INVESTIGATIONS INTO LEARNING ALGORITHMS IN INTELLIGENT MACHINES

A THESIS TO BE SUBMITTED TO
THE UNIVERSITY OF TRANS-DISCIPLINARY HEALTH SCIENCES AND TECHNOLOGY

THE UNIVERSITY OF TRANS-DISCIPLINARY
HEALTH SCIENCES & TECHNOLOGY

FOR THE AWARD OF THE DEGREE OF
DOCTOR OF PHILOSOPHY

BY

HARIKRISHNAN N. B.

UNDER THE GUIDANCE OF

DR. NITHIN NAGARAJ
ASSOCIATE PROFESSOR
CONSCIOUSNESS STUDIES PROGRAMME
SCHOOL OF HUMANITIES

NATIONAL INSTITUTE OF ADVANCED STUDIES
INDIAN INSTITUTE OF SCIENCE CAMPUS, 560012

JULY 2022

THE UNIVERSITY OF TRANS-DISCIPLINARY HEALTH SCIENCES AND TECHNOLOGY

Private University Established in Karnataka by ACT 35 of 2013
BENGALURU-560064

DECLARATION BY THE CANDIDATE

I declare that this thesis entitled "**Investigations into Learning Algorithms in Intelligent Machines**" submitted for the award of Doctor of Philosophy to THE UNIVERSITY OF TRANS-DISCIPLINARY HEALTH SCIENCES AND TECHNOLOGY, Bengaluru, is my original work, conducted under the supervision of my guide **Dr. Nithin Nagaraj**. I also wish to inform that no part of the research has been submitted for a degree or examination at any university. References, help and material obtained from other sources have been duly acknowledged.

I hereby confirm the originality of the work and that there is no plagiarism in any part of the dissertation.

Place: Bengaluru **Signature of the Candidate**
Date:

Name of the Candidate: Harikrishnan N. B.
Reg. No.: 21218011230
Date of Reg.: 06 December 2018

THE UNIVERSITY OF TRANS-DISCIPLINARY HEALTH SCIENCES AND TECHNOLOGY

Private University Established in Karnataka by ACT 35 of 2013
BENGALURU-560064

CERTIFICATE

This is to certify that the work incorporated in this thesis "**Investigations into Learning Algorithms in Intelligent Machines**" submitted by Mr. Harikrishnan N.B. was carried out under my supervision. No part of this thesis has been submitted for a degree or examination at any university. References, help and material obtained from other sources have been duly acknowledged. I hereby confirm the originality of the work and that there is no plagiarism in any part of the dissertation.

Research Supervisor: Dr. Nithin Nagaraj
Date:
Associate Professor
National Institute of Advanced Studies
Bengaluru

Acknowledgement

First and foremost, I express my sincere gratitude and thankfulness to my PhD supervisor, Dr. Nithin Nagaraj. I first met Dr. Nithin Nagaraj in 2013 during my B-Tech. Interaction with him during my B-Tech, helped me to question my assumptions about the world, people, my interests and gave me the confidence to pursue my interest. In 2018, I got the opportunity to do my PhD under his guidance. This was one of the best moments in my life. I thank him for accepting me as his PhD student and helping me grow both as a researcher and as a person. I will always cherish the time he has given me and discussions over long walks in IISc, JP Park etc. The success of my PhD dissertation would not have been possible without the guidance and continuous support of Dr. Nithin Nagaraj.

I express my thankfulness to Prof. Sangeetha Menon, for giving me an opportunity to be part of Consciousness Studies Programme (CSP) at National Institute of Advanced Studies (NIAS). The programs conceptualized by Prof. Sangeetha Menon such as Reading Glass, and other workshops helped me to develop my leadership skills and other competencies such as public speaking, research discussion etc. I am also very much grateful to 'The Tata Trusts CSP Project of the NIAS Consciousness Studies Programme' for funding my research for four years. My thanks to the 'The University of Trans-Discplinary Health Sciences and Technology' (TDU) for supporting my research as part of the PhD programme. I thank Mr. Ravikumar G, Assistant Registrar, TDU for navigating me through the academic formalities of TDU.

The constant interactions with my Doctoral Advisory Committee (DAC) members: Prof. L.M Patnaik, Prof. Sangeetha Menon, Dr. Snehanshu Saha and Dr. Nithin Nagaraj has helped me to critically examine my research work. Their valuable comments have helped me to improve my work.

I express my thankfulness to our lab mates: Dr. Aditi Kathpalia, Dr. Pranay SY, Rampriya Darshini, Anand Ganesh for the wonderful and insightful discussions. I thank the undergraduate research interns: Deeksha Sethi and Darshan Bharadwaj, together we were able to do some exciting work. The work I did with Deeksha Sethi

has contributed to many results in Chapter 5 of my dissertation.

My thanks to NIAS Director: Prof. Shailesh Nayak, Late Prof. B.V. Sreekantan (founder of NIAS CSP). I would also like to thank the entire administrative team at NIAS, especially Mr. Srinivasa Aithal (Head Administrator at NIAS); library staff: Ms. Hamsa and Ms. Vijayalakshmi, for helping me use the library resources; mess staff members at NIAS. I am also grateful to IISc for letting me avail their various courses and workshops.

I express my thankfulness to the NIAS CSP team members particularly Dr. Aditi, Rakesh, Amrutha, Pushya, Sukanya, Ashwini, and Dr. Shankar. My interactions with all of them have helped me open to learn new fields. I also express my thankfulness to my friends outside NIAS: Manoj, Harikumar, Nagesh, Murali, Akif Khan, Megha, Anuja V, Rahul V, and Dr. KR Sahasranand.

I am grateful to my previous mentors who were instrumental in pursuit of my higher studies: Dr. KP Soman, Dr. Sundararaman Gopalan, Gayathri Narayanan, Shriram Nandakumar, Dr. Vinayakumar R. I thank my parents (Dr. N S Balakrishnan Nair and S Ushakumari), brother (Dr. N Anand Balu) and sister-in-law (Anju V Nair) for the continuous support, unbounded love, and honest criticisms.

I dedicate my dissertation to my late grandfather Kunjunni Pillai P.

List of Tables

List of Figures

List of Acronyms

AI	Artificial Intelligence
NL	Neurochaos Learning
ML	Machine Learning
GLS	Generalized Lüroth Series
DL	Deep Learning
LL	Lifelong Learning
NLL	Neurochaos Lifelong Learning
NFL	No Free Lunch Theorem
CFX	Chaos based features
ANN	Artificial Neural Network
DNN	Deep Neural Network
SR	Stochastic Resonance
ρ	Cross Correlation Coefficient
CCD	Concentric Circle Data
OCCD	Overlapping Concentric Circle Data
FSDD	Free Spoken Digit Dataset
DT	Decision Tree
RF	Random Forest
AB	AdaBoost
SVM	Support Vector Machine
k-NN	k-Nearest Neighbors
GNB	Gaussian Naive Bayes
q	Initial Neural Activity
b	Discrimination Threshold
ϵ	Epsilon
UAT	Universal Approximation Theorem
GC	Granger Causality
CCC	Compression-Complexity Causality

Table of Contents

Synopsis

Alan Turing, a famous mathematician, computer scientist, and a World War II code breaker asked a profound question "Can Machines Think?" in his 1950 paper titled "Computing Machinery and Intelligence" published in the journal *Mind*. In this work, Turing designed a test to determine a machine's ability to exhibit intelligent behaviour. This test is popularly known as the Turing Test. This foundational work of Turing has inspired researchers to delve deeper into the notion of intelligence. The term Artificial Intelligence (AI) was first coined by John McCarthy who was the organiser of the 1956 Dartmouth Summer Research Project on Artificial Intelligence. Since then, there have been several methods developed to solve real-world problems intelligently by reducing human interventions. But none of them have succeeded in developing a replica of human intelligence. Today, we realise the enormous complexity involved in defining the term intelligence. At present, the field of AI can be seen as a unifying theme of a diverse set of methods/algorithms. These algorithms can be broadly classified as (1) Symbolic AI (logic based systems to replicate rational thought in humans), (2) Machine Learning (algorithms that learn from data), and (3) Sub-symbolic AI (biology/brain inspired learning). From 2010 onwards, researchers showcased the unreasonable effectiveness of brain inspired learning algorithms like deep neural networks, recurrent neural networks, convolutional neural networks etc. Despite their tremendous applications, there is a wide research gap between Artificial Neural Networks (ANNs) and Biological Neural Networks (BNNs). The neurons in BNNs are intrinsically nonlinear and exhibit a wide variety of firing patterns. There is empirical evidence of Deterministic Chaos (random-like behaviour from a deterministic nonlinear system with sensitive dependence on initial conditions) at different spatiotemporal scales in the brain. On the other hand, neurons in ANNs perform only a simple affine transformation followed by nonlinear activation. Also interestingly, the phenomena of stochastic resonance or noise enhanced signal processing is found in the brain. None of these properties are incorporated in current learning algorithms.

In this thesis, we address these research gaps by proposing a novel brain inspired

learning algorithm namely "Neurochaos Learning" (NL). NL is comprised of an input layer of chaotic 1D Generalized Lüroth Series (GLS) neurons. NL fundamentally uses the Topological Transitivity property of chaos and Stochastic Resonance to perform classification tasks. NL has mainly two architectures: (a) ChaosNet (an input layer of 1D GLS neurons followed by cosine similarity classifier), and (b) hybrid architecture: chaos based features + Machine Learning classifiers (features extracted from the chaotic neural trace followed by Machine Learning classifiers such as Decision Tree, Random Forest, AdaBoost, Support Vector Machine, k-Nearest Neighbours, Gaussian Naive Bayes etc.). We demonstrate the following rich properties of NL in this thesis: (1) NL satisfies the Universal Approximation Theorem, (2) NL supports the incorporation of chaotic biological neuronal models such as Hindmarsh-Rose neuronal model, (3) superior performance in the limited training sample regime when compared to ML algorithms (with training using just nine samples per class, NL gives an F1-score in the range [0.6, 0.98]), (4) the flexibility of NL allows for developing hybrid NL-ML algorithms to boost the performance of existing Machine Learning algorithms, (5) robustness to additive parametric noise, (6) exhibits stochastic resonance at the level of individual as well as layer of neurons, (7) in the context of continual learning, the rate of catastrophic forgetting in NL is much less compared to Deep Learning algorithms, and (8) the features extracted from the input layer of NL preserves the inherent causal structure in the time series data.

The proposed method has been rigorously tested on various datasets: *Iris, Ionosphere, Wine, Bank Note Authentication, Haberman's Survival, Breast Cancer Wisconsin, Statlog (Heart), Seeds, Free Spoken Digit Dataset, MNIST*. In the former, five out of nine datasets have shown a performance boost in terms of macro F1-score after using CFX features (features extracted from the input layer of NL). The highest performance boost obtained is 25.97% for *Statlog (Heart)* dataset using CFX+Decision Tree. In the low training sample regime (from just one to nine training samples per class), the highest performance boost of 144.38% is obtained for *Haberman's Survival* dataset using CFX+Random Forest.

Chapter 1

Introduction

This chapter provides a brief introduction to the field of Artificial Intelligence and the different approaches used in the field. Following that, the important research gaps in the field of brain inspired learning are highlighted. A brief synopsis of each chapter of the thesis are provided.

1.1 Artificial Intelligence

The quest for understanding the intricacies of nature is a timeless pursuit. Curious minds have always been fascinated by the intelligence that pervades this universe. From time to time philosophers, scientists, and poets attempted to unveil different facets of intelligence. In the mid of 1950s, scientist took an approach of creating intelligent machines in order to understand intelligence. The motivation for this comes from Alan Turing's phenomenal work titled as "Computing Machinery and Intelligence" [1] published in 1950 in the journal *Mind*. In this seminal 1950 paper, Alan Turing asked a profound question-*Can Machines Think?* He provided an operational definition of machine intelligence - *The Imitation Game* [1] (also popularly known as Turing Test). Even though the Turing Test was limited by anthropocentric notion of intelligence [2], it was an initial direction towards building intelligent systems. However, the acceleration in research to build intelligent machines was after the 1956 workshop organised by John McCarthy at Dartmouth college. John McCarthy was

the first to coin the term '*Artificial Intelligence*' (AI). The 1956 workshop brought divergent views on AI and approaches to creating intelligent machines. Mathematicians promoted the development of logic and deductive reasoning as the basis of rational thought. Statisticians stressed on extracting meaningful statistics from data and the usage of probabilistic methods for decision making under uncertainty. Other researchers suggested to emulate the properties of the brain and take inspiration from biology to harness intelligence. Interestingly, this resulted in different definitions for AI based on the approach one took. This also suggests our understanding of terms like 'intelligence', 'cognition', 'consciousness' etc. are very limited. Hence, the 2014 survey paper by Joel Lehman and Jeff Clune summarised AI as an '*An Anarchy of Methods*' [3]. This definition considers AI as a field which incorporates a wide range of methods to develop intelligent machines.

1.2 Anarchy of Methods in AI

The anarchy of methods in AI can be categorized as follows [4]:

1. Symbolic AI
2. Machine Learning
3. Sub-symbolic AI

Symbolic AI argued that symbol processing programs are the basis of developing general intelligence. General Problem Solver is one of the early symbolic AI programs that manipulates symbols in order to deduce logical reasoning. Machine Learning (ML) on the other hand was inspired from statistics and probability theory, with the goal of learning from data (data driven learning). ML has its own set of algorithms independent of Symbolic AI. Some of the popular ML algorithms are: k-Nearest Neighbours [5], Decision Tree [6], Random Forest [7], Support Vector Machine [8], Naive Bayes [9], and AdaBoost [10]. Sub-symbolic AI also learns from data. However, the motivation of Sub-symbolic AI is to emulate the properties of the brain. Sub-

symbolic AI has evolved over various stages. We briefly discuss the evolution of methods in Sub-symbolic AI.

The contribution of Donald O. Hebb, who is regarded as the father of neural networks, laid the foundation of neural networks by proposing Hebbian learning rule in the work titled "The Organization of Behavior: A Neuropsychological Theory" [11]. Even though there were limitations to Hebbian learning rule, it was a great beginning for research in neural networks. The other popular model of a primitive neural network was introduced by neurophysiologist Warren McCulloch and logician Walter Pitts [12]. Frank Rosenblatt further extended Hebbian Learning rule and McCulloch-Pitts model and introduced the "perceptron" [13]. Perceptron performs a linear decision by linearly combining the inputs with weights. With a single linear decision function NOT, AND, OR logical operators can be perfectly represented but not XOR which requires two linear decision boundaries. This limitation was famously pointed out by Minski and Papert [14]. With the introduction of Multilayer Perceptron (MLP) obtained by stacking perceptron side-ways and layer-ways, the problem of XOR was solved. One unique property of MLP is Universal Approximation Theorem which means that it can approximate any function and in this way the problem of representing XOR was solved. But in order to represent the function, the parameters (weights) of the neural network has to be updated while training. An efficient way to compute the weights is by the celebrated Back Propagation algorithm [15] which employs a gradient descent method [16]. Some of the other major breakthroughs were in the field of vision where Convolutional Neural Networks were widely used [17]. Recurrent Neural Networks (RNNs) [18] and its variants such as Long Short Term Memory (LSTM) [19] and Gated Recurrent Unit (GRU) [20], which are designed specifically to handle sequence data, are used in a number of applications such as sentiment analysis [21, 22], speech recognition [23], named entity recognition [24]. With the increase in computational capacity, the area of Deep Learning (DL) has witnessed significant progress in the last decade. Despite the tremendous applications of DL algorithms, Sub-symbolic AI methods are only loosely inspired from the brain. The research gaps are highlighted in the following section.

1.3 Research gap in Sub-symbolic AI

The core motivation of Sub-symbolic AI is to emulate the properties of human brain. Human brain which has a size of approximately 1500 cm^3 is a highly nonlinear system [25]. It has been estimated that the human brain has approximately 86 billion neurons [26] which interact with each other forming a complex network. Neurons are inherently nonlinear and found to exhibit chaos [27, 28]. An interesting counter-intuitive property of networks of neurons in the brain is their ability to learn in the presence of enormous amount of noise [29] and neural interference. Inspired by the biological brain, researchers have developed Artificial Intelligent systems which use learning algorithms such as DL and ML that loosely mimic the human brain.

DL and ML algorithms have a wide variety of practical applications in computer vision, natural language processing, speech processing [23], cyber-security [30], medical diagnosis [31] etc. However, these algorithms do not use the essential properties of human brain. One such property of brain is the presence of chaotic neurons [27, 28]. Even though Artificial Neural Networks are biologically inspired, none of its varied architectures have neurons which exhibit chaos though it has been shown that certain type of neural networks exhibit chaotic dynamics (for e.g., in Recurrent Neural Networks [32]). Chaotic regimes exhibit a wide range of behaviors and are beneficial for the brain to quickly adapt to changing conditions [28]. This is a major research gap in Sub-symbolic AI where the inspiration is to emulate the properties of the brain. Additionally, the study conducted by [33] gives experimental evidence supporting the presence of stochastic resonance in neuronal networks in brain. Stochastic resonance [34] refers to the counter-intuitive phenomenon of improved system's response for specific input signals in the presence of noise as compared to performance in the absence of noise. Another interesting property of the brain is its ability to learn from limited training samples. But this is not the case with current DL algorithms as they need enormous quantity of data to train. Learning from limited training samples comes under the regime of Few Shot Learning [35], which is a major research area in the current times. This PhD dissertation attempts to bridge the above mentioned

research gaps.

1.4 Research Objectives of this Dissertation

The research objectives of this dissertation are the following:

1. **Brain Inspired Learning**: To propose a novel brain-inspired learning algorithm that uses chaos and stochastic resonance at the level of individual neurons for classification.

2. **Learning from Limited Training Samples**: To boost the performance of Machine Learning (ML) algorithms in the low training sample regime by incorporating chaos based feature transformation.

3. **Cause-Effect Preservation**: To support the preservation of the inherent cause-effect structure in the data under chaotic feature transformation of the proposed brain inspired learning algorithm.

4. **Applications**: To rigorously test the proposed brain inspired learning algorithm for both simulated and real world datasets pertaining to medical diagnosis, banknote fraud detection, environmental applications, spoken-digit classification and others.

1.5 Structure of PhD Thesis

The thesis is organised in seven chapters. A brief synopsis of each chapter is provided below.

- **Chapter 1**- Chapter 1 (this chapter) provides a brief introduction to the field of Artificial Intelligence and the different approaches used in the field. Following that the important research gaps in the field of brain inspired learning are highlighted. A brief synopsis of each chapter of the thesis are provided.

- **Chapter 2**- This chapter provides a brief introduction to chaos theory and the research methodology followed in this thesis. A basic introduction to chaos theory is covered in this chapter. This involves a brief introduction to the history of chaos theory, mathematical definitions of deterministic chaos, chaos in $1D$ maps, symbolic sequence and its applications. The machine learning research methodology is addressed next. This involves a brief explanation regarding the principled way of doing machine learning research. The chapter also discusses the performance metrics used in this thesis to evaluate the machine learning models.

- **Chapter 3**- In this chapter, the foundational research objective of this dissertation is addressed - *To propose a brain inspired learning algorithm that uses chaos and stochastic resonance for classification.* The proposed algorithm is named as Neurochaos Learning (NL). NL has two architectures: ChaosNet and ChaosFEX (CFX) + Machine Learning (ML). ChaosNet is built using layers of neurons, each of which is a $1D$ chaotic map known as the Generalized Lüroth Series (GLS). GLS maps have been shown in earlier works to possess very useful properties for compression, cryptography and for computing XOR and other logical operations. CFX+ML is a hybrid NL-ML architecture combining Neurochaos features with ML classifiers. In this chapter, the mathematical foundations of ChaosNet and CFX + ML are described in detail.

- **Chapter 4-** This chapter provides a theoretical justification to the working of NL. The interplay of chaos and noise plays a crucial role in the working of NL. Inspired by the chaotic firing of neurons and the constructive role of noise in neuronal models, this chapter connects chaos, noise and learning for the first time. In this chapter, the Stochastic Resonance (SR) phenomenon in NL is demonstrated. SR manifests at the level of a single neuron of NL and enables efficient subthreshold signal detection. Furthermore, SR is shown to occur in single and multiple neuronal NL architectures for classification tasks - both on simulated and real-world spoken digit datasets, and in architectures with $1D$ chaotic maps

as well as in biologically plausible Hindmarsh Rose spiking neurons. Intermediate levels of noise in NL enables peak performance in classification tasks thus highlighting the role of SR in Artificial Intelligence applications, especially in brain inspired learning architectures.

- **Chapter 5**- This chapter focuses on the experiments and comparative study of NL with classical ML techniques. In this chapter, the efficacy of neurochaos feature transformation and extraction for classification are rigorously studied by testing ChaosNet and CFX+ML on various real-world datasets. The explored datasets in this study revolve around medical diagnosis, banknote fraud detection, environmental applications, spoken-digit classification and others.

- **Chapter 6**- This chapter investigates the wonderful properties of NL and summarises them. These include: (a) an input layer of GLS neurons in NL satisfies the Universal Approximation Theorem, (b) NL supports the incorporation of chaotic biological neuronal models, (c) superior performance in the limited training sample regime when compared to ML algorithms, (d) the flexibility of NL allows for developing hybrid NL-ML algorithms, (e) robustness to additive parametric noise, (f) exhibits stochastic resonance at the level of individual as well as layer of neurons, (g) the rate of catastrophic forgetting in NL is less compared to DL algorithms, and (h) the features extracted from the input layer of NL preserves cause-effect relationships.

- **Chapter 7**- The chapter provides a summary of the contribution of the thesis, limitations and future research directions.

Chapter 2

Foundations of Chaos Theory and Machine Learning Research Methodology

This chapter provides a brief introduction to chaos theory and the machine learning research methodology followed in this dissertation. The chapter contains a brief introduction to the history of chaos theory, mathematical definition of deterministic chaos, Generalized Lüorth Series - $1D$ piece-wise linear maps, and definition of a symbolic sequence. Throughout the research, the supervised machine learning research methodology is followed in conducting the computer experiments.

2.1 Introduction

The development of calculus and laws of classical mechanics by Sir Issac Newton in the 17^{th} century had a profound impact on the way nature was perceived and understood. Newtonian mechanics eased the modelling of dynamics of bodies with simple equations. Soon scientists started seeing the world through the lens of Newtonian mechanics. The 18^{th} and 19^{th} century science celebrated Newtonian mechanics. The French physicist Pierre-Simon, marquis de Laplace was one of those who was profoundly influenced by the Newtonian mechanics. His philosophical position on

deterministic universe can be attributed to the influence of Newtonian Mechanics. His famous quotation on determinism (referred as 'Laplace's Demon) is as follows:

"*We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at any given moment knew all of the forces that animate nature and the mutual positions of the beings that compose it, if this intellect were vast enough to submit the data to analysis, could condense into a single formula the movement of the greatest bodies of the universe and that of the lightest atom; for such an intellect nothing could be uncertain and the future just like the past would be present before its eyes." — Pierre-Simon, marquis de Laplace*

According to Laplace's world view, the universe is completely deterministic, there is no uncertainty and unpredictability. However, over the end of 19^{th} century, things took a turn. This was because of the birth of *Chaos Theory*. The birth of chaos theory goes back to Mittag-Leffler's[1] idea of organizing a mathematics international prize competition honouring the 60th birthday of King Oscar II of Sweden and Norway. The competition was announced in 1885 and winning prize consisted of a gold medal and 2,500 Swedish kronor. Out of the four questions, the first question attracted most attention.

"*For a system of arbitrarily many mass points that attract each other according to Newton's law, assuming that no two points ever collide, find a series expansion of the coordinates of each point in known functions of time converging uniformly for any period of time.*"

This question is the classical N-body problem and is related to the motion of celestial bodies. In other words, is our solar system stable? In 1890, Henri Poincaré, a polymath and french mathematician, in his 290 pages final memoir appeared in *Acta Mathematica* Volume 13, proved that the problem is insolvable because of chaotic behaviour [36].

This was the birth of chaos theory, however the term 'chaos' had to wait another 85 years to be coined. This was after the discovery of *sensitive dependence on initial conditions* by Edward Lorenz in 1961 [37]. Edward Lorenz designed a deterministic

[1]http://www.mittag-leffler.se/library/henri-poincare

nonlinear three dimensional weather model [38], where he showed two close-by initial conditions indicating the present weather conditions will quickly diverge in finite amount of time. This makes long term weather prediction an impossible task. This sensitive dependence on initial conditions making the long-term prediction impossible is popularly known as the Butterfly Effect - a butterfly flapping its wings in Bengaluru leading to unpredictable changes in Delhi weather. Later in 1975, James A. Yorke named this phenomena as *chaos theory*. This initiated numerous theoretical [39] and application [40] oriented research on chaos theory. In short, chaos theory revolutionized the 20^{th} century science.

Chaos theory is a vast field with various applications [40–42], where many ideas are conglomerated. For the purpose of this thesis, we will focus on the conceptual understanding of chaos. We shall also provide mathematical justifications wherever required. The same holds true for Machine Learning. A preliminary introduction to chaos theory and machine learning sufficient for understanding this thesis is provided in this chapter.

2.2 Dynamical System

A system is one that takes an input and manipulates it using definite rules so as to produce a response. A dynamical system is a deterministic system[2] that evolves with respect to time with a given fixed initial condition(s) and specific rule(s) [43]. A dynamical system is defined by mainly two factors:

1. Initial condition(s) of the system.

2. The rules that govern the dynamics of the dynamical system.

Based on the rules, the dynamical system can be classified as follows:

1. Linear Dynamical System

2. Nonlinear Dynamical System

[2]deterministic system is a system where there is no randomness or stochasticity involved in the governing equation(s) of the system.

A dynamical system that satisfies the properties of homogeneity and superposition is said to be a linear dynamical system [44]. On the other hand, nonlinear dynamical systems do not satisfies these properties. Chaos is exhibited by some nonlinear dynamical system. It is important to note that all nonlinear dynamical systems are not chaotic where as all system that exhibits chaos are nonlinear. Hence, it is important to understand what makes certain nonlinear systems special inorder to exhibit chaos. For this, we shall delve into the mathematical definition of chaos.

2.3 Chaos

A dynamical system D (D is $T : X \to X$, where T is a map from set X onto itself.) is chaotic [45] if it satisfies the following:

1. The periodic points in D are dense[3].
2. D is topologically transitive[4].
3. D is sensitive to initial conditions[5].

The consequence of these properties are the following:

- A chaotic system is nonlinear and bounded.
- A chaotic system has countably infinite number of periodic orbits[6], uncountably infinite number of nonperiodic orbits.
- The trajectories of a chaotic system looks/ feels like random, but has rich structure and order.

[3]Definition: A subset P is dense in set Q, if for any element $q \in Q$, there exists an element p in the subset P arbitrarily close to q.

[4]Definition: The topological transitivity property of a dynamical system is defined as: if for each two elements $x_1, y_1 \in \Sigma$ and for $\epsilon_1 > 0$, there exist a $z_1 \in \Sigma$ such that on finite number of iterations, z_1 reaches the ϵ_1 neighbourhood of x_1 and y_1.

[5]Definition: A dynamical system D is said to be sensitively dependent on initial values/conditions, if there exist a $\beta_1 > 0$ such that for any $x_1 \in D$ and $\epsilon_1 > 0$, there exist $y_1 \in D$ within the ϵ_1 of x_1 and there exist $k \in \mathbf{Z}^+$ such that $d(T^k(x_1), T^k(y_1)) \geq \beta$.

[6]The orbit/trajectory of x_1 under the map T is the set of points $\{x_1, T(x_1), T^2(x_1), ...\}$. The initial value of the orbit is the starting point x_1.

- A chaotic system is sensitively dependent on initial condition(s). A slight change in the initial condition(s) gives a completely different divergent trajectory. This is the reason for unpredictability even though the system we are dealing with is deterministic.

2.4 Generalized Lüroth Series

The chaotic dynamical system used in this research is the Generalized Lüroth Series (GLS). GLS [46] is a $1D$ piece-wise linear map that satisfies the properties of chaos. Tent map, Binary map and their skewed cousins are well known examples of GLS. Mathematically, the types of GLS maps we use in this research are described below.

2.4.1 GLS map: $T_{Skew-Tent}(x)$ and $T_{Skew-Binary}(x)$

A map $T_{Skew-Binary} : [0,1) \rightarrow [0,1)$ is defined as:

$$T_{Skew-Binary}(x) = \begin{cases} \frac{x}{b} & , \quad 0 \leq x < b, \\ \frac{(x-b)}{(1-b)} & , \quad b \leq x < 1, \end{cases}$$

where $x \in [0,1)$ and $0 < b < 1$. Refer to Figure 3.1(a).

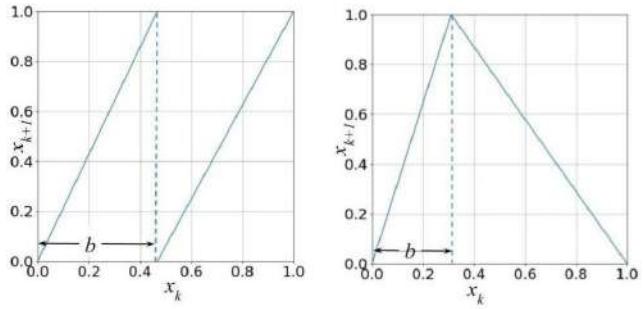
A map $T_{Skew-Tent} : [0,1) \rightarrow [0,1)$ is defined as:

$$T_{Skew-Tent}(x) = \begin{cases} \frac{x}{b} & , \quad 0 \leq x < b, \\ \frac{(1-x)}{(1-b)} & , \quad b \leq x < 1, \end{cases}$$

where $x \in [0,1)$ and $0 < b < 1$. Refer to Figure 3.1(b).

Let $X = \{x_0, x_1, x_2, \ldots\}$ be the orbit/trajectory of a chaotic map with initial condition x_0, where $x_i \in [U, V)$. The interval $[U, V)$ is partitioned into k sub intervals denoted as $I_0, I_1, \ldots, I_{k-1}$. If $x_i \in I_j$ then we denote x_i by the symbol $j \in \{0, 1, \ldots, k-1\}$. The new sequence of symbol $\{j_0, j_1, \ldots, j_{k-1}\}$ is the symbolic sequence of the trajectory of X. In the case of Figure 2.1, the symbols **L (or 0)** and **R (or 1)** are associated with the intervals $[0, b)$ and $[b, 1)$ respectively. We enumerate some salient properties of the GLS map:

1. Each GLS map has two parameters - an initial value x_0 and the skew value



(a) (b)

Figure 2.1: Generalized Lüroth Series, GLS map, is fundamentally of two types: (a) Left: Skew-Binary map $T_{Skew-Binary}$. (b) Right: Skew-Tent map $T_{Skew-Tent}$.

b. The parameter b defines the *generating Markov partition*[7] for the map and also acts as *internal discrimination threshold* of the map which will be used for feature extraction.

2. GLS map can fire either chaotically or in a periodic fashion (of any finite period length) depending on the initial activity value x_0. The degree of chaos is controlled by the skew parameter b. The lyapunov exponent of the map [48] is given by $\lambda_b = -b\ln(b) - (1-b)\ln(1-b)$. If the base of the logarithm is chosen as 2, then $\lambda_b = H(S_{x_0})$ (bits/iteration) where S_{x_0} is the symbolic sequence of the trajectory obtained by iterating the initial neural activity x_0 and $H(\cdot)$ is Shannon Entropy of symbolic sequence in bits/symbol. For $0 < b < 1$, $\lambda_b > 0$.

3. Any finite length input stream of bits can be *losslessly* compressed as an initial value x_0 on the GLS map by performing a backward iteration on the map. Further, such a lossless compression scheme has been previously shown to be *Shannon optimal* [49].

4. Error detection properties can be incorporated into GLS map to counter the effect of noise [50].

[7]Generating Markov Partition or GMP [47] is based on splitting the state space into a complete set of disjoint regions, namely, it covers all state space and enables associating a one-to-one correspondence between trajectories and itinerary sequences of symbols (L and R) without losing any information.

5. Owing to its excellent chaotic and ergodic (mixing) properties, GLS (and other related $1D$ maps) has been employed in cryptography [48, 49, 51].

6. GLS maps has been used to develop compression-based neural architecture for memory encoding and decoding [52].

7. GLS map can compute logical operations such as XOR, AND, NOT, OR etc. by switching between appropriate maps as described in a previous work [48, 52].

8. GLS map has the property of *topological transitivity* - defined in section 2.3

9. A single layer of a finite number of GLS maps satisfies a version of the *Universal Approximation Theorem* which we shall prove in chapter 3.

2.5 Machine Learning Framework

Machine Learning (ML) is a sub-field of Artificial Intelligence as already described in chapter 1. The ML goal is to develop learning algorithms that can learn from data. There are mainly two kinds of problems modeled using ML framework. They are (a) Classification, and (b) Regression. The goal of classification problems is to develop classification algorithms that can categorize a data instance to its respective class. A classification problem can be binary as well as multi-class depending on the number of pre-determined classes in the dataset. For eg., consider a binary classification problem, where class-0 represents the various images of cats and class-1 represents various images of dogs. In this examples the class labels are 0 and 1. The goal here is - given a test image of a cat or a dog, the classification algorithm has to correctly categorize the test image as belonging to class-0 or class-1. Since there are only two classes in this example, this problem is a binary classification problem. Regression can also be seen as a classification problem. In the case of regression, the number of classes are infinite. The data instances are mapped to real numbers. Where as in the case of classification tasks, the data instances are mapped to discrete numbers (label). In this thesis, the focus is on classification tasks.

2.5.1 Machine Learning Research Methodology

Throughout this research, we use the supervised ML research methodology. The supervised ML research framework has the following ingredients [53]:

- **Domain set**: Domain set $\mathcal{X} = \{X_1, X_2, \ldots, X_m\}$, consists of a set of data instances (X_i) which we wish to label. The domain set can be images, timeseries, text etc.

- **Label set**: Label set is the set of all possible labels. For eg, if we are considering a k-class classification problem, the label set $\mathcal{Y} = \{0, 1, 2, \ldots, k-1\}$.

- **Training Data**: Training data is a finite sequence of pairs in $\mathcal{X} \times \mathcal{Y}$. The training data $D = \{(X_1, y_1), (X_2, y_2), \ldots (X_n, y_n)\}$, where $X_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$.

- **Classification or Learning Algorithm**: The goal of the classification algorithm is to find an appropriate *function/ predictor/ hypothesis/ classifier/ model/ algorithm* h that maps $\mathcal{X}$ to $\mathcal{Y}$ ($h : \mathcal{X} \to \mathcal{Y}$) using the training data. This is known as learning.

- **Cross-validation to find hyperparameters**: Every algorithm has a set of hyperparameter(s)[8] to be tuned. The appropriate values for the hyperparameters has to be found using k-fold cross-validation. In k-fold crossvalidation, the training data is split into k folds. For each value of the hyperparameter(s), we train the algorithm k times independently. Each time, we use $k-1$ folds for training and predict the label for the remaining validation fold. The average performance (refer section 2.5.3 for the performance metric of the validation fold is found for each value of the hyperparameter(s). We choose that value of the

[8]These are configurations which are external to the model and typically not estimated from the data. The hyperparameters are tuned for a given classification or predictive modelling task in order to estimate the best model parameters. The hyperparameter(s) of a model are often arrived by heuristics and hence are different for different tasks. In the case of K-Nearest Neighbours (KNN) classification algorithm, the *number of nearest neighbours* (K) is a hyperparameter. In SVM, the *choice of kernel* is a hyperparameter. In Decision Tree, *depth of the tree* and the *least number of samples required to split the internal node* are the hyperparameters. In all these cases, the hyperparameters need to be fixed by cross-validation.

hyperparameter(s) which gives the best average performance for the validation folds in the k-fold cross-validation experiments.

- **Error Quantification**: The efficacy of the classifier is evaluated based on how well the classifier is able to correctly classify the data instances. There are several classification metrics to evaluate the performance of the classifier which are discussed in section 2.5.3.

- **Testing Data**: The goal of the ML classifier is to accurately predict the labels of new data (not visited by the algorithm while training) whose labels are unknown. This unseen new data whose labels we need to predict is the testdata.

The next important question is to ask "What is the best ML algorithm?" The No Free Lunch (NFL) theorem [54] in ML states that there is no perfect algorithm that works for all data. Every algorithm has some inherent assumptions and it is important to know the assumptions and the properties of the dataset being used. Each algorithm has a set of hyperparameters which has to be tuned. The principled way of doing supervised ML is as follows:

1. As a consequence of NFL theorem, there is no perfect ML algorithm that works for all datasets. This ensures one to have competing models to learn from data.

2. The goal of the classification task is to develop a classification algorithm that learns from the train data and predict the labels of testdata with high accuracy.

3. Use k-fold cross-validation for hyperparameter tuning. For each value of the hyperparamter, we do a k-fold crossvalidation on the traindata. The average performance of the classifier for the k-fold crossvalidation are found corresponding to each value of the hyperparameter. The hyperparameter corresponding to the best performance in the k-fold crossvalidation experiment is chosen for training.

4. Using the best hyperparameter the algorithm is retrained from scratch. The outcome of training is an ML model whose parameters[9] are learned from data.

5. The ML model with learnt parameters is tested once on testdata.

6. The performance of the ML model is evaluated using standard classification performance metric (refer to section 2.5.3).

2.5.2 Algorithms

Some of the popular ML algorithms are: AdaBoost [10], Random Forest [7], Support Vector Machine [8], Decision Tree [6], Gaussian Naive Bayes [9] and k-Nearest Neighbours [5]. We are not going into the details of each algorithm, which is outside the scope of this thesis. Readers may refer to standard textbooks used in the ML community [55, 56].

2.5.3 Evaluation Metrics

In this section, the performance metrics used in this dissertation to evaluate the performance of the ML classifiers are provided. We consider a binary classification problem where the labels are either positive or negative. The classifier predicts the outcome of the data instance. The predicted outcome can be either equal to actual outcome or not. Based on this, we have the following four cases[10]:

1. True Pos: The predicted and actual outcomes are both positive.

2. True Neg: The predicted and actual outcomes are both negative.

3. False Pos: The predicted outcome is positive and the actual outcome is negative.

4. False Neg: The predicted outcome is negative and the actual outcome is positive.

[9]This is an internal parameter which is estimated from the data. These internal parameters are what the model learns while training.

[10]Pos: Positive, Neg: Negative, ACC: Accuracy, Prec: Precision, Rec: Recall

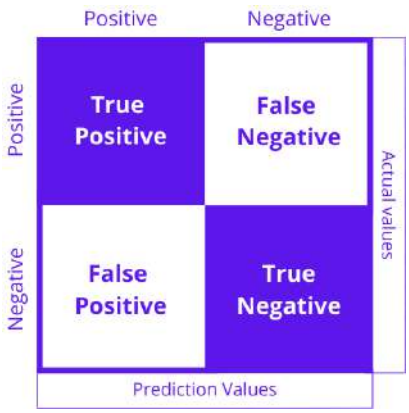


Figure 2.2: Pictorial representation of a confusion matrix corresponding to a binary classification problem.

The confusion of the classifier in classifying the data instances are recorded by True Pos, False Pos, True Neg, and False Neg. They are represented in the form of a matrix named as '*Confusion Matrix*' as provided in Figure 2.2. Using the confusion matrix, we derive the following performance measures for evaluation of the ML model.

The most common performance metric used in ML is *Accuracy* [57]. Accuracy is defined as follows:

$$ACC = \frac{True\ Neg + True\ Pos}{True\ Neg + True\ Pos + False\ Pos + False\ Neg}. \tag{2.1}$$

In the case of imbalanced datasets, this metric will lead to misinterpretation of results. To overcome this, the performance metric used for all experiments in this study is *Macro F1-Score*. This metric is computed from the confusion matrix. Figure 2.2 shows a confusion matrix for a binary classification problem. Prediction values represent the predictions made by the classifier. Actual values stand for the true/ target values for the predictions being made by the classifier. They both have two categories: Pos and Neg.

F1-Score depends on two parameters:

1. Prec - The ratio of the number of positives correctly classified as positive by the algorithm to the total number of instances classified as positive, given by -

$$Prec = \frac{True\ Pos}{False\ Pos + True\ Pos}. \tag{2.2}$$

2. Rec - The ratio of the number of positives correctly classified as positive by the algorithm to the total number of actual positives, given by -

$$Rec = \frac{True\ Pos}{False\ Neg + True\ Pos}. \tag{2.3}$$

Thus, F1-Score which is the harmonic mean of Prec and Rec is given by,

$$F1 = F1Score = \frac{2 \cdot Rec \cdot Prec}{Rec + Prec}. \tag{2.4}$$

Macro F1-Score is obtained by averaging the F1 scores for all classes in the dataset, given by

$$Macro\ F1Score = \frac{F1_{Class1} + F1_{Class2} + \ldots + F1_{Classn}}{n}, \tag{2.5}$$

where, n stands for the number of distinct classes in the dataset. F1-score of $Class_k$ considers the instances of $Class_k$ as positive and all remaining instances of classes as negative. Thus, the task is transformed to a binary classification problem. All instances of $Class_k$ correctly classified as positive or negative by the classifier are termed as true positives or true negatives respectively. All instances that are incorrectly classified by the classifier are placed under the false positive and false negative categories accordingly. Hence the associated equations of precision and recall for this example are given by,

$$Prec_{Class_k}(P_k) = \frac{True\ Pos_{Class_k}}{False\ Pos_{Class_k} + True\ Pos_{Class_k}}, \tag{2.6}$$

$$Rec_{Class_k}(R_k) = \frac{True\ Pos_{Class_k}}{False\ Neg_{Class_k} + True\ Pos_{Class_k}}. \tag{2.7}$$

The F1-score for $Class_k$ is computed by:

$$F1Score_{Class_k} = \frac{2 \cdot R_k \cdot P_k}{R_k + P_k}. \tag{2.8}$$

Similarly, the F1-scores for all classes in the classification problem are computed and applied in Eq 2.5.

In all the experiments, we report the macro F1-Score.

2.6 Conclusion

This chapter provided a brief introduction to chaos theory, properties of deterministic chaos, the machine learning research methodology followed in this dissertation, and the performance measures used in this dissertation to evaluate the ML models. The reader is referred to excellent references on chaos theory [45, 58] and machine learning [55, 56]. In the next chapter we shall, introduce Neurochaos Learning, a novel brain inspired algorithm for classification.

Chapter 3

Neurochaos Learning

This chapter introduces Neurochaos Learning (NL), a novel brain inspired algorithm for classification. NL has two architectures: ChaosNet and ChaosFEX (CFX) + Machine Learning (ML). ChaosNet is built using layers of neurons, each of which is composed of $1D$ chaotic map known as the Generalized Lüroth Series (GLS). GLS maps have been shown in earlier works to possess very useful properties for compression, cryptography and for computing XOR and other logical operations. CFX+ML is a hybrid NL-ML architecture combining Neurochaos features with ML classifiers. In this chapter, the mathematical foundations of ChaosNet and CFX + ML are described in detail.

3.1 Introduction

With the success of Artificial Intelligence, learning through algorithms such as Machine Learning (ML) and Deep Learning (DL) has become an area of intense activity and popularity with applications reaching almost every field known to humanity. These include – medical diagnosis [31], computer vision, cyber-security [30], natural language processing, speech processing [23], just to name a few. These algorithms, though inspired by the biological brain, are remotely related to the biological process of learning and memory encoding. The learning procedures used in these artificial neural networks (ANNs) to modify weights and biases are based on optimization

techniques and minimization of loss/error functions. The ANNs at present use an enormous number of hyperparamters which are fixed by an ad-hoc procedure for improving prediction as more and more new data is input into the system. These synaptic changes employed are solely data-driven and have little or no rigorous theoretical backing [59, 60]. Furthermore, for accurate prediction/classification, these methods require enormous amount of training data that capture the distribution of the target classes.

Despite their tremendous success, ANNs are nowhere close to the human brain/mind for accomplishing tasks such as natural language processing. To incorporate the excellent learning abilities of the human brain, as well as, to understand the brain better, researchers are now focusing on developing biologically inspired algorithms and architectures. This is being done both in the context of learning [61] and memory encoding [52, 62–71].

One of the most interesting properties of the brain is its ability to exhibit *Chaos* [72] – the phenomenon of complex unpredictable and random-like behaviour arising from simple deterministic nonlinear systems[1] as explained in chapter 2. The dynamics in electroencephalogram (EEG) signals is known to be chaotic [28]. The sensitivity to small shifts in internal functional parameters of a neuronal system helps to get desired response to different influences. This attribute resembles the dynamical properties of chaotic systems [73–75]. Moreover, it is seen that the brain may not reach a state of equilibrium after a transient, but is constantly alternating between different states. For this reason, it is suggested that with the change in functional parameters of the neurons, the brain is able to exhibit different behaviours – periodic orbits, weak chaos and strong chaos for different purposes. For example, there has been evidence to suggest that weak chaos may be good for learning [76] and periodic activity in the brain being useful for attention related tasks [77]. Thus, chaotic regimes exhibiting a wide variety of behaviors help the brain in quick adaptation to changing conditions.

Chaotic behaviour is exhibited not only by brain networks which are composed of

[1]Deterministic chaos is characterized by the *Butterfly Effect* – sensitive dependence of behaviour to minute changes in initial conditions. Refer to chapter 2 for more details.

billions of neurons, but the dynamics at the neuronal level (cellular and sub-cellular) are also chaotic [28]. Impulse trains produced by these neurons are actually responsible for the transmission and storage of information in the brain. These impulses or *action potentials* are generated when different ions cross the axonal membrane causing a change in the voltage across it. Hodgkin and Huxley were the first to propose a dynamical system's model for the interaction between the ion channels and axon membrane, that is capable of generating realistic action potentials [78]. Later, its simplified versions such as the Hindmarsh-Rose model [79] and the Fitzugh-Nagumo [80,81] model were proposed. All these models exhibit chaotic behaviour.

Although there exist some artificial neural networks which display chaotic dynamics (an example is Recurrent Neural Networks [32]), to the best of our knowledge, none of the architectures proposed for classification tasks till date exhibit chaos at the level of individual neurons. However, for a theoretical explanation of memory encoding in the brain, many chaotic neuron models have been suggested. These include the Aihara model [62] which has been utilized for memory encoding in unstable periodic orbits of the network [63]. Freeman, Kozma and group have developed chaotic models inspired from the mammalian olfactory network to explain the process of memorizing of odors [64–66]. Chaotic neural networks have been studied also by Tsuda et al. for their functional roles as short term memory generators as well a dynamic link for long term memory [67,68]. Kaneko has explored the dynamical properties of globally coupled chaotic maps suggesting possible biological information processing capabilities of these networks [69, 70]. Chaotic neurons have also been used to develop biologically inspired models for memory encoding [52].

In this chapter, a novel brain inspired learning algorithm namely *Neurochaos Learning* (NL) is proposed. NL uses chaos fundamentally at the level of artificial neuron. The proposed algorithm has two architectures (a) `ChaosNet` and (b) ChaosFEX (CFX) + ML. The mathematical foundations of these architectures are described in this chapter. NL can accomplish classification tasks by learning with limited training samples. `ChaosNet` and CFX+ML are developed as an attempt to use some of the best properties of biological neural networks arising as a result of rich chaotic behav-

ior of individual neurons and is shown to accomplish challenging classification tasks comparable to or better than conventional ANNs while requiring far less training samples.

Choice of $1D$ maps as neurons in NL helps to keep the processing simple and at the same time exploit the useful properties of chaos. The proposed classification scheme is rigorously tested on publicly available datasets. This is studied rigorously in chapter 5.

The contents of this chapter are organized as follows. GLS-neuron and its properties are described in section 3.2. Section 3.3, describes the mathematical foundations of the NL architectures: `ChaosNet` and CFX + ML. Section 3.4 provides the concluding remarks.

3.2 GLS neuron and its properties

The neuron proposed in this work is a piece-wise linear $1D$ chaotic map known as Generalized Lüroth Series or GLS [46]. The well known Tent map, Binary map and their skewed cousins are all examples of GLS. Mathematically, the types of GLS neurons we use in this work are described below.

3.2.1 GLS neuron types: $T_{Skew-Tent}(x)$ and $T_{Skew-Binary}(x)$

A map $T_{Skew-Binary} : [0, 1) \to [0, 1)$ is defined as:

$$T_{Skew-Binary}(x) = \begin{cases} \frac{x}{b} & , \quad 0 \leq x < b, \\ \frac{(x-b)}{(1-b)} & , \quad b \leq x < 1, \end{cases}$$

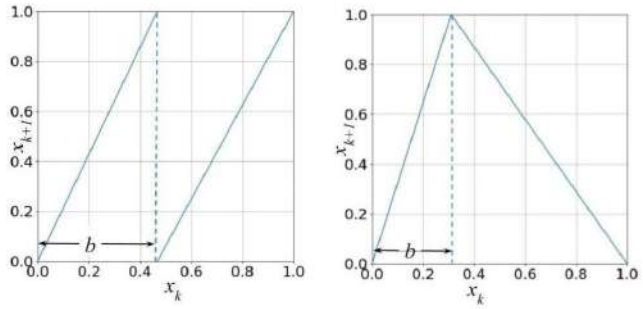
where $x \in [0, 1)$ and $0 < b < 1$. Refer to Figure 3.1(a).

A map $T_{Skew-Tent} : [0, 1) \to [0, 1)$ is defined as:

$$T_{Skew-Tent}(x) = \begin{cases} \frac{x}{b} & , \quad 0 \leq x < b, \\ \frac{(1-x)}{(1-b)} & , \quad b \leq x < 1, \end{cases}$$

where $x \in [0, 1)$ and $0 < b < 1$. Refer to Figure 3.1(b).

The symbols **L (or 0)** and **R (or 1)** are associated with the intervals $[0, b)$ and $(b, 1)$



(a) (b)

Figure 3.1: Generalized Lüroth Series, GLS neuron, is fundamentally of two types: (a) Left: Skew-Binary map $T_{Skew-Binary}$. (b) Right: Skew-Tent map $T_{Skew-Tent}$.

respectively, thereby defining the *symbolic sequence*[2] for every trajectory starting from an initial value on the map. We enumerate some salient properties of the GLS neuron:

1. Each GLS neuron has two parameters - an initial activity value x_0 and the skew value b. The parameter b defines the *generating Markov partition*[3] for the map and also acts as *internal discrimination threshold* of the neuron which will be used for feature extraction.

2. GLS neuron can fire either chaotically or in a periodic fashion (of any finite period length) depending on the initial activity value x_0. The degree of chaos is controlled by the skew parameter b. The lyapunov exponent of the map [48] is given by $\lambda_b = -b\ln(b) - (1-b)\ln(1-b)$. If the base of the logarithm is chosen as 2, then $\lambda_b = H(S_{x_0})$ (bits/iteration) where S_{x_0} is the symbolic sequence of the trajectory obtained by iterating the initial neural activity x_0 and $H(\cdot)$ is Shannon Entropy in bits/symbol. For $0 < b < 1$, $\lambda_b > 0$.

3. Any finite length input stream of bits can be *losslessly* compressed as an initial activity value x_0 on the neuron by performing a backward iteration on the map.

[2]Let $X = \{x_0, x_1, x_2, \ldots\}$ be the trajectory of a chaotic map with initial condition x_0, where $x_i \in [U, V)$. The interval $[U, V)$ is partitioned into k sub intervals denoted as $I_0, I_1, \ldots, I_{k-1}$. If $x_i \in I_j$ then we denote x_i by the symbol $j \in \{0, 1, \ldots, k-1\}$. The new sequence of symbol $\{j_0, j_1, \ldots, j_{k-1}\}$ is the symbolic sequence of the trajectory of X.

[3]Generating Markov Partition or GMP [47] is based on splitting the state space into a complete set of disjoint regions, namely, it covers all state space and enables associating a one-to-one correspondence between trajectories and itinerary sequences of symbols (L and R) without losing any information.

Further, such a lossless compression scheme has been previously shown to be *Shannon optimal* [49].

4. Error detection properties can be incorporated into GLS neuron to counter the effect of noise [50].

5. Owing to its excellent chaotic and ergodic (mixing) properties, GLS (and other related $1D$ maps) has been employed in cryptography [48, 49, 51, 82].

6. GLS neurons have been used to develop compression-based neural architecture for memory encoding and decoding [52].

7. GLS neuron can compute logical operations such as XOR, AND, NOT, OR etc. by switching between appropriate maps as described in a previous work [48,52].

8. GLS neuron has the property of *topological transitivity* - defined in a later section, which we will employ to perform classification.

9. A single layer of a finite number of GLS neurons satisfies a version of the *Universal Approximation Theorem* which we shall prove in a subsequent section of this paper.

The aforementioned properties make GLS neurons an ideal choice for building our novel algorithm for classification.

3.3 Neurochaos Learning: The Proposed Brain Inspired Learning Algorithm

In this section, the novel Neurochaos Learning algorithm the central contribution of this thesis is introduced. The key principles behind this algorithm, its parameters and hyperparameters, and a proof of the Universal Approximation Theorem (UAT) are discussed.

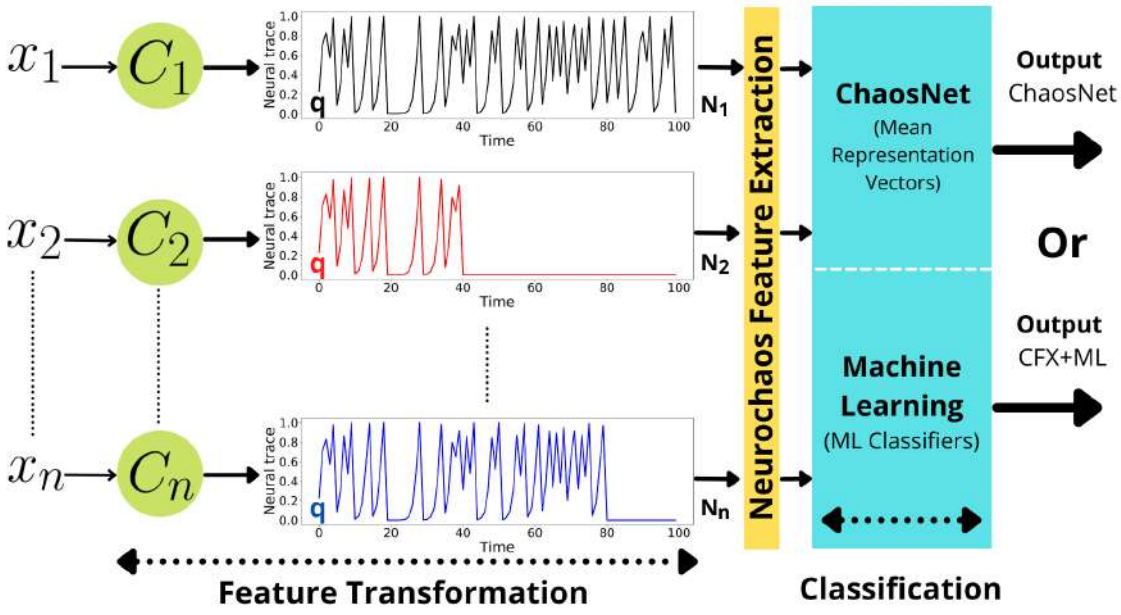


Figure 3.2: Neurochaos Learning involves three steps: (a) feature transformation, (b) neurochaos feature extraction, and (c) classification. The classifier chosen could either be ChaosNet or any other ML classifier.

3.3.1 Two Architectures of NL

NL architecture is depicted in Figure 3.2. NL has mainly three blocks - (a) feature transformation, (b) neurochaos feature extraction and (b) classification. A detailed description of each block is given below.

- **Input**: Input is the first and most significant step of any learning process. It contains input attributes obtained from the dataset $(x_1, x_2, \ldots, x_n)$. The algorithm requires the input attributes to be normalized in the range $[0, 1]$. After suitable normalization, the input attributes are further passed to the feature transformation block (refer Figure 3.2).

- **Feature Transformation**: The feature transformation block consists of an input layer of 1D Generalized Lüroth Series (GLS) neurons. The number of GLS neurons $(C_1, C_2, \ldots, C_n)$ in the input layer is equal to the number of input attributes (n) in the dataset. Each neuron $C_1, C_2, \ldots, C_n$ has an initial neural activity of q units. Upon arrival of the input attributes (also known as *stimuli*)

$x_1, x_2, \ldots, x_n$, each of the 1D GLS neurons $(C_1, C_2, \ldots, C_n)$ starts independently firing with an initial neural activity of q units.

The chaotic firing of each GLS neuron stops when their neural trace starting from the initial neural activity (q) reaches the epsilon neighbourhood of $x_1, x_2, \ldots, x_n$. The time at which each neuron stops firing can thus be different. The halting of firing of each GLS neuron is guaranteed by *Topological Transitivity* property of chaos. The formal definition of *Topological Transitivity* is as follows:

Definition 1: *Topological Transitivity* property for a map $T : R \to R$ states that for all non-empty open set pairs D and E in R, there exists an element $d \in D$ and a non negative finite integer n such that $T^n(d) \in E$.

For example, we consider a GLS 1D map (T) which is chaotic with $R : [0, 1)$. Let $D = (q - \epsilon, q + \epsilon)$ and $E = (x_k - \epsilon, x_k + \epsilon)$ where $\epsilon > 0$. From the definition of *Topological Transitivity*, the existence of an integer $N_k \geq 0$ and a real number $d \in D$ such that $T^{N_k}(d) \in E$ is ensured. We consider $d = q$ (initial neural activity of the GLS neuron) and x_k as the stimulus to the k-th GLS neuron (after normalizing). Thus, $N_k \geq 0$ will always exist. It is important to highlight that for certain values of q there may be no such N_k, for eg., initial values that lead to periodic orbits. However, we can always find a value for q for which N_k exists. This is because, for a chaotic map, there are an infinite number of initial values that lead to non-periodic ergodic orbits.

The neural trace corresponding to each stimulus is the transformed features. These transformed features are further passed to the neurochaos feature extraction block.

- **Neurochaos Feature Extraction**: The Neurochaos Feature Extraction block extracts the following features from the chaotic neural trace:

    1. *Firing time (N)*: The time taken (the number of iterations of the 1D map) by the chaotic neural trace to recognise the stimulus.

2. *Firing rate (R)*: Fraction of time for which the chaotic neural trace exceeds the discrimination threshold b so as to recognize the stimulus. Refer to Figure 3.3 for visual understanding of the firing rate feature extraction from the chaotic neural trace.

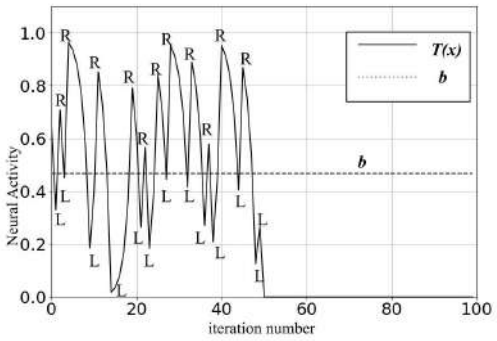


Figure 3.3: Illustration of Firing Rate feature extraction: The GLS neurons has a default initial neural activity of q units. The i-th GLS neuron fires for N_i iterations (chaotically) and stops by reaching the ϵ-neighbourhood of the i-th input stimulus. For the duration of the GLS neuron being active (chaotic firing), the fraction of the time when the neural activity exceeds the discrimination threshold b of the neuron (marked as R above) is the extracted Firing Rate.

3. *Energy (E)*: A chaotic neural trace $z(t)$ with a firing time N has an energy (E) defined as:

$$E = \sum_{t=1}^{N} z(t)^2. \tag{3.1}$$

4. *Entropy (H)*: The Shannon entropy of a chaotic neural trace $z(t)$ is computed using the symbolic sequence $SS(t)$ of $z(t)$. $SS(t)$ is defined as follows:

$$SS(t_i) = \begin{cases} 0, & z(t_i) < b, \\ 1, & b \leq z(t_i) < 1, \end{cases} \tag{3.2}$$

where $i = 1$ to N (firing time). From $SS(t)$, Shannon Entropy $H(SS)$ is

computed as follows:

$$H(SS) = -\sum_{i=1}^{2} p_i \log_2(p_i) \text{ bits}, \tag{3.3}$$

where, p_1 and p_2 are the probabilities of occurrence of symbols 0 and 1 in $SS(t)$ respectively.

An input stimulus x_k of a data instance visiting the k-th GLS neuron (C_k) is transformed to a 4D vector $[N_{x_k}, R_{x_k}, E_{x_k}, H_{x_k}]$. The CFX feature space contains a collection of all the 4D vectors after feature transformation. These chaos based features are passed to the third block of the NL architecture i.e, classification.

- **Classification**: There are mainly two kinds of NL architecture: (a) `ChaosNet`, (b) ChaosFEX (CFX) + ML. `ChaosNet` architecture computes the mean representation vector for each class. The training part of `ChaosNet` architecture involves the computation of mean representation vectors as described below: Let us assume an s class classification problem where classes are represented by $\mathscr{C}_1$, $\mathscr{C}_2$, ..., $\mathscr{C}_s$ and the corresponding true-labels be denoted as $1, 2, \ldots, s$ respectively. Let the normalized distinct data matrices be $U^1, U^2, \ldots, U^s$ of size $m \times n$. The matrices $U^1, U^2, \ldots, U^s$ denotes the data belonging to $\mathscr{C}_1$, $\mathscr{C}_2$, ..., $\mathscr{C}_s$ respectively. Training involves extracting features (Firing Time, Firing Rate, Energy and Entropy) from $U^1, U^2, \ldots, U^s$ using NL feature extraction so as to yield $V^1, V^2, \ldots, V^s$. Feature extraction using NL is applied on each stimulus, hence the size of $V^1, V^2, \ldots, V^s$ will be $m \times 4n$. Once the NL feature extraction is found for the data belonging to the s distinct classes, the average across row

is computed next:

$$\begin{aligned} M^1 &= \frac{1}{m}\Big[\sum_{i=1}^{m} V_{i1}^1, \sum_{i=1}^{m} V_{i2}^1, \ldots, \sum_{i=1}^{m} V_{i4n}^1\Big], \\ M^2 &= \frac{1}{m}\Big[\sum_{i=1}^{m} V_{i1}^2, \sum_{i=1}^{m} V_{i2}^2, \ldots, \sum_{i=1}^{m} V_{i4n}^2\Big], \\ &\vdots \\ M^s &= \frac{1}{m}\Big[\sum_{i=1}^{m} V_{i1}^s, \sum_{i=1}^{m} V_{i2}^s, \ldots, \sum_{i=1}^{m} V_{i4n}^s\Big]. \end{aligned}$$

The s row vectors $M^1, M^2, \ldots, M^s$ are termed as *mean representation vectors* corresponding to the s classes. The *average internal representation* of all the stimuli corresponding to k-th class is encoded in the vector M^k. As more and more input data are received the mean representation vectors get updated.

The mean representation vector of class-k contains the mean firing time, mean firing rate, mean energy and mean entropy of the k-th class. `ChaosNet` uses a simplistic decision rule, namely, the cosine similarity of testdata instances with the mean representation vectors. The testdata instance is assigned a label $= l$, if the cosine similarity of that testdata instance with l-th mean representation vector is the highest. The testing part for `ChaosNet` is provided below: Let Z denote the normalized test data matrix of size $r \times n$. The i-th row of Z represents i-th test data instance denoted as $z^i = [z_1^i, z_2^i, z_3^i, \ldots, z_n^i]$. The NL feature extraction step is applied to each row (each test data instance (z^i) where $i = 1, 2, 3, \ldots, r$). Let the NL feature extracted data of test samples be denoted as F where $f^i = [f_1^i, f_2^i, \ldots, f_{4n}^i]$ is the i-th row of F.

Feature extraction is followed by the computation of cosine similarity of f^i independently with each of the $M^1, M^2, ..., M^s$ (*mean representation* vectors) respectively:

$$
\begin{aligned}
\cos(\theta_1) &= \frac{f^i \cdot M^1}{\|f^i\|_2 \|M^1\|_2}, \\
\cos(\theta_2) &= \frac{f^i \cdot M^2}{\|f^i\|_2 \|M^2\|_2}, \\
&\vdots \\
\cos(\theta_s) &= \frac{f^i \cdot M^s}{\|f^i\|_2 \|M^s\|_2}.
\end{aligned}
$$

The scalar product between vectors f^i and M^k is represented by $f^i \cdot M^k$, $\|v\|_2$ denotes the l_2 norm of row-vector v. The above will give 's' scalar values which are the cosine similarity values between $\{M^k\}_{k=1}^{s}$ and f^i. Out of these 's' scalar values, the index (l) corresponding to the maximum cosine similarity ($\cos(\theta_l)$) is considered as the label for f^i:

$$\theta_l = \arg\max_{\theta_i}(\cos(\theta_1), \cos(\theta_2), \ldots, \cos(\theta_s)). \tag{3.4}$$

If there is more than one index with maximum cosine similarity, we take the smallest such index. The above procedure is continued until a unique label is assigned to each test data instance.

Alternatively, we have the flexibility of choosing any other ML classifier instead of `ChaosNet`. In this kind of NL architecture, the CFX features are fed directly to the ML classifier (CFX+ML). For CFX+ML, the following ML classifiers are used in this thesis – Decision Tree (DT), Random Forest (RF), AdaBoost (AB), Support Vector Machine (SVM), k-Nearest Neighbors (k-NN) and Gaussian Naive Bayes (GNB). CFX+ML is a hybrid NL architecture that combines the best of chaos and machine learning.

- **Output**: The output obtained from the classification block serves as output for that respective NL architecture. On using `ChaosNet` in the classification block, the output is an outcome of classification based on the mean representation vectors. The ML implementation in the classification block produces an output

dependent on the choice of the ML classifier.

3.3.2 Parameters vs. Hyperparameters

Distinguishing model parameters and model hyperparameters plays a crucial role in machine learning tasks. The model parameters and hyperparameters of the proposed method are as follows:

Model parameter: This is an internal parameter which is estimated from the data. These internal parameters are what the model learns while training. In the case of `ChaosNet`, the *mean representation vectors* $M^1, M^2, \ldots, M^s$ are the model parameters which are learned while training. The model parameters for deep learning (DL) are the weights and biases learnt during training the neural network. In Support Vector Machines (SVM), the support vectors are the parameters. In all these cases, the parameters are learned while training.

Model hyperparameters: These are configurations which are external to the model and typically not estimated from the data. The hyperparameters are tuned for a given classification or predictive modelling task in order to estimate the best model parameters. The hyperparameters of a model are often arrived by heuristics and hence are different for different tasks. In the case of NL, the hyperparameters are the *initial neural activity* (q), *discrimination threshold* (b), ϵ used in defining the neighbourhood interval of x_k^i ($I_k^i = (x_k^i - \epsilon, x_k^i + \epsilon)$) and the *chaotic map* chosen. In DL, the hyperparameters are the *number of hidden layers*, *number of neurons in the hidden layer*, *learning rate* and the *activation function* chosen. In the case of K-Nearest Neighbours (KNN) classification algorithm, the *number of nearest neighbours* (k) is a hyperparameter. In SVM, the *choice of kernel* is a hyperparameter. In Decision Tree, *depth of the tree* and the *least number of samples required to split the internal node* are the hyperparameters. In all these cases the hyperparameters need to be fixed by cross-validation.

3.3.3 Universal Approximation Theorem (UAT)

Cybenko (in 1989) proved one of the earliest versions of the *UAT*. *UAT* states that continuous functions on compact subsets of $\mathbb{R}^n$ can be approximated by an ANN with one hidden layer having finite number of neurons with sigmoidal activation functions [83]. Thus, simple neural networks with appropriately chosen parameters can approximate continuous functions of a wide variety. A single layer Neurochaos Learning can approximate a discrete time function with finite support. The UAT for NL is provided below:

UAT for NL

Theorem: Let $f(n)$ be a discrete time real valued function having a finite support L. The NL architecture consisting of a single layer with L chaotic neurons can approximate[4] $f(n)$. Assuming that we use a chaotic $1D$ map C_i for the i-th neuron in NL, and given any desired error $\epsilon > 0$, we have:

$$d(f, C) = \sum_{i=1}^{L} |f(i) - C_i^{N_i}(q)| < \epsilon, \tag{3.5}$$

where q is the initial neural activity for all the neurons in NL, N_i is the firing time of the i-th chaotic neuron and C_i is the $1D$ chaotic map with the chaotic trajectory starting from 'q' a dense orbit.

Proof. Design an NL with one layer with exactly L chaotic neurons. Let the initial neural activity of each neurons be initialized with q and let the input to this NL be the L real-valued samples of the function $f(n)$ which act as stimuli for the corresponding L chaotic neurons.

Now, for a given $\epsilon > 0$, we can always construct a neighbourhood of stimulus $I_k = (f(k) - \eta, f(k) + \eta)$, $0 < \eta < \frac{\epsilon}{2L}$ such that $C_k^{N_k}(q) \in I_k$ for the k-th chaotic neuron of NL. This is always possible because of the topological transitivity property

[4]For quantifying this approximation, we use the sum of absolute differences as the distance metric. In other words, for any two real-valued vectors $V, W \in \mathbb{R}^m$, $d(V, W) = \sum_{i=1}^{m} |V_i - W_i|$.

of chaos defined in Section 3.3 and since the chaotic trajectory starting from initial value q is dense. The topological transitivity property guarantees the chaotic firing to reach the η neighbourhood (I_k) of stimulus in finite number of iterations (N_k) for the dense orbit starting from 'q'. For any given ϵ, the following is true:

$$\begin{aligned} d(f,C) &= \sum_{i=1}^{L} d(f(i), C_i^{N_i}) = \sum_{i=1}^{L} |f(i) - C_i^{N_i}(q)| \\ &< \sum_{i=1}^{L} 2\eta = L(2\eta) < L(2\frac{\epsilon}{2L}) = \epsilon. \end{aligned}$$

□

Note that $\eta < \frac{\epsilon}{2L}$ since $C_i^{N_i}(q) \in (f(i)-\eta, f(i)+\eta)$ because N_i is the firing time for C_i and the orbit is dense. Hence, the set of chaotic neurons $\{C_i\}$ that constitute the input layer of NL can always approximate the function $f(n)$ with an ϵ error bound. This theorem holds true for NL constructed with chaotic neurons that satisfies the topological transitivity property and has a dense orbit. Further, having a single dense orbit implies countably infinite number of dense orbits.

A comparison of the properties of NL with Artificial Neural Networks is provided in Table 3.1.

Table 3.1: NL vs. ANN - a comparison of properties.

Properties	**ANN**	**NL**	**Remarks**
Neuron	Linear followed by a nonlinear activation	Non-linear and chaotic	Chaos allows for a rich set of properties to be exploited.
Output of a Neuron	Scalar	Variable length vector	Neurons in NL perform non-linear computations as compared with simple weighted linear addition in ANN.
Universal Approximation Theorem (UAT)	Satisfies UAT	Satisfies UAT	NL satisfies UAT with an exact specification on the number of neurons needed for approximating a real-valued discrete-time function with finite support.
Activation Functions	Yes	No	The nonlinearity in ANN is provided by the activation function which is not needed for NL.
Backpropagation	Yes	No	Not currently used. NL could employ backpropagation in the future if needed.

3.4 Conclusions

Chaos has been empirically found in the brain at several spatio-temporal scales [27,28]. In fact, individual neurons in the brain are known to exhibit chaotic bursting activity and several neuronal models (for eg., Hindmarsh-Rose neuron model) exhibit complex chaotic dynamics [84]. Though Artificial Neural Networks such as Recurrent Neural Networks exhibit chaos, to our knowledge, there have been no successful attempts in building an ANN for classification tasks that is entirely comprised of neurons which are individually chaotic and which harness the power of chaos for classification. In this chapter, Neurochaos Learning (NL) – a brain inspired learning algorithm for classification has been proposed. NL uses an input layer of 1-dimensional chaotic map known as Generalized Lüroth Series (GLS). GLS has been shown to have salient properties such as ability to encode and decode information losslessly with Shannon optimality, computing logical operations (XOR, AND etc.), universal approximation property and ergodicity (mixing) for cryptography applications. The property of Topological Transitivity is used ingeniously by NL for classification. In total, this chapter deals with the mathematical foundations of Neurochaos Learning.

Chapter 4

Stochastic Resonance in Neurochaos Learning

This chapter provides a theoretical justification to the working of NL. The interplay of chaos and noise forms the bedrock in the working of NL. Inspired by the chaotic firing of neurons and the constructive role of noise in neuronal models, this chapter connects chaos, noise and learning for the first time. In this chapter, the Stochastic Resonance (SR) phenomenon in NL is demonstrated. SR manifests at individual neurons in NL and enables efficient subthreshold signal detection. Furthermore, SR is shown to occur in single and multiple neuronal NL architectures for classification tasks - both on simulated and real-world spoken digit datasets, and in architectures with $1D$ *chaotic maps as well as Hindmarsh Rose spiking neurons. Intermediate levels of noise in NL enables peak performance in classification tasks thus highlighting the role of SR in Artificial Intelligence applications, especially in brain inspired learning architectures.*

4.1 Introduction

The discipline of 'Artificial Intelligence' (AI) originated with the aim of building computer systems that mimics the human brain. This involves the interplay of neuroscience and computational/mathematical models. Over the years since the inception

of AI, both neuroscience and computational approaches have expanded their boundaries. This in turn shifted the focus of AI from building systems by exploiting the properties of brain to mere engineering point of view i.e., 'what works is ultimately all that really matters' [85]. The engineering approaches like optimization and hyperparameter tuning evaluate AI from a performance point of view. This particular view greatly limits the original motivation of AI. In this research, two key ideas from neuroscience and physics namely Chaos and Stochastic Resonance are used to explain the working of NL.

With the current understanding, there are nearly 86 billion neurons [26] in the human brain. They interact with each other to form a complex network of neurons. These neurons are inherently non-linear and found to exhibit a fluctuating neural response for the same stimuli on different trials while doing experiments. The fluctuating neural response is in part due to (a) *inherent chaotic nature of neurons* [28]. Chaotic neurons are sensitive to initial states and thus show fluctuating behaviour to varying initial neural activity with the start of each trial. The second source of fluctuating behaviour can be attributed to (b) *neuronal noise and interference* [86]. Noise has its effect on the perception of sensory signals to the motor response generation [86]. Thus, noise poses a challenge as well as a benefit to information processing. The research in noise can be traced back to the experiment of Robert Brown in 1822 [87]. In the experiment, the Scottish botanist observed under a microscope the irregular movement of pollen on the surface of a film of water. Robert Brown tried to investigate the reason behind this irregular fluctuations. This phenomenon is known as Brownian motion. This phenomena was successfully explained by Albert Einstein in 1905 [88]. The noise produced by the Brownian motion is termed as brown noise or red noise. The term brown is indicated to give credit to Robert Brown for his key observations and laying out experiments to understand Brownian motion. Another interesting research in 1912 by Dutch Physicist and the first woman in noise theory, Geertruida de Haas-Lorentz viewed electrons as Brownian particles. This inspired the Swedish Physicist Gustav Adolf Ising in 1926 to explain why galvanometers cannot be cascaded indefinitely to increase amplification [89]. The next leap in noise

research was brought by J. B. Johnson and H. Nyquist. During the year 1927-1928 Johnson published his well known thermal voltage noise formula and derived the formula theoretically in collaboration with Nyquist [90, 91]. The world took a turn in 1948 by the ground breaking work of Claude Elwood Shannon who created the field called Information theory [92]. In his 1948 paper titled "A Mathematical Theory of Communication", Shannon solved how to reliably transmit a message through an unreliable (noisy) channel. Shannon showed that any communication channel can be modeled in terms of bandwidth and noise. Bandwidth is the range of electromagnetic frequencies required to transmit a signal and noise is an unwanted signal that disrupts the original signal. He further showed how to calculate the maximum rate at which data can be sent through a channel with a particular bandwidth and noise characteristics with zero error. This is called the rate of channel capacity or Shannon limit [92].

All these research, especially Shannon's work, considered noise as an unwanted signal that adversely affects the communication. But in the second half of 20th century, the constructive advantage of noise in signal detection and also the advantages of noise in physiological experiments lead to the birth of *Stochastic Resonance*. The term Stochastic Resonance (SR) was first used in the context of noise optimized systems by Roberto Benzi [93] in 1980 with regard to a discussion on climate variations and variability. Benzi introduced SR in connection with the explanation to a periodicity of 10^5 years found in the power spectrum of paleoclimatic variations for the last $700,000$ years [34]. The energy balance models failed to explain this phenomenon. Benzi, in his paper titled "Stochastic Resonance in Climate Change" [34], suggested that the combination of internal stochastic perturbations along with external periodic forcing due to earth's orbital variations are the reasons behind the ice age cycle. The paper further suggests that neither the stochastic perturbations nor periodic forcing alone can reproduce the strong peak found at a periodicity of 10^5 years. Thus, the effect produced by this co-operation of noise and periodic forcing was termed as Stochastic Resonance by Benzi. Even though this explanation is still a subject of debate, but the definition of the term SR continued to evolve in the coming years. SR

finds application in climate modelling [34], electronic circuits [94], neural models [95], chemical reactions [96], thermoacoustics [97] etc. The original use of the resonance part of SR comes from the plot of output signal to noise ratio (SNR) that exhibits a single maximum for an intermediate intensity of noise [98].

A motivating idea to develop SR-based electronic devices or neuromorphic systems comes from the brain, because we know that the brain is far better than electronic devices in terms of low computational power, robustness to noise and neural interference. SR in the brain and nervous system could serve as a motivation for the design of robust machine learning architectures and electronic systems. The observation of SR in neural models was first published in 1991 [95]. The research in SR accelerated when the 1993 Nature article reported the presence of SR in physiological experiments on crayfish mechanorecetors [99]. In the same year, yet another highly cited paper – SR on neuronal model [100] – became widely popular. These research studies triggered the expansion of SR in mathematical models of neurons, biological experiments, behavioural experiments especially in paddle fish [101], noise enhanced cochlear implants [102] and computations [103]. Recently SR has been studied in a triple cavity driven by noise [104]. Despite the wide popularity of SR, only a few papers have focused on the application of SR in machine learning (ML) and deep learning (DL) [105, 106]. The current ML and DL algorithms assume ideal working conditions, i.e. the input data is noiseless. But in practice, there is unavoidable noise that distorts measurements by sensors. Hence, there is a research gap from a theoretical and an implementation point of view as far as ML/DL algorithms are concerned.

We define SR as noise enhanced signal processing [98]. In a nutshell for SR to occur, the following four elements are required:

1. An information carrying input signal.

2. A noise added to the input signal.

3. A non-linear system that processes this noisy input signal.

4. A performance measure which captures the relationship between the output and input with respect to varying noise intensity. For classification tasks, we shall use F1-score (defined in chapter 2, section 2.5.3) as a measure to capture the performance of the non-linear system with respect to varying noise intensities.

This chapter focuses on how SR is inherently used in NL architecture. The sections in the chapter are arranged as follows: Section 4.2 describes how SR is naturally manifested in NL. Section 4.3 highlights the empirical evidence of SR in NL with $1D$ chaotic map for both simulated and real world datasets. The concluding remarks of this chapter are provided in Section 4.4.

4.2 Stochastic resonance in a single GLS neuron

Having described the `ChaosNet` NL architecture in chapter 3 of this dissertation (Figure 3.2), we begin our study by exploring the functioning of a single GLS neuron in this architecture. In order to understand the SR effect in a single GLS neuron, we consider a hypothetical example of the feeding pattern of a cannibalistic species with a food chain depicted in Figure 4.1. Specifically, we consider the problem of survival of a single organism X of this species whose size is 0.5 units. All those organisms whose size is ≤ 0.5 are a potential prey whereas those whose size is > 0.5 is a potential predator. For survival, this specific organism X needs to solve a binary classification problem. To this end, we assume that X has a single GLS neuron that fires upon receiving a stimulus from the environment. The stimulus, say light reflected off the approaching organism, encodes the size of that organism, and we assume that it could be any value in the range $[0, 1]$. The problem now reduces to determining whether the stimulus corresponds to the label 'prey' (Class-0) or label 'predator' (Class-1). An example of several stimuli received by the organism X is depicted in Figure 4.2a where Class-0 (Prey) and Class-1 (Predator) are marked with black stars and red circles respectively.

In order to solve the binary classification problem for survival, organism X employs the `ChaosNet` NL architecture. Briefly, this is accomplished as follows. The single

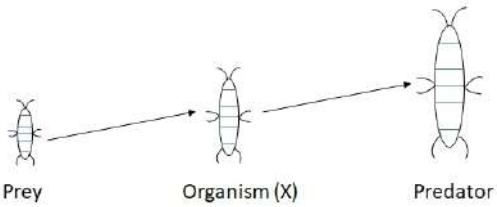


Figure 4.1: Food chain of a (hypothetical) cannibalistic species. The organism X feeds on other organisms of its species (prey) or is fed by other organisms of its species (predator) depending on their size. The organism X has to decide whether the approaching one is food (prey) or death (predator) – a binary classification problem. It has one internal GLS neuron which fires upon receiving stimulus which is light reflected of from the approaching organism (either predator or prey) that encodes the size of the organism.

GLS neuron of organism X fires chaotically from an initial neural activity (of q units) until it matches the value of the stimulus. This chaotic neuronal trace encodes all the necessary information which the organism X needs to determine whether the stimulus is Class-0 or Class-1. Particularly, the following four features are used – firing time, firing rate, energy and entropy of neural trace. In the training phase, where the labels are assumed to be known, the organism X learns the mean representation vector (the aforementioned 4 features) for each class (80 instances) and in the testing phase (20 instances per class), the organism X estimates the label. This $(80, 20)$ is the standard train-test distribution that we find in machine learning applications.

From an evolutionary point of view, the interesting question to ask is - *how can natural selection optimize the biological aspects of organism X in order for it to have the highest possible survival rate where it is able to correctly classify the stimulus as predator or prey?* Assuming that the organism can't afford to have more than one internal neuron (each neuron comes with a biological cost), the only available biological features are – A) the type of GLS neuron and B) initial neural activity (this corresponds to memory) of this single internal neuron. The type of GLS neuron is determined by the value of the parameter b (we assume the skew-tent map which is a type of $1D$ GLS map[1]) and the initial neural activity is represented by q. Both b and q in our example can take values in the range $(0, 1)$. In our discussion so far,

[1]In reality, it is possible for nature to evolve different types of neurons - which is very much the case with the brain. But for simplicity, we restrict to skew-tent map in this example.

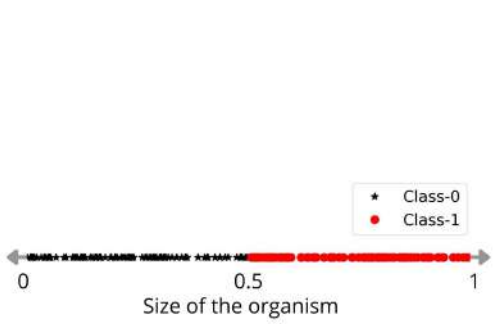
we have omitted a very important factor namely the *noise* that is inherent in the environment. Noise, as we very well know, is unavoidable and the stimulus (the light reflected off from the approaching organism) is inevitably distorted by ambient noise. In our efforts to optimize b and q to give the highest survival rate for organism X, we are also interested in asking the question – *is the presence of ambient noise always detrimental to the decision making process of the GLS neuron, irrespective of the level of noise?* In other words, one would naively expect that even the presence of a tiny amount of noise can only degrade the performance of the chaotic GLS neuron thereby reducing the survival rate of organism X (not to speak of high levels of noise where the performance is expected to be much worse).

To answer the above two questions – one pertaining to the best possible values of q and b that maximizes survival rate and the other to the role of the level of noise on the performance of GLS neuron for decision making – we need to solve an optimization problem. To this end, we performed a five fold cross-validation on the dataset in Figure 4.2a with $q = 0.25$, $b = 0.96$ and noise intensity varying from 0.001 to 1 in steps of 0.001. For each trial of the five fold cross-validation, we used 100 data instances per class with a $(80, 20)$ training-test split. The `ChaosNet` NL algorithm is run with this single GLS neuron (refer to chapter 3, section 3.3 for further details). An average accuracy of 100% is reported for noise intensities ranging from 0.248 to 0.253. Figure 4.2b represents the average accuracy vs. noise intensity.

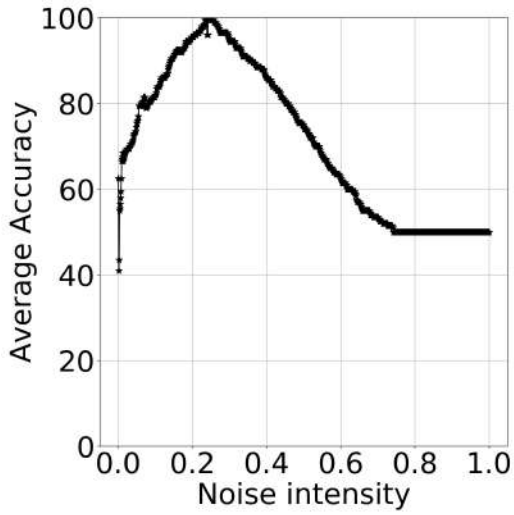
From Figure 4.2b we are able to see that the relationship of noise to survival rate (average accuracy) is not monotonic, as we had originally expected. Rather, we see the phenomenon of Stochastic Resonance exhibited by Neurochaos Learning architecture. In other words, ambient noise is not always detrimental to the performance of the GLS neuron for the survival of the organism X. An intermediate amount of noise, in fact, maximizes the survival rate of organism X. Our hypothetical example is not very far from reality. In the case of feeding behaviour of paddle fish [101] and in the physiological experiments on crayfish mechanorecetors, stochastic resonance has been reported. We conjecture that it may very well be the case of noise interacting with neurochaos leading to stochastic resonance in these real-world biological examples as

well.

It has been observed that SR is exhibited for several other settings of q and b as well, but are not reported here for lack of space. The particular choice of $q = 0.25$ and $b = 0.96$ is motivated by the fact that SR enables a 100% accuracy for these settings (with an intermediate level of noise as seen in Figure 4.2b). However, such a performance is not unique to this particular choice of q and b.



(a)



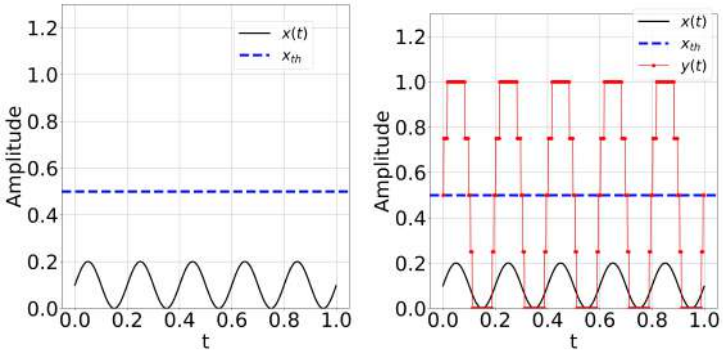
(b)

Figure 4.2: SR in ChaosNet NL with single GLS neuron: (a) Prey-Predator dataset for organism X. (b) Average Accuracy (%) vs. noise intensity shows stochastic resonance.

4.2.1 SR in Signal Detection using ChaosNet NL

Traditionally SR is shown to be exhibited in bistable system where there is an element of periodic forcing and stochastic component [107]. This notion is used to detect signals which are sub-threshold. A combination of a sub-threshold periodic and noisy signal provides a peak in detection of the frequency of the periodic signal. In this section, we demonstrate SR in signal detection using ChaosNet NL architecture.

There are different ways to quantify SR for signal detection. Some of the commonly used measures are cross correlation coefficient, signal-to-noise ratio, and mutual information [108]. In this work, we use cross correlation coefficient as a measure to



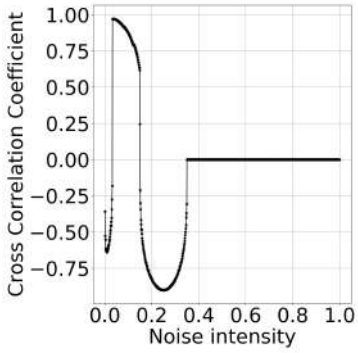


Figure 4.3: SR in signal detection using ChaosNet NL. (a) Sub-threshold signal $x(t)$ with threshold $x_{th} = 0.5$ (blue dotted). The signal cannot be detected since it is below the threshold. (b) In the presence of intermediate amount of noise ($\epsilon = 0.033$) added to the input signal, the normalized firing time of the single internal neuron in ChaosNet NL is given by $y(t)$ (red) which enables detection of the sub-threshold signal $x(t)$. (c) Cross correlation coefficient $\rho_{x,y}$ between $y(t)$ and $x(t)$ vs. noise intensity ϵ demonstrating SR. Whenever the variance of $y(t)$ was zero, we take $\rho_{x,y} = 0$.

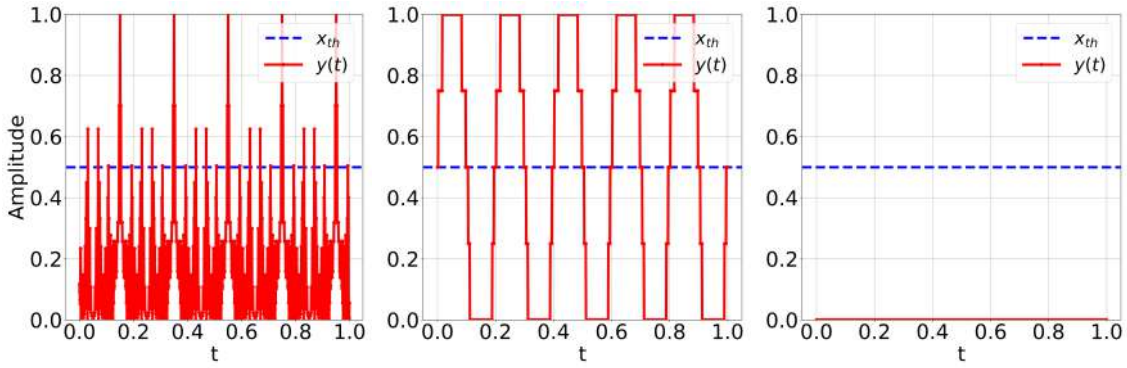
evaluate the SR behaviour in `ChaosNet` NL for signal detection.

Cross correlation coefficient between two discrete-time signals $P(t)$ and $Q(t)$ of length N is defined as follows:

$$\rho_{P,Q} = \frac{1}{N-1} \sum_{i=1}^{N} \left(\frac{P(i) - \mu_P}{\sigma_P} \right) \left(\frac{Q(i) - \mu_Q}{\sigma_Q} \right), \tag{4.1}$$

where μ_P, μ_Q are the mean of $P(t)$ and $Q(t)$ respectively and σ_P, σ_Q are the corresponding standard deviations. Cross correlation coefficient $\rho_{P,Q}$ is a scalar value in the range $[-1, 1]$. It measures the similarity between two signals $P(t)$ and $Q(t)$ (assuming only a linear relationship). A high positive value (close to 1) of $\rho_{P,Q}$ in-

dicates that $P(t)$ and $Q(t)$ are positively correlated. Similarly, a high negative value indicates that they are negatively correlated. A value of $\rho_{P,Q}$ close to zero indicates uncorrelatedness. Signal detection is deemed successful if $\rho_{P,Q}$ is highly positive between the input signal and the detected signal. SR is the phenomenon where signal detection is enhanced for an intermediate amount of noise.



(a) (b) (c)

Figure 4.4: Intermediate noise added to input signal is good for signal detection in NL. (a) With low noise ($\epsilon = 0.001$), signal detection is poor ($\rho_{x,y} = -0.357$). (b) With intermediate noise ($\epsilon = 0.033$), signal detection is good ($\rho_{x,y} = 0.971$). (c) With high noise ($\epsilon = 0.950$), signal detection is very poor ($\rho_{x,y} = 0$).

We consider a low amplitude information carrying periodic sinusoidal function:

$$x(t) = \frac{A(\sin(2\pi 5t) + 1)}{2}, \tag{4.2}$$

where $A = 0.2$, and $0 \leq t \leq 1$s. Let the threshold be defined as $x_{th} = 0.5$. Clearly, the low amplitude signal is below the threshold and hence the signal goes undetected (Figure 4.3a). We pass $x(t) + \eta(t)$ as input to the `ChaosNet` NL architecture with a single GLS neuron with $q = 0.35$ and $b = 0.65$. Here, $\eta(t)$ is noise which follows a uniform distribution with a range of $[-\epsilon, +\epsilon]$, where ϵ is varied from 0.001 to 1.0 in steps of 0.001. From the `ChaosNet` NL architecture, we extract the normalized firing time[2] $y(t)$ shown in Figure 4.3b which closely tracks $x(t)$. We then compute the cross correlation coefficient $\rho_{x,y}$ between $y(t)$ and $x(t)$ for various noise intensities (ϵ) as shown in Figure 4.3c. We see that, for an intermediate amount of noise intensity, namely

[2]Normalization is done as follows: $y(t) = \frac{\hat{y}(t) - min(\hat{y}(t))}{max(\hat{y}(t)) - min(\hat{y}(t))}$, $\hat{y}(t)$ is the firing time.

$\epsilon = 0.033$, we get a maximum cross correlation coefficient of $\rho_{x,y} = 0.971$ (Figure 4.3c and 4.4b) and for lower and higher noise intensities we get lower values of $\rho_{x,y}$ (this is because $y(t)$ fails to track $x(t)$ in these cases as demonstrated in Figure 4.4a, 4.4c). This is the classic phenomenon of stochastic resonance. Thus, we have demonstrated SR in NL (`ChaosNet`) with a single GLS neuron in both classification and signal detection scenarios.

4.3 SR in NL with multiple GLS Neurons

We now move on to demonstrating SR in NL with more than one GLS neuron in both simulated and real-world datasets.

4.3.1 Simulated Data

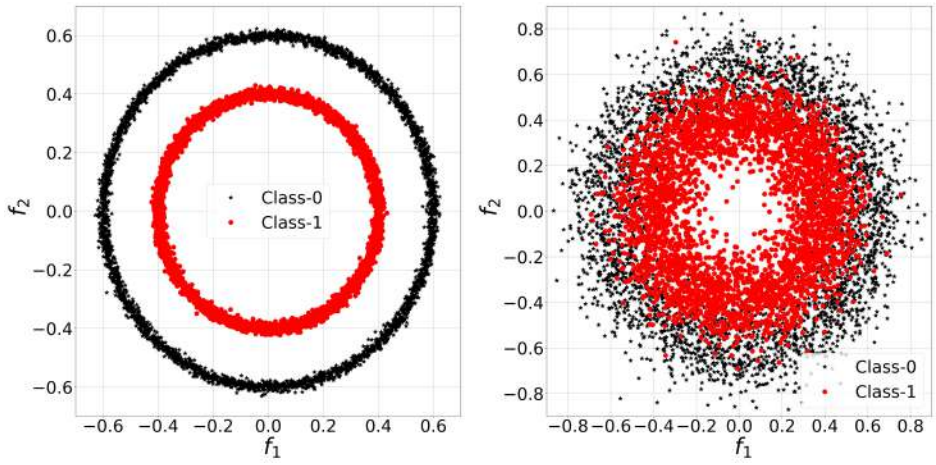
We consider a binary classification task of separating data instances belonging to two concentric circles which are either non-overlapping (CCD) or overlapping (OCCD). The governing equations for OCCD are as follows:

$$f_1 = r_i \cos(\theta) + \alpha\eta, \tag{4.3}$$

$$f_2 = r_i \sin(\theta) + \alpha\eta, \tag{4.4}$$

where $i = \{0, 1\}$ ($i = 0$ represents Class-0 and $i = 1$ represents Class-1), $r_0 = 0.6$, $r_1 = 0.4$, θ is varied from 0 to 360, $\alpha = 0.1$, $\eta \sim \mathcal{N}(\mu, \sigma)$, normal distribution, with $\mu = 0$ and $\sigma = 1$. For CCD, the value of α used in equation 4.3 and equation 4.4 is set to 0.01.

Figure 4.5a and Figure 4.5b depict the non-overlapping (CCD) and overlapping (OCCD) data respectively. The dataset details for CCD and OCCD experiments are provided in Table 4.1.



(a) (b)

Figure 4.5: Simulated datasets for binary classification. (a) Concentric Circle Data (CCD). (b) Overlapping Concentric Circle (OCCD).

Table 4.1: Dataset details of synthetically generated Concentric Circle Data (CCD) and Overlapping Concentric Circle Data (OCCD).

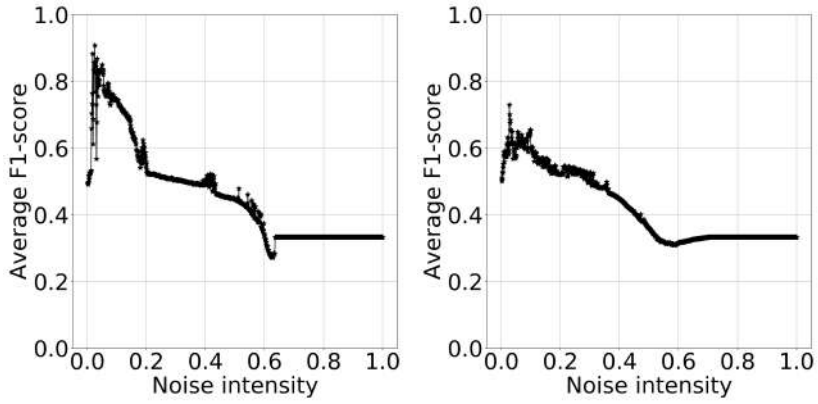
Dataset	CCD	OCCD
# Classes	2	2
# Training instances per class	(2513, 2527)	(2513, 2527)
# Testing instances per class	(1087, 1073)	(1087, 1073)

SR in ChaosNet NL on CCD and OCCD

We first perform a five fold cross-validation to determine the appropriate noise intensities for the CCD and OCCD classification tasks. The noise intensity (ϵ) was varied from 0.001 to 1 with a step size of 0.001. The initial neural activity and discrimination threshold were fixed to $q = 0.21$ and $b = 0.96$ respectively for CCD. In the case of OCCD, q and b are fixed to 0.23 and 0.97 respectively.

- **CCD**: For noise intensity $\epsilon = 0.025$, we get a maximum average F1-score = 0.907 in five fold cross-validation (Figure 4.6a).
- **OCCD**: For noise intensity $\epsilon = 0.027$, we get a maximum average F1-score = 0.73 in five fold cross-validation (Figure 4.6b).

Inspecting Figure 4.6a and 4.6b, we can make the following observations:



(a) (b)

Figure 4.6: SR in ChaosNet NL. (a) Average macro F1-score vs. Noise intensity for Concentric Circle Data (CCD). (b) Average macro F1-score vs. Noise intensity for Overlapping Concentric Circle (OCCD). Both plots indicate local as well as global SR.

1. Average F1-scores achieve global maxima for intermediate amount of noise intensities (ϵ) in both cases (CCD and OCCD) indicating *global SR*.

2. We observe several local maxima of average F1-scores in the above plots which we define as *local stochastic resonance* or *local SR*.

3. There could be multiple local SR in such tasks.

4. It is also possible that multiple global SR exist for different settings of b, q and ϵ.

5. Such a rich behaviour of multiple SR (local and global) is due to the properties of chaos.

4.3.2 SR in ChaosNet NL on real-world task

In this section, we empirically show that SR is exhibited in `ChaosNet` NL even for a real world classification task. The task is to identify spoken digits from a human speaker. Particularly, it is to classify the speech data into classes 0 to 9.

We considered the openly available Free Spoken Digit Dataset (FSDD)[3]. The

[3]https://github.com/Jakobovski/free-spoken-digit-dataset

dataset consists of voice recordings of spoken digits from 0 to 9 of six speakers. Out of the six speakers, for the purposes of this study, we considered the voice samples of only one speaker named 'Jackson'. We have 50 spoken digit recordings for each of the ten classes (0 to 9) sampled at 8 kHz. The audio recordings are trimmed to remove the silence at the beginnings and ends. The maximum and minimum length of data instances are 7038 and 2753 samples respectively. In order to have the same length of data across all instances, we only considered the first 2753 samples. The normalized Fourier coefficients of each data instance was extracted and input to `ChaosNet` NL.

We did a five fold cross-validation using 400 data instances (40 data instances for each class) to determine the optimum noise intensity that yields the best performance. The noise intensity was varied from 0.001 to 1.0 in steps of 0.001. For $q = 0.34$, $b = 0.499$ and a noise intensity = 0.178, we get a maximum macro average F1-score = 0.911. Figure 4.7 depicts the performance of `ChaosNet` NL with varying noise intensities and once again the familiar SR is seen (with local and global SR as defined in the previous section).

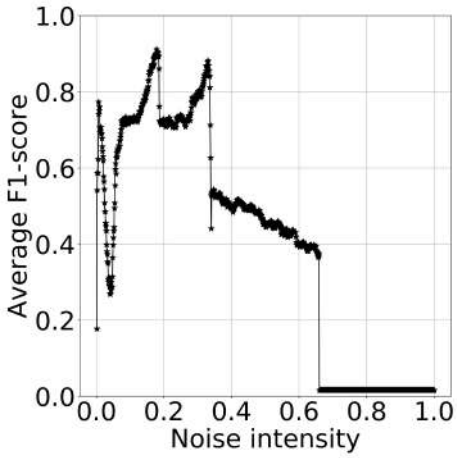


Figure 4.7: SR in ChaosNet NL for spoken digit classification task. Average macro F1-score vs. Noise intensity.

4.3.3 Why is there SR in NL?

Unlike traditional machine learning algorithms, stochastic resonance is not only found in neurochaos learning architecture (`ChaosNet`) but also seen to contribute to a peak

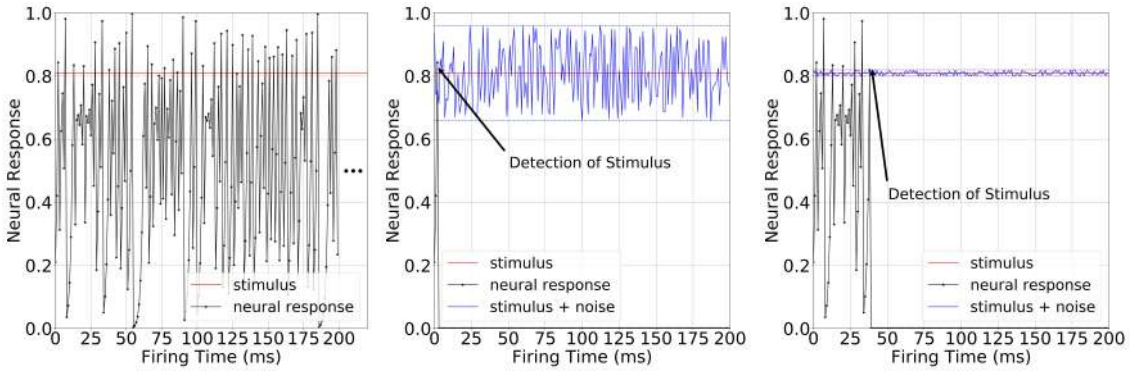
performance in classification tasks. This was demonstrated for NL with a single neuron as well as with multiple neurons, and for both simulated and real-world classification tasks. Even a simple sub-threshold signal detection task using NL exhibited strong SR.

The question that we now like to address is - *why is there SR in NL?* To answer this, we observe a single neural trace (or trajectory) of a single GLS neuron (of NL) corresponding to a single stimulus. Figure 4.8 shows the neural trace (in black) for three scenarios - (a) zero noise ($\epsilon = 0.0$), (b) high noise ($\epsilon = 0.15$) and (c) medium noise ($\epsilon = 0.01$). The target stimulus is indicated in red and the noisy stimulus (target + noise) in blue. The GLS neuron stops firing only when the neural activity matches the stimulus. The stopping time and the noise intensity is inversely proportional. For zero noise, the GLS neuron fires indefinitely without stopping since the probability of the neural activity becoming equal to the stimulus is zero. For a very high noise intensity ($\epsilon \approx 1$), the stopping time is very low (could be even zero) since the huge variation in the noisy stimulus ensures that the neural activity already matches the stimulus. It is only for a medium level of noise that there is a sufficient length of neural trace which enables meaningful features to be extracted for learning. This allows stimuli from different instances and different classes to have diverse length of neural traces with distinct features that enable efficient learning for classification. This is clearly demonstrated in the average F1-scores plotted in Figures 4.6a, 4.6b, and 4.7, where peak performance is obtained for intermediate noise intensities (the classic SR phenomenon).

From a biological point of view, functioning neurons in the brain can neither fire indefinitely nor not fire at all. It seems intuitive to posit that a rich pattern of firing is necessary for learning in the human brain.

4.3.4 Why GLS neurons in NL?

Even though spiking neuronal models could be used in NL, we have chosen GLS neurons for the following reasons:



(a) (b) (c)

Figure 4.8: Why SR in NL yields peak performance? (a) With zero noise, the neural trace is indefinite and hence firing time is undefined (infinite). (b) With very high noise, the neural trace matches the stimulus in a very short time yielding a very low firing time. (c) For medium level of noise, there is a sufficient length of neural trace which enables meaningful features to be extracted for learning. This allows stimuli from different instances and different classes to have diverse length of neural traces with distinct features that enable peak classification performance in NL.

1. GLS neuron is the simplest $1D$ dynamical system that exhibits all the rich properties of chaos [48].

2. Since GLS is a $1D$ dimensional piecewise linear map, the computational cost of implementing GLS neuron is less compared to $3D$ spiking neuronal models such as Hindmarsh Rose model.

3. A single layer of GLS neurons satisfies the Universal Approximation Theorem with a known value of the number of neurons required for the same (Refer chapter 3, section 6.1.1).

4. GLS neurons can implement logical operations such as OR, AND, NOT and XOR [48, 52].

5. GLS maps have been proven to have Shannon optimal lossless data compression properties [49].

For the aforementioned reasons, we stick to GLS neurons in NL for the spoken-digit classification task. However NL fundamentally exhibits SR irrespective of the type of chaotic neuronal model used. This is demonstrated in chapter 6.

4.4 Conclusions

Noise is always contextual. A universal mathematical definition of noise without contextual consideration does not exist. Noise cannot always be treated as an unwanted signal. Stochastic Resonance is one such counter intuitive phenomenon where the constructive role of noise is seen to contribute a performance boost in certain non-linear systems. In this study, we highlight for the first time how stochastic resonance is naturally manifesting in `ChaosNet` neurochaos learning architecture for classification and signal detection.

This study paves the way for potentially unravelling how learning happens in the human brain. There is ample amount of empirical evidence to suggest that chaos is inherent at the neuronal level in the brain, as well as at different spatiotemporal scales [27,28]. Also, given the enormous complexity of brain networks, neuronal noise and interference are unavoidable [29]. In spite of these challenges, the brain does a phenomenal job in learning - currently unmatched by Artificial Neural Networks if power consumption is factored (the brain is known to operate at ≈ 12.6 watts) [109].

Thus, noise has a definite role to play in cognitive systems that can be modeled as non-linear chaotic learning systems (both NL architectures like `ChaosNet` and the human brain). Noise optimizes the selection of diverse chaotic neural traces with distinct features that enables efficient learning. SR seems to be not only inevitable but indispensable in cognitive systems.

The code used for the study of SR in NL is available here: https://github.com/HarikrishnanNB/stochastic_resonance_and_nl.

Chapter 5

Neurochaos Feature Driven Machine Learning

Learning from limited and imbalanced data is a challenging problem in the Artificial Intelligence community. Real-time scenarios demand decision-making from rare events wherein the data are typically imbalanced. These situations commonly arise in medical applications, cybersecurity, natural disasters, financial markets etc. In this chapter, the efficacy of neurochaos feature transformation and extraction for classification in imbalanced learning and learning from limited training samples are studied. The explored datasets in this study revolve around medical diagnosis, banknote fraud detection, environmental applications and spoken-digit classification.

5.1 Introduction

Technological advancements have made a paradigm shift in the evolution of science. A driving force behind this shift is the high storage and computational capacity available in this era. This has given rise to computational techniques for analysis and pattern discovery from data, popularly known as *data-driven science*. Data can be structured or unstructured. In today's world, techniques for data analytics have to make sense from big data. This demands intelligent approaches towards meaningful feature extraction from data and thereby contribute to decision making.

The bedrock for this approach of data-driven science comes under the purview of Artificial Intelligence (AI). Artificial Intelligence (AI) can be defined as an anarchy of methods [3]. This anarchy involves (a) Symbolic AI (logic-based AI), (b) Statistical Learning (Machine Learning), and (c) Sub-Symbolic AI (brain-inspired learning) [4]. The aforementioned verticals of AI have seen a remarkable progress in recent past. On the other hand, these approaches are limited when it comes to decision making under the presence of rare events [110].

Rare events by definition are events whose frequency of occurrence are significantly less compared to more frequently occurring events [111]. Some examples of rare events are: outbreak of a pandemic [112, 113], cyber attack [114], fraudulent transactions [115] and natural disasters [111]. In this scenario, it is critical to provide an early warning response to take appropriate decisions. Hence, the accurate prediction or classification of rare events is of immense importance. Failing to do so can lead to a catastrophe. For example, in the case of COVID-19 pandemic, wrongly classifying a positive person can increase the spread rate of the viral infection. To curb the spread, it is important to correctly classify the infected people and take appropriate measures. Thus, the major challenge in the classification of all rare events is the lack of sufficient data for training. This boils down to the problem of learning from limited samples and imbalanced learning. Imbalanced learning deals with an unequal distribution of data instances amongst distinct classes of a dataset. This can mean either one or more classes in a dataset have relatively greater number of data instances as compared to the remaining classes [110]. Frequently implemented approaches to imbalanced learning can be categorized into (a) pre-processing strategies [110] and (b) cost sensitive learning [116]. The main idea behind pre-processing methods is to optimize the feature space and thereby perform classification by ensemble or algorithmic techniques [110]. On the other hand, cost-sensitive learning assigns a cost (penalty) for every misclassification, with the end goal of minimizing the overall cost [116].

One widely used pre-processing technique is resampling. Resampling restores balance in the sample space by modifying the samples as follows:

1. Over-sampling: This approach addresses resampling by generating new samples for the minority classes. Ex: SMOTE [117].

2. Under-sampling: This type of resampling is done by eliminating the innate samples in the majority classes. Ex: Random Under-Sampling [118].

Combinations of over-sampling and under-sampling methods are known as hybrid methods.

Another common pre-processing technique involves feature selection and feature extraction. Feature selection provides a subset of original input features. Whereas, feature extraction does a functional mapping of input features to generate new features [110]. Principal Component Analysis [119], Dynamic Mode Decomposition [120], Mel-scale Frequency Cepstral Coefficient [121], Non Negative Matrix Factorization [122], Empirical Mode Decomposition [123] etc. are examples of feature extraction algorithms. A detailed study on pre-processing, feature selection and feature extraction are provided in [110]. It should be noted that most of these methods make use of *linear* transformations.

In this chapter, the efficacy of nonlinear feature transformation and feature extraction using NL for classification is empirically studied. This is done by rigorously testing NL on various well-known bench-marking datasets: *Iris, Ionosphere, Wine, Bank Note Authentication, Haberman's Survival, Breast Cancer Wisconsin, Statlog (Heart), Seeds, Free Spoken Digit Dataset, MNIST*. The effectiveness of NL: `ChaosNet` architecture and CFX+ML are studied. In the case of CFX+ML, the NL features are combined with classical ML algorithms such as Decision Tree (DT), Random Forest (RF), AdaBoost (AB), Support Vector Machine (SVM), k-Nearest Neighbors (k-NN) and Gaussian Naive Bayes (GNB). A detailed analysis of this comparative study (NL: chaos-based-hybrid ML vs. stand-alone ML) is carried out in this chapter for both the high and low training sample regime.

The organization of this chapter is as follows: The description of all datasets used for the experiments are provided in section 5.2. The experiments and their corresponding results are available in 5.3. A detailed discussion on the inferences is

given in the 5.4 section. Section 5.5 provides the conclusion of this study.

5.2 Dataset Description

The CFX feature extraction part of NL requires normalization of the dataset and numeric codes for labels. Therefore, to maintain uniformity, all datasets are normalized[1] to $[0, 1]$ for both stand-alone algorithms and their integration with CFX. The labels are renamed to begin from zero in each dataset to ensure compatibility with CFX feature extraction. The rules followed for the numeric coding for the labels of all the datasets are provided in Appendix AA1 (Table A1 - A9). The description of datasets used in this dissertation are provided below:

5.2.1 Iris

Iris [124, 125] aids classification of three Iris plant variants: Iris Setosa, Iris Versicolour, and Iris Virginica. There are 150 data instances in this dataset with four attributes in each data instance: sepal length, sepal width, petal length, and petal width. All attributes are in *cms*. The specified class distribution provided in Table 5.1 is in the following order: (Setosa, Versicolour, Virginica).

5.2.2 Ionosphere

The *Ionosphere* [126,127] dataset enables a binary classification problem. The classes represent the status of returning a radar signal from the Ionosphere. Label 'g' (Good) denotes the return of the radar signal, and label 'b' (Bad) indicates no trace of return of the radar signal. The goal of this experiment is to identify the structure of the Ionosphere using radar signals. This dataset has 351 data instances and 34 attributes. The specified class distribution provided in Table 5.1 is as follows: (Bad, Good).

[1] $X\_norm = \frac{X - \min(X)}{\max(X) - \min(X)}$.

5.2.3 Wine

Wine [126, 128] dataset aims to identify the origin of different wines using chemical analysis. The classes are labeled '1', '2', and '3'. It has 178 data instances and 13 attributes ranging from alcohol, malic acid to hue and proline for the collected samples. The specified class distribution for classes $1, 2$ and 3 are provided in Table 5.1.

5.2.4 Bank Note Authentication

Bank-note Authentication [126, 129] is a binary classification dataset. The classes involve Genuine and Forgery. A Genuine class refers to an authentic banknote denoted by '0', while a Forgery class refers to a forged banknote denoted by '1'. The obtained dataset is from images of banknotes belonging to both classes, taken from an industrial camera. It contains 1372 total data instances. The dataset has four attributes retrieved from the images using wavelet transformation. The specified class distribution provided in Table 5.1 corresponds to (Genuine, Forgery).

5.2.5 Haberman's Survival

Haberman's Survival [126, 130] is a compilation of sections of a study investigating the lifespan of a patient after undergoing a breast cancer surgery. This is a binary classification problem and the dataset provides information for the prediction of the survival of patients beyond five years. Class '1' denotes survival of the patient for five years or longer after the surgery. Class '2' denotes the death of a patient within five years of the surgery. It contains 306 total data instances and three attributes. The specified class distribution provided in Table 5.1 corresponds to classes $(1, 2)$.

5.2.6 Breast Cancer Wisconsin

Breast Cancer Wisconsin [126,131] dataset deals with the classification of the intensity of the breast cancer. Class 'M' refers to a malignant level of infection and class 'B' refers to a benign level of infection. It contains a total of 569 data instances and 31

attributes such as radius, perimeter, texture, smoothness, etc. for each cell nucleus. The specified class distribution provided in Table 5.1 is as follows: (Malignant - M, Benign - B).

5.2.7 Statlog (Heart)

Statlog (Heart) [126] enables differentiation between presence and absence of a heart disease in a patient. Class '1' denotes the absence while class '2' denotes the presence of a heart disease. It contains 270 total data instances and 13 attributes including resting blood pressure, chest pain type, exercise induced angina and so on. The specified class distribution provided in Table 5.1 is as follows: (Absence, Presence).

5.2.8 Seeds

Seeds [126] dataset examines three classes of wheat: Kama, Rosa and Canadian using soft X-ray on the wheat kernels to retrieve relevant properties. It contains 210 total data instances and seven attributes namely compactness, length, width etc. of each wheat kernel. The specified class distribution provided in Table 5.1 is as follows: (Kama, Rosa, Canadian).

5.2.9 Free Spoken Digit Dataset

The *Free Spoken Digit Dataset* [132] is a time-series dataset comprising recordings of six speakers. Each speaker recites numbers from one to nine. For each number, every speaker makes 50 recordings. The speaker chosen for all experiments in this chapter is Jackson. The dataset undergoes preprocessing using a Fast Fourier Transform (FFT) technique. The dataset for speaker Jackson has 500 data instances, and only instances above a threshold of 3000 samples are considered to tackle the varying data length through the dataset. In these data instances, only the first 3005 data samples are examined. Finally, 480 data instances are filtered to feed into the algorithm. The specified class distribution provided in Table 5.1 is as follows: $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$.

5.2.10 MNIST

MNIST is a publicly available computer vision dataset corresponding to handwritten digits from 0 to 9. This is a 10 class classification task. Each image in their respective classes has a dimension of 28 pixels $\times$ 28 pixels. There is a total of $70,000$ images in MNIST database [133].

Table 5.1: Train-Test split in experiments. High training sample regime corresponds to an 80% and 20% split in training and testing respectively. In the Imbalanced data column, 'Y' implies yes and 'N' implies no.

Dataset	**Classes**	**Features**	**Training samples /class**	**Testing samples /class**	**Imbalanced data (Y/N)**
Iris	3	4	(40, 41, 39)	(10, 9, 11)	N
Ionosphere	2	34	(98, 182)	(28, 43)	Y
Wine	3	13	(45, 57, 40)	(14, 14, 8)	Y
Bank Note Authentication	2	4	(614, 483)	(148, 127)	Y
Haberman's Survival	2	3	(181, 63)	(44, 18)	Y
Breast Cancer Wisconsin	2	31	(169, 286)	(43, 71)	Y
Statlog (Heart)	2	13	(117, 99)	(33, 21)	Y
Seeds	3	7	(59, 56, 53)	(11, 14, 17)	N
FSDD	10	3005	(40, 35, 44, 42, 38, 34, 37, 44, 33, 37)	(10, 15, 6, 8, 8, 7, 13, 6, 10, 13)	Y
MNIST	10	784	(5923, 6742, 5958, 6131, 5842, 5421, 5918, 6265, 5851, 5949)	(980, 1135, 1032, 1010, 982, 892, 958, 1028, 974, 1009)	Y

5.3 Experiments & Results

In this section, we evaluate the efficacy of ChaosFEX (CFX) feature engineering on various classification tasks. For this, hybrid models are developed by combining CFX features extracted from the input data with the classical ML algorithms such as AdaBoost, Random Forest, Support Vector Machine, Decision Tree, Gaussian Naive Bayes and k-Nearest Neighbors. The experiments are performed for both low and high training sample regimes. The train-test distribution ($80\% - 20\%$) for each dataset

in the high training sample regime is available in Table 5.1. In low training sample regime, 150 independent random trials for training with $1, 2, \ldots, 9$ data instances per class are considered. The trends of all algorithms in the stand-alone form and their implementation using CFX features are conveyed in this section. For all experiments in this chapter, the following softwares: Python 3.8 and scikit-learn [124] are used.

5.3.1 Hyperparameter Tuning

Every ML algorithm has a set of optimal hyperparameters to be found by hyperparameter tuning. The hyperparameters for all the algorithms with respect to each dataset were tuned using five-fold cross-validation. Table 5.2 provides the set of hyperparameters tuned for all algorithms in this research.

Table 5.2: Tuned hyperparameters for all algorithms: Owing to space constraints, most of the tables are moved to the Appendix A (section A1).

Algorithm	**Acronym**	**Hyperparameters Tuned**	**Reference**
`ChaosNet`	-	q, b, ϵ	Tables 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11and 5.12
Decision Tree	DT	$min\_samples\_leaf$, max_depth, ccp_alpha	Tables A11 and A12 in Appendix A
Random Forest	RF	$n\_estimators$, max_depth	Tables A13 and A14 in Appendix A
AdaBoost	AB	$n\_estimators$	Table A15 in Appendix A
Support Vector Machine	SVM	C, $kernel$	Table A16 in Appendix A
k-Nearest Neighbors	k-NN, KNN	k	Table A17 in Appendix A
Gaussian Naive Bayes	GNB	-	-

`ChaosNet` has three hyperparameters – initial neural activity (q), discrimination threshold (b), and noise intensity (ϵ). Specifics of the same are provided in Table 5.2. In the case of CFX+ML for *Iris, Ionosphere* and *Wine*, both CFX (q, b, ϵ) and ML hyperparameters were tuned. For the remaining datasets, the hyperparameters tuned for `ChaosNet` (q, b, ϵ) were retained and only the ML hyperparameters were tuned.

Table 5.3: ChaosNet results: High training sample regime results for the nine datasets used in the experiments.

Dataset	**Macro-F1 Score (Test Data)**
Iris	1.000
Ionosphere	0.860
Wine	0.976
Bank Note Authentication	0.845
Haberman's Survival	0.560
Breast Cancer Wisconsin	0.927
Statlog (Heart)	0.738
Seeds	0.845
FSDD	0.897
MNIST	0.808

5.3.2 Performance of ChaosNet

5.3.3 Comparative Performance Evaluation

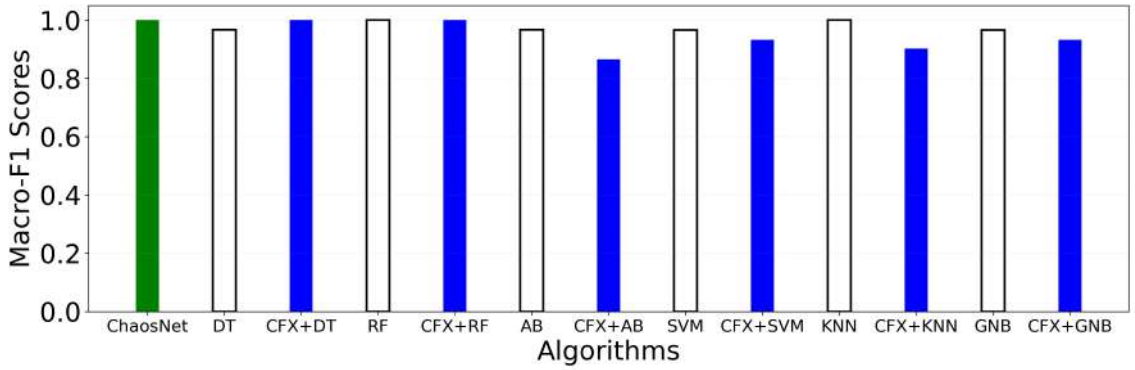
In this section, we represent the results in two formats (a) bar graph and (b) line graph. The comparative results for `ChaosNet`, CFX+ML and stand-alone ML in the high training sample regime are depicted using bar graph. All values plotted in the bar graphs for each dataset are provided in the Appendix A (section A1), Tables A18 - A23. On the other hand, the line graph depicts the comparative performance of `ChaosNet`, CFX+ML and stand-alone ML in the regime of low number of training instances.

Results for Iris

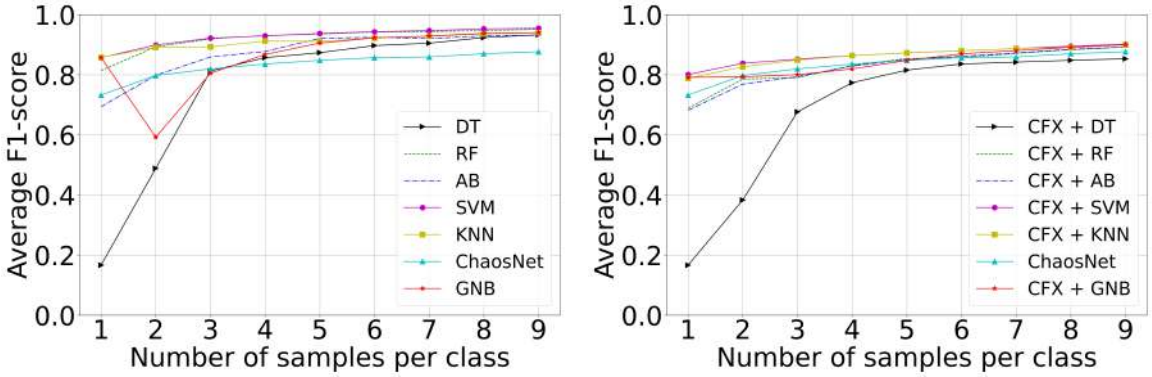
The tuned hyperparameters used and all experimental results for the *Iris* dataset are available in Table 5.4 and Figure 5.1 respectively.

Table 5.4: Hyperparameters for ChaosNet used for *Iris* dataset for high and low training sample regime experiments.

Hyperparameter	**Tuned Value**
q	0.141
b	0.499
ϵ	0.147



(a)



(b) (c)

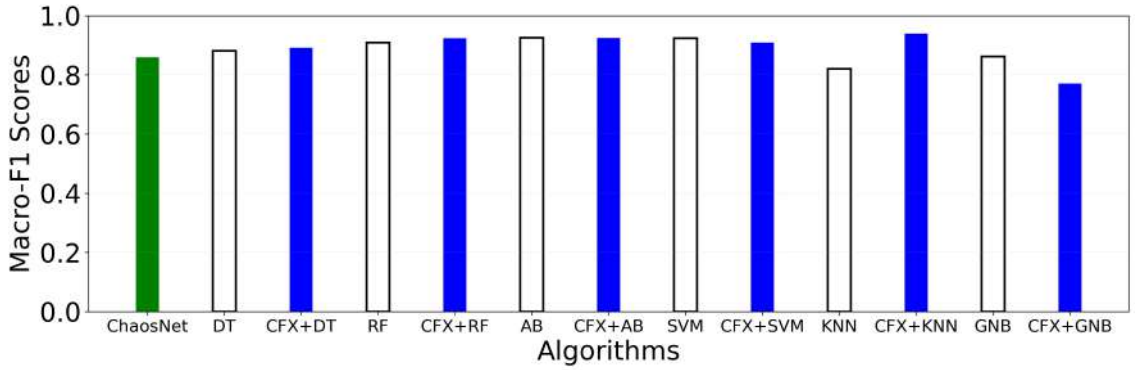
Figure 5.1: *Iris*. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for Ionosphere

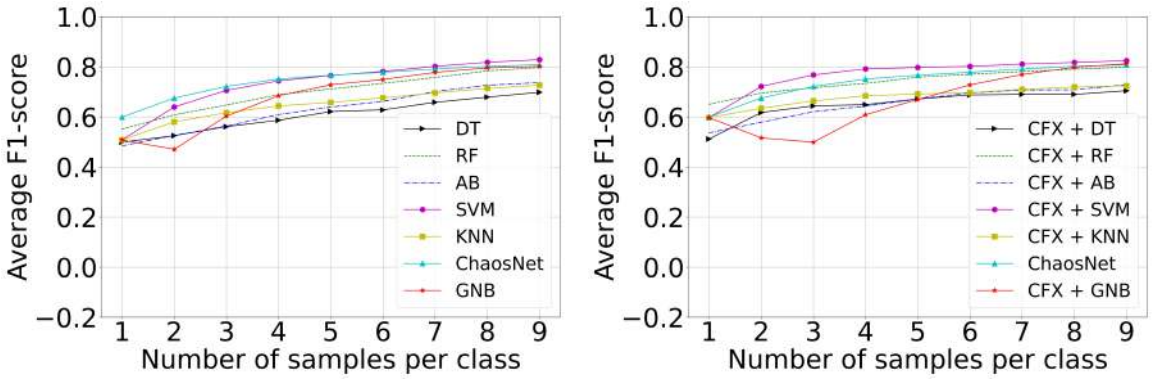
The tuned hyperparameters used and all experimental results for the *Ionosphere* dataset are available in Table 5.5 and Figure 5.2 respectively.

Table 5.5: Hyperparameters for ChaosNet used for *Ionosphere* dataset for high and low training sample regime experiments.

Hyperparameter	**Tuned Value**
q	0.680
b	0.969
ϵ	0.164



(a)



(b) (c)

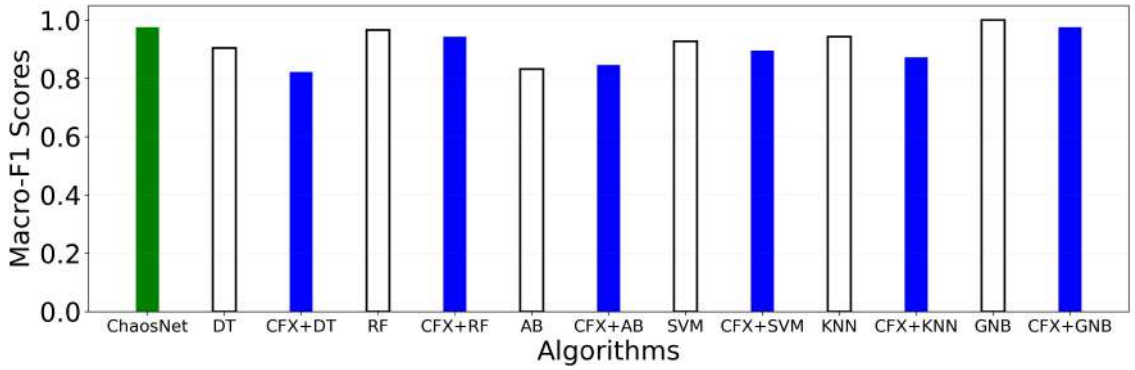
Figure 5.2: *Ionosphere.* (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for Wine

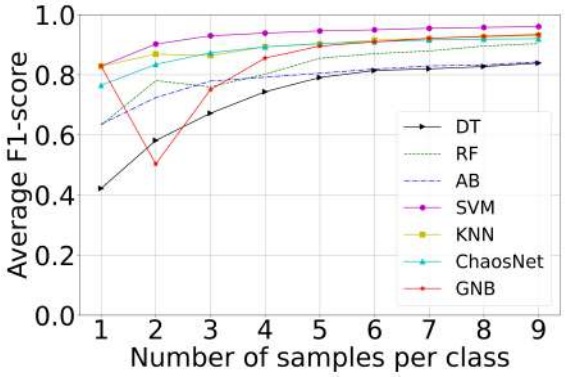
The tuned hyperparameters used and all experimental results for the *Wine* dataset are available in Table 5.6 and Figure 5.3 respectively.

Table 5.6: Hyperparameters for ChaosNet used for *Wine* dataset for high and low training sample regime experiments.

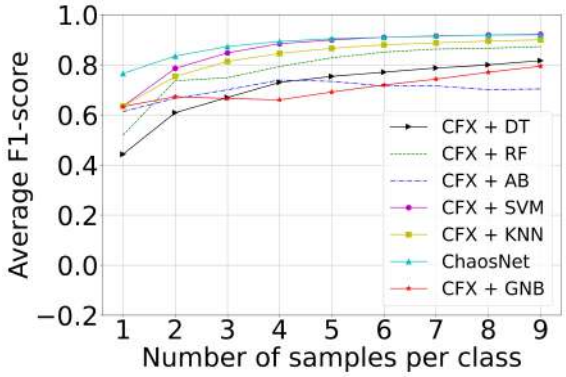
Hyperparameter	**Tuned Value**
q	0.790
b	0.499
ϵ	0.262



(a)



(b)



(c)

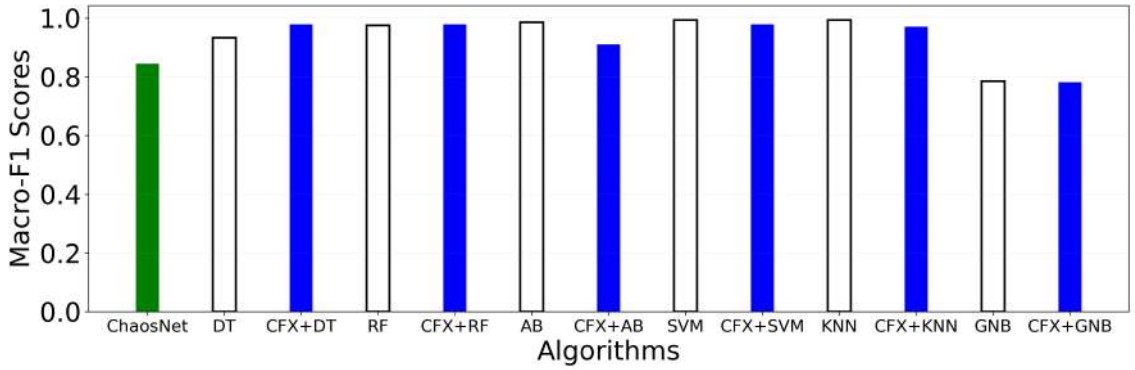
Figure 5.3: *Wine*. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for Bank Note Authentication

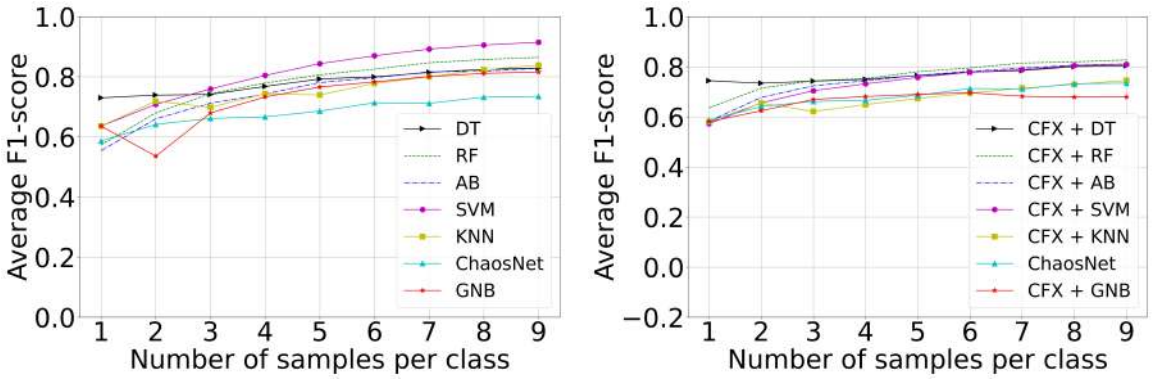
The tuned hyperparameters used and all experimental results for the *Bank Note Authentication* dataset are available in Table 5.7 and Figure 5.4 respectively.

Table 5.7: Hyperparameters for ChaosNet used for *Bank Note Authentication* dataset for high and low training sample regime experiments.

Hyperparameter	**Tuned Value**
q	0.080
b	0.250
ϵ	0.233



(a)



(b) (c)

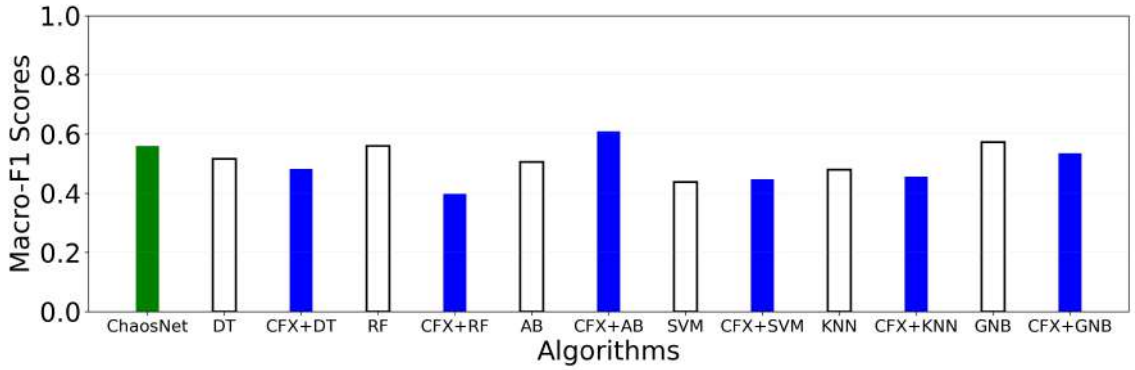
Figure 5.4: *Bank Note Authentication.* (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for Haberman's Survival

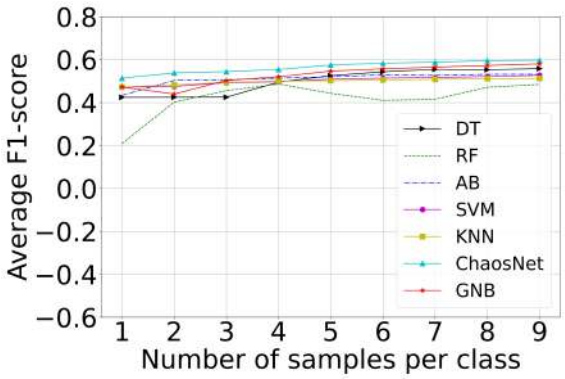
The tuned hyperparameters used and all experimental results for the *Haberman's Survival* dataset are available in Table 5.8 and Figure 5.5 respectively.

Table 5.8: Hyperparameters for ChaosNet used for *Haberman's Survival* dataset for high and low training sample regime experiments.

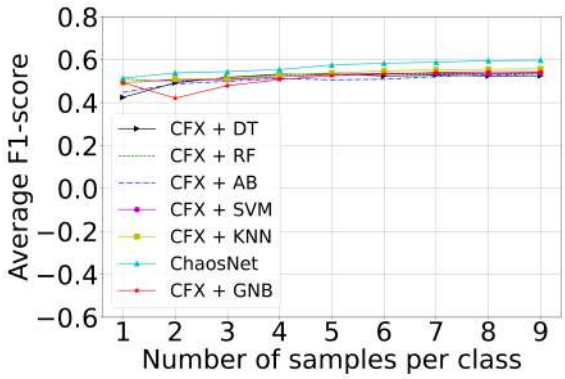
Hyperparameter	Tuned Value
q	0.810
b	0.140
ϵ	0.003



(a)



(b)



(c)

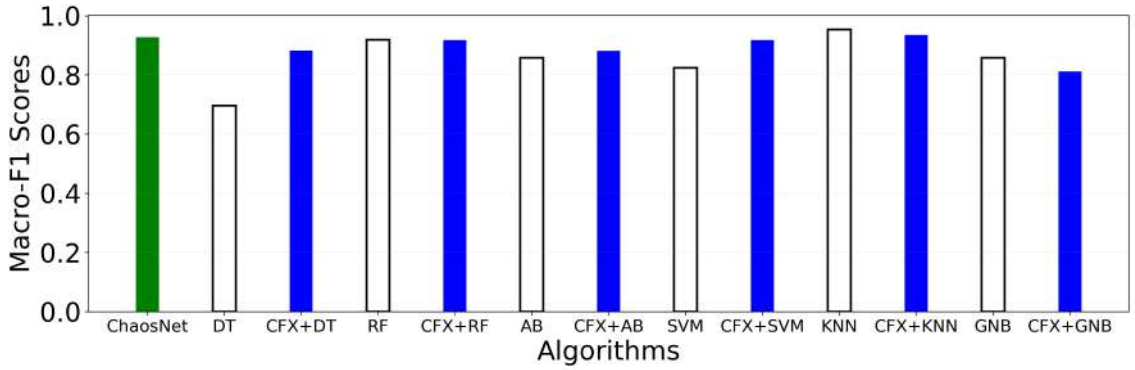
Figure 5.5: *Haberman's Survival.* (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for Breast Cancer Wisconsin

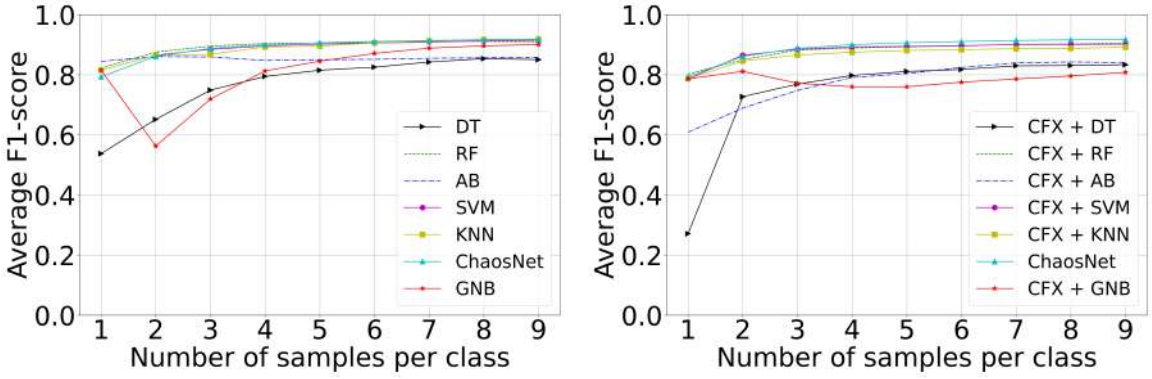
The tuned hyperparameters used and all experimental results for the *Breast Cancer Wisconsin* dataset are available in Table 5.9 and Figure 5.6 respectively.

Table 5.9: Hyperparameters for ChaosNet used for *Breast Cancer Wisconsin* dataset for high and low training sample regime experiments.

Hyperparameter	Tuned Value
q	0.930
b	0.490
ϵ	0.159



(a)



(b) (c)

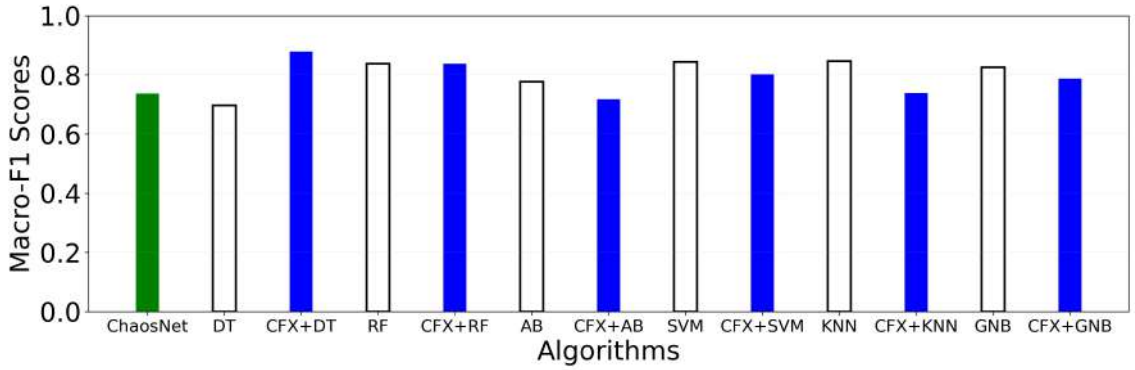
Figure 5.6: *Breast Cancer Wisconsin.* (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for Statlog (Heart)

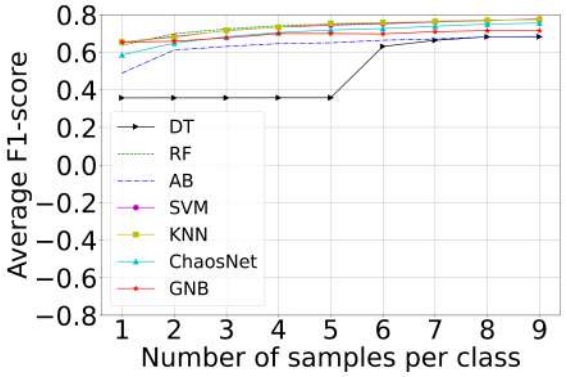
The tuned hyperparameters used and all experimental results for the *Statlog (Heart)* dataset are available in Table 5.10 and Figure 5.7 respectively.

Table 5.10: Hyperparameters for ChaosNet used for *Statlog (Heart)* dataset for high and low training sample regime experiments.

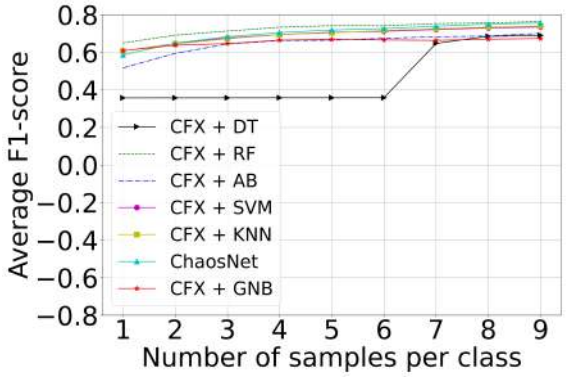
Hyperparameter	Tuned Value
q	0.080
b	0.060
ϵ	0.170



(a)



(b)



(c)

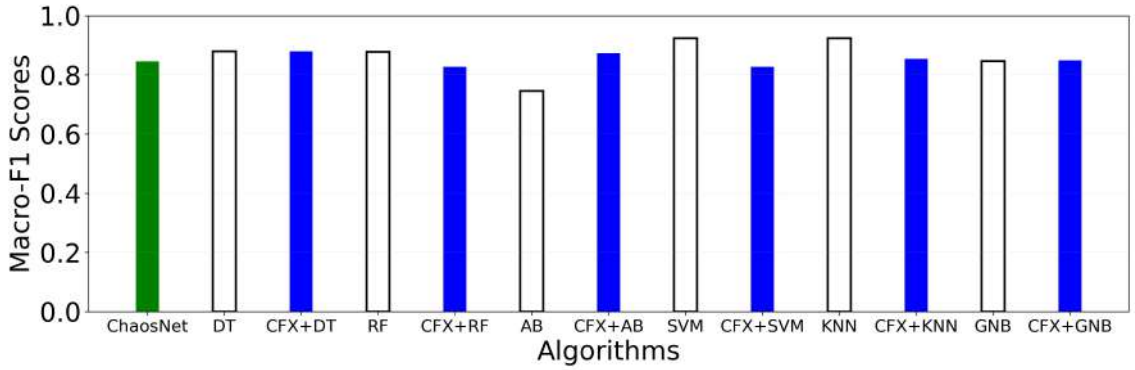
Figure 5.7: *Statlog (Heart).* (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the low training sample regime.

Results for Seeds

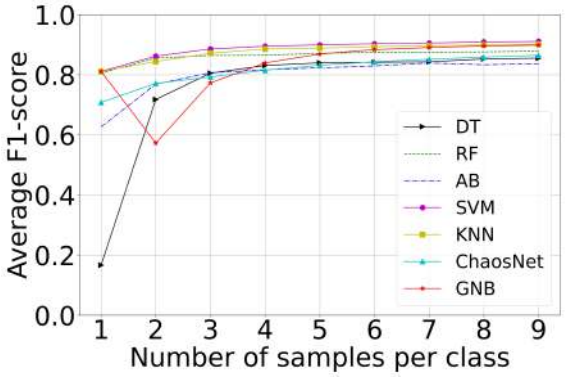
The tuned hyperparameters used and all experimental results for the *Seeds* dataset are available in Table 5.11 and Figure 5.8 respectively.

Table 5.11: Hyperparameters for ChaosNet used for *Seeds* dataset for high and low training sample regime experiments.

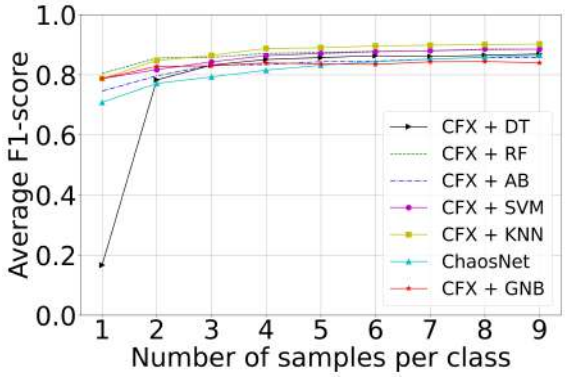
Hyperparameter	**Tuned Value**
q	0.020
b	0.070
ϵ	0.238



(a)



(b)



(c)

Figure 5.8: *Seeds*. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for Free Spoken Digit Dataset (FSDD)

The tuned hyperparameters used and all experimental results for the *FSDD* dataset are available in Table 5.12 and Figure 5.9 respectively.

Table 5.12: Hyperparameters for ChaosNet used for *FSDD* dataset for high and low training sample regime experiments.

Hyperparameter	**Tuned Value**
q	0.340
b	0.499
ϵ	0.178

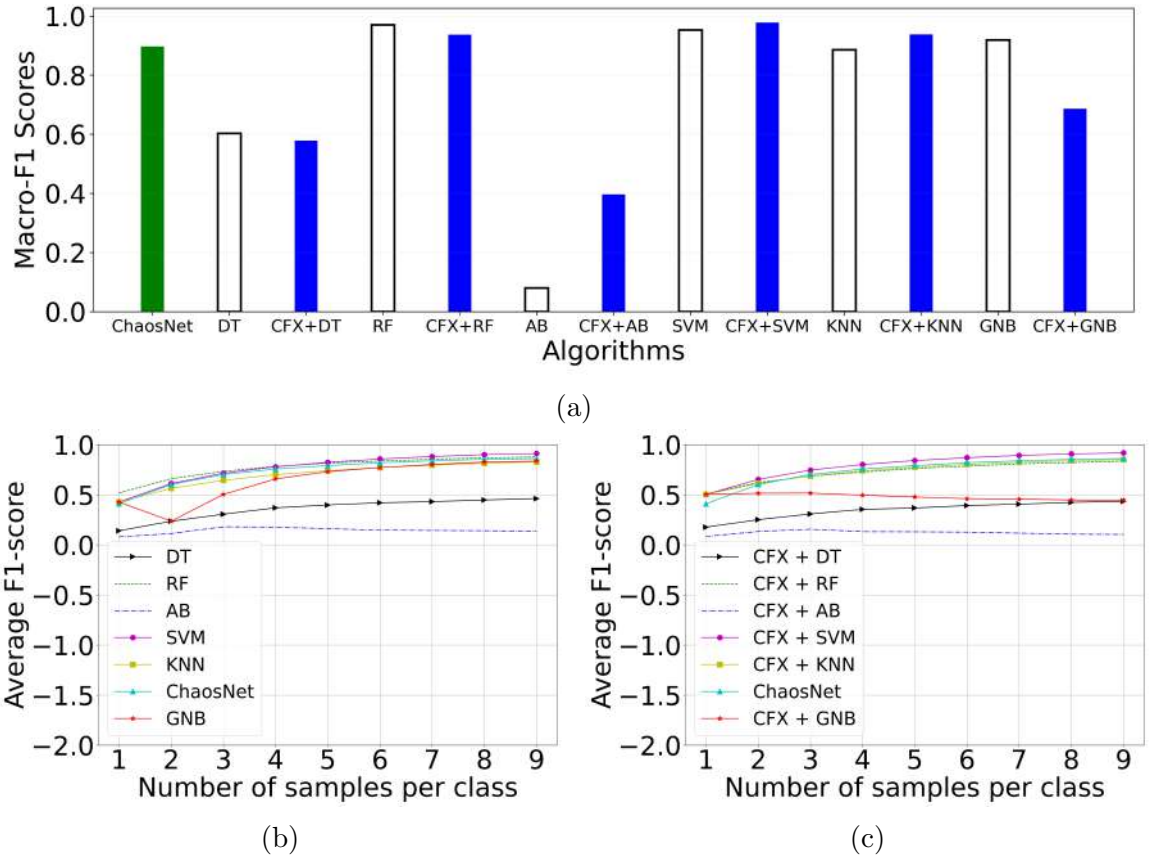


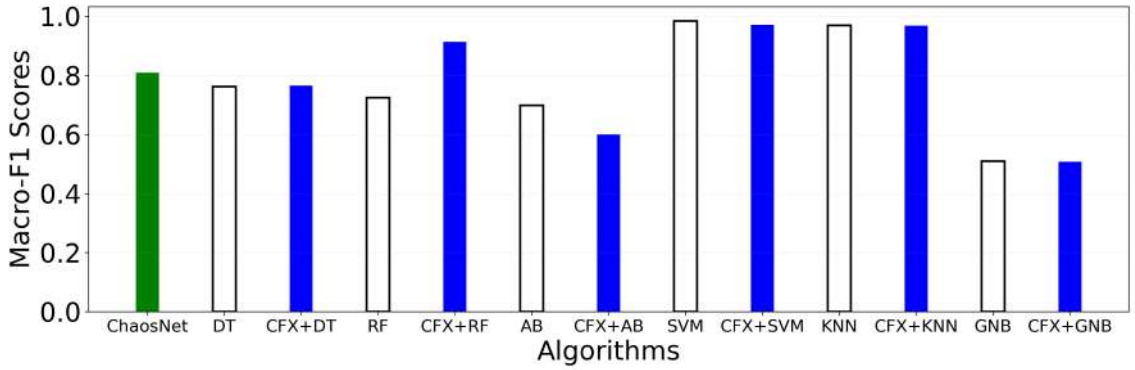
Figure 5.9: *Free Spoken Digit Dataset.* (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

Results for MNIST

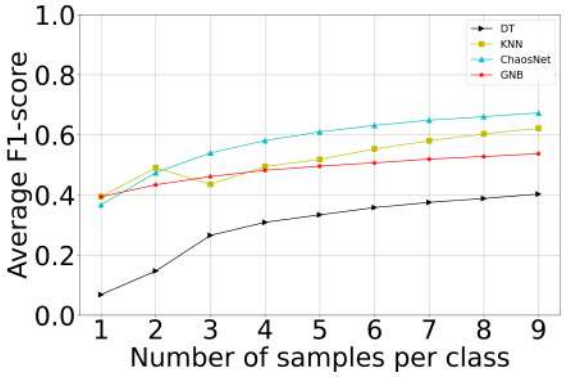
The tuned hyperparameters used and all experimental results for the *MNIST* dataset are available in Table 5.13 and Figure 5.10 respectively. In the case of low training sample regime we used 200 samples per class. This is because of the computational constraints.

Table 5.13: Hyperparameters for ChaosNet used for *MNIST* dataset for high and low training sample regime experiments.

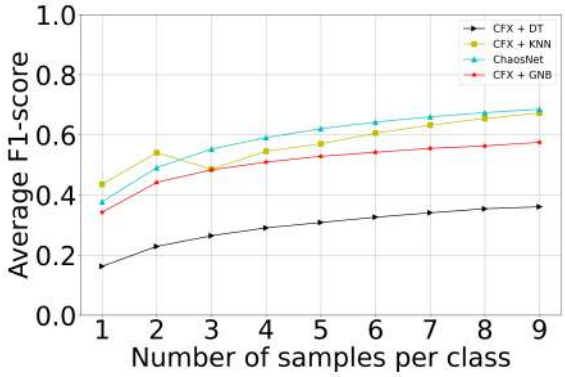
Hyperparameter	Tuned Value
q	0.340
b	0.499
ϵ	0.399



(a)



(b)



(c)

Figure 5.10: *MNIST*. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the regime of low number of training instances. (c) Comparative performance of CFX+ML algorithms in the regime of low number of training instances.

5.4 Discussion

5.4.1 High Training Sample Regime

The overall comparative performance of different algorithms is provided in Table 5.14. In the high training sample regime, the efficacy of using CFX features is evident from Table 5.14 for *Iris, Ionosphere, Haberman's Survival, Statlog (Heart)* and *FSDD*.

While *Iris* is a balanced dataset, *Ionosphere, Haberman's Survival* and *Statlog (Heart)* are imbalanced. Through this, a versatility in the algorithm's ability to perform with both balanced and imbalanced datasets can be established.

The performance boost after using CFX features is calculated as follows:

$$Boost = \left(\frac{F1_{\text{CFX+ML}} - F1_{\text{ML}}}{F1_{\text{ML}}} \right) \cdot 100\%, \tag{5.1}$$

where $F1_{\text{ML}}$ and $F1_{\text{CFX+ML}}$ refers to the macro F1-scores for the stand-alone ML algorithm and the hybrid NL (CFX+ML) algorithm respectively.

Table 5.14: Overall comparative performance of different algorithms in the high training sample regime. Percentages in parenthesis indicate the *Boost* computed using

$$Boost = \left(\frac{F1_{\text{CFX+ML}} - F1_{\text{ML}}}{F1_{\text{ML}}} \right) \cdot 100\%, \tag{5.3}$$

Dataset	`ChaosNet` (Macro F1-Score)	Best Algorithm	Best Macro F1-Score
Iris	**1.0**	`ChaosNet`, RF, KNN **CFX+DT (3.41%)**, **CFX+RF (0%)**	**1.0**
Ionosphere	0.860	**CFX+KNN (14.37%)**	**0.939**
Wine	0.976	GNB	**1.0**
Bank Note Authentication	0.845	SVM, KNN	**0.993**
Haberman's Survival	0.560	**CFX+AB (20.59%)**	**0.609**
Breast Cancer Wisconsin	0.927	k-NN	**0.954**
Statlog (Heart)	0.738	**CFX+DT (25.97%)**	**0.878**
Seeds	0.845	KNN	**0.924**
FSDD	0.897	**CFX+SVM (2.73%)**	**0.978**
MNIST	0.808	SVM	**0.983**

5.4.2 Low Training Sample Regime

Table 6.1 indicates the overall performance of all datasets in the regime of low number of training instances after employing the CFX features. A ✓refers to a performance boost of ML algorithms after using CFX features as computed using Eq. 5.3. CFX features have shown an evident improvement in performance for *Ionosphere, Haberman's Survival, Statlog (Heart), Seeds* and *FSDD*. 150 independent random trials of training in the regime of low number of training instances with $1, 2, \ldots, 9$ samples per class are performed. The is done to ensure that the model does not overfit.

Table 5.15: Overall performance boost using CFX features in the regime of low number of training instances. A ✓refers to a performance boost of ML algorithms after using CFX features. We report $(Minimum, Maximum)$ of *Boost* values only in these instances.

Dataset	DT	RF	AB	SVM	KNN	GNB
Iris						
Ionosphere	✓ (0.88%, 17.63%)	✓ (0.15%, 18.01%)	✓ (0.83%, 10.71%)	✓ (1.15%, 17.60%)	✓ (0.80%, 17.60%)	
Wine						
Bank Note Authentication						
Haberman's Survival	✓ (0.0%, 21.59%)	✓ (8.43%, 144.38%)		✓ (2.19%, 6.43%)	✓ (3.16%, 8.85%)	
Breast Cancer Wisconsin						
Statlog (Heart)	✓ (0.0%, 1.05%)		✓ (0.69%, 5.58%)			
Seeds	✓ (0.0%, 9.08%)	✓ (0.01%, 0.84%)	✓ (1.56%, 19.02%)		✓ (0.04%, 0.43%)	
FSDD				✓ (0.76%, 17.63%)	✓ (2.76%, 17.63%)	

5.4.3 Inferences based on consistency of algorithms across all datasets

The consistency of an algorithm can be inferred by evaluating the range of minimum and maximum macro F1-scores produced by it in the high training sample regime. Table 5.16 shows the ranges of macro F1-scores of different algorithms, measured across all nine datasets used in the research. The provided format for the range of macro F1-scores in Table 5.16 is [Minimum, Maximum]. `ChaosNet` ranks second in the least difference of 0.44 between the maximum and minimum macro F1-scores. This observation owes to the consistency and tolerance towards dataset diversity in `ChaosNet`. In algorithms such as AdaBoost (AB) and Support Vector Machine (SVM), a relatively low difference between F1-scores is realised after the usage of CFX features. Gaussian Naive Bayes (GNB) shows the least difference of 0.428 between F1-scores. Along datasets of different domains considered in this study, the performance of `ChaosNet` can be seen as comparable with GNB.

Table 5.16: Depiction of consistency of algorithms across all nine datasets used in this study (high training sample regime). The minimum and maximum macro F1-scores for each algorithm corresponding to all nine datasets are provided in the format: $[Minimum, Maximum]$.

Stand-alone ML		**CFX + ML**	
Algorithm	**F1-Score Range**	**Algorithm**	**F1-Score Range**
ChaosNet	[0.56, 1.0]	-	-
DT	[0.516, 0.967]	CFX + DT	[0.482, 1.0]
RF	[0.56, 1.0]	CFX + RF	[0.398, 1.0]
AB	[0.08, 0.985]	CFX + AB	[0.397, 0.925]
SVM	[0.437, 0.993]	CFX + SVM	[0.447, 0.978]
KNN	[0.48, 1.0]	CFX + KNN	[0.455, 0.971]
GNB	[0.572, 1.0]	CFX + GNB	[0.535, 0.976]

5.4.4 Limitations

With the current implementation of the ChaosFEX algorithm, computation of image datasets is a costly process. This may limit the use of NL architectures in practical sit-

uations involving images. This can be overcome by incorporating parallel computing methods in the implementation part of the algorithm.

Furthermore, NL architectures have been based on certain assumptions. One assumption revolves around the separability of data. NL assumes that applying a nonlinear chaotic transformation will result in separable data suitable for classification, which may not be true in all cases. As it stands, the input layer of NL treats input attributes as independent of each other. It establishes no connection between the neurons for each input attribute. This limitation can be addressed by using coupled chaotic neurons in the input layer. Currently, we have not considered multi-layered NL (Deep-NL) which could significantly enhance performance (by careful choice of coupling between adjacent layers). Another limitation of the NL architectures is the lack of a principled approach to tune the best hyperparameters for a classification task. Currently, cross-validation experiments are used to tune the hyperparameters. The connection between the degree of chaos as measured by lyapunov exponent [134]) and learnability is also worth exploring for future research.

5.5 Conclusion

Decision making under the presence of rare events is a challenging problem in the ML community. This is because rare events have limited data instances, and this problem boils down to imbalanced learning. In this work, we have evaluated the effectiveness of *ChaosFEX (CFX)* feature transformation used in *Neurochaos Learning (NL)* architectures for imbalanced learning. Nine benchmark datasets were used in this study to bring out this evaluation. Seven out of nine datasets used are imbalanced (Refer to Table 5.1). This paper accomplishes a comparative study on the performance of NL architecture: `ChaosNet` and CFX+ML with classical Machine Learning (ML) algorithms. The obtained results reflect an evident performance boost in terms of macro F1-score after a *nonlinear chaotic transformation* (ChaosFEX or CFX features). Additionally, the efficacy of CFX features can be observed in five out of nine balanced and imbalanced datasets in the high training sample regime, with a boost ranging

from **2.73%** (*Free Spoken Digit Dataset*) to **25.97%** (*Statlog - Heart*). In the regime of low number of training instances, the integration of CFX features has boosted the performance of classical ML algorithms in five datasets, from a total of nine datasets. A maximum boost of **144.38%** on the *Haberman's Survival* dataset using CFX+RF is obtained. Refer to Table 5.14 and 6.1 for the detailed performance boost using CFX features.

NL is a unique combination of chaos and noise-enhanced classification. The enormous flexibility of NL offers endless possibilities for development of novel NL: chaos-based-hybrid ML models that suit the application at hand. As new ML algorithms get invented, they can be readily combined in the NL framework. We forsee exciting combinations of CFX with DL and other ML algorithms in the future.

Chapter 6

Properties of Neurochaos Learning

This chapter investigates the properties of NL and summarises them. These include: (1) an input layer of GLS neurons in NL satisfies the Universal Approximation Theorem, (2) NL supports the incorporation of chaotic biological neuronal models, (3) superior performance in the limited training sample regime when compared to ML algorithms, (4) the flexibility of NL allows for developing hybrid NL-ML algorithms, (5) robustness to additive parametric noise, (6) exhibits stochastic resonance at the level of individual as well as layer of neurons, (7) the rate of catastrophic forgetting in NL is less compared to DL algorithms, and (8) the features extracted from the input layer of NL preserves cause-effect relationships.

6.1 NL Satisfies Universal Approximation Theorem (UAT)

A one layer NL (input layer) with exactly L chaotic neurons can approximate L real valued samples of the function $f(n)$. This is because of the topological transitivity property of chaos and due to the existence of countably infinite number of dense orbits. Unlike the UAT for Neural Networks, NL has a bound on the number of chaotic neurons required to approximate a function. The detailed proof is provided below:

6.1.1 UAT for NL

Theorem: Let $f(n)$ be a discrete time real valued function having a finite support L. The NL architecture consisting of a single layer with L chaotic neurons can approximate[1] $f(n)$. Assuming that we use a chaotic $1D$ map C_i for the i-th neuron in NL, and given any desired error $\epsilon > 0$, we have:

$$d(f, C) = \sum_{i=1}^{L} |f(i) - C_i^{N_i}(q)| < \epsilon, \tag{6.1}$$

where q is the initial neural activity for all the neurons in NL, N_i is the firing time of the i-th chaotic neuron and C_i is the $1D$ chaotic map with the chaotic trajectory starting from 'q' a dense orbit.

Proof. Design an NL with one layer with exactly L chaotic neurons. Let the initial neural activity of each neurons be initialized with q and let the input to this NL be the L real-valued samples of the function $f(n)$ which act as stimuli for the corresponding L chaotic neurons.

Now, for a given $\epsilon > 0$, we can always construct a neighbourhood of stimulus $I_k = (f(k) - \eta, f(k) + \eta)$, $0 < \eta < \frac{\epsilon}{2L}$ such that $C_k^{N_k}(q) \in I_k$ for the k-th chaotic neuron of NL. This is always possible because of the topological transitivity property of chaos defined in Section 3.3 and since the chaotic trajectory starting from initial value q is dense. The topological transitivity property guarantees the chaotic firing to reach the η neighbourhood (I_k) of stimulus in finite number of iterations (N_k) for the dense orbit starting from 'q'. For any given ϵ, the following is true:

$$\begin{aligned} d(f, C) &= \sum_{i=1}^{L} d(f(i), C_i^{N_i}) = \sum_{i=1}^{L} |f(i) - C_i^{N_i}(q)| \\ &< \sum_{i=1}^{L} 2\eta = L(2\eta) < L(2\frac{\epsilon}{2L}) = \epsilon. \end{aligned}$$

[1]For quantifying this approximation, we use the sum of absolute differences as the distance metric. In other words, for any two real-valued vectors $V, W \in \mathbb{R}^m$, $d(V, W) = \sum_{i=1}^{m} |V_i - W_i|$.

□

Note that $\eta < \frac{\epsilon}{2L}$ since $C_i^{N_i}(q) \in (f(i)-\eta, f(i)+\eta)$ because N_i is the firing time for C_i and the orbit is dense. Hence, the set of chaotic neurons $\{C_i\}$ that constitute the input layer of NL can always approximate the function $f(n)$ with an ϵ error bound. This theorem holds true for NL constructed with chaotic neurons that satisfies the topological transitivity property and has a dense orbit. Further, having a single dense orbit implies countably infinite number of dense orbits.

6.2 NL Supports Chaotic Biological Neuronal Models

Neuromorphic computing systems are biologically inspired with an aim to understand the rich structure and behaviour of biological neural networks so that novel learning architectures can be designed in both software and hardware. The flexibility of NL supports the incoporation of biological neuronal models in NL architecture.

6.2.1 SR in NL using Hindmarsh Rose Neuronal Model

So far we have considered NL to be constituted of only $1D$ chaotic GLS maps as neurons. However, this is not necessary and NL could employ other chaotic maps and flows as neurons. In this section, we incorporate Hindmarsh Rose Neuronal Model in NL. The Hindmarsh Rose neuronal model was constructed by J. L. Hindmarsh and R. M. Rose in their 1984 paper titled '*A model of neuronal bursting using three coupled first order differential equations*' [79]. Hindmarsh and Rose used voltage clamp experiments on the brain of a pond snail (*Lymnaea*) [79, 135]. From the neuronal cell data, they observed a bursting behaviour which outlived the stimulus when depolarized by a short electrical pulse. The Hindmarsh Rose Model exhibits different behaviours such as bursting, spiking, and chaotic regime for various parameters [136].

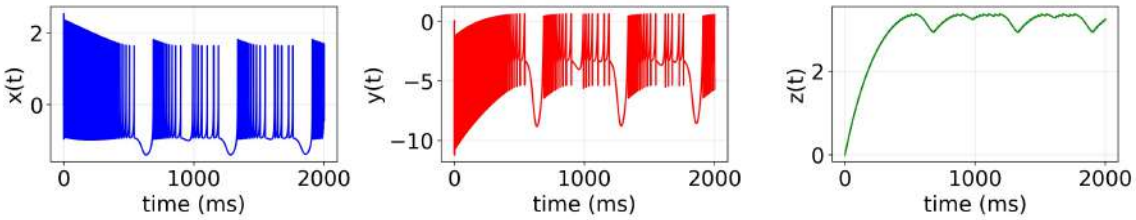


Figure 6.1: Hindmarsh Rose spiking neuronal model. The dynamics of $x(t)$, $y(t)$, and $z(t)$ for $r = 0.0021$, $s = 4$, and $I = 3.28$.

The governing equations of the Hindmarsh Rose model are as follows:

$$\frac{dx}{dt} = y - ax^3 + bx^2 - z + I, \tag{6.2}$$

$$\frac{dy}{dt} = c - dx^2 - y, \tag{6.3}$$

$$\frac{dz}{dt} = r\Big(s(x - x_1) - z\Big). \tag{6.4}$$

The $x(t)$, $y(t)$, and $z(t)$ correspond to membrane potential, recovery variable and adaptation variable respectively. There are eight parameters in this set of non-linear ordinary differential equations. These parameters are: a, b, I (applied current), c, d, r, s, and x_1. The model exhibits different behaviours such as regular and chaotic for different parameters. The parameters of the system are set to the following values: $a = 1$, $b = 3$, $c = 1$, $d = 5$, $x_1 = -1.6$. The values of I, r and s are critical in controlling the behaviour of the system. We choose $I = 3.28$, $r = 0.0021$, and $s = 4$ which makes the Hindmarsh Rose model exhibit chaotic behaviour [136]. The trajectory of Hindmarsh Rose model for this setting is provided in Figure 6.1 for the initial conditions set to $x(0) = 0, y(0) = 0, z(0) = 0$. We normalize $x(t)$ provided in Figure 6.1 so as to lie in the range $[0, 1]$. The normalized trajectory is used in the NL architecture. In order to demonstrate the presence of SR in NL using Hindmarsh Rose neuronal model we use the prey-predator dataset as described in chapter 4, section 4.2.

We performed a five fold cross-validation to find the best noise intensity for the above mentioned parameters of Hindmarsh Rose model (in the chaotic regime) and a discrimination threshold = 0.89. The noise intensity was varied from 0.01 to 1 with

a step size of 0.01. For a noise intensity (ϵ) = 0.14, we get an average macro F1-score = 1.0 for the five fold cross-validation. The SR plot is provided in Figure 6.2.

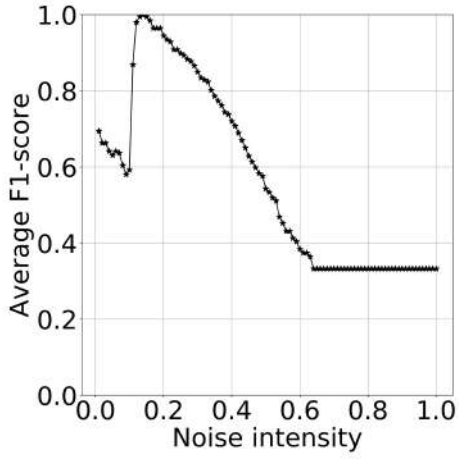


Figure 6.2: SR using Hindmarsh Rose Neuronal model in NL for the classification of prey-predator dataset provided in chapter 4, section 4.2.

Figure 6.2 demonstrates the occurrence of SR phenomenon even when the GLS neurons in NL are replaced by a computational model of biological neuron (Hindmarsh Rose). Thus, demonstrating the inherent SR effect in NL.

6.3 Learning from Limited Samples

Learning from limited training samples is a challenging problem in the ML community. It has been found that using NL features provides a superior performance when training dataset is limited. This has been rigorously shown in chapter 5, section 5.3. The results for the low training sample regime has been reproduced below:

Table 6.1: Overall performance boost using CFX features in the regime of low number of training instances. A ✓refers to a performance boost of ML algorithms after using CFX features. We report $(Minimum, Maximum)$ of *Boost* values only in these instances.

Dataset	DT	RF	AB	SVM	KNN	GNB
Iris						
Ionosphere	✓ (0.88%, 17.63%)	✓ (0.15%, 18.01%)	✓ (0.83%, 10.71%)	✓ (1.15%, 17.60%)	✓ (0.80%, 17.60%)	
Wine						
Bank Note Authentication						
Haberman's Survival	✓ (0.0%, 21.59%)	✓ (8.43%, 144.38%)		✓ (2.19%, 6.43%)	✓ (3.16%, 8.85%)	
Breast Cancer Wisconsin						
Statlog (Heart)	✓ (0.0%, 1.05%)		✓ (0.69%, 5.58%)			
Seeds	✓ (0.0%, 9.08%)	✓ (0.01%, 0.84%)	✓ (1.56%, 19.02%)		✓ (0.04%, 0.43%)	
FSDD				✓ (0.76%, 17.63%)	✓ (2.76%, 17.63%)	

6.4 Hybrid NL-ML models

The flexibility of NL allows developing hybrid NL-ML models. NL features such as firing time, firing rate, energy, entropy can be passed to ML classifiers. This has been discussed in detail in chapter 5, section 5.3. In future work, the efficacy of NL features in reinforcement learning shall be evaluated.

6.5 Robustness to Additive Noise to Learned Parameters

6.5.1 ChaosNet in the presence of Noise

One of the key ways to identify the robustness of a ML algorithm is by testing its efficiency in classification or prediction in the presence of noise. Robustness needs to be tested under the following scenarios: noisy test data, training data attributes affected by noise, inaccurate training data labels due to the influence of noise and noise affected model parameters[2]. Amongst these, *noise affected model parameters* is

[2]Hyperparameters are rarely subjected to noise and hence we ignore this scenario. It is always possible to protect the hyperparameters by using strong error correction codes.

the most challenging since it can significantly impact performance of the algorithm. We consider a scenario where the parameters learned by the model while training are passed through a noisy channel. As an example, we compare the performance of `ChaosNet` architecture and two layer neural network (DL architecture) for Iris data. The parameters for the both algorithms are modified by Additive White Gaussian Noise (AWGN) with zero mean and increasing variance. The Iris dataset consists of three classes with four attributes per data instance. We considered the specific case of training with only seven samples per class.

Parameters settings for `ChaosNet` *for Iris data:* Corresponding to the three classes for the Iris data, the output layer of `ChaosNet` consists of three nodes O_1, O_2, and O_3 which store the mean representation vectors. Since each representation vector contains four components (corresponding to the four input attributes), the total number of learnable parameters are 12. These parameters are passed through a channel corrupted by AWGN with zero mean and increasing standard deviation (from 0.0001 to 0.0456). This results in a variation of the Signal-to-Noise Ratio (SNR) from -6.98 dB to 45.36 dB.

Parameter settings for the two layer neural network for Iris data: The two layer neural network has four nodes in the input layer, four neurons in the hidden layer and three neurons in the output layer. Thus, the total number of learnable parameters (weights and biases) for this architecture are: $(4 \times 4) + 4 + (3 \times 4) + 3 = 35$. These parameters are passed through a channel corrupted by AWGN with zero mean and increasing standard deviation (from 0.0003 to 0.15). This results in a variation of the Signal-to-Noise Ratio (SNR) from -8.78 dB to 45.48 dB.

Parameter Noise Analysis: The results corresponding to additive gaussian parameter noise for `ChaosNet` architecture and two layer neural network are provided in Figure 6.3 and Figure 6.5 respectively. Figure 6.4 and Figure 6.6 depict the variation of SNR (dB) with σ of the AWGN for the two algorithms. Firstly, we can observe that the performance of `ChaosNet` architecture degrades gracefully with increasing σ, whereas for the two layer neural network there is a sudden and drastic fall in performance (as $\sigma > 0.02$.). A closer observation of the variation of accuracy for different

SNRs is provided in Table 6.2. In the case of `ChaosNet` architecture, for the SNR in the range 4.79 dB to 45.36 dB, the accuracy remains unchanged at 95.83%. Whereas for the same SNR, the accuracy for the two layer neural network reduced from 67.5% to 31.66%. This preliminary analysis on parameter noise indicates the better performance of our method when compared with two layer neural network. However, more extensive analysis will be performed for other datasets and other noise scenarios in the near future.

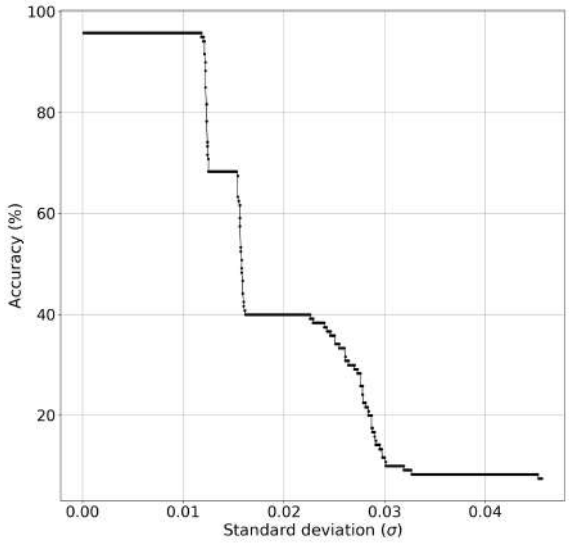


Figure 6.3: Parameter noise analysis: Accuracy of ChaosNet for Iris data in the presence of AWGN noise with zero mean and increasing standard deviation (σ).

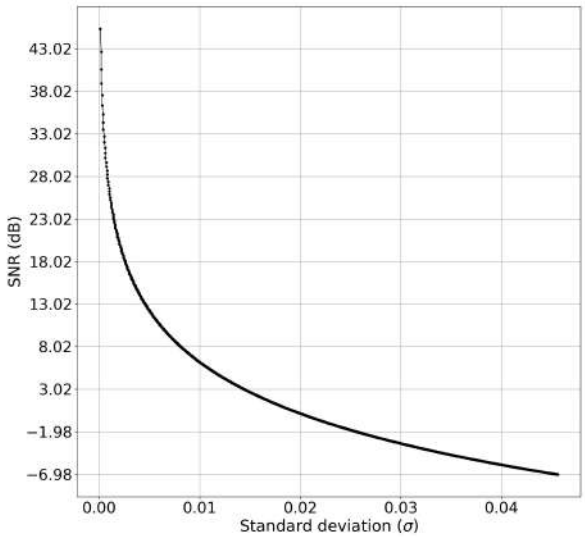


Figure 6.4: SNR vs. standard deviation (σ) of AWGN corresponding to Figure 6.3.

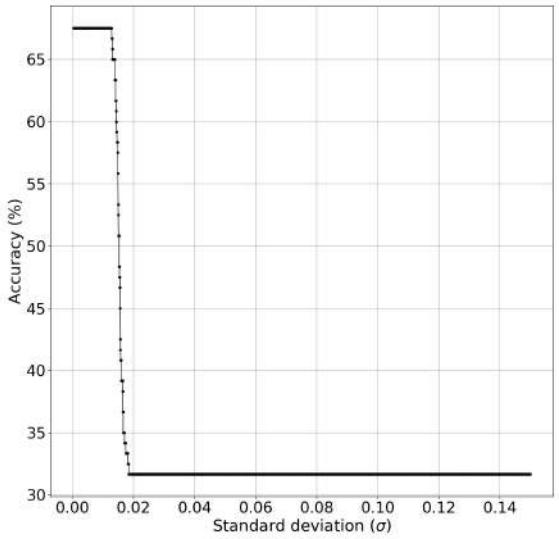


Figure 6.5: Parameter noise analysis: Accuracy of two layer neural network for Iris data in the presence of AWGN noise with zero mean and increasing standard deviation (σ).

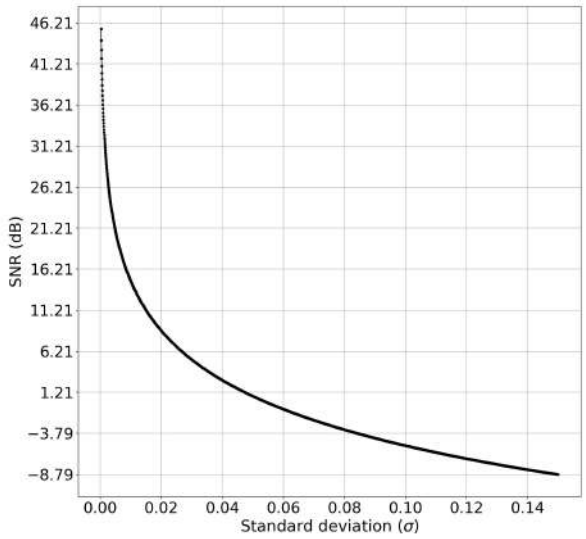


Figure 6.6: SNR vs. standard deviation (σ) of AWGN corresponding to Figure 6.5.

Table 6.2: Parameter Noise Analysis for AWGN noise: comparison of accuracies for ChaosNet architecture and two layer neural network for various SNR ranges.

SNR (dB) ChaosNet)	4.79 – 45.36	4.61 – 4.76	4.41 – 4.59	4.27 – 4.39
Accuracy (%) (ChaosNet)	95.83	95.00	81.66 – 94.16	70.83 – 78.33
SNR (dB) (two layer NN)	12.56 – 45.48	11.94 – 12.36	11.57 – 11.91	10.65 – 11.54
Accuracy (%) (two layer NN)	67.50	65.00	60.00 – 63.33	40.83 – 59.16

6.6 NL exhibits Stochastic Resonance

The working of NL can be attributed to the unique combination of Chaos and Stochastic Resonance. NL provides an optimal performance for an intermediate value of noise intensity (ϵ). The value of the noise intensity depends on the dataset. Chapter 4 extensively deals with SR property found in NL.

6.7 Neurochaos Lifelong Learning

The success of DL and ML algorithms is under the purview of a close world assumption i.e, the test data distribution matches the train data distribution. This assumption

limits DL to be brain inspired, because humans have the ability to continuously learn new tasks over time without forgetting what has previously learned. This aspect of learning is popularly known as continual learning. In the case of DL, the progressive learning of new tasks adversely affects the previously learned tasks. This problem of forgetting the previously learned tasks upon arrival of new tasks is termed as catastrophic forgetting [137, 138]. Inorder to address catastrophic forgetting, novel learning frameworks have to be developed that support *Lifelong Learning* (LL) [139]. LL was first proposed by Sebastian Thrun and Tom. M. Mitchell around 1995 [140]. The inspiration for LL comes from the effective LL mechanism found in humans and animals. Inspired from the neurophysiological aspects of LL, several ML and neural networks approaches were developed. A detailed review of the methods are provided in [141].

6.7.1 Lifelong Learning

In LL framework, the learner has learned a sequence of N tasks given as T_1, T_2, ..., T_N at a given time point. These are tasks already learnt by the learner. When a new task T_{N+1} and its corresponding datasets have arrived, the learner uses the past knowledge from the knowledge base (KB) to learn the new task T_{N+1}. This new task can be identified or given to the learner [139]. In a nutsell, the LL framework involves the following:

1. A continuously learning environment.
2. Knowledge base (KB) of the previously learned tasks.
3. Identify and learn the new tasks based on learning of previous tasks.
4. Update the KB.

6.7.2 Datasets

MNIST

MNIST is a publicly available computer vision dataset corresponding to handwritten digits from 0 to 9. This is a 10 class classification task. Each image in their respective classes has a dimension of 28 pixels $\times$ 28 pixels. There is a total of $70,000$ images in MNIST database [133].

Spoken Digit Dataset

The *Free Spoken Digit Dataset* [132] is a time-series dataset comprising recordings of six speakers. Each speaker recites numbers from one to nine. For each number, every speaker makes 50 recordings. The speaker chosen for all experiments in this chapter is Jackson. The dataset undergoes preprocessing using a Fast Fourier Transform (FFT) technique. The dataset for speaker Jackson has 500 data instances, and only instances above a threshold of 3000 samples are considered to tackle the varying data length through the dataset. In these data instances, only the first 3005 data samples are examined. Finally, 480 data instances are filtered to feed into the algorithm. The specified class distribution provided in Table 5.1 is as follows: $(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)$.

6.7.3 Experiments and Results

This section is dedicated to the experiments carried out MNIST and spoken digit dataset. We have used ChaosNet to demonstrate the efficacy of NL in LL framework. In this framework, NL progressively learns tasks $T_1, T_2, \ldots, T_5$. Each tasks contains data instances belonging to two classes. For eg., in the case of MNIST dataset, task T_1 contains the hand written images belonging to classes 0 and 1. Similarly task T_2 contains the hand written images belonging to classes 2 and 3. The 10 classes in MNIST dataset are grouped into five tasks $T_1, T_2, \ldots, T_5$. The same procedure is followed for spoken digit dataset (SDD).

6.7.4 Hyperparameter tuning

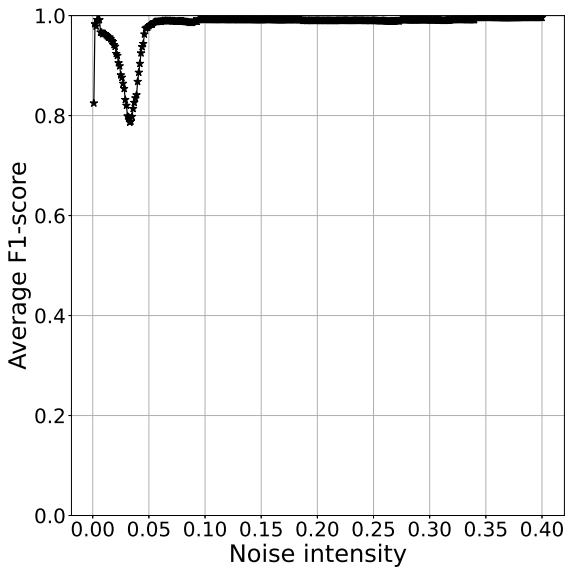


Figure 6.7: Hyperparameter tuning for MNIST: A five fold crossvalidation is carried out on T_1 using $q = 0.340$, $b = 0.499$ and noise intensity (ϵ) is varied in the range 0.001 to 0.400 with a step size of 0.001.

The hyperparameters of ChaosNet for MNIST and SDD are tuned using the data instances belonging to task T_1 of MNIST and SDD respectively.

1. From the MNIST dataset, 80% data instances were considered as training set and 20% data instances were considered as the test set. From the training set we choose a subset of class-0 (5560) and class-1 (6277) data instances for five fold crossvalidation. The q and b were fixed to 0.340 and 0.499 respectively. Noise intensity (ϵ) was varied from 0.001 to 0.400 with a step size of 0.001. A maximum average macro F1-score = 0.996 was obtained for $\epsilon = [0.398, 0.399, 0.400]$ in five fold crossvalidation using the data instances belonging to the training set of T_1. We choose $\epsilon = 0.399$ for further experiments. Figure 6.7 depicts the crossvalidation plot (average macro F1-score vs. noise intensity (ϵ)) for MNIST dataset.

2. From the SDD dataset, 80% data instances were considered as training set and 20% data instances were considered as the test set. From the training set

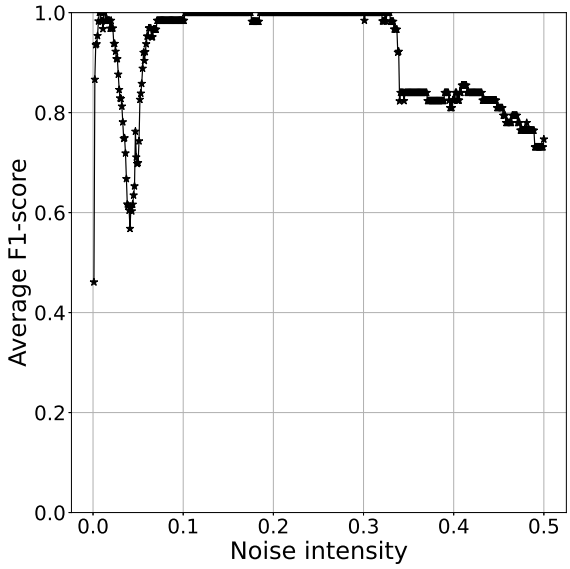


Figure 6.8: Hyperparameter tuning for SDD: A five fold crossvalidation is carried out on T_1 using $q = 0.340$, $b = 0.499$ and noise intensity (ϵ) is varied in the range 0.001 to 0.500 with a step size of 0.001.

we choose a subset of class-0 (30) and class-1 (34) data instances for five fold crossvalidation. For SDD, q and b were fixed to 0.340 and 0.499 respectively. Noise intensity (ϵ) was varied from 0.001 to 0.500 with a step size of 0.001. A maximum average macro F1-score = 1 was obtained for several values of ϵ in the range starting from 0.007 to 0.328 with a step size of 0.001 in five fold crossvalidation using the data instances belonging to the training set of T_1. We choose $\epsilon = 0.178$ for further experiments. Figure 6.8 depicts the crossvalidation plot (average macro F1-score vs. noise intensity (ϵ)) for SDD.

We use the tuned hyperparameters to evaluate the efficacy of Neurochaos Lifelong Learning (NLL) on MNIST and SDD. The results for progressive learning of a sequence of tasks T_1, T_2, T_3, T_4, T_5 for MNIST and SDD using NLL in the high training sample regime is provided in Table 6.3. Figure 6.9 and 6.10 depicts the performance of NLL for sequential learning of tasks T_1, T_2, T_3, T_4, T_5 corresponding to MNIST and SDD.

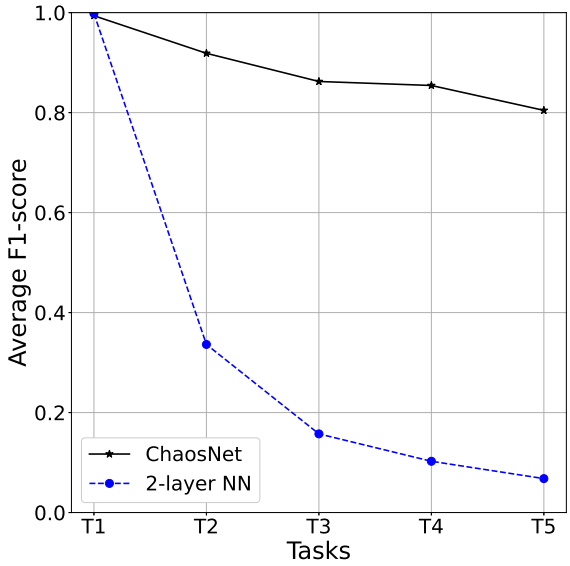


Figure 6.9: Neurochaos Lifelong Learning for MNIST for sequential learning of tasks T_1, T_2, T_3, T_4, T_5 compared with performance of a 2-layer neural network.

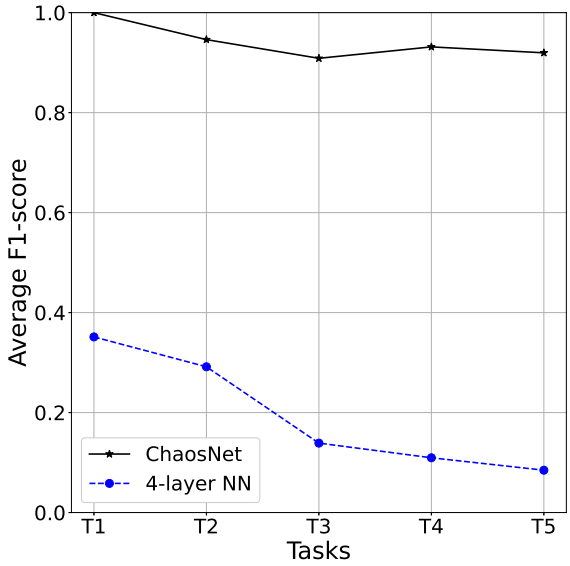


Figure 6.10: Neurochaos Lifelong Learning for SDD for sequential learning of tasks T_1, T_2, T_3, T_4, T_5 compared with performance of a 4-layer neural network.

Table 6.3: Results for MNIST and SDD using NLL.

Tasks	Train Classes per Tasks	Test Classes per Tasks	F1-score MNIST	F1-Score SDD
T_1	0, 1	0, 1	0.994	1.000
T_2	2, 3	0, 1, 2, 3	0.919	0.968
T_3	4, 5	0, 1, 2, 3, 4, 5	0.862	0.979
T_4	6, 7	0, 1, 2, 3, 4, 5, 6, 7	0.854	0.964
T_5	8, 9	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	0.804	0.971

6.7.5 Deep Neural Network

We compare the catastrophic forgetting in NL with DL for sequential learning. We use the same protocol for training NL and DL. The following DL architecture was used to train MNIST and SDD respectively.

- MNIST- A neural network consisting of 2 layers having 784 artificial neurons (first hidden layer), activation ($ReLu$), and 10 output layer neurons. Training is done for 30 epochs.

- SDD - A neural network consisting of 4 layers having 2000 artificial neurons (first hidden layer), activation ($ReLu$); 1000 artificial neurons (second hidden layer), activation (sigmoid function); 500 artificial neurons (third hidden layer), activation ($ReLu$); and 10 output layer neurons. Training is done for 60 epochs.

It is very evident from Figure 6.9 and 6.10 that the catastrophic forgetting in NL is much lesser than DL. However, more rigorous testing on various datasets are required for making general comments.

6.8 Cause-Effect Preservation and Classification using NL

Despite the success of Machine Learning (ML) and Deep Learning (DL) algorithms in the field of natural language processing [22], computer vision [142], speech recognition [143], these algorithms face difficulty in interpretability and trustworthiness. One of the main reasons for this is the fact that they are merely discovering associations between 'input' and 'output' in the name of 'learning'. However, discovering associations alone is insufficient as an explanation to aid in the decision making process. Decision-making in everyday human existence relies heavily on reasoning and causal inference. Hence, incorporation of *causal* reasoning, which goes over and beyond associations (or correlations), has become an important objective in current machine learning research [144].

For time-series data, *causality* has been mathematically defined by Wiener [145]. This mathematical definition was later realized through a number of mathematical/algorithmic formulations. Granger Causality (GC) [146] was the pioneering causality estimation technique developed for time-series data. GC has been followed up with a number of extensions and inspirations, some of which are, information-theoretic causality as defined by Transfer Entropy [147] and data-compression based causality, as defined by Compression-Complexity Causality [148]. While GC has primarily been formulated for linear time-series, the latter are valid for non-linear applications. These methods have been successfully employed in econometrics [149,150], climatology [151, 152], neuroscience [153, 154] etc.

Conventional ML algorithms have not had much success in causal learning or causality based classification from time-series data [155]. They are either employed in combination with an existing time-series based causality detection technique [156,157] or specialized methods assuming underlying causal models have been developed [158–161], in order to incorporate causality learning abilities in ML.

In this chapter, we focus on causality detection from time-series data and NL to learn generalized causal patterns. NL maps the input data into a high dimen-

sional space which enables efficient classification. In this sense, NL shows similarity to Reservoir Computing [162] which also employs nonlinear mappings of input data. However, NL is fundamentally different from Reservoir Computing in terms of motivation, methodology, and working. NL fundamentally uses the *Topological Transitivity* property of chaos and *Stochastic Resonance* for learning classification tasks as described in chapters 3, 4, and 5. Given that NL is still at a nascent stage of development, it is not well known in the AI community and most of its properties are largely unexplored. Hence, the objectives of this present study are to investigate the following:

O1: The efficacy of NL in *cause-effect classification* and compare the same with Deep Neural Network (DNN).

O2: Does success in cause-effect classification imply preservation of causality?

O3: Can NL use a *transfer learning* framework for cause-effect classification?

We do not build a novel causal ML algorithm, but rather explore if the existing NL architecture which does classification, has the capability to classify based on 'causal' information in the data. As a first step, we check its ability to distinguish between *cause* and *effect* time-series data. We do not use any existing causality estimation method as an aid to the NL algorithm for this purpose. Further, for **O2**, we check whether features extracted from the learning architecture (in the above cause-effect classification task), preserve causality as measured by an existing time-series causality estimation method. This is important because it determines whether NL is doing a causality informed classification or not.

It is found that a general NL architecture slightly outperforms a general DL architecture (five layer DNN) in cause-effect classification. Further, the features extracted using NL are found to preserve the cause-effect relationship present in input data. This, however, was not the case for DNN, probably because the classification results were not *causally* informed. The findings demonstrate that even a general NL architecture is capable of some basic causal learning and hence promising for developing more sophisticated causal ML algorithms required for different tasks.

6.8.1 Datasets

To evaluate the efficacy of `ChaosNet` and deep learning for the classification of cause-effect, we used simulated datasets from (a) Coupled autoregressive (AR) processes, (b) Coupled 1D chaotic maps in master-slave configuration (1D skew tent maps and the 1D logistic maps) and real-world dataset from a (c) prey-predator system.

Coupled AR processes

The governing equations for the coupled AR processes are the following:

$$M(t) = a_1 M(t-1) + \gamma r(t), \tag{6.5}$$

$$S(t) = a_2 S(t-1) + \eta M(t-1) + \gamma r(t), \tag{6.6}$$

where $M(t)$ and $S(t)$ are the independent and dependent (or cause and effect) time series respectively; $a_1 = 0.8$, $a_2 = 0.9$, the noise intensity $\gamma = 0.03$ and $r(t)$ is i.i.d additive gaussian noise drawn from a standard normal distribution. The coupling-coefficient η is varied from 0 to 1 in steps of 0.1. We generated 1000 independent random trials for each value of η. Each of the data instances are of length 2000, after removing the initial 500 samples (transients) from the time series.

Coupled Skew-tent maps

The governing equations used to generate the master and slave time series for the coupled 1D skew-tent maps are the following:

$$M(n) = T_1(M(n-1)), \tag{6.7}$$

$$S(n) = (1-\eta) T_2(S(n-1)) + \eta M(n-1), \tag{6.8}$$

where $M(n)$ is the master (cause) and $S(n)$ is the slave (effect) system. $M(n)$ influences the dynamics of the slave system (equation 6.8). The coupling coefficient given by η, is varied from 0 to 0.9 with a step size of 0.1. $T_1(n)$, and $T_2(n)$ are iterates of skew tent maps with skewness $b_1 = 0.65$, and $b_2 = 0.47$ respectively. The

initial values are chosen randomly for the master-slave system in the interval $(0, 1)$. We generated 1000 independent random trials for each value of η. Each of the data instances are of length 2000, after removing the initial 500 samples (transients) from the time series.

Coupled Logistic maps

The 1D Logistic map is a widely used model to study population dynamics [163]. The governing dynamics for coupled logistic maps in master-slave configuration is given by:

$$M(n) = L_1(M(n-1)), \tag{6.9}$$

$$S(n) = (1-\eta)L_2(S(n-1)) + \eta M(n-1), \tag{6.10}$$

where the coupling coefficient η is varied from 0 to 0.9. $L_1(n) = 4 \cdot L_1(n-1)(1 - L_1(n-1))$, and $L_2(n) = 3.82 \cdot L_2(n-1)(1 - L_2(n-1))$. The attractor for this coupled dynamical system is provided in Figure 6.15b.

For both systems, 1000 data instances $(M(n), S(n))$ are generated and grouped as class-0 ($M(n)$: Cause) and class-1 ($S(n)$: Effect) respectively. Table 6.4 gives details of the train-test split for the classification task.

Table 6.4: Train-Test distribution for the simulated datasets.

Class	Traindata	Testdata
Class-0	801	199
Class-1	799	201
Total	1600	400

6.8.2 Experiments, Results and Discussions

In this section, we begin with a description of hyperparameter tuning for NL and DL followed by a demonstration of causality preservation by ChaosFEX for coupled AR processes (and the failure of DL). Subsequently, macro F1-scores for `ChaosNet` and DL for the cause-effect classification for each η are plotted. For all results in this chapter,

software implementation is performed using Python 3, scikit-learn [124], keras [164], ChaosFEX toolbox, Multivariate Granger Causality (MVGC) toolbox [165], CCC toolbox [148] and MATLAB.

Hyperparameter tuning for NL

The hyperparameter tuning is done only once with the traindata corresponding to $\eta = 0.4$ (Table 6.4) separately for the coupled AR processes and coupled skew tent maps.

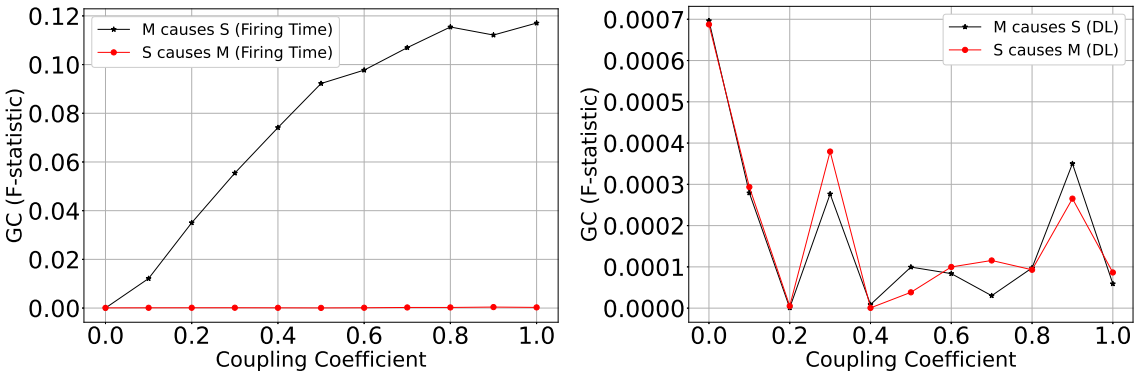
For a fixed value of $b = 0.499$, and $\epsilon = 0.171$, q was varied from 0.01 to 0.98 with a stepsize of 0.01 for both coupled AR processes and coupled chaotic skew tent maps. In the case of coupled AR processes, a maximum average macro F1-score = 0.605 is obtained for $q = 0.78$. In the case of coupled skew tent maps, a maximum average macro F1-score = 1.0 was obtained for the following values of q = [0.16, 0.26, 0.27, 0.28, 0.29, 0.30, 0.31, 0.32, 0.34, 0.36, 0.37, 0.38, 0.48, 0.51, 0.52, 0.56, 0.57, 0.72, 0.76, 0.77, 0.78, 0.79, 0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.98] in a five-fold cross validation using traindata. We choose $q = 0.56$ for further experiments.

Deep Learning Parameters

A five layer Deep Learning architecture was used to evaluate the efficacy of cause effect classification. The number of nodes in the input layer = 2000, followed by first hidden layer with 5000 neurons and sigmoid activation function. The output from this layer is passed to second hidden layer with 500 neurons and $ReLU$ activation function. This is followed by 100 neurons with $ReLu$ activation function in the third hidden layer. The fourth hidden layer contains 30 neurons with $ReLu$ activation function. The output layer contains 2 neurons with softmax activation function. Training was done for 30 epochs.

Preservation of Granger Causality for coupled AR processes under a chaotic transformation

Accurate estimation of causality for coupled AR processes is ideally obtained by Granger Causality (GC) since GC models time series as AR processes. This is the reason GC is very popular in causal analysis of financial time series, climatology and neuroscience. ChaosFEX features are extracted after a chaotic transformation of the input time series. It is important to verify whether Granger Causality is preserved under such a nonlinear transformation. To test this, we perform the following experiment. For $q = 0.78$, $b = 0.499$, and $\epsilon = 0.171$, the firing time has been extracted from ChaosFEX for time series from coupled AR processes. The GC vs. coupling coefficient plot for firing time depicted in Figure 6.11a reveals that indeed Granger Causality is nicely preserved. The GC values shown here are obtained from 50 random trials[3]. This indicates the reliability of the chaotic transformation of NL in preserving granger causality and hence very desirable in applications which employ GC. Note that such a property is not available for DL (Figure 6.11b) making NL a very attractive candidate for causal ML applications.



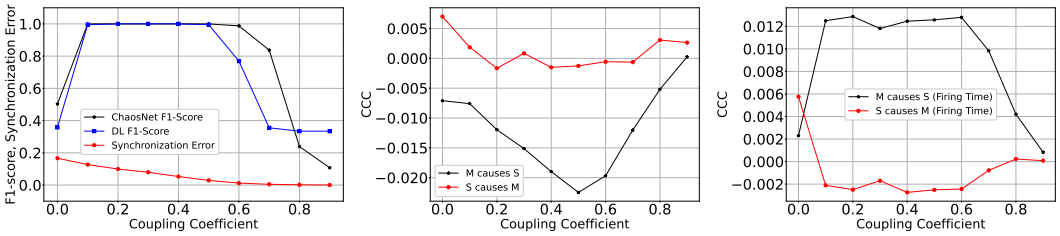
(a) (b)

Figure 6.11: (a) GC vs. coupling coefficient for the firing time feature extracted from the coupled AR processes. The ChaosFEX settings are $q = 0.78$, $b = 0.499$, and $\epsilon = 0.171$. The GC F-statistic is computed from 50 trials. (b) GC vs. coupling coefficient for DL features extracted from the fourth hidden layer of a five layer neural network. The GC F-statistic is computed from 50 trials.

[3]The maximum model order setting in the MVGC toolbox [166] was set to 30 for ChaosFEX features and to 20 for DL features.

Classification of Cause Effect for Coupled Skew-Tent maps in Master-Slave Configuration

In this section, we compare the efficacy of NL - `ChaosNet` architecture with a five layer DNN architecture in cause-effect classification (objective **O1**). A binary classification problem is formulated, to classify whether a given time-series is a cause or an effect. The performance of `ChaosNet` and five layer DNN (DL) for varying coupling coefficient (η) is depicted in Figure 6.12a.



(a) (b) (c)

Figure 6.12: (a) Performance comparison of ChaosNet and five layer DNN for the classification of cause-effect for 1D coupled skew tent map in master-slave configuration. (b) CCC vs Coupling Coefficient for the raw data corresponding to 1D chaotic skew tent map in master-slave configuration. (c) CCC vs Coupling Coefficient for firing time (ChaosFEX feature) corresponding to 1D chaotic coupled skew tent maps in master-slave configuration.

`ChaosNet` and DL give identical performance (a macro F1-score = 1.0) for η values up to 0.5. However, for $\eta = [0.6, 0.7]$, `ChaosNet` outperforms DL. Beyond $\eta > 0.6$, the synchronization error < 0.013 indicating that the two time series are practically identical. Hence, classification fails as there is essentially nothing to distinguish between the two time series owing to synchronization.

Preservation of causality in ChaosFEX feature space

To check if causality is preserved in the ChaosFEX feature space of unidirectionally coupled skew-tent maps, we use the measure Compression-Complexity Causality (CCC) [148]. CCC is ideal for application to non-linear time series, where often GC can face issues. Figure 6.12b shows the CCC estimates for original (raw) time se-

ries. The estimates plotted are averaged over 50 trials with CCC parameters[4] set to $L = 100, w = 15, \delta = 50, B = 4$. As expected, the magnitude of CCC values from the master to the slave increases with increasing coupling and begins to decrease as the time-series become synchronized and effectively no transfer of information can be detected. As discussed in [148], CCC can take negative values and its magnitude denotes the strength of causation. CCC values in the direction of causation from slave to master are much lower in magnitude and remain close to zero.

CCC for the corresponding firing time feature of ChaosFEX for these coupled maps is depicted in Figure 6.12c. These values are also averaged over 50 trials and computed with CCC parameters set to $L = 120, w = 15, \delta = 60, B = 2$. Here, master to slave CCC does not perfectly preserve the increasing trend with increasing values of coupling, however decreases just as the estimates for raw data, when the processes proceed to synchronization. The slave to master estimates for the coupling range $0.1 - 0.9$ are quite low in magnitude and remain close to zero as expected. Even though the estimates for zero coupling are not very close to zero or take exactly the same value and the increasing trend for increasing coupling is not perfectly preserved, CCC estimates in the direction for which coupling exists and the its opposite are well differentiated and hence it can be said that ChaosFEX features do a reasonably good job in preserving causality even for skew chaotic tent maps. Surrogate based causality analysis might help to reveal a more adequate picture of the differentiation and of the existence of causality, but is out of the scope of this work.

Transfer Learning for Cause-Effect Classification

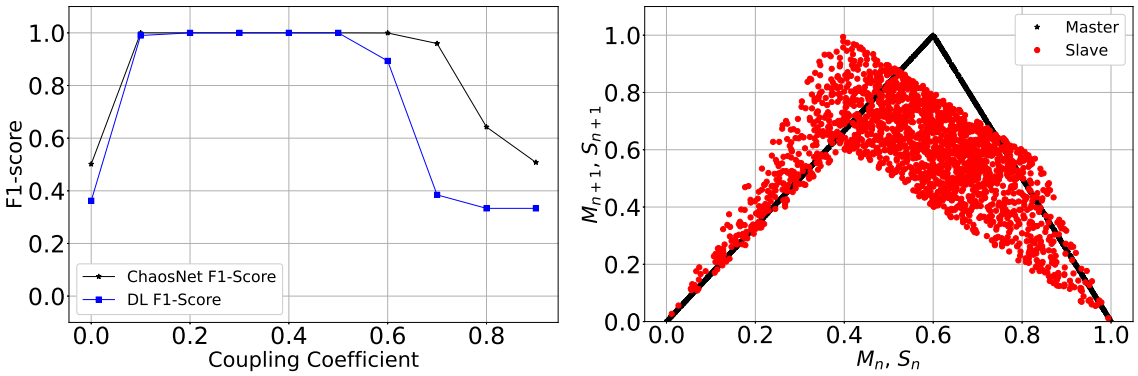
We have demonstrated the possibility of cause-effect classification for coupled chaotic maps in master-slave configuration. However, it is interesting to explore whether we can transfer this 'learning' to scenarios where the master-slave systems are different from the ones for which the method was trained. Specifically, we shall change the skewness of both the master and slave systems from the original parameter values used in the training phase. A more drastic case of transfer learning would be to test

[4]These were chosen using the selection criteria described in [148].

on an entirely different nonlinear map, for example, coupled logistic maps without training afresh (using the same learned parameters as the coupled skew tent maps). These would help us determine to what extent the learning is generalizable for both NL and DL.

We consider the following cases for transfer causal learning:

- **Case I:** Train with master-slave coupled skew tent map system ($b_1 = 0.65$, $b_2 = 0.47$) and test with master-slave coupled skew tent map system with $b_1 = 0.6$ and $b_2 = 0.4$ (classification results are in Figure 6.13a). The attractor for skew tent map master slave testdata with $b_1 = 0.6$ and $b_2 = 0.4$ is provided in Figure 6.13b.

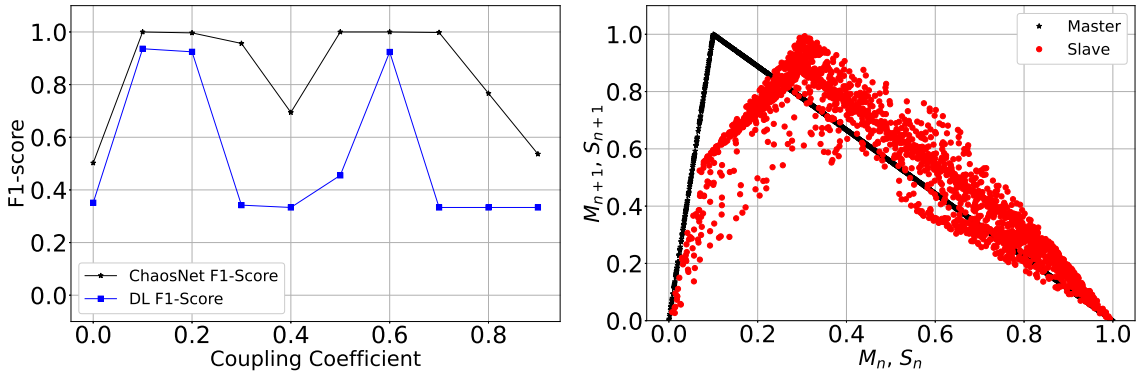


(a) (b)

Figure 6.13: (a) Transfer learning for case I: comparative performance of ChaosNet and five layer DNN evaluated using macro F1-score for η in the range 0 to 0.9 with a stepsize of 0.1. (b) Case I: Attractor for the coupled 1D chaotic skew tent maps in master slave configuration with $b_1 = 0.6$ and $b_2 = 0.4$.

- **Case II:** Train with master-slave coupled skew tent map system ($b_1 = 0.65$, $b_2 = 0.47$) and test with master-slave coupled skew tent map system with $b_1 = 0.1$ and $b_2 = 0.3$ (classification results are in Figure 6.14a). The attractor for skew tent map master slave testdata with $b_1 = 0.1$ and $b_2 = 0.3$ is provided in Figure 6.14b.

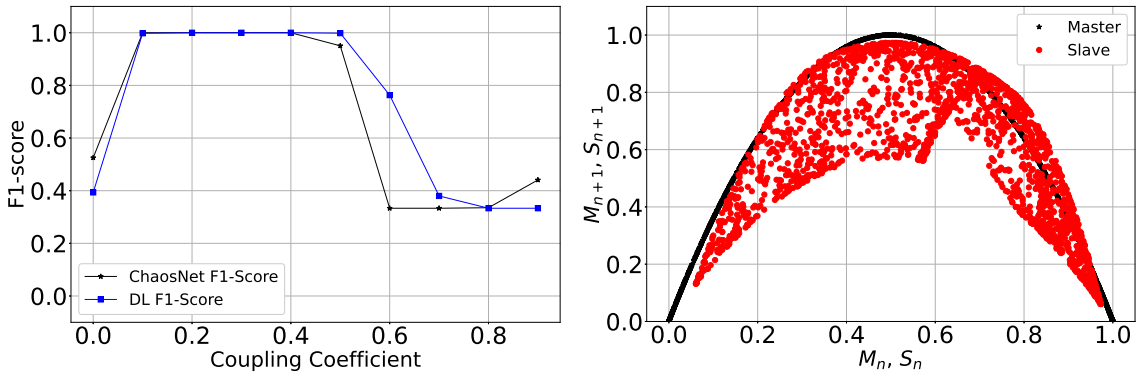
- **Case III:** Train with skew tent map master-slave coupled skew tent map system ($b_1 = 0.65$, $b_2 = 0.47$) and test with logistic map master-slave system with



(a) (b)

Figure 6.14: (a) Transfer Learning for Case II: comparative performance of ChaosNet and five layer DNN evaluated using macro F1-score for η in the range 0 to 0.9 with a stepsize of 0.1. (b) Case II: Attractor for the coupled 1D chaotic skew tent maps in master slave configuration with $b_1 = 0.1$ and $b_2 = 0.3$.

$A_1 = 4.0$ and $A_2 = 3.82$ (Figure 6.15a). The attractor for logistic map master slave testdata with $A_1 = 4.0$ and $A_2 = 3.82$ is provided in Figure 6.15b.



(a) (b)

Figure 6.15: (a) Transfer Learning for Case III: comparative performance of ChaosNet and five layer DNN evaluated using macro F1-score for η in the range 0 to 0.9 with a stepsize of 0.1. (b) Case III: Attractor for the coupled 1D logistic maps in master slave configuration with $A_1 = 4.0$ and $A_2 = 3.82$.

In the case of testing with data generated from different models, `ChaosNet` completely outperforms DL for Case I (Figure 6.13a) and Case II (Figure 6.14a) for the entire range of η. For Case III (Figure 6.15a), `ChaosNet` and DL shows similar trends (with DL outperforming `ChaosNet` for some values of η). A high performance of `ChaosNet` in classification shows the separability of the mean representation vectors

of cause and effect.

Real Data

The efficacy of ChaosFEX features in cause-effect preservation was evaluated on a real world dataset from a prey-predator system as well. The data consists of 71 data points of predator (Didinium nasutum) and prey (Paramecium aurelia) populations [167, 168]. This is a system of bidirectional causation as the predator population directly influences the prey population and then itself gets influenced by a change in the prey population. It is expected that the direct causal influence from the predator to the prey should be higher than in the opposite direction.

For our analysis, initial 9 transients were removed. With the remaining 62 data points, CCC values are computed for the raw data and ChaosFEX feature (firing time). The parameters of CCC[5] used for the raw data are $L = 40, w = 15, \delta = 4, B = 8$. In the case of ChaosFEX, firing time feature was extracted for the following NL hyperparameters: $q = 0.56$, $b = 0.499$, and $\epsilon = 0.1$. The CCC parameters chosen for ChaosFEX are $L = 40, w = 15, \delta = 4, B = 4$. The results for the cause-effect preservation for the raw data and ChaosFEX firing time feature is provided in Table 6.5. CCC rightly captures the higher causal influence from predator to prey population and finds a lower influence in the opposite direction, for both raw data and ChaosFEX feature - firing time.

Table 6.5: Cause-effect preservation of the prey-predator real world data using CCC.

Class	CCC (rawdata)	CCC (firing time)
Predator $\rightarrow$ Prey	0.1160	0.0484
Prey $\rightarrow$ Predator	-0.0210	0.0050

6.8.3 Limitations

In the case of coupled 1D chaotic maps, NL consistently performed well up to $\eta = 0.5$ for classification. However, the same is not true for the classification of data generated

[5]These were chosen using the selection criteria described in [148].

from coupled AR processes. For $q = 0.78$, $b = 0.499$, and $\epsilon = 0.171$, the classification results are depicted in Figure 6.16. In the same figure, it can be seen that DL performance is worse than NL[6]. We have used the exact same architecture for DL as we have used for the cause-effect classification of data from coupled chaotic skew-tent maps in master-slave configure (section 4.2).

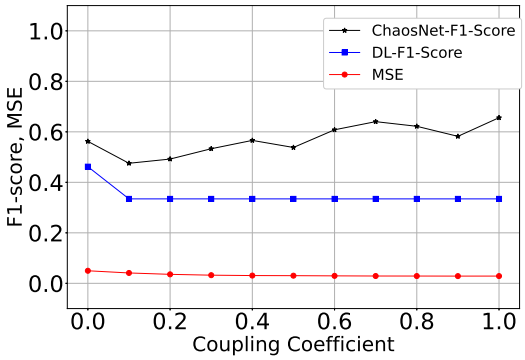


Figure 6.16: F1-score vs. coupling coefficient for the classification of the coupled AR processes using ChaosNet.

A maximum macro F1-score = 0.656 was obtained for $\eta = 1.0$ implying that `ChaosNet` was not able to find mean representation vectors that could separate the two classes. Choosing a more sophisticated classifier for NL (instead of the simplistic cosine-similarity metric) could solve this problem and improve classification results. We shall explore these possibilities in a future study.

In this research, we have shown the classification and preservation of causality for unidirectional causation of two variables. A detailed study needs to be undertaken for the classification and causal discovery for coupled high dimensional systems and real world datasets in the future.

6.8.4 Concluding Remarks On Causality and NL

In this work, Neurochaos Learning architecture - `ChaosNet` has been used in the classification of cause-effect for data generated from coupled chaotic maps. `ChaosNet` outperforms a five layer deep learning architecture in several cases (objective **O1**).

[6]We have performed some amount of hyperparmater tuning for DL architecture, however a more extensive tuning needs to be performed.

In the experiments, `ChaosNet` consistently outperformed a five layer DNN upto a coupling coefficient of 0.7. Causality testing using Granger Causality (for coupled AR processes) and Compression-Complexity Causality (for coupled chaotic systems and for a real-world prey-predator system) on the firing times extracted from the chaotic neural traces reveals the preservation of cause-effect in the NL feature extracted space (objective **O2**). Further, the efficacy of the proposed method was observed in *transfer learning* of the classification of cause-effect from the master-slave time series generated from different chaotic unimodal maps (skew-tent maps with different skews and logistic map with different parameters) (objective **O3**). This motivates future research direction of NL in lifelong learning framework, classification of cause-effect using `ChaosNet` on real world datasets and building more sophisticated causal learning algorithms using NL for specific tasks.

The preservation of causality can be attributed to the rich properties of the non-linear chaotic transformation of GLS neurons in NL (`ChaosNet`). Unlike traditional ANNs, NL is intrinsically a nonlinear deterministic algorithm that performs a point-by-point chaotic transformation, in fact, a non-linear embedding of the input raw features in to a high dimensional space. *Deterministic Chaos* combines the best of both the worlds - *pseudo-randomness* and *determinism*. The ergodic, 'random-like' structure of the chaotic neural traces enables an effective transformation of the input data (stimuli) preserving causality that is inherent in the input space and at the same time ensuring separability in the chaotic feature space for efficient classification.

This concludes the description of eight properties of NL.

Chapter 7

Conclusions and Future Work

This chapter summarises the contributions of this PhD dissertation. It also discusses the future directions and limitations of this research.

7.1 Summary of Research

Artificial Intelligence (AI) is a unifying theme that unifies different methods which are inspired from anthropocentric notion of intelligence. These methods include (a) Symbolic AI - using symbolic processing to harness intelligence, (b) Statistical/Machine Learning - using algorithms to learn from data (Machine Learning) (c) Sub-symbolic AI-brain inspired learning algorithms. Hence, AI is concisely defined as "An Anarchy of Methods" [3].

From 2010 onwards, the research in AI has largely been confined to Neural Networks and Deep Learning (DL) (Sub-symbolic AI). This is because of the availability of high performance computing systems and computational resources. DL has numerous applications in the field of computer vision, speech processing, text classification, cyber security etc. and provides state-of-the-art performance. Despite their tremendous success, DL algorithms are only loosely inspired from the brain. A key thing that researchers missed is to fundamentally make use of the presence of chaos in the brain [28] in Artificial Neural Networks. This aspect has been explored in this PhD dissertation.

The foundational idea for this dissertation is to fundamentally use chaos and stochastic resonance in machine learning which to our knowledge has not done before. The inspiration for chaos and stochastic resonance comes from the brain [28,33]. Inspired by the chaotic neuronal firing in the brain, a novel learning algorithm namely *Neurochaos Learning* (NL) has been proposed in this dissertation. The trans-disciplinary approach used in the thesis involves ideas from chaos theory, stochastic resonance, neuroscience, and machine learning (Figure 7.1).

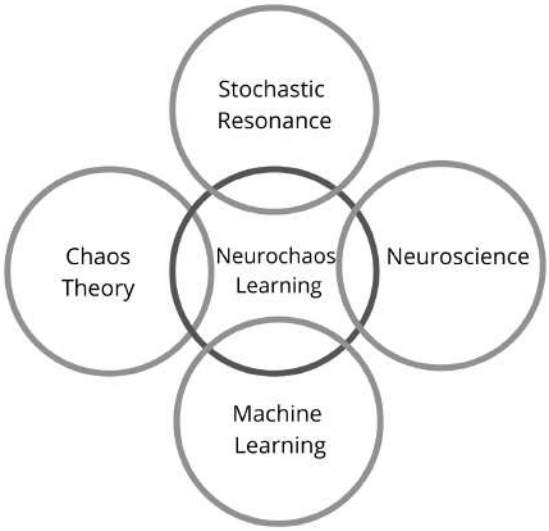


Figure 7.1: Trans-disciplinary approach used in this dissertation.

NL has the following properties: (a) fundamentally uses chaos and stochastic resonance for learning classification tasks, (b) NL - `ChaosNet` architecture has only four hyperparameters to be tuned, (c) superior performance in the limited training sample regime when compared to ML algorithms, (d) the rate of catastrophic forgetting in NL is less compared to DL algorithms, (d) the deterministic nature of NL is highly desirable for reproducible research and explainability, (e) the flexibility of NL allows for developing hybrid NL-ML algorithms, and (f) the features extracted from the input layer of NL preserves cause-effect relationships.

NL has been tested on several real world problems ranging from image classification, cyber security, medical applications, cause-effect classification and classification of several simulated datasets as demonstrated in earlier chapters. NL is a promising

way of connecting nonlinear dynamics and chaos to machine learning.

7.2 Contribution of this PhD Dissertation

The unique contribution of this PhD dissertation are as follows:

1. **Neurochaos Learning**: NL is a data driven learning algorithm which uses chaotic neurons and nonlinear feature transformation for decision making. Some of the benefits of NL are:

    - NL has three blocks - feature transformation, feature extraction and classification. Most of the feature extraction techniques used in the ML community (sush as Principal Component Analysis, Dynamic Mode Decomposition, Non-Negative Matrix Factorization etc.) are linear operators. Whereas NL perform a nonlinear feature transformation. This is unique when compared to commonly used pre-processing techniques in the ML community. The hyperparameters of NL are selected in such a way that the NL features (firing time, firing rate, energy and entropy) exhibits distinct pattern for data belonging to distinct classes.

2. **Stochastic Resonance in NL**: NL is the first learning algorithm to the best of our knowledge where both chaos and stochastic resonance plays an equally important role in classification tasks. This paves the way for developing more brain-inspired architectures in the ML community.

3. **Mathematical Guarantees in NL**: NL satisfies a version of Universal Approximation Theorem with a bound on the number of chaotic neurons to approximate a discrete time finite support function $f(n)$.

4. **Efficient Learning**: The ability to learn new concepts from limited examples is a unique feature to human cognition. Hence, developing algorithms which learn from limited training samples plays an integral part in building intelligent systems. The efficacy of NL in learning from limited training samples has been

thoroughly studied in Chapter 5. In several datasets, NL provides a superior performance in learning from limited samples when compared to classical ML techniques.

5. **Causality Informed Classification**: NL preserves cause-effect relationship in timeseries data under chaotic feature transformation and extraction. Whereas, in the case of Deep Neural Networks, the cause-effect relationship is not preserved across layers. Preserving causal relationship is highly desirable in causal machine learning applications.

6. **Neurochaos Lifelong Learning**: The robustness and flexibility of human cognition allows for continuously learning new tasks without forgetting what was previously learnt. Forgetting what was previously learnt when trying to learn a new task is called Catastrophic Forgetting. Neural Networks suffers from catastrophic forgetting. In Chapter 6, we show that the extent of catastrophic forgetting in NL is less when compared to that of Neural Networks.

7. **Flexibility of NL architecture**: The flexibility of NL architecture permits one to develop hybrid NL-ML/DL architectures for learning. In Chapter 5, the performance of hybrid NL-ML architectures have been thoroughly studied. In several cases, the hybrid NL-ML architectures gave performance boost compared to standalone ML methods.

7.3 Limitations of the Research

With the current implementation of the ChaosFEX algorithm, computation of image datasets is a costly process. This may limit the use of NL architectures in practical situations involving images.

Furthermore, NL architectures have been based on certain assumptions. One assumption revolves around the nonlinear separability of data. NL assumes that applying a nonlinear chaotic transformation will result in separable or near separable data suitable for classification, which may not be true in all cases. As it stands, the

input layer of NL treats input attributes as independent of each other. It establishes no connection between the neurons for each input attribute. This limitation can be addressed by using coupled chaotic neurons in the input layer. Currently, we have not considered multi-layered NL (Deep-NL) which could significantly enhance performance (by careful choice of coupling between adjacent layers). Another limitation of the NL architecture is the lack of a principled approach to tune the best hyperparameters for a classification task. Currently, cross-validation experiments are used to tune the hyperparameters. The connection between the degree of chaos as measured by lyapunov exponent [134] and learnability is also worth exploring for future research.

7.4 Future Research Directions

We would like to explore the following research directions in future:

1. **Learning from limited data:** Learning from limited data instances is a challenging problem across science and engineering. Recently, a Structured Hankel Approximation View of Koopman (sHAVOK [169]) algorithm has been developed by Seth M. Hrish et al. This algorithm has shown promising results to identify stable and accurate models from limited data. Inspired by this research, we plan to investigate the effectiveness of sHAVOK in Neurochaos Learning for robust classification under noisy and limited training sample regime.

2. **Intersection of Neurochaos Learning and Causality:** In this work, we plan to propose a novel measure for complexity of learning (Effort to Learn) for Neurochaos Learning (NL). We plan to use Network Causal Activity (NCA) [170] - a quantitative consciousness measure to capture significant causal influence activity between all the subsystems of the given system. We investigate the efficacy of NCA as a candidate for developing the Effort to Learn for Neurochaos Learning. Inorder to compute NCA, the causal functional connectivity of NL has to be estimated from the data. In this way, we fundamentally connect causality, learning (classification), chaos and complex systems.

3. **Degree of Chaos and Learning**: The hyperparameter b (discrimination threshold) in the GLS maps controls the degree of chaos which is measured by lyapunov exponent [134]. Future work involves the theoretical investigation of the relationship between degree of chaos and learning classification tasks using NL.

List of Publications

International Peer Reviewed Journal Publications

1. Harikrishnan N. B., and Nithin Nagaraj. "**When Noise Meets Chaos: Stochastic Resonance In Neurochaos Learning**." *Neural Networks* (2021), (Impact Factor: 9.657), DOI: https://doi.org/10.1016/j.neunet.2021.06.025.

2. Harikrishnan N. B., Aditi Kathpalia, Snehanshu Saha, and Nithin Nagaraj. "**ChaosNet: A Chaos Based Artificial Neural Network Architecture For Classification**." *Chaos: An Interdisciplinary Journal of Nonlinear Science 29*, no. 11 (2019): 113125, (Impact Factor: 3.741), DOI: https://doi.org/10.1063/1.5120831.

3. Harikrishnan N. B., Pranay S. Y., Nithin Nagaraj. "**Classification Of SARS-CoV-2 Viral Genome Sequences Using Neurochaos Learning**." *Medical Biological Engineering Computing* (2022), (Impact Factor: 2.602), DOI: https://doi.org/10.1007/s11517-022-02591-3.

Conference Proceedings/ Presentations

1. Harikrishnan, N. B., and Nithin Nagaraj. "**A Novel Chaos Theory Inspired Neuronal Architecture**." *Global Conference for Advancement in Technology (GCAT)*, pp. 1-6. IEEE, 2019, DOI: 10.1109/GCAT47503.2019.8978360.

2. Harikrishnan, N. B., and Nithin Nagaraj. "**Neurochaos Inspired Hybrid Machine Learning Architecture For Classification**." *International Conference on Signal Processing and Communications (SPCOM)*. IEEE, 2020, DOI: 10.1109/SPCOM50965.2020.9179632.

3. Harikrishnan N. B., and Nithin Nagaraj, "**Why Does Neurochaos Learning Work? The Role Of Chaos And Noise In Neurochaos Learning**." *13th Conference on Nonlinear Systems Dynamics (CNSD)* 2021.

ArXiv Reports

1. Harikrishnan N. B., Aditi Kathpalia, and Nithin Nagaraj. "**Cause-Effect Preservation and Classification Using Neurochaos Learning**." arXiv preprint arXiv:2201.12181 (2022) [cs.LG].

2. Deeksha Sethi, Nithin Nagaraj, and Harikrishnan N. B. "**Neurochaos Feature Transformation and Classification for Imbalanced Learning**." arXiv preprint arXiv:2205.06742 (2022) [cs.NE].

References

[1] Alan M Turing. Computing machinery and intelligence (1950). *The Essential Turing: the Ideas That Gave Birth to the Computer Age*, pages 433–464, 2012.

[2] Peter Millican and Andy Clark. *Machines and Thought: The Legacy of Alan Turing, Volume I*. Oxford University Press, 1996.

[3] Joel Lehman, Jeff Clune, and Sebastian Risi. An anarchy of methods: Current trends in how intelligence is abstracted in ai. *IEEE Intelligent Systems*, 29(6):56–62, 2014.

[4] Melanie Mitchell. *Artificial intelligence: A guide for thinking humans*. Penguin UK, 2019.

[5] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[6] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[9] Daniel Berrar. Bayes' theorem and naive bayes classifier. *Encyclopedia of Bioinformatics and Computational Biology: ABC of Bioinformatics*, 403, 2018.

[10] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[11] Donald Olding Hebb. *The Organizations of Behavior: a Neuropsychological Theory*. Lawrence Erlbaum, 1963.

[12] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[13] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[14] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[15] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[16] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[21] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[22] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligence magazine*, 13(3):55–75, 2018.

[23] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[24] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*, 2015.

[25] Gee Wah Ng. *Brain-mind machinery: brain-inspired computing and mind opening*. World scientific, 2009.

[26] Frederico AC Azevedo, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

[27] Philippe Faure and Henri Korn. Is there chaos in the brain? i. concepts of nonlinear dynamics and methods of investigation. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 324(9):773–793, 2001.

[28] Henri Korn and Philippe Faure. Is there chaos in the brain? ii. experimental evidence and related models. *Comptes rendus biologies*, 326(9):787–840, 2003.

[29] Gabriela Czanner, Sridevi V Sarma, Demba Ba, Uri T Eden, Wei Wu, Emad Eskandar, Hubert H Lim, Simona Temereanca, Wendy A Suzuki, and Emery N Brown. Measuring the signal-to-noise ratio of a neuron. *Proceedings of the National Academy of Sciences*, 112(23):7141–7146, 2015.

[30] NB Harikrishnan, R Vinayakumar, and KP Soman. A machine learning approach towards phishing email detection. In *Proceedings of the Anti-Phishing Pilot at ACM International Workshop on Security and Privacy Analytics (IWSPA AP)*, volume 2013, pages 455–468, 2018.

[31] Yiming Ding, Jae Ho Sohn, Michael G Kawczynski, Hari Trivedi, Roy Harnish, Nathaniel W Jenkins, Dmytro Lituiev, Timothy P Copeland, Mariam S Aboian, Carina Mari Aparici, et al. A deep learning model to predict a diagnosis of alzheimer disease by using 18f-fdg pet of the brain. *Radiology*, 290(2):456, 2019.

[32] A Zerroug, L Terrissa, and A Faure. Chaotic dynamical behavior of recurrent neural network. *Annu. Rev. Chaos Theory Bifurc. Dyn. Syst*, 4:55–66, 2013.

[33] Bruce J Gluckman, Theoden I Netoff, Emily J Neel, William L Ditto, Mark L Spano, and Steven J Schiff. Stochastic resonance in a neuronal network from mammalian brain. *Physical Review Letters*, 77(19):4098, 1996.

[34] Roberto Benzi, Giorgio Parisi, Alfonso Sutera, and Angelo Vulpiani. Stochastic resonance in climatic change. *Tellus*, 34(1):10–16, 1982.

[35] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

[36] Karl G Andersson. Poincaré's discovery of homoclinic points. *Archive for history of exact sciences*, pages 133–147, 1994.

[37] Christian Oestreicher. A history of chaos theory. *Dialogues in clinical neuroscience*, 2022.

[38] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[39] Tien-Yien Li and James A Yorke. Period three implies chaos. In *The theory of chaotic attractors*, pages 77–84. Springer, 2004.

[40] Christos H Skiadas and Charilaos Skiadas. *Handbook of applications of chaos theory*. crc Press, 2017.

[41] Pooja Rani Sharma and Manish Dev Shrimali. Computing with coupled chaotic neuronal maps. *Int. J. Unconv. Comput.*, 7(1-2):115–123, 2011.

[42] Hai-Peng Ren, Murilo S Baptista, and Celso Grebogi. Wireless communication with chaos. *Physical Review Letters*, 110(18):184101, 2013.

[43] Kathleen T.. Alligood, Tim Sauer, and James A Yorke. *Chaos: an introduction to dynamical systems*. Springer New York, 2000.

[44] Simon Haykin and Barry Van Veen. *Signals and systems*. John Wiley & Sons, 2007.

[45] Robert L Devaney. *A first course in chaotic dynamical systems: theory and experiment*. CRC Press, 2018.

[46] Karma Dajani and Cor Kraaikamp. *Ergodic theory of numbers*, volume 29. American Mathematical Soc., 2002.

[47] Nicolás Rubido, Celso Grebogi, and Murilo S Baptista. Entropy-based generating markov partitions for complex systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(3):033611, 2018.

[48] Nithin Nagaraj. *Novel applications of chaos theory to coding and cryptography*. PhD thesis, NIAS, 2008.

[49] Nithin Nagaraj, Prabhakar G Vaidya, and Kishor G Bhat. Arithmetic coding as a non-linear dynamical system. *Communications in Nonlinear Science and Numerical Simulation*, 14(4):1013–1020, 2009.

[50] Nithin Nagaraj. Using cantor sets for error detection. *PeerJ Computer Science*, 5:e171, 2019.

[51] Kwok-Wo Wong, Qiuzhen Lin, and Jianyong Chen. Simultaneous arithmetic coding and encryption using chaotic maps. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(2):146–150, 2010.

[52] Aditi Kathpalia and Nithin Nagaraj. A novel compression based neuronal architecture for memory encoding. In *Proceedings of the 20th international conference on distributed computing and networking*, pages 365–370, 2019.

[53] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[54] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[55] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[56] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[57] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[58] Steven H Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[59] Andrew M Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan D Tracey, and David D Cox. On the information bottleneck theory of deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124020, 2019.

[60] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

[61] Charles B Delahunt and J Nathan Kutz. Putting a bug in ml: The moth olfactory network learns to read mnist. *Neural Networks*, 118:54–64, 2019.

[62] Kazuyuki Aihara, T Takabe, and Masashi Toyoda. Chaotic neural networks. *Physics letters A*, 144(6-7):333–340, 1990.

[63] Nigel T Crook and Tjeerd Olde Scheper. A novel chaotic neural network architecture. In *ESANN*, pages 295–300, 2001.

[64] Walter J Freeman et al. *Mass action in the nervous system*, volume 2004. Citeseer, 1975.

[65] Hung-Jen Chang and Walter J Freeman. Parameter optimization in models of the olfactory neural system. *Neural Networks*, 9(1):1–14, 1996.

[66] Robert Kozma and Walter J Freeman. A possible mechanism for intermittent oscillations in the kiii model of dynamic memories-the case study of olfaction. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 1, pages 52–57. IEEE, 1999.

[67] Ichiro Tsuda. Dynamic link of memory—chaotic memory map in nonequilibrium neural networks. *Neural networks*, 5(2):313–326, 1992.

[68] John S Nicolis and Ichiro Tsuda. Chaotic dynamics of information processing: The "magic number seven plus-minus two" revisited. *Bulletin of Mathematical Biology*, 47(3):343–365, 1985.

[69] Kunihiko Kaneko. Lyapunov analysis and information flow in coupled map lattices. *Physica D: Nonlinear Phenomena*, 23(1-3):436–447, 1986.

[70] Kunihiko Kaneko. Clustering, coding, switching, hierarchical ordering, and control in a network of chaotic elements. *Physica D: Nonlinear Phenomena*, 41(2):137–172, 1990.

[71] Zainab Aram, Sajad Jafari, Jun Ma, Julien C Sprott, Sareh Zendehrouh, and Viet-Thanh Pham. Using chaotic artificial neural networks to model memory in the brain. *Communications in Nonlinear Science and Numerical Simulation*, 44:449–459, 2017.

[72] Kathleen T Alligood, Tim D Sauer, and James A Yorke. Chaos: An introduction to dynamical systems. 1996, 1997.

[73] Agnessa Babloyantz and Carlos Lourenço. Brain chaos and computation. *International Journal of Neural Systems*, 7(04):461–471, 1996.

[74] Colin Barras. Mind maths: Brainquakes on the edge of chaos. *New Scientist*, 217(2903):36, 2013.

[75] Thomas Elbert, Brigitte Rockstroh, Zbigniew J Kowalik, Manfried Hoke, Mark Molnar, James E Skinner, and Niels Birbaumer. Chaotic brain activity. *Electroencephalography and Clinical Neurophysiology/Supplement*, 44:441–449, 1995.

[76] JC Sprott. Is chaos good for learning. *Nonlinear dynamics, psychology, and life sciences*, 17(2):223–232, 2013.

[77] Golnaz Baghdadi, Sajad Jafari, Julien Clinton Sprott, Farzad Towhidkhah, and MR Hashemi Golpayegani. A chaotic model of sustaining attention problem in attention deficit disorder. *Communications in Nonlinear Science and Numerical Simulation*, 20(1):174–185, 2015.

[78] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500, 1952.

[79] James L Hindmarsh and RM Rose. A model of neuronal bursting using three coupled first order differential equations. *Proceedings of the Royal society of London. Series B. Biological sciences*, 221(1222):87–102, 1984.

[80] Richard FitzHugh. Impulses and physiological states in theoretical models of nerve membrane. *Biophysical journal*, 1(6):445–466, 1961.

[81] Jinichi Nagumo, Suguru Arimoto, and Shuji Yoshizawa. An active pulse transmission line simulating nerve axon. *Proceedings of the IRE*, 50(10):2061–2070, 1962.

[82] NK Pareek, Vinod Patidar, and KK Sud. Cryptography using multiple one-dimensional chaotic maps. *Communications in Nonlinear Science and Numerical Simulation*, 10(7):715–723, 2005.

[83] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[84] Yinshui Fan and Arun V Holden. Bifurcations, burstings, chaos and crises in the rose-hindmarsh model for neuronal activity. *Chaos, Solitons & Fractals*, 3(4):439–449, 1993.

[85] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[86] A Aldo Faisal, Luc PJ Selen, and Daniel M Wolpert. Noise in the nervous system. *Nature reviews neuroscience*, 9(4):292–303, 2008.

[87] Robert Brown. Xxvii. a brief account of microscopical observations made in the months of june, july and august 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *The philosophical magazine*, 4(21):161–173, 1828.

[88] Albert Einstein. Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen. *Annalen der physik*, 4, 1905.

[89] Gustaf Ising. Lxxiii. a natural limit for the sensibility of galvanometers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1(4):827–834, 1926.

[90] Harry Nyquist. Thermal agitation of electric charge in conductors. *Physical review*, 32(1):110, 1928.

[91] John Bertrand Johnson. Thermal agitation of electricity in conductors. *Physical review*, 32(1):97, 1928.

[92] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[93] André L Berger. *Climatic Variations and Variability: Facts and Theories: NATO Advanced Study Institute First Course of the International School of Climatology, Ettore Majorana Center for Scientific Culture, Erice, Italy, March 9–21, 1980*, volume 72. Springer Science & Business Media, 2012.

[94] Stéphan Fauve and F Heslot. Stochastic resonance in a bistable system. *Physics Letters A*, 97(1-2):5–7, 1983.

[95] Adi Bulsara, EW Jacobs, Ting Zhou, Frank Moss, and Laszlo Kiss. Stochastic resonance in a single neuron model: Theory and analog simulation. *Journal of Theoretical Biology*, 152(4):531–555, 1991.

[96] David S Leonard and LE Reichl. Stochastic resonance in a chemical reaction. *Physical Review E*, 49(2):1734, 1994.

[97] XiaoYu Zhang, Yong Xu, Qi Liu, and Jürgen Kurths. Rate-dependent tipping-delay phenomenon in a thermoacoustic system with colored noise. *Science China Technological Sciences*, 63(11):2315–2327, 2020.

[98] Mark D McDonnell, Nigel G Stocks, Charles EM Pearce, and Derek Abbott. Stochastic resonance. *Stochastic Resonance*, 2008.

[99] John K Douglass, Lon Wilkens, Eleni Pantazelou, and Frank Moss. Noise enhancement of information transfer in crayfish mechanoreceptors by stochastic resonance. *Nature*, 365(6444):337–340, 1993.

[100] André Longtin. Stochastic resonance in neuron models. *Journal of statistical physics*, 70(1):309–327, 1993.

[101] David F Russell, Lon A Wilkens, and Frank Moss. Use of behavioural stochastic resonance by paddle fish for feeding. *Nature*, 402(6759):291–294, 1999.

[102] Monita Chatterjee and Mark E Robert. Noise enhances modulation sensitivity in cochlear implant listeners: Stochastic resonance in a prosthetic sensory system? *Journal of the Association for Research in Otolaryngology*, 2(2):159–171, 2001.

[103] Adi R Bulsara, Anna Dari, William L Ditto, K Murali, and Sudeshna Sinha. Logical stochastic resonance. *Chemical Physics*, 375(2-3):424–434, 2010.

[104] Ruoxing Mei, Yong Xu, Yongge Li, and Jürgen Kurths. Characterizing stochastic resonance in a triple cavity. *Philosophical Transactions of the Royal Society A*, 379(2198):20200230, 2021.

[105] Shuhei Ikemoto, Fabio DallaLibera, and Koh Hosoda. Noise-modulated neural networks as an application of stochastic resonance. *Neurocomputing*, 277:29–37, 2018.

[106] Achim Schilling, Richard Gerum, Claus Metzner, Andreas Maier, and Patrick Krauss. Intrinsic noise improves speech recognition in a computational model of the auditory pathway. *Frontiers in Neuroscience*, page 795, 2022.

[107] Bruno Andò and Salvatore Graziani. *Stochastic resonance: theory and applications*. Springer Science & Business Media, 2000.

[108] Aruneema Das, NG Stocks, A Nikitin, and EL Hines. Quantifying stochastic resonance in a single threshold detector for random aperiodic signals. *Fluctuation and Noise Letters*, 4(02):L247–L265, 2004.

[109] Ferris Jabr. Does thinking really hard burn more calories. *scientific american*, 18, 2012.

[110] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert systems with applications*, 73:220–239, 2017.

[111] Maher Maalouf and Theodore B Trafalis. Robust weighted kernel logistic regression in imbalanced and rare events data. *Computational Statistics & Data Analysis*, 55(1):168–183, 2011.

[112] Marco Ciotti, Massimo Ciccozzi, Alessandro Terrinoni, Wen-Can Jiang, Cheng-Bin Wang, and Sergio Bernardini. The covid-19 pandemic. *Critical reviews in clinical laboratory sciences*, 57(6):365–388, 2020.

[113] Antoni Trilla, Guillem Trilla, and Carolyn Daer. The 1918 "spanish flu" in spain. *Clinical infectious diseases*, 47(5):668–673, 2008.

[114] Craig Beaman, Ashley Barkworth, Toluwalope David Akande, Saqib Hakak, and Muhammad Khurram Khan. Ransomware: Recent advances, analysis, challenges and future research directions. *Computers & Security*, 111:102490, 2021.

[115] Suvasini Panigrahi, Amlan Kundu, Shamik Sural, and Arun K Majumdar. Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning. *Information Fusion*, 10(4):354–363, 2009.

[116] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[117] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[118] Muhammad Atif Tahir, Josef Kittler, Krystian Mikolajczyk, and Fei Yan. A multiple expert approach to the class imbalance problem using inverse random under sampling. In *International workshop on multiple classifier systems*, pages 82–91. Springer, 2009.

[119] Ludmila I Kuncheva and William J Faithfull. Pca feature extraction for change detection in multidimensional unlabeled data. *IEEE transactions on neural networks and learning systems*, 25(1):69–80, 2013.

[120] Olayinka O Ogundile, Ayinde M Usman, Oluwaseyi P Babalola, and Daniel JJ Versfeld. Dynamic mode decomposition: A feature extraction technique based hidden markov model for detection of mysticetes' vocalisations. *Ecological Informatics*, 63:101306, 2021.

[121] Md Afzal Hossan, Sheeraz Memon, and Mark A Gregory. A novel approach for mfcc feature extraction. In *2010 4th International Conference on Signal Processing and Communication Systems*, pages 1–5. IEEE, 2010.

[122] Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on knowledge and data engineering*, 25(6):1336–1353, 2012.

[123] Vaijayanthi Nagarajan, Elizabeth Caroline Britto, and Senthilvel Murugan Veeraputhiran. Feature extraction based on empirical mode decomposition for automatic mass classification of mammogram images. *Medicine in Novel Technology and Devices*, 1:100004, 2019.

[124] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[125] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[126] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[127] Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

[128] B Vandeginste. Parvus: An extendable package of programs for data exploration, classification and correlation, m. forina, r. leardi, c. armanino and s. lanteri, elsevier, amsterdam, 1988, price: Us $645 isbn 0-444-43012-1. *Journal of Chemometrics*, 4(2):191–193, 1990.

[129] Eugen Gillich and Volker Lohweg. Banknote authentication. *1. Jahreskolloquium Bild. Der Autom*, pages 1–8, 2010.

[130] Shelby J Haberman. The analysis of residuals in cross-classified tables. *Biometrics*, pages 205–220, 1973.

[131] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. SPIE, 1993.

[132] Free Spoken Digit Dataset. Available online: https://github. com. *Jakobovski/free-spoken-digit-dataset (accessed on 17 May 2022)*, 2021.

[133] Yann LeCun and Corinna Cortes. Mnist handwritten dataset. *: http://yann.lecun.com/exdb/mnist/*.

[134] Jonathan B Dingwell. Lyapunov exponents. *Wiley encyclopedia of biomedical engineering*, 2006.

[135] Kaijun Wu, Tianqi Luo, Huaiwei Lu, and Yang Wang. Bifurcation study of neuron firing activity of the modified hindmarsh–rose model. *Neural Computing and Applications*, 27(3):739–747, 2016.

[136] Arun V Holden and Yin-Shui Fan. From simple to simple bursting oscillatory behaviour via chaos in the rose-hindmarsh model for neuronal activity. *Chaos, Solitons & Fractals*, 2(3):221–236, 1992.

[137] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[138] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[139] Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–207, 2018.

[140] Luc Steels. *The biology and technology of intelligent autonomous agents*, volume 144. Springer Science & Business Media, 2012.

[141] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[142] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[143] Zixing Zhang, Jürgen Geiger, Jouni Pohjalainen, Amr El-Desoky Mousa, Wenyu Jin, and Björn Schuller. Deep learning for environmentally robust speech recognition: An overview of recent developments. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(5):1–28, 2018.

[144] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic books, 2018.

[145] Norbert Wiener. The theory of prediction. *Modern mathematics for engineers*, 1:125–139, 1956.

[146] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pages 424–438, 1969.

[147] Thomas Schreiber. Measuring information transfer. *Physical review letters*, 85(2):461, 2000.

[148] Aditi Kathpalia and Nithin Nagaraj. Data-based intervention approach for complexity-causality measure. *PeerJ Computer Science*, 5:e196, 2019.

[149] Craig Hiemstra and Jonathan D Jones. Testing for linear and nonlinear granger causality in the stock price-volume relation. *The Journal of Finance*, 49(5):1639–1664, 1994.

[150] Song Zan Chiou-Wei, Ching-Fu Chen, and Zhen Zhu. Economic growth and energy consumption revisited—evidence from linear and nonlinear granger causality. *Energy Economics*, 30(6):3063–3076, 2008.

[151] Timothy J Mosedale, David B Stephenson, Matthew Collins, and Terence C Mills. Granger causality of coupled climate processes: Ocean feedback on the north atlantic oscillation. *Journal of climate*, 19(7):1182–1194, 2006.

[152] Adolf Stips, Diego Macias, Clare Coughlan, Elisa Garcia-Gorriz, and X San Liang. On the causal structure between co 2 and global temperature. *Scientific reports*, 6(1):1–9, 2016.

[153] Anil K Seth, Adam B Barrett, and Lionel Barnett. Granger causality analysis in neuroscience and neuroimaging. *Journal of Neuroscience*, 35(8):3293–3297, 2015.

[154] Raul Vicente, Michael Wibral, Michael Lindner, and Gordon Pipa. Transfer entropy—a model-free measure of effective connectivity for the neurosciences. *Journal of computational neuroscience*, 30(1):45–67, 2011.

[155] Raha Moraffah, Paras Sheth, Mansooreh Karami, Anchit Bhattacharya, Qianru Wang, Anique Tahir, Adrienne Raglin, and Huan Liu. Causal inference for time series analysis: Problems, methods and evaluation. *Knowledge and Information Systems*, pages 1–45, 2021.

[156] Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily Fox. Neural granger causality. *arXiv preprint arXiv:1802.05842*, 2018.

[157] Ričards Marcinkevičs and Julia E Vogt. Interpretable models for granger causality using self-explaining neural networks. *arXiv preprint arXiv:2101.07600*, 2021.

[158] Judea Pearl et al. Causality: Models, reasoning and inference. *Cambridge, UK: CambridgeUniversityPress*, 19:2, 2000.

[159] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

[160] Patrik Hoyer, Dominik Janzing, Joris M Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. *Advances in neural information processing systems*, 21, 2008.

[161] Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109(5):612–634, 2021.

[162] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[163] Robert M May. Simple mathematical models with very complicated dynamics. In *The Theory of Chaotic Attractors*, pages 85–93. Springer, 2004.

[164] François Chollet. Keras. https://github.com/fchollet/keras, 2015.

[165] Lionel Barnett and Anil K Seth. The mvgc multivariate granger causality toolbox: a new approach to granger-causal inference. *Journal of neuroscience methods*, 223:50–68, 2014.

[166] Lionel Barnett and Anil K Seth. The mvgc multivariate granger causality toolbox: a new approach to granger-causal inference. *Journal of neuroscience methods*, 223:50–68, 2014.

[167] Brendan G Veilleux. The analysis of a predatory interaction between didinium and paramecium. *Master's thesis. University of Alberta, Edmondton*, 1976.

[168] Christian Jost and Stephen P Ellner. Testing for predator dependence in predator-prey dynamics: a non-parametric approach. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 267(1453):1611–1620, 2000.

[169] Seth M Hirsh, Sara M Ichinaga, Steven L Brunton, J Nathan Kutz, and Bingni W Brunton. Structured time-delay models for dynamical systems with connections to frenet–serret frame. *Proceedings of the Royal Society A*, 477(2254):20210097, 2021.

[170] Nikita Agarwal, Aditi Kathpalia, and Nithin Nagaraj. Distinguishing different levels of consciousness using a novel network causal activity measure. In *2019 Global Conference for Advancement in Technology (GCAT)*, pages 1–5. IEEE, 2019.

A1 Appendix A

Appendix A contains information pertaining to chapter 5. It contains the following – (1) description of datasets used in our study including the coding rule for the labels of different classes, (2) hyperparamter tuning details for each dataset and for each ML algorithm (Decision Tree, Random Forest, AdaBoost, SVM, k-NN) and NL algorithm (`ChaosNet`) used in the study, (3) the test data macro F1-scores for each algorithm in the high training sample regime for each dataset.

A1.1 Dataset Description

Iris

Table A1: ***Iris***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
Iris-Setosa	0	50
Iris-Versicolour	1	50
Iris-Virginica	2	50

Ionosphere

Table A2: ***Ionosphere***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
b (Bad)	0	126
g (Good)	1	225

Wine

Table A3: ***Wine***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
1	0	59
2	1	71
3	2	48

Bank Note Authentication

Table A4: ***Bank Note Authentication***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
0 (Genuine)	0	762
1 (Forgery)	1	610

Haberman's Survival

Table A5: ***Haberman's Survival***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
1 ($<$ 5yrs)	0	225
2 ($\geq$ 5yrs)	1	81

Breast Cancer Wisconsin

Table A6: ***Breast Cancer Wisconsin***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
M (Malignant)	0	212
B (Benign)	1	357

Statlog (Heart)

Table A7: ***Statlog (Heart)***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
1 (Absence)	0	150
2 (Presence)	1	120

Seeds

Table A8: ***Seeds***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
1.0 (Kama)	0	70
2.0 (Rosa)	1	70
3.0 (Canadian)	2	70

FSDD

Table A9: ***FSDD***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Complete Dataset)**
0	0	50
1	1	50
2	2	50
3	3	50
4	4	46
5	5	41
6	6	50
7	7	50
8	8	43
9	9	50

A1.2 MNIST

Table A10: ***MNIST***: Rule followed for renaming of the class labels.

Class Label	**Numeric Code**	**Number of Instances (In Entire Dataset)**
0	0	6903
1	1	7877
2	2	6990
3	3	7141
4	4	6824
5	5	6313
6	6	6876
7	7	7293
8	8	6825
9	9	6958

A1.3 Hyperparameter Tuning

The hyperparameter tuning for all algorithms for the respective datasets are provided below:

Decision Tree

Following are the hyperparameters tuned for Decision Tree:

1. **min_samples_leaf**: Defines the minimum number of samples required for a leaf node in the decision tree. It is tuned from 1 to 10 with a step-size of 1.

2. **max_depth**: Declares the maximum depth to which a decision tree can be grown. It is tuned from 1 to 10 with a step-size of 1.

3. **ccp_alpha**: A numpy array of alpha values obtained by devising Cost Complexity Pruning on the original decision tree. This array is obtained using the *cost_complexity_pruning_path*.

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for Decision Tree are available in Table A11 and A12.

Table A11: ***Decision Tree***: Tuned hyperparameters for all nine datasets (Part I). The performance metric used for the provided results is macro F1-score.

Dataset	Implementation	Tuned Hyperparameters	F1 Score
Iris	Stand-Alone Decision Tree	$min\_samples\_leaf = 3$	0.931
		$max\_depth = 3$	
		$ccp\_alpha = 0.0$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 3$	0.955
		$max\_depth = 4$	
		$ccp\_alpha = 0.0$	
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.13$	
Ionosphere	Stand-Alone Decision Tree	$min\_samples\_leaf = 1$	0.882
		$max\_depth = 2$	
		$ccp\_alpha = 0.0$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 1$	0.921
		$max\_depth = 7$	
		$ccp\_alpha = 0.0$	
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.22$	
Wine	Stand-Alone Decision Tree	$min\_samples\_leaf = 1$	0.916
		$max\_depth = 3$	
		$ccp\_alpha = 0.0$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 1$	0.949
		$max\_depth = 6$	
		$ccp\_alpha = 0.0$	
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.07$	
Bank Note Authentication	Stand-Alone Decision Tree	$min\_samples\_leaf = 1$	0.977
		$max\_depth = 8$	
		$ccp\_alpha = 0.0$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 1$	0.961
		$max\_depth = 6$	
		$ccp\_alpha = 0.000836$	
		$q = 0.080$	
		$b = 0.250$	
		$\epsilon = 0.233$	
Haberman's Survival	Stand-Alone Decision Tree	$min\_samples\_leaf = 4$	0.614
		$max\_depth = 6$	
		$ccp\_alpha = 0.005389$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 2$	0.648
		$max\_depth = 8$	
		$ccp\_alpha = 0.005389$	
		$q = 0.81$	
		$b = 0.14$	
		$\epsilon = 0.003$	

Random Forest

Following are the hyperparameters tuned for Random Forest:

1. **n_estimators**: Defines the number of trees in the forest being grown. The values are tuned from the array $[1, 10, 100, 1000, 10000]$.

2. **max_depth**: Declares the maximum depth each tree the the forest is allowed

Table A12: ***Decision Tree***: Tuned hyperparameters for all nine datasets (Part II). The performance metric used for the provided results is macro F1-score.

Dataset	Implementation	Tuned Hyperparameters	F1 Score
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone Decision Tree	$min\_samples\_leaf = 1$	0.920
		$max\_depth = 4$	
		$ccp\_alpha = 0.0$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 2$	0.945
		$max\_depth = 8$	
		$ccp\_alpha = 0.005595$	
		$q = 0.930$	
		$b = 0.490$	
		$\epsilon = 0.159$	
Statlog (Heart)	Stand-Alone Decision Tree	$min\_samples\_leaf = 6$	0.779
		$max\_depth = 3$	
		$ccp\_alpha = 0.006818$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 7$	0.786
		$max\_depth = 4$	
		$ccp\_alpha = 0.0$	
		$q = 0.08$	
		$b = 0.06$	
		$\epsilon = 0.17$	
Seeds	Stand-Alone Decision Tree	$min\_samples\_leaf = 2$	0.918
		$max\_depth = 4$	
		$ccp\_alpha = 0.005844$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 2$	0.949
		$max\_depth = 5$	
		$ccp\_alpha = 0.0$	
		$q = 0.02$	
		$b = 0.07$	
		$\epsilon = 0.238$	
FSDD	Stand-Alone Decision Tree	$min\_samples\_leaf = 1$	0.536
		$max\_depth = 9$	
		$ccp\_alpha = 0.0099206$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 1$	0.537
		$max\_depth = 10$	
		$ccp\_alpha = 0.0061384$	
		$q = 0.34$	
		$b = 0.499$	
		$\epsilon = 0.178$	
MNIST	Stand-Alone Decision Tree	$min\_samples\_leaf = 3$	0.735
		$max\_depth = 10$	
		$ccp\_alpha = 0.0015$	
	ChaosFEX + Decision Tree	$min\_samples\_leaf = 1$	0.616
		$max\_depth = 10$	
		$ccp\_alpha = 0.0$	
		$q = 0.34$	
		$b = 0.499$	
		$\epsilon = 0.399$	

to have. It is tuned from 1 to 10 with a step-size of 1.

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for Random Forest are available in

Table A13 and A14

Table A13: ***Random Forest***: Tuned hyperparameters for all nine datasets (Part I). The performance metric used for the provided results is macro F1-score.

Dataset	Implementation	Tuned Hyperparameters	F1 Score
Iris	Stand-Alone Random Forest	$n\_estimators = 100$	0.944
		$max\_depth = 3$	
	ChaosFEX + Random Forest	$n\_estimators = 10$	0.944
		$max\_depth = 5$	
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.11$	
Ionosphere	Stand-Alone Random Forest	$n\_estimators = 10000$	0.923
		$max\_depth = 4$	
	ChaosFEX + Random Forest	$n\_estimators = 100$	0.928
		$max\_depth = 10$	
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.23$	
Wine	Stand-Alone Random Forest	$n\_estimators = 10$	0.980
		$max\_depth = 4$	
	ChaosFEX + Random Forest	$n\_estimators = 10$	0.987
		$max\_depth = 5$	
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.05$	
Bank Note Authentication	Stand-Alone Random Forest	$n\_estimators = 100$	0.991
		$max\_depth = 8$	
	ChaosFEX + Random Forest	$n\_estimators = 100$	0.967
		$max\_depth = 7$	
		$q = 0.080$	
		$b = 0.250$	
		$\epsilon = 0.233$	
Haberman's Survival	Stand-Alone Random Forest	$n\_estimators = 1$	0.621
		$max\_depth = 3$	
	ChaosFEX + Random Forest	$n\_estimators = 1000$	0.651
		$max\_depth = 9$	
		$q = 0.810$	
		$b = 0.140$	
		$\epsilon = 0.003$	
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone Random Forest	$n\_estimators = 1000$	0.956
		$max\_depth = 9$	
	ChaosFEX + Random Forest	$n\_estimators = 100$	0.955
		$max\_depth = 7$	
		$q = 0.930$	
		$b = 0.490$	
		$\epsilon = 0.159$	

Table A14: ***Random Forest***: Tuned hyperparameters for all nine datasets (Part II). The performance metric used for the provided results is macro F1-score.

Dataset	Implementation	Tuned Hyperparameters	F1 Score
Statlog (Heart)	Stand-Alone Random Forest	$n\_estimators = 1000$	0.839
		$max\_depth = 2$	
	ChaosFEX + Random Forest	$n\_estimators = 10000$	0.830
		$max\_depth = 4$	
		$q = 0.08$	
		$b = 0.06$	
		$\epsilon = 0.17$	
Seeds	Stand-Alone Random Forest	$n\_estimators = 100$	0.918
		$max\_depth = 5$	
	ChaosFEX + Random Forest	$n\_estimators = 1000$	0.941
		$max\_depth = 5$	
		$q = 0.020$	
		$b = 0.070$	
		$\epsilon = 0.238$	
FSDD	Stand-Alone Random Forest	$n\_estimators = 1000$	0.9479
		$max\_depth = 9$	
	ChaosFEX + Random Forest	$n\_estimators = 1000$	0.921
		$max\_depth = 8$	
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.178$	
MNIST	Stand-Alone Random Forest	$n\_estimators = 10000$	0.944
		$max\_depth = 10$	
	ChaosFEX + Random Forest	$n\_estimators = 10000$	0.872
		$max\_depth = 10$	
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.399$	

AdaBoost

Following are the hyperparameters tuned for Adaptive Boosting (AdaBoost):

1. **n_estimators**: An upper limit for the number of stumps being grown. The values are tuned from the array
   $[1, 10, 50, 100, 500, 1000, 5000, 10000]$.

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for AdaBoost are available in Table A15.

Table A15: ***AdaBoost***: Tuned hyperparameters for all nine datasets. The performance metric used for the provided results is macro F1-score.

Dataset	Implementation	Tuned Hyperparameters	F1 Score
Iris	Stand-Alone AdaBoost	$n\_estimators = 50$	0.929
	ChaosFEX + AdaBoost	$n\_estimators = 10$	0.915
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.03$	
Ionosphere	Stand-Alone AdaBoost	$n\_estimators = 50$	0.929
	ChaosFEX + AdaBoost	$n\_estimators = 500$	0.921
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.02$	
Wine	Stand-Alone AdaBoost	$n\_estimators = 10$	0.896
	ChaosFEX + AdaBoost	$n\_estimators = 10$	0.893
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.05$	
Bank Note Authentication	Stand-Alone AdaBoost	$n\_estimators = 500$	0.999
	ChaosFEX + AdaBoost	$n\_estimators = 500$	0.934
		$q = 0.080$	
		$b = 0.250$	
		$\epsilon = 0.233$	
Haberman's Survival	Stand-Alone AdaBoost	$n\_estimators = 10$	0.620
	ChaosFEX + AdaBoost	$n\_estimators = 1$	0.639
		$q = 0.81$	
		$b = 0.14$	
		$\epsilon = 0.003$	
Breast Cancer Wisconsin (Heart)	Stand-Alone AdaBoost	$n\_estimators = 1000$	0.962
	ChaosFEX + AdaBoost	$n\_estimators = 100$	0.952
		$q = 0.930$	
		$b = 0.490$	
		$\epsilon = 0.159$	
Statlog (Heart)	Stand-Alone AdaBoost	$n\_estimators = 10$	0.816
	ChaosFEX + AdaBoost	$n\_estimators = 50$	0.815
		$q = 0.08$	
		$b = 0.06$	
		$\epsilon = 0.17$	
Seeds	Stand-Alone AdaBoost	$n\_estimators = 500$	0.563
	ChaosFEX + AdaBoost	$n\_estimators = 10$	0.918
		$q = 0.020$	
		$b = 0.070$	
		$\epsilon = 0.238$	
FSDD	Stand-Alone AdaBoost	$n\_estimators = 50$	0.086
	ChaosFEX + AdaBoost	$n\_estimators = 50$	0.170
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.178$	
MNIST	Stand-Alone AdaBoost	$n\_estimators = 5000$	0.630
	ChaosFEX + AdaBoost	$n\_estimators = 10000$	0.525
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.399$	

Support Vector Machine

Following are the hyperparameters tuned for Support Vector Machine (SVM):

1. **C**: Controls the bias-variance trade-off of the algorithm, known as the *"Regularization Parameter"*. It is tuned from 0.1 to 100.0 with a step-size of 0.1

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for Support Vector Machine are available in Table A16.

Table A16: ***Support Vector Machine (SVM)***: Tuned hyperparameters for all nine datasets. Only FSDD uses the 'linear' kernel. All remaining datasets, use the 'rbf' kernel. The performance metric used for the provided results is macro F1-score.

Dataset	Implementation	Tuned Hyperparameters	F1 Score
Iris	Stand-Alone SVM	$C = 4.4$	0.954
	ChaosFEX + SVM	$C = 38.7$	0.958
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.13$	
Ionosphere	Stand-Alone SVM	$C = 2.6$	0.934
	ChaosFEX + SVM	$C = 3.4$	0.926
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.37$	
Wine	Stand-Alone SVM	$C = 0.6$	0.975
	ChaosFEX + SVM	$C = 16.0$	0.958
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.01$	
Bank Note Authentication	Stand-Alone SVM	$C = 1.3$	1.0
	ChaosFEX + SVM	$C = 16.8$	0.965
		$q = 0.080$	
		$b = 0.250$	
		$\epsilon = 0.233$	
Haberman's Survival	Stand-Alone SVM	$C = 19.8$	0.597
	ChaosFEX + SVM	$C = 15.7$	0.609
		$q = 0.81$	
		$b = 0.14$	
		$\epsilon = 0.003$	
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone SVM	$C = 16.0$	0.981
	ChaosFEX + SVM	$C = 20.3$	0.948
		$q = 0.930$	
		$b = 0.490$	
		$\epsilon = 0.159$	
Statlog (Heart)	Stand-Alone SVM	$C = 0.2$	0.852
	ChaosFEX + SVM	$C = 2.5$	0.825
		$q = 0.08$	
		$b = 0.06$	
		$\epsilon = 0.17$	
Seeds	Stand-Alone SVM	$C = 0.9$	0.948
	ChaosFEX + SVM	$C = 2.4$	0.909
		$q = 0.020$	
		$b = 0.070$	
		$\epsilon = 0.238$	
FSDD	Stand-Alone SVM (linear kernel)	$C = 0.7$	0.952
	ChaosFEX + SVM (linear kernel)	$C = 6.1$	0.978
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.178$	
MNIST	Stand-Alone SVM (radial kernel)	$C = 2.7$	0.930
	ChaosFEX + SVM (radial kernel)	$C = 2.0$	0.896
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.399$	

k-Nearest Neighbors

Following are the hyperparameters tuned for k-Nearest Neighbors:

1. k: Defines the number of nearest training data samples to the testing data sample to be chosen. It is tuned from 1 to 6 with a step-size of 2.

All remaining hyperparameters offered by scikit-learn are retained in their default forms.The results of hyperparameter tuning for k-Nearest Neighbors are available in Table A17.

Table A17: ***k-Nearest Neighbors (k-NN)***: Tuned hyperparameters for all nine datasets. The performance metric used for the provided results is macro F1-score.

Dataset	Implementation	Tuned Hyperparameters	F1 Score
Iris	Stand-Alone k-NN	$k = 5$	0.944
	ChaosFEX + k-NN	$k = 1$	0.936
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.12$	
Ionosphere	Stand-Alone k-NN	$k = 1$	0.814
	ChaosFEX + k-NN	$k = 1$	0.886
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.11$	
Wine	Stand-Alone k-NN	$k = 5$	0.957
	ChaosFEX + k-NN	$k = 1$	0.966
		$q = 0.21$	
		$b = 0.969$	
		$\epsilon = 0.10$	
Bank Note Authentication	Stand-Alone k-NN	$k = 5$	0.998
	ChaosFEX + k-NN	$k = 3$	0.967
		$q = 0.080$	
		$b = 0.250$	
		$\epsilon = 0.233$	
Haberman's Survival	Stand-Alone k-NN	$k = 1$	0.562
	ChaosFEX + k-NN	$k = 5$	0.659
		$q = 0.81$	
		$b = 0.14$	
		$\epsilon = 0.003$	
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone k-NN	$k = 5$	0.964
	ChaosFEX + k-NN	$k = 1$	0.932
		$q = 0.930$	
		$b = 0.490$	
		$\epsilon = 0.159$	
Statlog (Heart)	Stand-Alone k-NN	$k = 5$	0.803
	ChaosFEX + k-NN	$k = 5$	0.791
		$q = 0.08$	
		$b = 0.06$	
		$\epsilon = 0.17$	
Seeds	Stand-Alone k-NN	$k = 5$	0.930
	ChaosFEX + k-NN	$k = 1$	0.876
		$q = 0.020$	
		$b = 0.070$	
		$\epsilon = 0.238$	
FSDD	Stand-Alone k-NN	$k = 1$	0.834
	ChaosFEX + k-NN	$k = 1$	0.909
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.178$	
MNIST	Stand-Alone k-NN	$k = 1$	0.898
	ChaosFEX + k-NN	$k = 1$	0.922
		$q = 0.340$	
		$b = 0.499$	
		$\epsilon = 0.3382$	

ChaosNet

1. q: Initial Neural Activity, it is varied from 0.01 to 0.49 with a step-size of 0.01.

2. b: Discrimination Threshold, it is varied from 0.01 to 0.49 with a step-size of 0.01.

3. ϵ: Noise Intensity, it is varied from 0.001 to 0.499 with a step-size of 0.001.

A1.4 Results

Decision Tree

The results of all experiments using Decision Tree in the high training sample regime are shown in Table A18.

Table A18: ***Decision Tree***: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

Dataset	Implementation	F1 Score (Test Data)
Iris	Stand-Alone Decision Tree	0.967
	ChaosFEX + Decision Tree	1.0
Ionosphere	Stand-Alone Decision Tree	0.881
	ChaosFEX + Decision Tree	0.891
Wine	Stand-Alone Decision Tree	0.904
	ChaosFEX + Decision Tree	0.822
Bank Note Authentication	Stand-Alone Decision Tree	0.933
	ChaosFEX + Decision Tree	0.978
Haberman's Survival	Stand-Alone Decision Tree	0.516
	ChaosFEX + Decision Tree	0.482
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone Decision Tree	0.696
	ChaosFEX + Decision Tree	0.882
Statlog (Heart)	Stand-Alone Decision Tree	0.697
	ChaosFEX + Decision Tree	0.878
Seeds	Stand-Alone Decision Tree	0.880
	ChaosFEX + Decision Tree	0.880
FSDD	Stand-Alone Decision Tree	0.603
	ChaosFEX + Decision Tree	0.579
MNIST	Stand-Alone Decision Tree	0.762
	ChaosFEX + Decision Tree	0.765

Random Forest

The results of all experiments using Random Forest in the high training sample regime are shown in Table A19.

Table A19: ***Random Forest***: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

Dataset	Implementation	F1 Score (Test Data)
Iris	Stand-Alone Random Forest	1.0
	ChaosFEX + Random Forest	1.0
Ionosphere	Stand-Alone Random Forest	0.909
	ChaosFEX + Random Forest	0.924
Wine	Stand-Alone Random Forest	0.966
	ChaosFEX + Random Forest	0.943
Bank Note Authentication	Stand-Alone Random Forest	0.974
	ChaosFEX + Random Forest	0.978
Haberman's Survival	Stand-Alone Random Forest	0.560
	ChaosFEX + Random Forest	0.398
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone Random Forest	0.919
	ChaosFEX + Random Forest	0.918
Statlog (Heart)	Stand-Alone Random Forest	0.838
	ChaosFEX + Random Forest	0.838
Seeds	Stand-Alone Random Forest	0.877
	ChaosFEX + Random Forest	0.828
FSDD	Stand-Alone Random Forest	0.970
	ChaosFEX + Random Forest	0.937
MNIST	Stand-Alone Random Forest	0.724
	ChaosFEX + Random Forest	0.914

Adaptive Boosting (AdaBoost)

The results of all experiments using AdaBoost in the high training sample regime are shown in Table A20.

Table A20: ***AdaBoost***: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

Dataset	Implementation	F1 Score (Test Data)
Iris	Stand-Alone AdaBoost	0.967
	ChaosFEX + AdaBoost	0.865
Ionosphere	Stand-Alone AdaBoost	0.926
	ChaosFEX + AdaBoost	0.925
Wine	Stand-Alone AdaBoost	0.833
	ChaosFEX + AdaBoost	0.846
Bank Note Authentication	Stand-Alone AdaBoost	0.985
	ChaosFEX + AdaBoost	0.910
Haberman's Survival	Stand-Alone AdaBoost	0.505
	ChaosFEX + AdaBoost	0.609
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone AdaBoost	0.858
	ChaosFEX + AdaBoost	0.881
Statlog (Heart)	Stand-Alone AdaBoost	0.777
	ChaosFEX + AdaBoost	0.717
Seeds	Stand-Alone AdaBoost	0.746
	ChaosFEX + AdaBoost	0.873
FSDD	Stand-Alone AdaBoost	0.080
	ChaosFEX + AdaBoost	0.397
MNIST	Stand-Alone AdaBoost	0.698
	ChaosFEX + AdaBoost	0.600

Support Vector Machine (SVM)

The results of all experiments using Support Vector Machine (SVM) in the high training sample regime are shown in Table A21.

Table A21: ***Support Vector Machine (SVM)***: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

Dataset	Implementation	F1 Score (Test Data)
Iris	Stand-Alone SVM	0.966
	ChaosFEX + SVM	0.933
Ionosphere	Stand-Alone SVM	0.924
	ChaosFEX + SVM	0.909
Wine	Stand-Alone SVM	0.928
	ChaosFEX + SVM	0.896
Bank Note Authentication	Stand-Alone SVM	0.993
	ChaosFEX + SVM	0.978
Haberman's Survival	Stand-Alone SVM	0.437
	ChaosFEX + SVM	0.447
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone SVM	0.824
	ChaosFEX + SVM	0.918
Statlog (Heart)	Stand-Alone SVM	0.844
	ChaosFEX + SVM	0.801
Seeds	Stand-Alone SVM	0.924
	ChaosFEX + SVM	0.827
FSDD	Stand-Alone SVM	0.952
	ChaosFEX + SVM	0.978
MNIST	Stand-Alone SVM	0.967
	ChaosFEX + SVM	0.970

k - Nearest Neighbors

The results of all experiments using k - Nearest Neighbors in the high training sample regime are shown in Table A22.

Table A22: ***k-Nearest Neighbors (k-NN)***: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

Dataset	Implementation	F1 Score (Test Data)
Iris	Stand-Alone k-NN	1.0
	ChaosFEX + k-NN	0.902
Ionosphere	Stand-Alone k-NN	0.821
	ChaosFEX + k-NN	0.939
Wine	Stand-Alone k-NN	0.943
	ChaosFEX + k-NN	0.873
Bank Note Authentication	Stand-Alone k-NN	0.993
	ChaosFEX + k-NN	0.971
Haberman's Survival	Stand-Alone k-NN	0.480
	ChaosFEX + k-NN	0.455
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone k-NN	0.954
	ChaosFEX + k-NN	0.935
Statlog (Heart)	Stand-Alone k-NN	0.847
	ChaosFEX + k-NN	0.739
Seeds	Stand-Alone k-NN	0.924
	ChaosFEX + k-NN	0.854
FSDD	Stand-Alone k-NN	0.885
	ChaosFEX + k-NN	0.938
MNIST	Stand-Alone k-NN	0.9689
	ChaosFEX + k-NN	0.9682

Gaussian Naive Bayes

The results of all experiments using Gaussian Naive Bayes (GNB) in the high training sample regime are shown in Table A23.

Table A23: ***Gaussian Naive Bayes (GNB)***: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

Dataset	Implementation	F1 Score (Test Data)
Iris	Stand-Alone GNB	0.966
	ChaosFEX + GNB	0.933
Ionosphere	Stand-Alone GNB	0.862
	ChaosFEX + GNB	0.771
Wine	Stand-Alone GNB	1.0
	ChaosFEX + GNB	0.976
Bank Note Authentication	Stand-Alone GNB	0.785
	ChaosFEX + GNB	0.781
Haberman's Survival	Stand-Alone GNB	0.572
	ChaosFEX + GNB	0.535
Breast Cancer Wisconsin (Diagnostic)	Stand-Alone GNB	0.858
	ChaosFEX + GNB	0.812
Statlog (Heart)	Stand-Alone GNB	0.826
	ChaosFEX + GNB	0.788
Seeds	Stand-Alone GNB	0.846
	ChaosFEX + GNB	0.849
FSDD	Stand-Alone GNB	0.919
	ChaosFEX + GNB	0.687
MNIST	Stand-Alone GNB	0.508
	ChaosFEX + GNB	0.507

A1.5 Code Availability

The codes developed as a part of this thesis are available below:

- CFX package: https://github.com/pranaysy/ChaosFEX.
- CFX+ML: https://github.com/deeksha-sethi03/nl-imbalanced-learning.
- SR in NL: https://github.com/HarikrishnanNB/stochastic_resonance_and_nl.