition, randomized learning [214] is applied to all three models to minimize training times. Moreover, as in [10] and [273], we also explored SISO (Single Input Single Output), MISO (Multiple Input Single Output) and MIMO (Multiple Input Multiple Output) designs in implementing our models. In the methodology section, more detailed descriptions of these methods and how to use them in the implementation of our models will be provided. The proposed approaches are applied to one chaotic system (a realization of the Rössler system of equations with three variates [274, 275]) and two large-scale time series datasets (a condition monitoring dataset with six variates [6] and a gas sensor data set with four variates [276, 277], and their results are compared against a few well-known machine-learning algorithms (Multilayer Perceptron (MLP), Radial Basis Function Network (RBFN), Support Vector Regression (SVR)) as well as the results obtained from other recent related publications. We used two statistical evaluation, the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE), to compare our results.

6.2. Methodology

6.2.1. Proposed Models

As mentioned earlier, in this chapter we intend to extend the three proposed RCNFIS models of Chapter 5 that were proposed to predict univariate time series using CFS.

6.2.1.1. SISO RCNFIS

This RCNFIS design considers each variate as an independent univariate time series. There are therefore *N* complete and separate RCNFIS systems for *N*-variate time series. For example, for a time series with two variates of x and y in the form of $(x_1, x_{2,} \dots, \boldsymbol{x_t})$ and $(y_1, y_{2,} \dots, \boldsymbol{y_t})$, the first system is fed by the variate x and the second system by the variate y, while x_t and y_t are the respective one-step-ahead values that must be predicted. Finally, the overall performance of this method in forecasting multivariate time series is computed from the combined accuracy obtained from all these systems. The three proposed SISO RCNFIS models are evaluated in the same way as in Chapter 5, i.e. the three modes of activation functions; the purely sinusoidal CFS, purely Gaussian CFS, or a combination of the two. Figure 38 shows the three SISO Simple/Hybrid RCNFIS models in these three modes for a sample variate x. Note that in this method all variates in a time series will be forecast using the same model choice. The neuron transfer function in each layer of the basic models, are the same as their counterparts in Chapter 5.

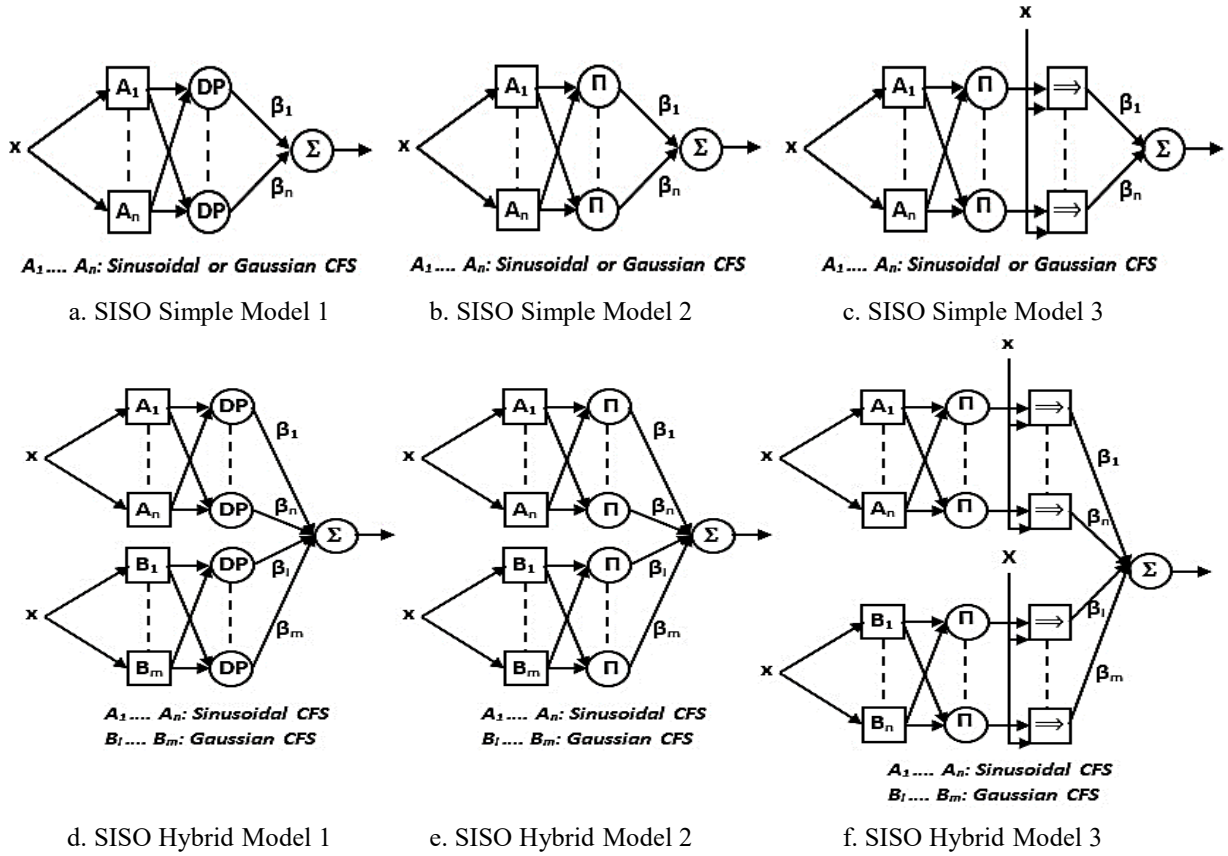
6.2.1.2. MISO RCNFIS

In this RCNFIS design, as in the previous design, a separate system is considered for each variate, but the input of each consists of the concatenated delay vectors from all variates. For example, for a time series with two variates of x and y in the form of $(x_1, x_{2,} \dots, x_t)$ and $(y_1, y_{2,} \dots, y_t)$, the first system receives $(x_1, x_{2,} \dots, \boldsymbol{x_t}, y_1, y_{2,} \dots, y_t)$ and the one-step-ahead target that must be predicted is x_t.The second system receives $(x_1, x_{2,} \dots, x_t, y_1, y_{2,} \dots, \boldsymbol{y_t})$ and y_t is the one-step-ahead target

that must be predicted. The overall performance of this method is likewise measured by the combined accuracy of all systems. The design of MISO Simple/Hybrid RCNFIS models of this method is shown in Figure 39.

6.2.1.3. MIMO RCNFIS

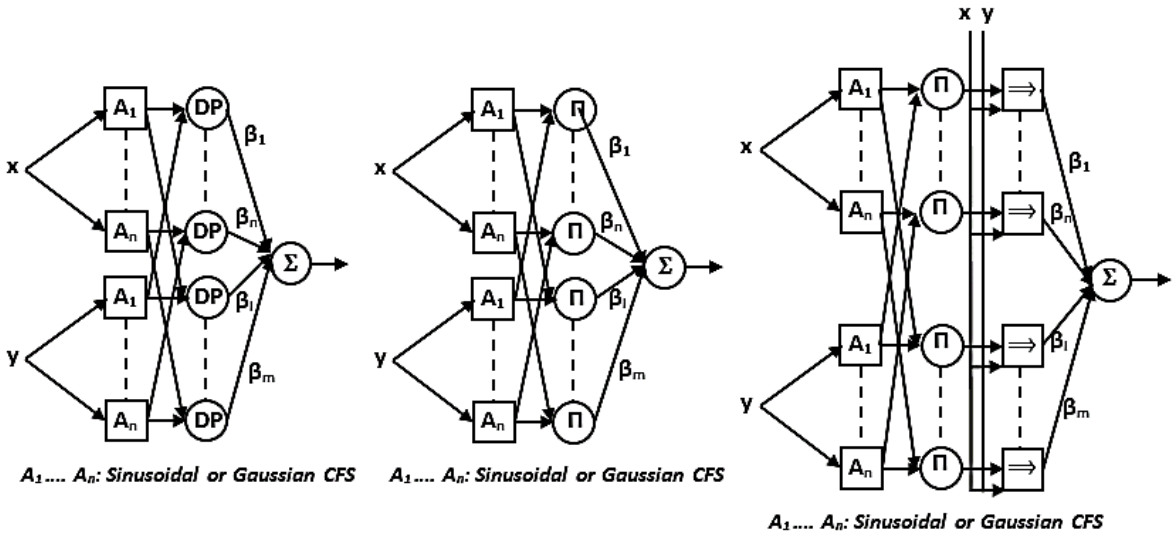
This RCNFIS design uses a single network that receives the concatenated delays vectors from an *N*-variate time series at the same time; thus there are *N* input and *N* output nodes. For example, for a time series with two variates of x and y in the form of $(x_1, x_{2,} \dots, x_t)$ and $(y_1, y_{2,} \dots, y_t)$, the first is fed by $(x_1, x_{2,} \dots, \boldsymbol{x_t}, y_1, y_{2,} \dots, \boldsymbol{y_t})$ and x_t and y_t are the one-step-ahead targets that must be predicted at the same time. Figure 40 shows the three MIMO Simple/Hybrid RCNFIS models in the three modes for a sample time series with two variates x and y.



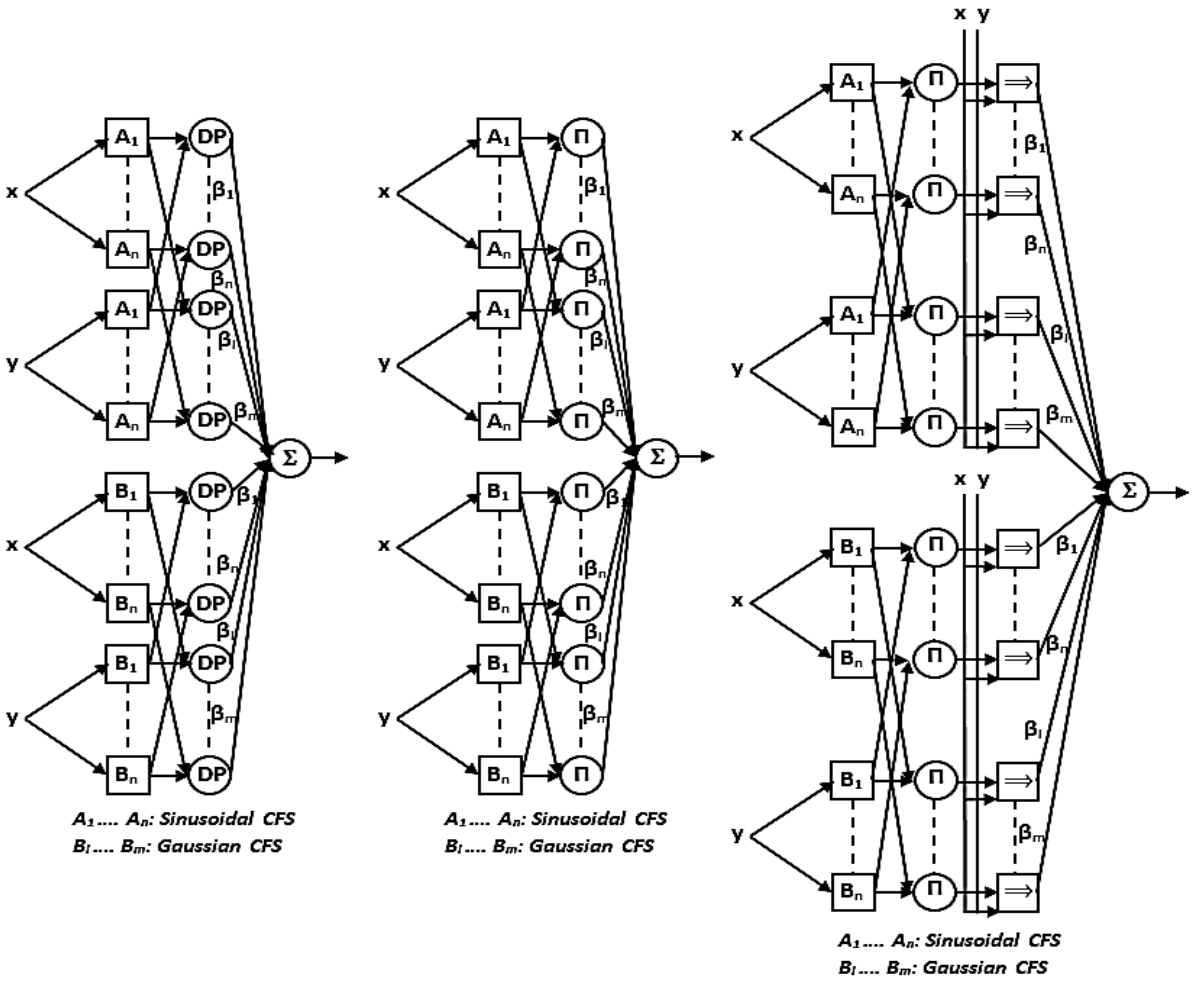
a. SISO Simple Model 1 b. SISO Simple Model 2 c. SISO Simple Model 3

d. SISO Hybrid Model 1 e. SISO Hybrid Model 2 f. SISO Hybrid Model 3

Figure 38 - Three SISO Simple/Hybrid RCNFIS models in three modes for a sample variate x [278]

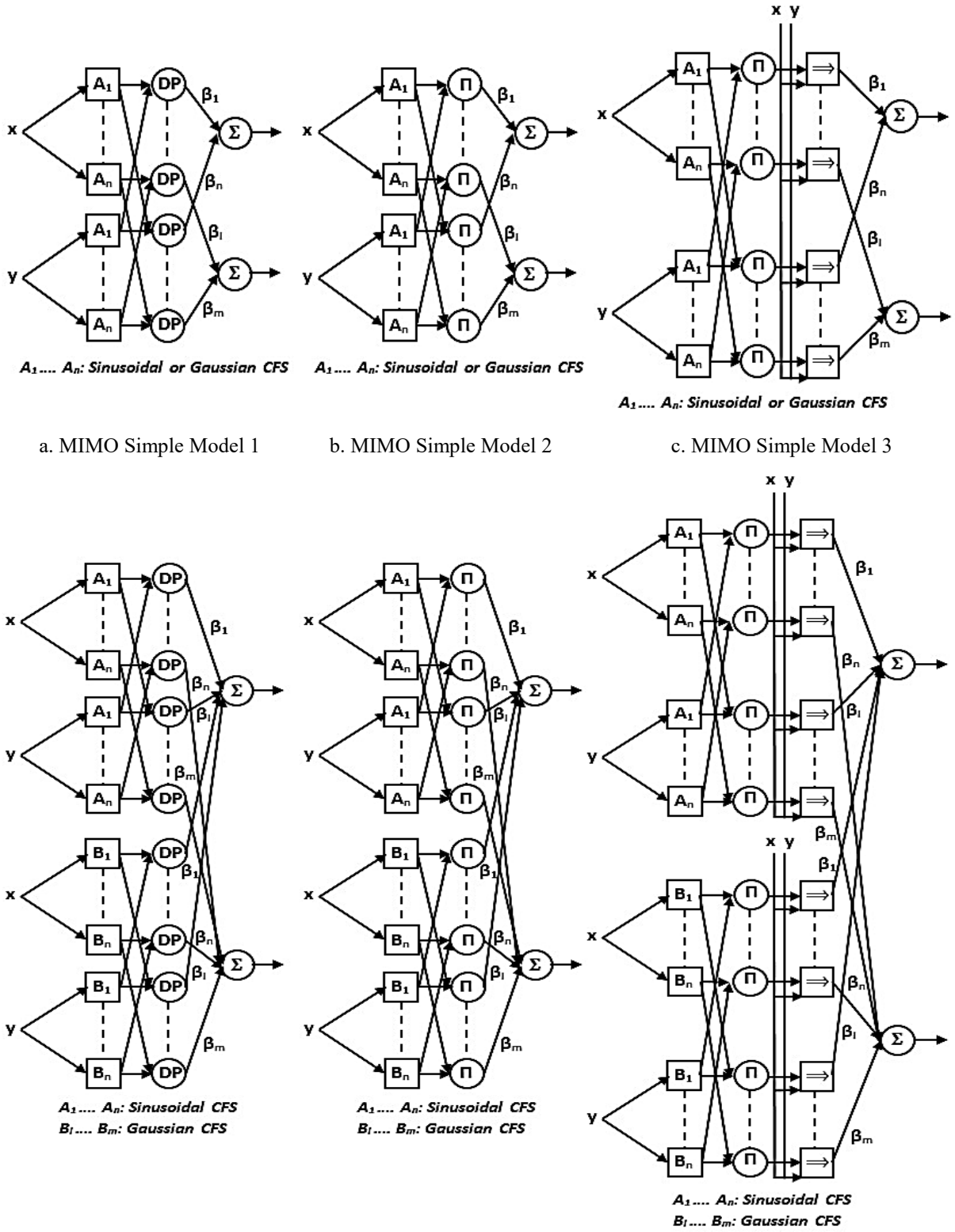


a. MISO Simple Model 1 b. MISO Simple Model 2 c. MISO Simple Model 3



d. MISO Hybrid Model 1 e. MISO Hybrid Model 2 f. MISO Hybrid Model 3

Figure 39 - Three MISO Simple/Hybrid RCNFIS models in the three modes for a sample time series with two variates x and y



a. MIMO Simple Model 1 b. MIMO Simple Model 2 c. MIMO Simple Model 3

d. MIMO Hybrid Model 1 e. MIMO Hybrid Model 2 f. MIMO Hybrid Model 3

Figure 40 - Three MIMO Simple/Hybrid RCNFIS models in the three modes for a sample time series with two variates x and y

6.2.2. Datasets

In order to assess the overall performance of the proposed algorithms, we carry out our prediction experiments on three multivariate datasets of medium and large scale, which are described below.

6.2.2.1. Rössler Attractor

Chaotic systems are nonlinear dynamic systems that are very sensitive to their initial conditions [279]. A small change in the initial conditions of such systems will cause many changes in the future. This phenomenon is known in butterfly theory as the butterfly effect [280]. Chaotic behavior occurs in many natural systems, including fluid flow, palpitations, and climate [281-283], as well as in some systems with artificial components, such as the stock market [284] and road traffic [285]. The Rössler system is another example of such a system. The Rössler system, first studied by Otto Rössler, consists of three standard nonlinear differential equations as follows [274, 275]:

$$\frac{dx}{dt} = -y - z, \quad \frac{dy}{dt} = x + ay, \quad \frac{dz}{dt} = b + z(x - c) \tag{112}$$

These differential equations describe a chaotic dynamic system that affords both complex and simple output based on the parameters a, b, and c. The integration of these equations with respect to time leads to the formation of a three-dimensional object known as an attractor [286, 287]. A set of common coefficients for studying Rössler equations reported as the main set of coefficients studied by Rössler [274] are a=b=0.2 and c=5.7. Using these standard values, we obtained a database with 3 properties (variables) and 17177 records for each one. The graph of Rössler attractor in terms of time for the Rössler attractor is plotted in Figure 41.

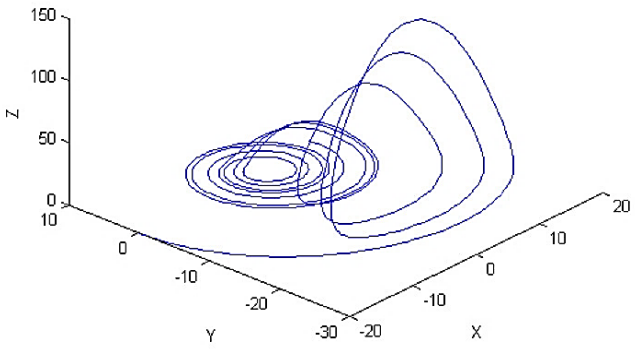


Figure 41 - The Rössler attractor graph for a=b=0.2, c=5.7

6.2.2.2. Condition Monitoring Dataset

We discussed this dataset in detail in Chapter 4 and briefly in Chapter 5, therefore we will not repeat ourselves here, but just to note that this dataset has 6 features and about 1.3 million records, and unlike the previous chapter, we consider all its variables in this chapter, which is about multivariate time series experiments.

6.2.2.3. Gas Sensor Dataset

The Gas sensor dataset [276, 277] contains the recordings of 16 chemical sensors of 4 different types that have been exposed to two dynamic gas mixtures at varying concentration levels that have been continuously obtained during 12 hours without interruption. The data is presented in two different datasets, so that each dataset contains a mixture of data, that is, the ethylene_CO dataset contains the recordings of sensors exposed to a mixture of ethylene and CO in the air and the ethylene_methane dataset includes recording of sensors exposed to a mixture of ethylene and methane in the air.

The structure of the datasets is the same for the first and second mixture. Each dataset distributed in 19 columns. First column is time (in seconds), second and third columns present

concentration set point of Methane (or CO) and Ethylene, respectively. Also, the remaining 16 columns are the recordings of the sensor array. Lastly, each database contains about 4 million records.

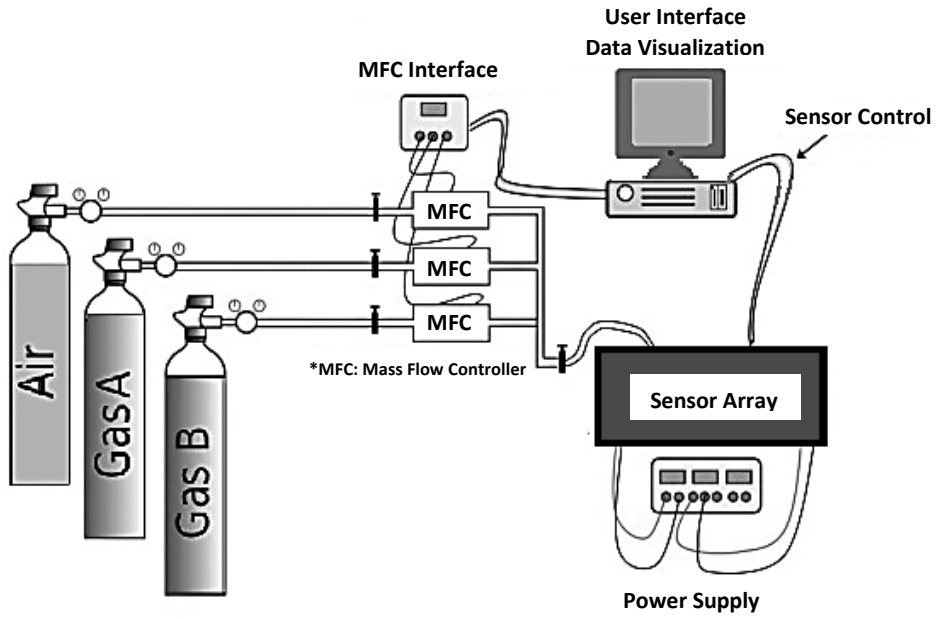


Figure 42 - A chemical sensing system for measuring the conductivity of a sensor array exposed to different gas conditions [276]

Note that in order to perform our experiments, we used only one of the two datasets mentioned, namely ethylene_methane, and also to simplify and speed up the experiments, we considered only the values of 4 sensors with half the number of records, i.e. 2 Million records.

6.3. Preprocessing

Data preprocessing is the process of converting raw data into a machine-understandable format [288] which includes operations such as cleaning [289], instance selection [290], normalization [291], transformation [292], feature extraction [293], and so on. The important steps we took to prepare our data are shown in Figure 43.

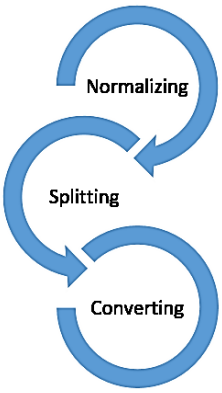


Figure 43 - Data preprocessing steps in our project

As shown in Figure 43, we began by normalizing each variate in the data using the min-max normalization [294], in which the values of each feature are mapped to a value between 0 and 1 using the Formula (113):

$$X_n = \frac{X - X_{min}}{X_{max} - X_{min}} \qquad (113)$$

where x_{min} and x_{max} are the minimum and maximum values for the X variate, respectively, and x is the current value.

The next stage of our preprocessing involves the train-test split procedure to estimate the performance of our proposed models. To this end, we divided our datasets into training and testing sets and assigned 2/3 data points to the former and the remaining one-third to the latter. Of course, this division was a chronologically ordered single split of our data, it means the order in which the events occurred, from first to last, and not randomly [295].

Last but not least step is to convert our multivariate time series into a suitable form for machine learning algorithms. A conventional method for this purpose is the delay embedding developed in Takens [15]. The vital concept is that since a time series is a sequence of observations on a system, the forecasting of the time series is actually to predict the state vector of system based

on previous states. In other words, time-delay embedding refers to the inclusion of historic data into the dynamical system models [296]. In the following, we will consider the delay embedding in univariate and multivariate time series.

A delay vector in a univariate time series is defined by [248]:

$$S_n = (S_{n-(m-1)d}, S_{n-(m-2)d}, \dots, S_n) \quad (114)$$

where S_n is the n-th delay vector with two main parameters, namely m (dimension) and d (delay), both of which must be determined heuristically to generate the delay vector. Finding d allows us to get an optimal level of autocorrelation in each delay vector and it is usually estimated by taking the first minimum of the mutual function [247], while m can be estimated by the false-nearest neighborhood algorithm [250].

A very important point to remember is that in Formula (114), the delay vectors are relative to a moment in time. This means that our input, like ANCFIS, is a sliding window, not orthogonal delay vectors, because in order to match a sinusoidal membership function to an observation, we must keep phase information in our inputs. The phase information is destroyed by definition when orthogonal lagged inputs are used. To match the membership functions to the sliding window, we take the variate as a single input. As a result, ANCFIS/RCNFIS only requires a single input per variate, while systems using lagged inputs require $\prod_{i=1}^{n} r_i$ inputs, where n is the number of variates and r_i is the number of lags. In this way, ANCFIS/RCNFIS significantly reduces the combinatorial explosion inherent in time-series forecasting [3, 278].

In multivariate time series, a delay vector can be described as follows [297]:

$$V_n = \begin{pmatrix} x_{1,n}, x_{1,n-d_1}, \dots, x_{1,n-(m_1-1)d_1}, \\ x_{2,n}, x_{2,n-d_2}, \dots, x_{2,n-(m_2-1)d_2}, \\ \dots, \\ x_{t,n}, x_{t,n-d_t}, \dots, x_{t,n-(m_t-1)d_t} \end{pmatrix} \quad (115)$$

where V_n is the n-th delay vector including t variates of length n, each in general form of $X_i = (x_{1,i}, x_{2,i}, \dots, x_{t,i})$ and i-th variate has delay d_i and dimension m_i (i=1,2,…,t). To obtain d and m, each variate of the multivariate time series is considered separately, and the same approach as the univariate time series are applied on them [297-299]. Using the functions in TISEAN package [266], we obtained the best values for *d* and *m* in the same manner as in Chapter 5 Figures 44 to 46 show the plots of finding the best parameters *m* and *d* using mutual, phase and false-nearest diagrams (Note that these diagrams are plotted for one of the datasets' variate, and the process is similar for other variates, leading to similar results).

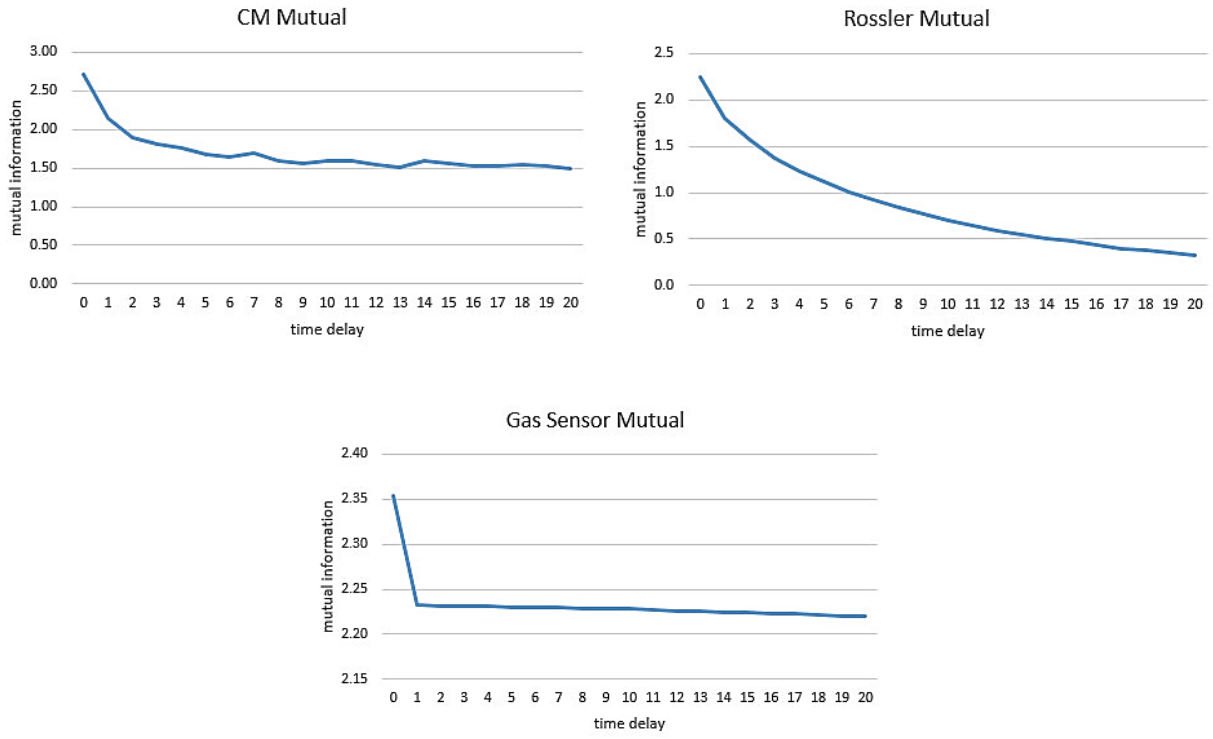


Figure 44 - Mutual Diagrams for the datasets

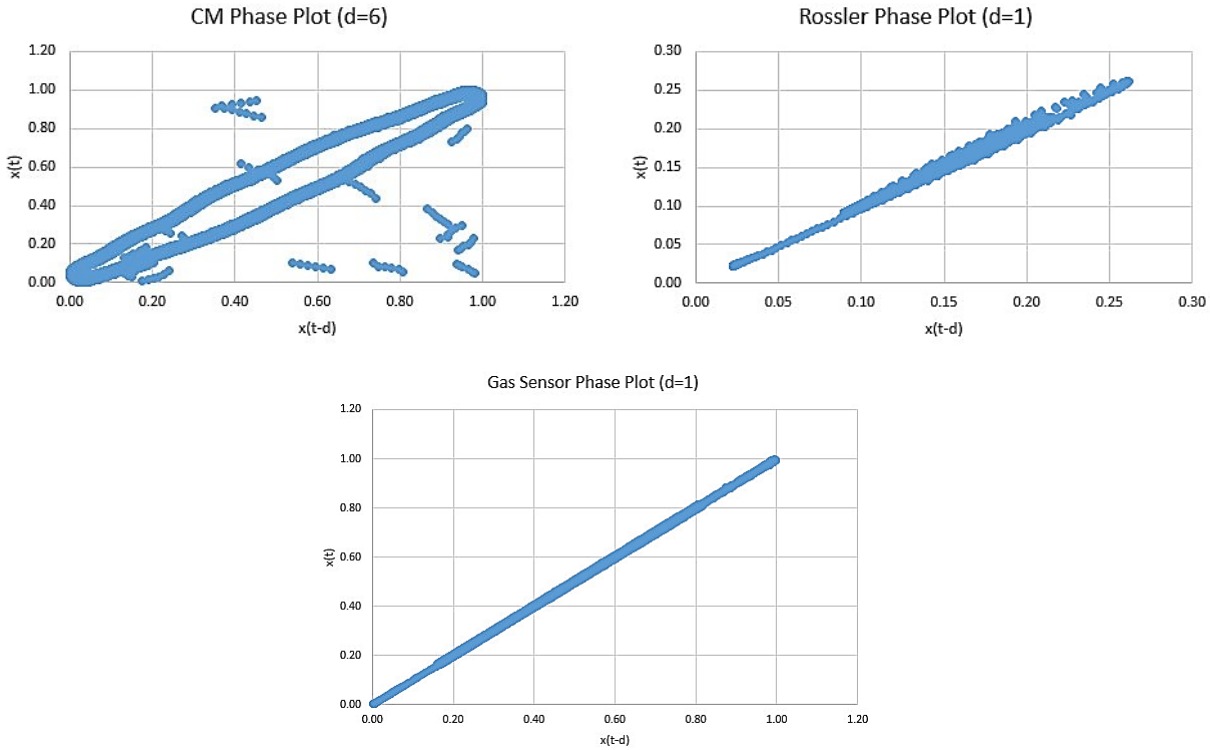


Figure 45 - Phase Plot for the datasets based on their obtained delay

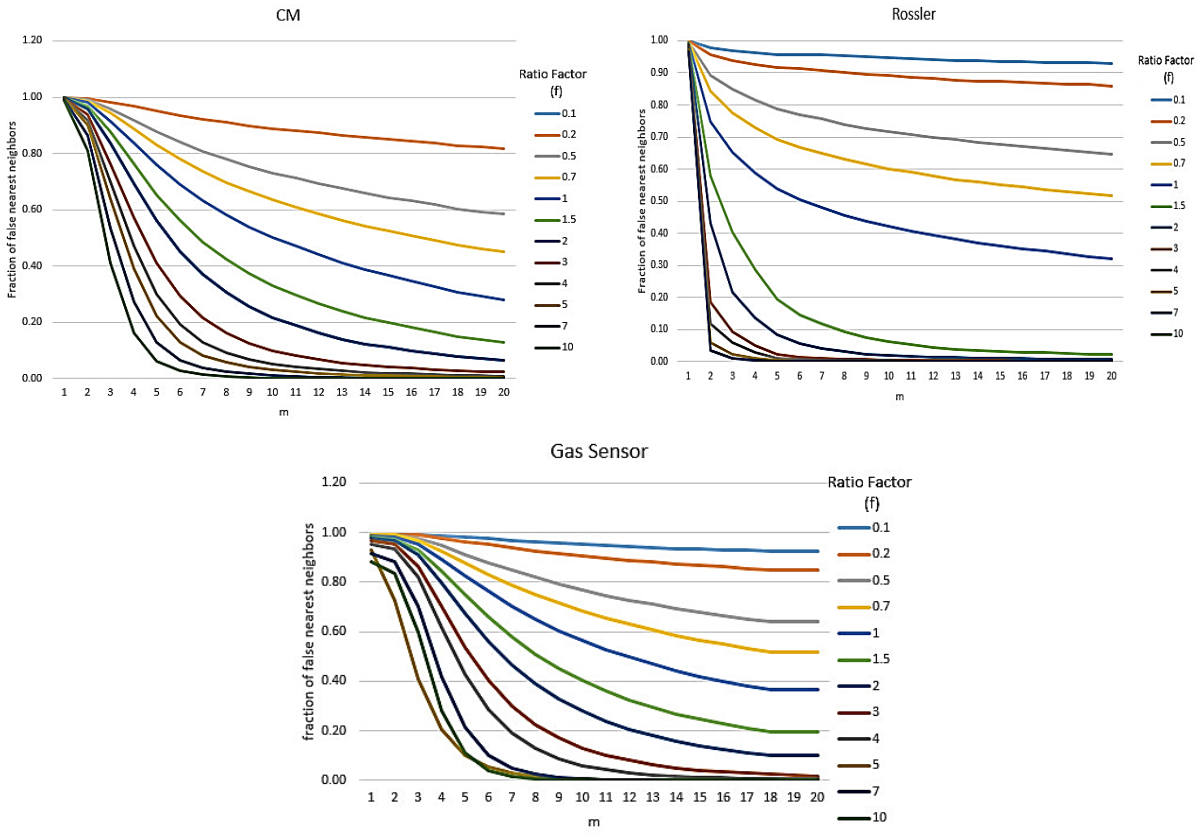


Figure 46 - Proportion of False-Nearest Neighbors for different values of ratio factor (f)

6.4. Results

6.4.1. Experimental Design

According to what was discussed in the previous section and the related graphs, the obtained delay vector parameters for each dataset are shown in Table 11. Also, as a rule of thumb (ROT) and the result we obtained in [278] it can be said that the lower lag with lower dimension gave a better result. Therefore, we also considered d = 1 and m = 3 for CM dataset.

Table 11 - Obtained delay vectors parameters

Dataset	Delay Vector Parameters	
	Delay (d)	Dimension (m)
CM	6 Based on the graphs	7 Based on the graphs
	1 Based on the ROT	3 Based on the ROT
Rössler	1	3
Gas Sensor	1	6

In all experiments, we used the Root Mean Square Error (RMSE) to evaluate our proposed models and compare them with other approaches and models:

$$RMSE = \sqrt{\frac{1}{K}\sum_{i=1}^{K} MSE_i} \quad , \quad MSE_i = \frac{\sum_{j=1}^{n}(y_j - \hat{y}_1)^2}{n} \tag{116}$$

where K is the number of variates in the time series, MSE_i is mean squared one step-ahead of *i-th* variate, y_j is predicted value, and $\hat{y}_1$ is desired value.

6.4.2. Experimental Results

Exploration of hyper-parameters in our experiments was repeated 1000 times per d and m obtained in Table 11 (to obtain the best RMSE with the least standard deviation), the number of neurons in the second layer of the models changed from 4 to 50 neurons, the value of forgetting factor (λ) in the RLS is selected from the set {0.01, 0.1, 0.5} in each experiment, and finally, as discussed in the previous sections, the experiments are performed for three modes of activation functions, namely Sinusoidal CFS, Gaussian CFS, and a combination of the two, and three implementation methods, namely SISO, MISO and MIMO. Therefore, we performed several experiments with different combinations of hyper-parameters, the best results of which are described in the next section (4.3). It should be noted that all models have been implemented in the Visual Studio C++ environment. Tables 12 to 15 show a comparison between the results of exploring the parameters for the different modes of our proposed models for each of the CM, Rössler, and Gas Sensor datasets.

Table 12 - Explore and compare the best results of the three proposed models in different modes & methods (CM dataset – d6m7)

d6 m7	Unit	λ	RMSE	SD
SISO RCNFIS Model 1 (Sin)	4	0.5	1.23974E-01	9.06050E-02
SISO RCNFIS Model 1 (Gauss.)	50	0.5	1.13600E-02	2.39100E-03
SISO RCNFIS Model 1 (Mixed)	4	0.5	1.02510E-01	8.51410E-02
MISO RCNFIS Model 1 (Sin)	20	0.01	1.37664E-01	9.09740E-02
MISO RCNFIS Model 1 (Gauss.)	50	0.5	2.50500E-02	2.33600E-03
MISO RCNFIS Model 1 (Mixed)	20	0.5	1.16230E-01	8.52510E-02
MIMO RCNFIS Model 1 (Sin)	4	0.1	1.22605E-01	9.06590E-02
MIMO RCNFIS Model 1 (Gauss.)	50	0.5	9.99100E-03	2.34000E-03
MIMO RCNFIS Model 1 (Mixed)	4	0.5	1.01141E-01	8.51010E-02
SISO RCNFIS Model 2 (Sin)	4	0.1	2.46715E+00	9.64550E-02
SISO RCNFIS Model 2 (Gauss.)	50	0.5	1.52320E-02	4.46400E-03
SISO RCNFIS Model 2 (Mixed)	50	0.5	1.52530E-02	4.45600E-03
MISO RCNFIS Model 2 (Sin)	4	0.1	2.46852E+00	9.67150E-02
MISO RCNFIS Model 2 (Gauss.)	50	0.5	1.66010E-02	4.23200E-03
MISO RCNFIS Model 2 (Mixed)	20	0.01	1.66220E-02	4.25300E-03
MIMO RCNFIS Model 2 (Sin)	4	0.5	2.48221E+00	9.65190E-02
MIMO RCNFIS Model 2 (Gauss.)	50	0.5	3.02910E-02	4.60100E-03
MIMO RCNFIS Model 2 (Mixed)	4	0.1	3.03120E-02	4.62200E-03
SISO RCNFIS Model 3 (Sin)	4	0.5	6.33855E-01	6.76849E+00
SISO RCNFIS Model 3 (Gauss.)	50	0.5	8.54980E-02	5.47129E-01
SISO RCNFIS Model 3 (Mixed)	20	0.01	3.74724E-01	3.65436E+00
MISO RCNFIS Model 3 (Sin)	4	0.1	6.47545E-01	6.76886E+00
MISO RCNFIS Model 3 (Gauss.)	50	0.5	9.91880E-02	5.47498E-01
MISO RCNFIS Model 3 (Mixed)	50	0.5	3.88414E-01	3.65472E+00
MIMO RCNFIS Model 3 (Sin)	4	0.01	6.32486E-01	6.76804E+00
MIMO RCNFIS Model 3 (Gauss.)	4	0.5	8.41290E-02	5.47356E-01
MIMO RCNFIS Model 3 (Mixed)	4	0.5	3.73355E-01	3.65415E+00

Table 13 - Explore and compare the best results of the three proposed models in different modes & methods (CM dataset – d1m3)

d1 m3	Unit	λ	RMSE	SD
SISO RCNFIS Model 1 (Sin)	4	0.5	1.55892E-01	8.91000E-02
SISO RCNFIS Model 1 (Gauss.)	50	0.5	2.32800E-03	4.25000E-04
SISO RCNFIS Model 1 (Mixed)	4	0.5	8.76130E-02	1.03891E-01
MISO RCNFIS Model 1 (Sin)	4	0.5	1.57261E-01	8.98920E-02
MISO RCNFIS Model 1 (Gauss.)	50	0.5	6.01800E-03	3.28000E-04
MISO RCNFIS Model 1 (Mixed)	20	0.1	8.89820E-02	1.03613E-01
MIMO RCNFIS Model 1 (Sin)	4	0.01	1.70951E-01	8.92610E-02
MIMO RCNFIS Model 1 (Gauss.)	50	0.5	1.97080E-02	1.80000E-05
MIMO RCNFIS Model 1 (Mixed)	4	0.1	1.02672E-01	1.03982E-01
SISO RCNFIS Model 2 (Sin)	4	0.1	6.11232E-01	9.88630E-02
SISO RCNFIS Model 2 (Gauss.)	50	0.5	2.87000E-03	3.50100E-03
SISO RCNFIS Model 2 (Mixed)	4	0.5	2.88000E-03	4.51100E-03
MISO RCNFIS Model 2 (Sin)	4	0.5	6.24922E-01	9.82320E-02
MISO RCNFIS Model 2 (Gauss.)	50	0.5	1.65600E-02	3.28700E-03
MISO RCNFIS Model 2 (Mixed)	20	0.01	1.65700E-02	4.28800E-03
MIMO RCNFIS Model 2 (Sin)	4	0.5	6.09863E-01	9.86350E-02
MIMO RCNFIS Model 2 (Gauss.)	50	0.5	1.50100E-03	3.99500E-03
MIMO RCNFIS Model 2 (Mixed)	50	0.5	1.51100E-03	4.01800E-03
SISO RCNFIS Model 3 (Sin)	4	0.01	7.01657E-01	1.48932E+00
SISO RCNFIS Model 3 (Gauss.)	50	0.5	1.79600E-03	4.21600E-03
SISO RCNFIS Model 3 (Mixed)	4	0.5	5.17368E-01	1.04914E+00
MISO RCNFIS Model 3 (Sin)	4	0.1	7.03026E-01	1.48972E+00
MISO RCNFIS Model 3 (Gauss.)	50	0.5	3.16500E-03	4.85500E-03
MISO RCNFIS Model 3 (Mixed)	20	0.1	5.18737E-01	1.04943E+00
MIMO RCNFIS Model 3 (Sin)	4	0.1	7.16716E-01	1.48966E+00
MIMO RCNFIS Model 3 (Gauss.)	50	0.5	1.68550E-02	4.79600E-03
MIMO RCNFIS Model 3 (Mixed)	50	0.01	5.32427E-01	1.04937E+00

Table 14 - Explore and compare the best results of the three proposed models in different modes & methods (Rössler dataset)

d1 m3	Unit	λ	RMSE	SD
SISO RCNFIS Model 1 (Sin)	4	0.5	3.48250E-02	3.05360E-02
SISO RCNFIS Model 1 (Gauss.)	50	0.5	4.42600E-03	8.37000E-05
SISO RCNFIS Model 1 (Mixed)	4	0.5	1.97950E-02	2.72940E-02
MISO RCNFIS Model 1 (Sin)	4	0.1	3.61940E-02	3.08840E-02
MISO RCNFIS Model 1 (Gauss.)	50	0.5	5.79500E-03	8.38000E-05
MISO RCNFIS Model 1 (Mixed)	20	0.01	2.11640E-02	2.78540E-02
MIMO RCNFIS Model 1 (Sin)	4	0.5	4.98840E-02	3.08250E-02
MIMO RCNFIS Model 1 (Gauss.)	50	0.5	1.94850E-02	8.39000E-05
MIMO RCNFIS Model 1 (Mixed)	50	0.1	3.48540E-02	2.77950E-02
SISO RCNFIS Model 2 (Sin)	4	0.5	1.41500E-02	5.23410E-02
SISO RCNFIS Model 2 (Gauss.)	50	0.5	4.44300E-03	2.81600E-03
SISO RCNFIS Model 2 (Mixed)	4	0.5	4.45200E-03	2.74900E-03
MISO RCNFIS Model 2 (Sin)	4	0.1	1.44364E-01	5.20540E-02
MISO RCNFIS Model 2 (Gauss.)	50	0.5	5.80900E-03	2.49900E-03
MISO RCNFIS Model 2 (Mixed)	20	0.01	7.65270E-02	2.21700E-03
MIMO RCNFIS Model 2 (Sin)	4	0.1	1.58054E-01	5.24150E-02
MIMO RCNFIS Model 2 (Gauss.)	50	0.5	1.94990E-02	2.44300E-03
MIMO RCNFIS Model 2 (Mixed)	4	0.5	9.02170E-02	2.45200E-03
SISO RCNFIS Model 3 (Sin)	4	0.1	1.42995E-01	7.06003E-01
SISO RCNFIS Model 3 (Gauss.)	50	0.5	4.44000E-03	1.93000E-04
SISO RCNFIS Model 3 (Mixed)	4	0.5	7.51580E-02	3.70306E-01
MISO RCNFIS Model 3 (Sin)	4	0.1	1.44364E-01	7.06054E-01
MISO RCNFIS Model 3 (Gauss.)	50	0.5	5.80900E-03	4.99000E-04
MISO RCNFIS Model 3 (Mixed)	20	0.01	7.65270E-02	3.70217E-01
MIMO RCNFIS Model 3 (Sin)	4	0.5	1.58054E-01	7.06364E-01
MIMO RCNFIS Model 3 (Gauss.)	50	0.5	1.94990E-02	8.09000E-04
MIMO RCNFIS Model 3 (Mixed)	50	0.1	9.02170E-02	3.70527E-01

Table 15 - Explore and compare the best results of the three proposed models in different modes & methods (Gas Sensor dataset)

d1 m3	Unit	λ	RMSE	SD
SISO RCNFIS Model 1 (Sin)	4	0.5	1.36695E-01	7.28438E-02
SISO RCNFIS Model 1 (Gauss.)	50	0.5	1.89490E-03	4.49356E-04
SISO RCNFIS Model 1 (Mixed)	20	0.1	7.34578E-02	8.35699E-02
MISO RCNFIS Model 1 (Sin)	20	0.01	1.50385E-01	7.32008E-02
MISO RCNFIS Model 1 (Gauss.)	50	0.5	1.55849E-02	8.06356E-04
MISO RCNFIS Model 1 (Mixed)	20	0.5	8.71478E-02	8.39269E-02
MIMO RCNFIS Model 1 (Sin)	20	0.1	1.38064E-01	7.30028E-02
MIMO RCNFIS Model 1 (Gauss.)	50	0.5	3.26390E-03	6.08356E-04
MIMO RCNFIS Model 1 (Mixed)	4	0.01	7.48268E-02	8.37289E-02
SISO RCNFIS Model 2 (Sin)	4	0.01	5.24225E-01	5.47093E+00
SISO RCNFIS Model 2 (Gauss.)	50	0.5	1.72012E-03	1.57224E-05
SISO RCNFIS Model 2 (Mixed)	4	0.1	2.74946E-01	2.89007E+00
MISO RCNFIS Model 2 (Sin)	4	0.1	5.37915E-01	5.47129E+00
MISO RCNFIS Model 2 (Gauss.)	50	0.5	1.54101E-02	3.72722E-04
MISO RCNFIS Model 2 (Mixed)	20	0.5	2.88636E-01	2.89043E+00
MIMO RCNFIS Model 2 (Sin)	4	0.1	5.25594E-01	5.47109E+00
MIMO RCNFIS Model 2 (Gauss.)	50	0.5	3.08912E-03	1.74722E-04
MIMO RCNFIS Model 2 (Mixed)	20	0.1	2.76315E-01	2.89023E+00
SISO RCNFIS Model 3 (Sin)	20	0.5	5.18624E-01	2.36552E+00
SISO RCNFIS Model 3 (Gauss.)	50	0.5	1.73281E-03	1.20399E-05
SISO RCNFIS Model 3 (Mixed)	20	0.1	2.66200E-01	1.52224E+00
MISO RCNFIS Model 3 (Sin)	20	0.01	5.32314E-01	2.36588E+00
MISO RCNFIS Model 3 (Gauss.)	50	0.5	1.54228E-02	3.69040E-04
MISO RCNFIS Model 3 (Mixed)	4	0.001	2.79890E-01	1.52260E+00
MIMO RCNFIS Model 3 (Sin)	4	0.1	5.19993E-01	2.36568E+00
MIMO RCNFIS Model 3 (Gauss.)	50	0.5	3.10181E-03	1.71040E-04
MIMO RCNFIS Model 3 (Mixed)	4	0.1	2.67569E-01	1.52240E+00

6.4.3. Out-of-Sample Results

Table 16 presents the results of out-of-sample error for all three datasets for our proposed models against the other architectures and well-known approaches that were applied recently on each time series. To achieve the results of the Table 16, for the CM dataset, the best parameters were obtained on the first proposed model (MIMO RCNFIS Model 1) for d=6 and m=7 (N=50, λ=0.5, AF=$Gauss.CFS$). It should be noted that although we obtained a better result than other architectures and methods with this delay vector (d = 6 and m = 7) and the parameters of the first proposed model, to obtain a lower RMSE (as shown in Table 16), we also performed our experiments on the delay vector (d = 1 and m = 3), which gave the best result on the second proposed model (MIMO RCNFIS Model 2) and the parameters (N=50, λ=0.5, AF=$Gauss.CFS$). Moreover, for the Rössler dataset, the best parameters were gained on the first proposed model (SISO RCNFIS Model 1) with the parameters (N=50, λ=0.5, AF=$Gauss.CFS$). Finally, for the Gas Sensor dataset, the best parameters were discovered on the second proposed model (SISO RCNFIS Model 2) with the parameters (N=50, λ=0.5, AF=$Gauss.CFS$).

Additionally, we used WEKA [254] to implement the RBFN with the logistic function as the activation function and performed our experiments with 1 to 400 neurons in the hidden layer and the standard deviation selecting from the set $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. The best result (lowest RMSE) as shown in Table 16 was obtained for the CM dataset with hyper parameter (N_n =100, SD=10^{-5}) in MIMO implementation method, for the Rössler dataset with hyper parameter (N_n =50, SD=10^{-5}) in SISO implementation method, and for the Gas Sensor dataset with hyper parameter (N_n =200, SD=10^{-5}) in SISO implementation method. Also, to implement the MLP, we applied both WEKA and Scikit-learn [257] to compare the results and get the best one, the number of hidden layers (N_{hl}) is considered to 2 and the number of neurons

(N_n) in each layer is changed from 50 to 1000 neurons in each layer. Also, the learning rate (η) value is selected from the set {0.001, 0.01, 0.1, 0.3} in each experiment, the value of Momentum (m) is assumed to 0.2 and the value of Batch Size (BS) is equal to 100. Thus, the best hyper parameters for the least RMSE are obtained for the CM dataset {N_{hl} =2, N_n=500, η=0.001, m=0.2, BS=100} in MIMO implementation method, for the Rössler dataset {N_{hl} =2, N_n=1000, η=0.1, m=0.2, BS=100} in SISO implementation method, and for the Gas Sensor dataset {N_{hl} =2, N_n=500, η=0.01, m=0.2, BS=100} in SISO implementation method. Lastly, to implement the SMOreg on WEKA, we used RBF kernel, γ selecting from the set {0.01, 0.05, 0.005}, complexity (C) selecting from the set {1, 5, 10, 15, 20, 25}, and BS = 100. The best hyper parameters for the least RMSE are obtained for the CM dataset {γ=0.005, C=25, BS=100} in MIMO implementation method, for the Rössler dataset {γ=0.005, C=10, BS=100} in SISO implementation method, and for the Gas Sensor dataset for the Rössler Dataset {γ=0.05, C=15, BS=100} in SISO implementation method.

Therefore, as it can be seen in the Table 16, our proposed models yielded the lowest forecast error compared to other architectures.

Table 16 - Compare results in different architectures in terms of RMSE

Architecture	Dataset/RMSE		
	CM	Rössler	GAS Sensor
RCNFIS	(d6 m7) MIMO Model 1 (Gauss.) **9.991E-03** (d1 m3) MIMO Model 2 (Gauss.) **1.501E-03**	SISO Model 1 (Gauss.) **4.426E-03**	SISO Model 2 (Gauss.) **1.720E-03**
LSTM [300]	-	5.6E-01	-
GA [301]	-	1.21E+00	-
PSO [301]	-	4.94E-01	-
WOA [301]	-	9.72E-01	-
MFO [301]	-	4.85E-01	-
FIS [302] (with uniform embedding)	-	4.92E-01	-
FIS [302] (without uniform embedding)	-	4.84E-01	-
ARIMA (2,0,3) [302]	-	1.27E+00	-
EGRNN++ [303]	-	-	2.48E-01
Multi-GAN [304]	-	-	8.54E-02
Rec-TSA [304]	-	-	1.66E-01
TreNet [305]	-	-	9.57E+00
RBF	MIMO (d6 m7) 6.01E-02 (d1 m3) 5.03E-02	SISO 2.0E-02	SISO 1.93E-01
MLP	MIMO (d6 m7) 1.08E-02 (d1 m3) 2.50E-03	SISO 2.63E-02	SISO 3.15E-03
SMOreg	MIMO (d6 m7) 4.20E-02 (d1 m3) 5.20E-02	SISO 4.5E-02	SISO 3.90E-02

In order to determine whether there are a significant differences between the results obtained from our proposed models and the results obtained from other models and published papers, the Z-test is again applied [270, 271].

A. For the CM dataset we have:

H_0: $\mu_0 \geq 0.0025$

H_a: $\mu_0 < 0.0025$ (This is a left-tailed (one tailed) test)

0.0025 is the lowest value obtained for RMSE in the methods and results of other papers (which is obtained for CM from MLP) and the rest of the RMSE values provided are greater than or equal to this value. Therefore, since we have reached a lower value of it (that is 0.001501), we have written our hypothesis as above so that we can reject the null hypothesis claim. So we have:

n = 1000

$\bar{X}$ = 0.001501

s = 0.003995

We run our statistical test on a 98% confidence level, then the significance level (α) is:

c = 0.98 → α = 1-c = 0.02

From Formula (110) we have: $Z_c = \frac{0.001501-0.0025}{0.003995/\sqrt{1000}} = -0.1264$

Using the Z-table, we find the p-value corresponding to the z-value obtained (≅ 0.00003) and then by comparing the p-value with the α-value and that its value is much smaller (0.00003<0.02), then our null hypothesis (H_0) is rejected.

B. **For the Rössler dataset we have:**

H_0: $\mu_0 \geq 0.02$

H_a: $\mu_0 < 0.02$ (This is a left-tailed (one tailed) test)

n = 1000

$\bar{X}$ = 0.004426

s = 0.0000837

c = 0.98 → α = 1-c = 0.02

From Formula (110) we have: $Z_c = \frac{0.004426-0.02}{0.0000837/\sqrt{1000}} = -5.88E+03$

Similar to the previous case (A) and using the Z-table, the p-value corresponding to the z-value obtained (≅ 0.00003) and comparing with the α-value, it can be concluded that its value is much smaller (0.00003<0.02), then our null hypothesis (H_0) is rejected.

C. **For the Gas Sensor dataset we have:**

H_0: $\mu_0 \geq 0.00315$

H_a: $\mu_0 < 0.00315$ (This is a left-tailed (one tailed) test)

n = 1000

$\bar{X}$ = 0.00172

s = 0.0000157

c = 0.98 → α = 1-c = 0.02

From Formula (110) we have: $Z_c = \frac{0.00172-0.00315}{0.0000157/\sqrt{1000}} = -2.88E+03$

Alike to the former cases (A and B), by applying the Z-table, the p-value corresponding to the z-value obtained (≅ 0.00003) and comparing with the α-value, it is possible to deduce that its value is substantially lower (0.00003<0.02), then our null hypothesis (H_0) is rejected.

As in the previous chapter, by applying the Friedman test, we determined which CFS statistically improved accuracy. Therefore, we have:

k= 27

n= 3

α (level of significance) = 0.05

Based on the Formula (111), the calcukated p-value is equal to: 0.0001

A small p-value compares to α indicates that H_0 is rejected, which means that there are significant differences between the models, or in other words, different CFS improve accuracy significantly. Having rejected H_0, we can compare models with different CFS using MCB method. For α = 0.05, k=3, and df=2 we obtained $q_\alpha = 5.418$ giving $r_{\alpha,k,n} = 1.042$.

We present the differences in average ranks for our three CFS types in Table 17. Gaussian fuzzy sets were clearly the superior alternative to sinusoidal ones; however, the differences between either sinusoids or Gaussians and the mixed-CFS design were not significant. What we can say, however, is that using mixed CFS was not superior to just using Gaussian CFS.

Table 17 - The average rank of the measures

	$\overline{\text{Sin}}$	$\overline{\text{Gauss}}$	$\overline{\text{Mixed}}$
$\overline{\text{Sin}}$	0	2	1
$\overline{\text{Gauss}}$		0	1
$\overline{\text{Mixed}}$			0

6.5. Conclusion

In this chapter, we extended the RCNFIS architecture multivariate time series forecasting. Then we tested this architecture on large and medium datasets, and found that our architecture was superior compared to other previous published papers and work done, as shown in Table 16.

To summarize, the three proposed RCNFIS models are based on CFS&L and they are very similar to ANCFIS architecture, but by eliminating some extra layers, it leads to higher efficiency in terms of the measured error of the algorithm. Also, like RANCFIS, they are feed-forward neuro-fuzzy algorithms which employ randomized learning. This approach eliminates the backpropagation learning step, and leads to a fast learning rate.

Our investigation has focused on implementing rule interference using the inner product of firing weights in the first proposed model. In the second proposed model, we employed a weighted complex summation instead. Finally, in our third proposed model we tried to examine the effect of the consequents parameters. Moreover, in all three proposed models, we examined the efficiency of two activation functions, the sinusoidal MF and the complex Gaussian MF. In addition, for each proposed model we considered a combination of these two activation functions (a bank of sinusoidal MFs, and another bank of complex Gaussian MFs, in the input layer) to perceive the complex Gaussian fuzzy sets singular modifiers that can modify the behavior of rules built from sinusoidal fuzzy sets. In addition, to implement our models, we have used three implementation methods, namely SISO, MISO and MIMO, according to what is stated in section 6.2.1.

7. Chapter VII: Summary, Conclusion, and Future Work

7.1. Summary and Conclusion

In this dissertation, we reviewed a variety of time series forecasting methods and algorithms, which can be categorized into two main types: statistical models and machine learning models. A brief overview of statistical methods and models is presented in the second chapter, and then one of the most important theories of machine learning, FS&L, is examined, followed by a major extension, CFS&L, which forms the basis of this dissertation's studies, architectures, and experiments.

In the third chapter, we reviewed the various types of CFS&L-based architectures, each of which tried to achieve two important goals by structural modification or by using different methods and algorithms: 1. achieving more accurate and parsimonious results, and 2. speeding up the learning algorithm. ANCFIS was designed to work with both univariate and multivariate time series datasets. The main drawback of the ANCFIS architecture was its slow learning algorithm. So to speed up this neuro-fuzzy system, another algorithm called RANCFIS was designed. This algorithm, which was a feed-forward complex neuro-fuzzy system, took advantage of the randomized learning algorithm in its structure. This modification in structure eliminated the back propagation learning and led to faster learning. The experimental results showed that the RANCFIS learning rate was faster than ANCFIS but slower than ANCFIS-ELM (An ANCFIS extended architecture inspired by the Extreme Learning Machine (ELM) algorithm and the SLFN architecture). Next, a fast and compact learning algorithm based on CFS&L called FANCFIS was designed, which is especially suitable for data stream mining.

FANCFIS consisted of two steps of learning: the initial step and the incremental stage, and the membership functions were determined using FFT. It should be noted that in all the mentioned architectures that were from ANCFIS family, the complex sinusoidal membership function is used, but in another family of CFS&L based architectures, i.e. CNFS, the complex Gaussian membership function is used.

While the architectures and algorithms mentioned are very significant and valuable, they still have problems, one of the most important of which is dealing with big data. Accordingly, one of the main focus of our research in this dissertation has been on this issue. As a result, in addition to the existing datasets that were used in this thesis, we also obtained a large dataset ourselves, which is about one of the most important topics in the world, namely condition monitoring of induction motors, which was explained in detail in Chapter 4 (It should be noted that one of the remarkable aspects of this thesis is that all our experiments on large-scale datasets are based on real sensor data). Then, in the fifth chapter, we introduced a new architecture based on CFS and inspired by the ANCFIS architecture and in three variants, which also took advantage of previous architectures such as randomized learning. Another important feature was the reduction in the number of layers in order to achieve a better speed in the learning algorithm. Also, as another major focus of this dissertation, we used both types of complex membership functions used in CFS&L-based architectures, sinusoidal and complex-valued Gaussian, as well as combining them in all three variants of our proposed architecture (RCNFIS) to have a systematic comparison (for the first time) of the application of these complex membership functions to a common architecture. Our results showed that Gaussian CFS showed superior accuracy over sinusoidal CFS or a combination of the two. Furthermore, the proposed architecture was more accurate and faster to train than existing literature and three other shallow

learning algorithms (MLP, RBFN, and SMOreg). Then, in the sixth chapter, we tried to extend the architecture proposed in the fifth chapter to support multivariate time series in the large scale as well. For this purpose, we examined SISO, MISO and MIMO designs for implementing our three proposed architectural variants. The results of our experiments on two large-scale datasets and one chaotic dataset indicated that the RCNFIS architecture is significantly superior to previous architectures as well as other well-known algorithms. The results of statistically significant tests given at the end of both chapters also emphasize this statement.

7.2. Future Work

A strong performance of the proposed architecture suggests that this could be a promising future development avenue. Further structural and algorithmic changes could be made to improve the accuracy of the results, as well as the speed of the learning algorithm (for example, by applying Evolutionary Algorithms), or the proposed architecture could be applied to other areas of machine learning. Time series forecasting was the main focus of this dissertation, but the architecture can also be applied to other areas such as classification, clustering, image processing, and data mining.

Moreover, according to what was discussed and reviewed in chapter II, the effect of different architectures and approaches on XAI and how they affect the perception of explainability was investigated. Finally, we investigated the effect of combining different machine learning algorithms such as evolutionary algorithms and deep neural networks with fuzzy logic. The conclusion was that by using linguistic variable structures, artificial intelligence systems based on fuzzy sets and logic act very transparently. This advantage is more evident for hybrid neuro-fuzzy and fuzzy-evolutionary systems. Thus, the necessity of designing dedicated explanatory interfaces (EIs) was emphasized. Although in our proposed models consider the

parameters of the layer-1 fuzzy set as adaptive parameters for learning, it is quite possible that the fuzzy sets in our trained models deviate from the "intuitive" meaning associated with the linguistic term that may be assigned to them. Therefore, in future work, we believe that EI design should apply to our RCNFIS model to improve the explanation (transparency) further and consequently increase user trust.

Additionally, regarding the last discussion we had in Chapter II, AI brittleness, we concluded that companies should have clear criteria and establish test protocols before purchasing or validating AI-enabled systems, especially on safety-critical systems, in order to ensure that these systems can operate effectively in their intended operational areas. Also, it was discussed how data/distribution shifts may significantly reduce model performance in real-world scenarios and may lead to brittleness in machine learning models. Therefore, as one of the future work and in order to avoid brittleness in our proposed RCNFIS model, we should generalize it in such a way that it adapt to conditions outside a limited set of assumptions. By using a method that increases the model's tolerance for dealing with data/distribution shifts that can affect input features, target variables, and their relationships, we can prevent brittleness from occurring in our model.

Bibliography

1. Zadeh, L.A., *Information and control.* Fuzzy sets, 1965. **8**(3): p. 338-353.
2. Ramot, D., et al., *Complex fuzzy sets.* IEEE Transactions on Fuzzy Systems, 2002. **10**(2): p. 171-186.
3. Chen, Z., et al., *ANCFIS: A neurofuzzy architecture employing complex fuzzy sets.* IEEE Transactions on Fuzzy Systems, 2010. **19**(2): p. 305-322.
4. Li, C. and T.-W. Chiang. *Complex neuro-fuzzy self-learning approach to function approximation*. in *Asian conference on intelligent information and database systems*. 2010. Springer.
5. Yazdanbakhsh, O. and S. Dick, *A systematic review of complex fuzzy sets and logic.* Fuzzy Sets and Systems, 2018. **338**: p. 1-22.
6. Sobhi, S., M. Reshadi, and S. Dick, *Condition Monitoring and Fault Detection of Small Induction Motors using Machine Learning Algorithms.* Submitted to the IEEE Access, 2022.
7. Ebadati, O. and M. Mortazavi, *An efficient hybrid machine learning method for time series stock market forecasting.* Neural Network World, 2018. **28**(1): p. 41-55.
8. Kumari, P., et al., *Autoregressive Integrated Moving Average (ARIMA) approach for prediction of rice (Oryza sativa l.) yield in India.* The Bioscan, 2014. **9**(3): p. 1063-1066.
9. Jang, J.-S., *ANFIS: adaptive-network-based fuzzy inference system.* IEEE transactions on systems, man, and cybernetics, 1993. **23**(3): p. 665-685.
10. Yazdanbakhsh, O. and S. Dick, *Forecasting of multivariate time series via complex fuzzy logic.* IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2017. **47**(8): p. 2160-2171.
11. Mills, T.C. and T.C. Mills, *Time series techniques for economists*. 1990: Cambridge University Press.

12. Chen, J.-F., W.-M. Wang, and C.-M. Huang, *Analysis of an adaptive time-series autoregressive moving-average (ARMA) model for short-term load forecasting.* Electric Power Systems Research, 1995. **34**(3): p. 187-196.

13. Che, J. and J. Wang, *Short-term electricity prices forecasting based on support vector regression and auto-regressive integrated moving average modeling.* Energy Conversion and Management, 2010. **51**(10): p. 1911-1917.

14. Elgar, E., *Handbook of Research Methods and Applications in Empirical Macroeconomics.* 2013.

15. Takens, F., *Detecting strange attractors in turbulence*, in *Dynamical systems and turbulence, Warwick 1980*. 1981, Springer. p. 366-381.

16. Jang, J.-S.R. *Neuro-fuzzy modeling for dynamic system identification*. in *Soft Computing in Intelligent Systems and Information Processing. Proceedings of the 1996 Asian Fuzzy Systems Symposium*. 1996. IEEE.

17. Klir, G.J. and B. Yuan, *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*. Vol. 6. 1996: World Scientific.

18. Wang, D. and L. Acar. *An analysis of type-1 and type-2 fuzzy logic systems*. in *Proceedings of the 1999 IEEE International Symposium on Intelligent Control Intelligent Systems and Semiotics (Cat. No. 99CH37014)*. 1999. IEEE.

19. Moses, D., et al. *Linguistic coordinate transformations for complex fuzzy sets*. in *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No. 99CH36315)*. 1999. IEEE.

20. Moses, D., et al., *Complex Membership Grades with an Application to the Design of Adaptive Filters.* Comput. Sci. J. Moldova, 1999. **7**(3): p. 253-283.

21. Nguyen, H.T., A. Kandel, and V. Kreinovich. *Complex fuzzy sets: towards new foundations*. in *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063)*. 2000. IEEE.

22. Ramot, D., et al., *Complex fuzzy logic.* IEEE Transactions on Fuzzy Systems, 2003. **11**(4): p. 450-461.

23. Dick, S., *Toward complex fuzzy logic.* IEEE Transactions on Fuzzy Systems, 2005. **13**(3): p. 405-414.

24. Zhang, G., et al., *Operation properties and δ-equalities of complex fuzzy sets.* International journal of approximate reasoning, 2009. **50**(8): p. 1227-1249.

25. Man, J., Z. Chen, and S. Dick. *Towards inductive learning of complex fuzzy inference systems*. in *NAFIPS 2007-2007 Annual Meeting of the North American Fuzzy Information Processing Society*. 2007. IEEE.

26. Armstrong, J.S., *Principles of forecasting: a handbook for researchers and practitioners*. Vol. 30. 2001: Springer.

27. Montgomery, D.C., C.L. Jennings, and M. Kulahci, *Introduction to time series analysis and forecasting*. 2015: John Wiley & Sons.

28. Song, H., *Review of Time Series Analysis and Its Applications With R Examples , by Robert H. Shumway & David S. Stoffer: New York, NY: Springer, 2011, 596 pp., $99.00 (hardcover)*. 2017, Taylor & Francis.

29. Majid, R. and S.A. Mir, *Advances in statistical forecasting methods: An overview.* Economic Affairs, 2018. **63**(4): p. 815-831.

30. Pal, A. and P. Prakash, *Practical time series analysis: master time series data processing, visualization, and modeling using python*. 2017: Packt Publishing Ltd.

31. Udny Yule, G., *On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers.* Philosophical Transactions of the Royal Society of London Series A, 1927. **226**: p. 267-298.

32. Slutzky, E., *The summation of random causes as the source of cyclic processes.* Econometrica: Journal of the Econometric Society, 1937: p. 105-146.

33. Wold, H., *A study in the analysis of stationary time series*. 1938, Almqvist & Wiksell.

34. Parzen, E., *ARARMA models for time series analysis and forecasting.* Journal of Forecasting, 1982. **1**(1): p. 67-82.

35. Mandal, B., *Forecasting sugarcane production in India with ARIMA model.* Inter Stat, October, 2005.

36. Nochai, R. and T. Nochai. *ARIMA model for forecasting oil palm price*. in *Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and applications*. 2006.

37. Paul, R., *Stochastic modeling of wholesale price of rohu in West Bengal, India.* 2010.

38. Burark, S., H. Sharma, and G. Meena, *Market integration and price volatility in domestic markets of coriander in Rajasthan.* Indian Journal of Agricultural Marketing, 2013. **27**(1): p. 121-131.

39. Paul, R.K., *Forecasting wholesale price of pigeon pea using long memory time-series models.* Agricultural Economics Research Review, 2014. **27**(2): p. 167-176.

40. Kim, S., et al., *Statistical model for forecasting uranium prices to estimate the nuclear fuel cycle cost.* Nuclear Engineering and Technology, 2017. **49**(5): p. 1063-1070.

41. Volterra, V., *Theory of functions and of integral and integro-differential equations", Dover Publications, New York.* 1930.

42. Tong, H. and K.S. Lim, *Threshold autoregression, limit cycles and cyclical data*, in *Exploration Of A Nonlinear World: An Appreciation of Howell Tong's Contributions to Statistics*. 2009, World Scientific. p. 9-56.

43. Box, G.E., *GM Jenkins Time Series Analysis: Forecasting and Control.* San Francisco, Holdan-Day, 1970.

44. Terasvirta, T. and H.M. Anderson, *Characterizing nonlinearities in business cycles using smooth transition autoregressive models.* Journal of applied econometrics, 1992. **7**(S1): p. S119-S136.

45. Zivot, E. and J. Wang, *Modeling financial time series with S-PLUS*. Vol. 2. 2006: Springer.

46. Engle, R.F., *Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation.* Econometrica: Journal of the econometric society, 1982: p. 987-1007.

47. Engle, R.F. and D. Kraft, *Multiperiod forecast error variances of inflation estimated from ARCH models.* Applied time series analysis of economic data, 1983: p. 293-302.

48. Bollerslev, T., *Generalized autoregressive conditional heteroskedasticity.* Journal of econometrics, 1986. **31**(3): p. 307-327.

49. Brooks, C.H., M.J., *Episodic non-stationarity in exchange rates.* Applied Economics Letters, 1998. **5**: p. 719-772.

50. Engle, R., *GARCH 101: The use of ARCH/GARCH models in applied econometrics.* Journal of economic perspectives, 2001. **15**(4): p. 157-168.

51. Poon, S.-H. and C.W. Granger, *Forecasting volatility in financial markets: A review.* Journal of economic literature, 2003. **41**(2): p. 478-539.

52. Wilhelmsson, A., *GARCH forecasting performance under different distribution assumptions.* Journal of Forecasting, 2006. **25**(8): p. 561-578.

53. Liu, H.-C., Y.-H. Lee, and M.-C. Lee, *Forecasting China stock markets volatility via GARCH models under skewed-GED distribution.* Journal of money, Investment and Banking, 2009. **7**(1): p. 542-547.

54. Lee, Y.-H. and T.-Y. Pai, *REIT volatility prediction for skew-GED distribution of the GARCH model.* Expert Systems with Applications, 2010. **37**(7): p. 4737-4741.

55. Lama, A., G.K. Jha, and R.K. Paul, *Modelling and forecasting of price volatility: An application of GARCH and EGARCH models.* 2015.

56. Schmidt, W.F., M.A. Kraaijveld, and R.P. Duin. *Feed forward neural networks with random weights*. in *International conference on pattern recognition*. 1992. IEEE Computer Society Press.

57. Igelnik, B. and Y.-H. Pao, *Stochastic choice of basis functions in adaptive function approximation and the functional-link net.* IEEE transactions on Neural Networks, 1995. **6**(6): p. 1320-1329.

58. Pao, Y.-H., G.-H. Park, and D.J. Sobajic, *Learning and generalization characteristics of the random vector functional-link net.* Neurocomputing, 1994. **6**(2): p. 163-180.

59. Pao, Y.-H. and Y. Takefuji, *Functional-link net computing: theory, system architecture, and functionalities.* Computer, 1992. **25**(5): p. 76-79.

60. Li, M. and D. Wang, *Insights into randomized algorithms for neural networks: Practical issues and common pitfalls.* Information Sciences, 2017. **382**: p. 170-178.

61. Gorban, A.N., et al., *Approximation with random bases: Pro et contra.* Information Sciences, 2016. **364**: p. 129-145.

62. Li, M., et al., *Improved randomized learning algorithms for imbalanced and noisy educational data classification.* Computing, 2019. **101**(6): p. 571-585.

63. Lancaster, P. and M. Tismenetsky, *The theory of matrices: with applications*. 1985: Elsevier.

64. Li, M., C. Huang, and D. Wang, *Robust stochastic configuration networks with maximum correntropy criterion for uncertain data regression.* Information Sciences, 2019. **473**: p. 73-86.

65. Wang, D. and M. Li, *Robust stochastic configuration networks with kernel density estimation for uncertain data regression.* Information Sciences, 2017. **412**: p. 210-222.

66. Wang, D. and M. Li, *Stochastic configuration networks: Fundamentals and algorithms.* IEEE transactions on cybernetics, 2017. **47**(10): p. 3466-3479.

67. Wang, D. and M. Li. *Deep stochastic configuration networks with universal approximation property*. in *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018. IEEE.

68. Wang, D. and C. Cui, *Stochastic configuration networks ensemble with heterogeneous features for large-scale data analytics.* Information Sciences, 2017. **417**: p. 55-71.

69. Li, M. and D. Wang, *2-D stochastic configuration networks for image data analytics.* IEEE Transactions on Cybernetics, 2019. **51**(1): p. 359-372.

70. Maass, W., *Liquid state machines: motivation, theory, and applications.* Computability in context: computation and logic in the real world, 2011: p. 275-296.

71. Zhang, Y., et al., *A digital liquid state machine with biologically inspired learning and its application to speech recognition.* IEEE transactions on neural networks and learning systems, 2015. **26**(11): p. 2635-2649.

72. Maass, W., T. Natschläger, and H. Markram, *Real-time computing without stable states: A new framework for neural computation based on perturbations.* Neural computation, 2002. **14**(11): p. 2531-2560.

73. Wang, Q., Y. Jin, and P. Li. *General-purpose LSM learning processor architecture and theoretically guided design space exploration*. in *2015 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 2015. IEEE.

74. Dutta, S., et al., *Leaky integrate and fire neuron by charge-discharge dynamics in floating-body MOSFET.* Scientific reports, 2017. **7**(1): p. 1-7.

75. Jin, Y. and P. Li, *Performance and robustness of bio-inspired digital liquid state machines: A case study of speech recognition.* Neurocomputing, 2017. **226**: p. 145-160.

76. Avesani, P., et al., *Non-parametric temporal modeling of the hemodynamic response function via a liquid state machine.* Neural Networks, 2015. **70**: p. 61-73.

77. Jaeger, H., *Echo state network.* scholarpedia, 2007. **2**(9): p. 2330.

78. Jang, J.S., *R., Sun, CT., and Mizutani, E.(1997): Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence.* Prentice Hall, Inc., Simon & Schuster/A Viacom Company, Upper Saddle River, NJ, 1997. **7458**: p. 23.

79. Zadeh, L.A., *Quantitative fuzzy semantics.* Information sciences, 1971. **3**(2): p. 159-176.

80. Castillo, O. and P. Melin, *Soft computing for control of non-linear dynamical systems*. Vol. 63. 2012: Physica.

81. Wang, L.-X., *Adaptive fuzzy systems and control. Prentice Hall.* Inglewood Cliffs, New Jersey, 1994.

82. Mamdani, E.H., *Application of fuzzy logic to approximate reasoning using linguistic synthesis.* IEEE transactions on computers, 1977. **26**(12): p. 1182-1191.

83. Mamdani, E.H. and S. Assilian, *An experiment in linguistic synthesis with a fuzzy logic controller.* International journal of man-machine studies, 1975. **7**(1): p. 1-13.

84. Takagi, T. and M. Sugeno, *Fuzzy identification of systems and its applications to modeling and control.* IEEE transactions on systems, man, and cybernetics, 1985(1): p. 116-132.

85. Sugeno, M. and G. Kang, *Structure identification of fuzzy model.* Fuzzy sets and systems, 1988. **28**(1): p. 15-33.

86. Nürnberger, A., D. Nauck, and R. Kruse, *Neuro-fuzzy control based on the NEFCON-model: recent developments.* Soft Computing, 1999. **2**(4): p. 168-182.

87. Lin, C.-T. and C.S.G. Lee, *Neural-network-based fuzzy logic control and decision system.* IEEE Transactions on computers, 1991. **40**(12): p. 1320-1336.

88. Sulzberger, S.M., N. Tschichold-Gurman, and S.J. Vestli. *FUN: Optimization of fuzzy rule based systems using neural networks*. in *IEEE International Conference on Neural Networks*. 1993. IEEE.

89. Jang, J.-S.R., C.-T. Sun, and E. Mizutani, *Neuro-fuzzy and soft computing-a computational approach to learning and machine intelligence [Book Review].* IEEE Transactions on automatic control, 1997. **42**(10): p. 1482-1484.

90. Jang, J.-S. and C.-T. Sun, *Neuro-fuzzy modeling and control.* Proceedings of the IEEE, 1995. **83**(3): p. 378-406.

91. Kaufman, A. and M.M. Gupta, *Introduction to fuzzy arithmetic*. 1991: Van Nostrand Reinhold Company New York.

92. Li, C. and T.-W. Chiang. *Complex fuzzy computing to time series prediction—A multi-swarm PSO learning approach*. in *Asian Conference on Intelligent Information and Database Systems*. 2011. Springer.

93. Hollander, M., D.A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. 2013: John Wiley & Sons.

94. Ribeiro, M.T., S. Singh, and C. Guestrin. " *Why should i trust you?" Explaining the predictions of any classifier*. in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016.

95. Caruana, R., et al. *Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission*. in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015.

96. Hoffman, R.R., et al., *Metrics for explainable AI: Challenges and prospects.* arXiv preprint arXiv:1812.04608, 2018.

97. Samek, W., et al., *Explainable AI: interpreting, explaining and visualizing deep learning*. Vol. 11700. 2019: Springer Nature.

98. Escalante, H.J., et al., *Explainable and interpretable models in computer vision and machine learning*. 2018: Springer.

99. Biran, O. and C. Cotton. *Explanation and justification in machine learning: A survey*. in *IJCAI-17 workshop on explainable AI (XAI)*. 2017.

100. Gunning, D., et al., *XAI—Explainable artificial intelligence.* Science robotics, 2019. **4**(37): p. eaay7120.

101. Fazzolari, M. and R. Alcala, *Nojima Yu., Ishibuchi H., Herrera F*. 2013, A review of the application of multiobjective evolutionary fuzzy systems ….

102. Haas Jr, J., *The Intelligibility of Nature: How Science Makes Sense of the World.* Perspectives on Science and Christian Faith, 2007. **59**(2): p. 154-156.

103. Lindsay, R.K., *Applications of artificial intelligence for organic chemistry: the DENDRAL project*. 1980: McGraw-Hill Companies.

104. Van Melle, W.J., *A domain-independent system that aids in constructing knowledge-based consultation programs*. 1980: Stanford University.

105. McCulloch, W.S. and W. Pitts, *A logical calculus of the ideas immanent in nervous activity.* The bulletin of mathematical biophysics, 1943. **5**(4): p. 115-133.

106. Nilsson, N.J., *The quest for artificial intelligence*. 2009: Cambridge University Press.

107. Minsky, M. and S.A. Papert, *Perceptrons, Reissue of the 1988 Expanded Edition with a new foreword by Léon Bottou: An Introduction to Computational Geometry*. 2017: MIT press.

108. Fukushima, K., *Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron.* IEICE Technical Report, A, 1979. **62**(10): p. 658-665.

109. LeCun, Y., *The MNIST database of handwritten digits.* http://yann. lecun. com/exdb/mnist/, 1998.

110. Wilamowski, B.M., Y. Chen, and A. Malinowski. *Efficient algorithm for training neural networks with one hidden layer*. in *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. 1999. IEEE.

111. Hopfield, J.J., *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the national academy of sciences, 1982. **79**(8): p. 2554-2558.

112. Simon, H., *Neural networks and machine learning*. 2009, Machine Industry Press, Beijing.

113. Johansson, E.M., F.U. Dowla, and D.M. Goodman, *Backpropagation learning for multilayer feed-forward neural networks using the conjugate gradient method.* International Journal of Neural Systems, 1991. **2**(04): p. 291-301.

114. Marquardt, D.W., *An algorithm for least-squares estimation of nonlinear parameters.* Journal of the society for Industrial and Applied Mathematics, 1963. **11**(2): p. 431-441.

115. Towell, G.G., *Symbolic knowledge and neural networks: Insertion, refinement and extraction.* 1993.

116. Hassibi, B. and D. Stork, *Second order derivatives for network pruning: Optimal brain surgeon.* Advances in neural information processing systems, 1992. **5**.

117. Broomhead, D.S., *Multivariable function interpolation and adaptive networks.* Complex Syst., 1988. **2**(3): p. 321-355.

118. Bell, A.J. and T.J. Sejnowski, *An information-maximization approach to blind separation and blind deconvolution.* Neural computation, 1995. **7**(6): p. 1129-1159.

119. Linsker, R., *Self-organization in a perceptual network.* Computer, 1988. **21**(3): p. 105-117.

120. Tesauro, G., *Temporal difference learning and TD-Gammon.* Communications of the ACM, 1995. **38**(3): p. 58-68.

121. Ackley, D.H., G.E. Hinton, and T.J. Sejnowski, *A learning algorithm for Boltzmann machines.* Cognitive science, 1985. **9**(1): p. 147-169.

122. Andrews, D., *Tickle, 1995 Andrews R., Diederich J., Tickle AB.* Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-Based Systems, 1995. **8**(6): p. 373-389.

123. Craven, M.W., *Extracting comprehensible models from trained neural networks*. 1996: The University of Wisconsin-Madison.

124. Thrun, S., *Extracting rules from artificial neural networks with distributed representations.* Advances in neural information processing systems, 1994. **7**.

125. Humphrey, M., S.J. Cunningham, and I.H. Witten, *Knowledge visualization techniques for machine learning.* Intelligent Data Analysis, 1998. **2**(1-4): p. 333-347.

126. Barakat, N. and A.P. Bradley, *Rule extraction from support vector machines: a review.* Neurocomputing, 2010. **74**(1-3): p. 178-190.

127. Lacave, C. and F.J. Díez, *A review of explanation methods for Bayesian networks.* The Knowledge Engineering Review, 2002. **17**(2): p. 107-127.

128. Druzdzel, M.J. and M. Henrion. *Using scenarios to explain probabilistic inference*. in *Working notes of the AAAI-90 Workshop on Explanation*. 1990.

129. Goyal, Y., et al. *Making the v in vqa matter: Elevating the role of image understanding in visual question answering*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

130. Goyal, Y., et al., *Towards transparent ai systems: Interpreting visual question answering models.* arXiv preprint arXiv:1608.08974, 2016.

131. Sadeghi, F., S.K. Kumar Divvala, and A. Farhadi. *Viske: Visual knowledge extraction and question answering by visual verification of relation phrases*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

132. Selvaraju, R.R., et al., *Grad-CAM: Why did you say that?* arXiv preprint arXiv:1611.07450, 2016.

133. Yosinski, J., et al., *Understanding neural networks through deep visualization.* arXiv preprint arXiv:1506.06579, 2015.

134. Zahavy, T., N. Ben-Zrihem, and S. Mannor. *Graying the black box: Understanding dqns*. in *International conference on machine learning*. 2016. PMLR.

135. Zeiler, M.D. and R. Fergus. *Visualizing and understanding convolutional networks*. in *European conference on computer vision*. 2014. Springer.

136. Gunning, D. and D. Aha, *DARPA's explainable artificial intelligence (XAI) program.* AI magazine, 2019. **40**(2): p. 44-58.

137. Glomsrud, J.A., et al. *Trustworthy versus explainable AI in autonomous vessels*. in *Proceedings of the International Seminar on Safety and Security of Autonomous Vessels (ISSAV) and European STAMP Workshop and Conference (ESWC)*. 2019.

138. Nothdurft, F., F. Richter, and W. Minker. *Probabilistic human-computer trust handling*. in *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*. 2014.

139. Bach, S., et al., *Jul. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation.* PLoS One. **10**(7).

140. Kim, B., et al. *Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)*. in *International conference on machine learning*. 2018. PMLR.

141. Ramon, Y., et al., *Metafeatures-based rule-extraction for classifiers on behavioral and textual data.* arXiv preprint arXiv:2003.04792, 2020.

142. Humphreys, P., *The philosophical novelty of computer simulation methods.* Synthese, 2009. **169**(3): p. 615-626.

143. Buckley, J.J. and H. Yoichi, *Neural nets for fuzzy systems.* Fuzzy Sets and Systems, 1995. **71**(3): p. 265-276.

144. Nauck, D. and R. Kruse, *Designing neuro-fuzzy systems through backpropagation*, in *Fuzzy Modelling*. 1996, Springer. p. 203-228.

145. Zadeh, L., R. Kalman, and N. DeClaris, *Aspects of network and system theory*. 1971, RE Kalman, N DeClaris, New York: Rinehart & Winston.

146. Zadeh, L.A., *A rationale for fuzzy control*, in *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*. 1996, World Scientific. p. 123-126.

147. Zadeh, L., *January 1973. Outline of a new approach to the analysis of complex systems and decision processes.* IEEE Transactions on Systems, Man and Cybernetics, SMC-3 (1): p. 28-44.

148. Zadeh, L.A., *On the analysis of large-scale systems*, in *Fuzzy Sets, Fuzzy Logic, And Fuzzy Systems: Selected Papers by Lotfi A Zadeh*. 1996, World Scientific. p. 195-209.

149. Zadeh, L.A., *Linguistic variables, approximate reasoning and dispositions.* Medical Informatics, 1983. **8**(3): p. 173-186.

150. Mucientes, M., et al., *Learning weighted linguistic rules to control an autonomous robot.* International Journal of Intelligent Systems, 2009. **24**(3): p. 226-251.

151. Babuška, R., *Fuzzy systems, modeling and identification.* Delft University of Technology, Department of Electrical Engineering Control Laboratory, Mekelweg, 1996. **4**.

152. Lee, S.C. and E.T. Lee, *Fuzzy sets and neural networks.* 1974.

153. Keller, J.M. and D.J. Hunt, *Incorporating fuzzy membership functions into the perceptron algorithm.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1985(6): p. 693-699.

154. Wang, L.-X. and J.M. Mendel, *Generating fuzzy rules by learning from examples.* IEEE Transactions on systems, man, and cybernetics, 1992. **22**(6): p. 1414-1427.

155. 高木英行. *Fusion technology of fuzzy theory and neural networks: Survey and future directions*. in *Proceedings of International Conference on Fuzzy Logic & Neural Networks, Iizuka'90*. 1990. Fuzzy Logic Systems Institute.

156. Hayashi, Y., J.J. Buckley, and E. Czogala, *Fuzzy neural network with fuzzy signals and weights.* International Journal of Intelligent Systems, 1993. **8**(4): p. 527-537.

157. Pal, S.K. and S. Mitra, *Multilayer perceptron, fuzzy sets, classifiaction.* 1992.

158. Klir, G. and B. Yuan, *Fuzzy sets and fuzzy logic*. Vol. 4. 1995: Prentice hall New Jersey.

159. Valenzuela-Rendón, M. *The fuzzy classifier system: Motivations and first results*. in *International Conference on Parallel Problem Solving from Nature*. 1990. Springer.

160. Thrift, P.R. *Fuzzy Logic Synthesis with Genetic Algorithms*. in *ICGA*. 1991.

161. Pham, D. and D. Karaboga, *Optimum design of fuzzy logic controllers using genetic algorithms.* Journal of Systems Engineering, 1991. **1**(2): p. 114-118.

162. Sarabakha, A. and E. Kayacan, *Online deep fuzzy learning for control of nonlinear systems using expert knowledge.* IEEE Transactions on Fuzzy Systems, 2019. **28**(7): p. 1492-1503.

163. Chen, C., et al., *Fuzzy restricted Boltzmann machine and its fast learning algorithm.* IEEE Trans. Fuzzy Syst., 2015. **23**(6): p. 2163-2173.

164. Zhang, S., et al., *Deep fuzzy echo state networks for machinery fault diagnosis.* IEEE Transactions on Fuzzy Systems, 2019. **28**(7): p. 1205-1218.

165. Feng, Q., et al., *Deep fuzzy clustering—a representation learning approach.* IEEE Transactions on Fuzzy Systems, 2020. **28**(7): p. 1420-1433.

166. Hurtik, P., V. Molek, and J. Hula, *Data preprocessing technique for neural networks based on image represented by a fuzzy function.* IEEE Transactions on Fuzzy Systems, 2019. **28**(7): p. 1195-1204.

167. Guan, C., S. Wang, and A.W.-C. Liew, *Lip image segmentation based on a fuzzy convolutional neural network.* IEEE Transactions on Fuzzy Systems, 2019. **28**(7): p. 1242-1251.

168. Wang, Y., et al., *Deep fuzzy tree for large-scale hierarchical visual classification.* IEEE Transactions on Fuzzy Systems, 2019. **28**(7): p. 1395-1406.

169. Aviles, A.I., et al. *A deep-neuro-fuzzy approach for estimating the interaction forces in robotic surgery*. in *2016 IEEE international conference on fuzzy systems (FUZZ-IEEE)*. 2016. IEEE.

170. Risen, T., *Tesla Updates Radar in Wake of Autonomous Car Crashes.* US News & World Report, 2016.

171. Crowe, S., *Tesla Autopilot Causes 2 More Accidents.* Robotics Trends, 2016.

172. Griggs, T. and D. Wakabayashi, *How a self-driving Uber killed a pedestrian in Arizona.* The New York Times, 2018. **13**.

173. Bubbers, M., *Don't hold your breath-fully autonomous cars are still decades away.* The Globe and Mail, 2019. **27**.

174. Elias, J., *Alphabet exec says self-driving cars 'have gone through a lot of hype,'but Google helped drive that hy.* CNBC, accessed, 2019. **22**.

175. Walch, K., *Are we heading for another AI winter soon? Forbes*. 2019.

176. Zhu, Z., et al., *Deep learning for autonomous vehicle and pedestrian interaction safety.* Safety science, 2022. **145**: p. 105479.

177. Gupta, R.K., et al., *COVID-19 Lesion Segmentation and Classification of Lung CTs Using GMM-Based Hidden Markov Random Field and ResNet 18.* International Journal of Fuzzy System Applications (IJFSA), 2022. **11**(2): p. 1-21.

178. Harwell, D., *The accent gap. The Washington Post*. 2018.

179. Marathe, A., et al., *In Rain or Shine: Understanding and Overcoming Dataset Bias for Improving Robustness Against Weather Corruptions for Autonomous Vehicles.* arXiv preprint arXiv:2204.01062, 2022.

180. Cummings, M., *The Surprising Brittleness of AI.* Women Corporate Directors, 2020.

181. Lewis, R., *Reality is going to stall for some time the advent of driverless cars.* The Washington Post. https://www. washingtonpost. com/realestate/reality-is-going-to-stallfor-some-time-the-advent-of-driverless-cars/2019/08/01/343c9458-afa8-11e9-a0c9-6d2d7818f3da_story. html, 2019.

182. Shorten, C. and T.M. Khoshgoftaar, *A survey on image data augmentation for deep learning.* Journal of big data, 2019. **6**(1): p. 1-48.

183. Dijk, T.v. and G.d. Croon. *How do neural networks see depth in single images?* in *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

184. Hernández-Castro, C.J., et al., *Adversarial machine learning*, in *Security and Artificial Intelligence*. 2022, Springer. p. 287-312.

185. Eykholt, K., et al. *Robust physical-world attacks on deep learning visual classification*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

186. Su, J., D.V. Vargas, and K. Sakurai, *One pixel attack for fooling deep neural networks.* IEEE Transactions on Evolutionary Computation, 2019. **23**(5): p. 828-841.

187. Quinonero-Candela, J., et al., *Dataset shift in machine learning*. 2008: Mit Press.

188. Gretton, A., et al., *Covariate shift by kernel mean matching.* Dataset shift in machine learning, 2009. **3**(4): p. 5.

189. Trinh, T.Q., et al. *Tackling covariate shift with node-based Bayesian neural networks*. in *International Conference on Machine Learning*. 2022. PMLR.

190. Tasche, D., *Minimising quantifier variance under prior probability shift.* arXiv preprint arXiv:2107.08209, 2021.

191. Biswas, A. and S. Mukherjee. *Ensuring fairness under prior probability shifts*. in *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 2021.

192. Jain, M., G. Kaur, and V. Saxena, *A K-Means clustering and SVM based hybrid concept drift detection technique for network anomaly detection.* Expert Systems with Applications, 2022. **193**: p. 116510.

193. Koh, P.W., et al. *Wilds: A benchmark of in-the-wild distribution shifts*. in *International Conference on Machine Learning*. 2021. PMLR.

194. Zech, J.R., et al., *Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: a cross-sectional study.* PLoS medicine, 2018. **15**(11): p. e1002683.

195. Beery, S., G. Van Horn, and P. Perona. *Recognition in terra incognita*. in *Proceedings of the European conference on computer vision (ECCV)*. 2018.

196. Li, H. and Y. Guan, *Leopard: fast decoding cell type-specific transcription factor binding landscape at single-nucleotide resolution.* bioRxiv, 2019: p. 856823.

197. Jean, N., et al., *Combining satellite imagery and machine learning to predict poverty.* Science, 2016. **353**(6301): p. 790-794.

198. Buolamwini, J. and T. Gebru. *Gender shades: Intersectional accuracy disparities in commercial gender classification*. in *Conference on fairness, accountability and transparency*. 2018. PMLR.

199. Geirhos, R., et al., *Shortcut learning in deep neural networks.* Nature Machine Intelligence, 2020. **2**(11): p. 665-673.

200. Arjovsky, M., et al., *Invariant risk minimization.* arXiv preprint arXiv:1907.02893, 2019.

201. Li, D., et al. *Deeper, broader and artier domain generalization*. in *Proceedings of the IEEE international conference on computer vision*. 2017.

202. Beery, S., et al., *The iwildcam 2021 competition dataset.* arXiv preprint arXiv:2105.03494, 2021.

203. Bandi, P., et al., *From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge.* IEEE transactions on medical imaging, 2018. **38**(2): p. 550-560.

204. Wu, M., et al., *Variational item response theory: Fast, accurate, and expressive.* arXiv preprint arXiv:2002.00276, 2020.

205. Hu, W., et al., *Open graph benchmark: Datasets for machine learning on graphs.* Advances in neural information processing systems, 2020. **33**: p. 22118-22133.

206. Taylor, J., et al. *Rxrx1: An image set for cellular morphological variation across many experimental batches*. in *International Conference on Learning Representations (ICLR)*. 2019.

207. David, E., et al., *Global Wheat Head Detection (GWHD) dataset: a large and diverse dataset of high-resolution RGB-labelled images to develop and benchmark wheat head detection methods.* Plant Phenomics, 2020. **2020**.

208. Borkan, D., et al. *Nuanced metrics for measuring unintended bias with real data for text classification*. in *Companion proceedings of the 2019 world wide web conference*. 2019.

209. Christie, G., et al. *Functional map of the world*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.

210. Yeh, C., et al., *Using publicly available satellite imagery and deep learning to understand economic well-being in Africa.* Nature communications, 2020. **11**(1): p. 1-11.

211. Ni, J., J. Li, and J. McAuley. *Justifying recommendations using distantly-labeled reviews and fine-grained aspects*. in *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 2019.

212. Raychev, V., P. Bielik, and M. Vechev, *Probabilistic model for code with decision trees.* ACM SIGPLAN Notices, 2016. **51**(10): p. 731-747.

213. Lu, S., et al., *Codexglue: A machine learning benchmark dataset for code understanding and generation.* arXiv preprint arXiv:2102.04664, 2021.

214. Yazdanbakhsh, O. and S. Dick. *ANCFIS-ELM: A machine learning algorithm based on complex fuzzy sets*. in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. 2016. IEEE.

215. Yazdanbakhsh, O. and S. Dick, *Induction of complex fuzzy inferential systems via randomized learning.* IEEE Transactions on Fuzzy Systems, Submitted, 2016.

216. Yazdanbakhsh, O. and S. Dick, *FANCFIS: Fast adaptive neuro-complex fuzzy inference system.* International Journal of Approximate Reasoning, 2019. **105**: p. 417-430.

217. Huang, G.-B., Q.-Y. Zhu, and C.-K. Siew, *Extreme learning machine: theory and applications.* Neurocomputing, 2006. **70**(1-3): p. 489-501.

218. Zhang, L. and P.N. Suganthan, *A survey of randomized algorithms for training neural networks.* Information Sciences, 2016. **364**: p. 146-155.

219. Stull, R.B., *An introduction to boundary layer meteorology*. Vol. 13. 1988: Springer Science & Business Media.

220. Li, C. and T.-W. Chiang, *Function approximation with complex neuro-fuzzy system using complex fuzzy sets–a new approach.* New generation computing, 2011. **29**(3): p. 261-276.

221. Li, C. and F.-T. Chan. *Knowledge discovery by an intelligent approach using complex fuzzy sets*. in *Asian Conference on Intelligent Information and Database Systems*. 2012. Springer.

222. Li, C., T.-W. Chiang, and L.-C. Yeh, *A novel self-organizing complex neuro-fuzzy approach to the problem of time series forecasting.* Neurocomputing, 2013. **99**: p. 467-476.

223. Sun, H., S. Wang, and Q. Jiang, *FCM-based model selection algorithms for determining the number of clusters.* Pattern recognition, 2004. **37**(10): p. 2027-2037.

224. Kim, K. and A.G. Parlos, *Induction motor fault diagnosis based on neuropredictors and wavelet signal processing.* IEEE/ASME Transactions on mechatronics, 2002. **7**(2): p. 201-219.

225. Jawadekar, A., et al., *Artificial neural network-based induction motor fault classifier using continuous wavelet transform.* Systems Science & Control Engineering: An Open Access Journal, 2014. **2**(1): p. 684-690.

226. Bonaldi, E.L., et al., *Predictive maintenance by electrical signature analysis to induction motors*, in *Induction Motors-Modelling and Control*. 2012, IntechOpen.

227. Lin, F., et al., *Fault diagnosis of power components in electric vehicles.* Journal of Asian Electric Vehicles, 2013. **11**(2): p. 1659-1666.

228. Gupta, K. and A. Kaur, *A review on fault diagnosis of induction motor using artificial neural networks.* International journal of science and research, 2014. **3**(7): p. 680-684.

229. Carden, E.P. and P. Fanning, *Vibration based condition monitoring: a review.* Structural health monitoring, 2004. **3**(4): p. 355-377.

230. Joseph, B. and C.B. Brosilow, *Inferential control of processes: Part I. steady state analysis and design.* AIChE Journal, 1978. **24**(3): p. 485-492.

231. Jutan, A., J. Wright, and J. MacGregor, *Multivariable computer control of a butane hydrogenolysis reactor: Part II: Design and on-line implementation of a stochastic controller using an identified multivariate noise model.* AIChE journal, 1984. **30**(2): p. 220-226.

232. Qin, S.J., H. Yue, and R. Dunia, *Self-validating inferential sensors with application to air emission monitoring.* Industrial & engineering chemistry research, 1997. **36**(5): p. 1675-1685.

233. Booth, C. and J.R. McDonald, *The use of artificial neural networks for condition monitoring of electrical power transformers.* Neurocomputing, 1998. **23**(1-3): p. 97-109.

234. Kamohara, H., et al., *Product quality estimation and operating condition monitoring for industrial ethylene fractionator.* Journal of chemical engineering of Japan, 2004. **37**(3): p. 422-428.

235. Lamberson, R.E., *Apparatus and method for the remote monitoring of machine condition*. 1998, Google Patents.

236. Daroogheh, N., et al., *Prognosis and health monitoring of nonlinear systems using a hybrid scheme through integration of PFs and neural networks.* IEEE Transactions on Systems, Man, and Cybernetics: Systems, 2016. **47**(8): p. 1990-2004.

237. Henriquez, P., et al., *Review of automatic fault diagnosis systems using audio and vibration signals.* IEEE Transactions on systems, man, and cybernetics: Systems, 2013. **44**(5): p. 642-652.

238. Wu, S.-j., et al., *A neural network integrated decision support system for condition-based optimal predictive maintenance policy.* IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2007. **37**(2): p. 226-236.

239. Nandi, S., H.A. Toliyat, and X. Li, *Condition monitoring and fault diagnosis of electrical motors—A review.* IEEE transactions on energy conversion, 2005. **20**(4): p. 719-729.

240. Trachi, Y., et al., *Induction machines fault detection based on subspace spectral estimation.* IEEE Transactions on Industrial Electronics, 2016. **63**(9): p. 5641-5651.

241. Sun, J., et al., *BLDC motor speed control system fault diagnosis based on LRGF neural network and adaptive lifting scheme.* Applied Soft Computing, 2014. **14**: p. 609-622.

242. Ince, T., et al., *Real-time motor fault detection by 1-D convolutional neural networks.* IEEE Transactions on Industrial Electronics, 2016. **63**(11): p. 7067-7075.

243. Li, K. and Q. Wang. *Study on signal recognition and diagnosis for spacecraft based on deep learning method*. in *2015 Prognostics and System Health Management Conference (PHM)*. 2015. IEEE.

244. Reddy, K.K., et al. *Anomaly detection and fault disambiguation in large flight data: A multi-modal deep auto-encoder approach*. in *Annual Conference of the PHM Society*. 2016.

245. Sun, W., et al., *A sparse auto-encoder-based deep neural network approach for induction motor faults classification.* Measurement, 2016. **89**: p. 171-178.

246. Hill, D.J. and B.S. Minsker, *Anomaly detection in streaming environmental sensor data: A data-driven modeling approach.* Environmental Modelling & Software, 2010. **25**(9): p. 1014-1022.

247. Kantz, H. and T. Schreiber, *Nonlinear time series analysis*. Vol. 7. 2004: Cambridge university press.

248. Hegger, R., H. Kantz, and T. Schreiber, *Practical implementation of nonlinear time series methods: The TISEAN package.* Chaos: An Interdisciplinary Journal of Nonlinear Science, 1999. **9**(2): p. 413-435.

249. Abarbanel, H., *Analysis of observed chaotic data*. 2012: Springer Science & Business Media.

250. Kennel, M.B., R. Brown, and H.D. Abarbanel, *Determining embedding dimension for phase-space reconstruction using a geometrical construction.* Physical review A, 1992. **45**(6): p. 3403.

251. Haykin, S.S., *Neural networks and learning machines/Simon Haykin*. 2009: New York: Prentice Hall.

252. Witten, I.H., et al., *Weka: Practical machine learning tools and techniques with Java implementations.* 1999.

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

254. Dimov, R., et al., *Weka: Practical machine learning tools and techniques with java implementations.* AI Tools SeminarUniversity of Saarland, WS, 2007. **6**(07).

255. Breiman, L., et al., *Classification and regression trees*. 2017: Routledge.

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

257. Pedregosa, F., et al., *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research, 2011. **12**: p. 2825-2830.

258. Staff. *Lowry Range Solar Station (LRSS)*. 2013; Available from: http://www.nrel.gov/midc/lrss/.

259. Gruber, M.H., *Regression estimators: A comparative study*. 2010: JHU Press.

260. Hsia, T., *On multistage least squares approach to systems identification.* IFAC Proceedings Volumes, 1975. **8**(1): p. 288-293.

261. Palmer, T., *Edward Norton Lorenz. 23 May 1917—16 April 2008*. 2009, The Royal Society.

262. Weeks, D.E.R. *Chaotic Time Series Analysis*. 2021; Available from: http://www.physics.emory.edu/faculty/weeks//research/tseries1.html.

263. Bellini, A., et al. *Simplified model of a photovoltaic module*. in *2009 Applied Electronics*. 2009. IEEE.

264. Yazdanbaksh, O., A. Krahn, and S. Dick. *Predicting solar power output using complex fuzzy logic*. in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*. 2013. IEEE.

265. Yazdanbakhsh, O. and S. Dick, *Time-series forecasting via complex fuzzy logic*, in *Frontiers of higher order fuzzy sets*. 2015, Springer. p. 147-165.

266. TISEAN. *Nonlinear Time Series Analysis*. 2021; Available from: http://www.mpipksdresden.mpg.de/~tisean/.

267. Rojas, I., et al., *Time series analysis using normalized PG-RBF network with regression weights.* Neurocomputing, 2002. **42**(1-4): p. 267-285.

268. Rojas, I., et al., *Soft-computing techniques and ARMA model for time series prediction.* Neurocomputing, 2008. **71**(4-6): p. 519-537.

269. Abdulkadir, S.J. and S.-P. Yong. *Lorenz time-series analysis using a scaled hybrid model*. in *2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC)*. 2015. IEEE.

270. Wasserstein, R.L. and N.A. Lazar, *The ASA statement on p-values: context, process, and purpose*. 2016, Taylor & Francis. p. 129-133.

271. Greenland, S., et al., *Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations.* European journal of epidemiology, 2016. **31**(4): p. 337-350.

272. Hsu, J., *Multiple comparisons: theory and methods*. 1996: CRC Press.

273. Yazdanbakhsh, O. and S. Dick. *Multi-variate timeseries forecasting using complex fuzzy logic*. in *2015 Annual Conference of the North American Fuzzy Information Processing Society (NAFIPS) held jointly with 2015 5th World Conference on Soft Computing (WConSC)*. 2015. IEEE.

274. Rössler, O.E., *An equation for continuous chaos.* Physics Letters A, 1976. **57**(5): p. 397-398.

275. Rossler, O., *An equation for hyperchaos.* Physics Letters A, 1979. **71**(2-3): p. 155-157.

276. Fonollosa, J., et al., *Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring.* Sensors and Actuators B: Chemical, 2015. **215**: p. 618-629.

277. Fonollosa, J. and R. Huerta, *Gas sensor array under dynamic gas mixtures data set*. 2015, UCI Machine Learning Repository, http://archive. ics. uci. edu/ml/datasets ….

278. Sobhi, S. and S. Dick, *An Investigation of Complex Fuzzy Sets for Large-Scale Learning.* Submitted to Transactions on Fuzzy Systems, 2022.

279. Su, F. *The chaos theory and its application*. in *Journal of Physics: Conference Series*. 2021. IOP Publishing.

280. Ghys, É. *The butterfly effect*. in *The Proceedings of the 12th International Congress on Mathematical Education*. 2015. Springer, Cham.

281. Lorenz, E.N., *Deterministic nonperiodic flow.* Journal of atmospheric sciences, 1963. **20**(2): p. 130-141.

282. Ivancevic, V.G. and T.T. Ivancevic, *Complex nonlinearity: chaos, phase transitions, topology change and path integrals*. 2008: Springer Science & Business Media.

283. Zalta, E.N., *Metaphysics Research Lab.* Center for the Study of Language and Information, Stanford University, Stanford, CA, 2015.

284. Albulescu, C.T., A.K. Tiwari, and P. Kyophilavong, *Nonlinearities and chaos: A new analysis of CEE stock markets.* Mathematics, 2021. **9**(7): p. 707.

285. Safonov, L.A., et al., *Multifractal chaotic attractors in a system of delay-differential equations modeling road traffic.* Chaos: An Interdisciplinary Journal of Nonlinear Science, 2002. **12**(4): p. 1006-1014.

286. Peitgen, H.-O., et al., *Chaos and fractals: new frontiers of science*. Vol. 7. 1992: Springer.

287. Lainscsek, C., et al., *Non-linear dynamical classification of short time series of the Rössler system in high noise regimes.* Frontiers in Neurology, 2013. **4**: p. 182.

288. Gao, J., *Data preprocessing.* 2012.

289. Ilyas, I.F. and X. Chu, *Data cleaning*. 2019: Morgan & Claypool.

290. García, S., J. Luengo, and F. Herrera, *Data preprocessing in data mining*. Vol. 72. 2015: Springer.

291. Dutka, A.F. and H.H. Hansen, *Fundamentals of data normalization*. 1991: Addison-Wesley Longman Publishing Co., Inc.

292. Manikandan, S., *Data transformation.* Journal of Pharmacology and Pharmacotherapeutics, 2010. **1**(2): p. 126.

293. Barandas, M., et al., *TSFEL: Time series feature extraction library.* SoftwareX, 2020. **11**: p. 100456.

294. Patro, S. and K.K. Sahu, *Normalization: A preprocessing stage.* arXiv preprint arXiv:1503.06462, 2015.

295. Weigend, A.S., *Time series prediction: forecasting the future and understanding the past*. 2018: Routledge.

296. Pan, S. and K. Duraisamy, *On the structure of time-delay embedding in linear models of non-linear dynamical systems.* Chaos: An Interdisciplinary Journal of Nonlinear Science, 2020. **30**(7): p. 073135.

297. Cao, L., A. Mees, and K. Judd, *Dynamics from multivariate time series.* Physica D: Nonlinear Phenomena, 1998. **121**(1-2): p. 75-88.

298. Boccaletti, S., et al., *Reconstructing embedding spaces of coupled dynamical systems from multivariate data.* Physical Review E, 2002. **65**(3): p. 035204.

299. Su, L.-y., *Prediction of multivariate chaotic time series with local polynomial fitting.* Computers & Mathematics with Applications, 2010. **59**(2): p. 737-744.

300. Terziyska, M., Z. Terziyski, and Y. Todorov. *A Long-Short Term Memory Network for Chaotic Time Series Prediction*. in *2021 Big Data, Knowledge and Control Systems Engineering (BdKCSE)*. 2021. IEEE.

301. Canayaz, M., *Training anfis system with moth-flame optimization algorithm.* International Journal of Intelligent Systems and Applications in Engineering, 2019. **7**(3): p. 133-144.

302. Lukoseviciute, K. and M. Ragulskis, *Evolutionary algorithms for the selection of time lags for time series forecasting by fuzzy inference systems.* Neurocomputing, 2010. **73**(10-12): p. 2077-2088.

303. Kamaruddin, S. and V. Ravi, *EGRNN++ and PNN++: Parallel and Distributed Neural Networks for Big Data Regression and Classification.* SN Computer Science, 2021. **2**(2): p. 1-19.

304. Koesdwiady, A., A. El Khatib, and F. Karray. *Methods to improve multi-step time series prediction*. in *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018. IEEE.

305. Lin, T., T. Guo, and K. Aberer, *Hybrid neural networks over time series for trend forecasting.* 2017.