

AMERICAN UNIVERSITY OF BEIRUT

A STUDY ON LIQUID STATE MACHINE FOR PATTERN
RECOGNITION

by
OBADA MOHAMMAD YASSER AL ZOUBI

A thesis
submitted in partial fulfillment of the requirements
for the degree of Master of Engineering
to the Department of Electrical and Computer Engineering
of the Faculty of Engineering and Architecture
at the American University of Beirut

Beirut, Lebanon
January 2016

AMERICAN UNIVERSITY OF BEIRUT

A STUDY ON LIQUID STATE MACHINE FOR PATTERN
RECOGNITION

by
OBADA MOHAMMAD YASSER AL ZOUBI

Approved by:

Dr. Mariette Awad, Associate Professor
Department of Electrical and Computer Engineering

Advisor

Prof. Mohamad Adnan Al-Alaoui, Professor
Department of Electrical and Computer Engineering

Member of Committee

M. A. Al-Alaoui

Prof. Nikola Kasabov, Professor
KEDRI, Auckland University of Technology

Member of Committee

4/0

//

Date of thesis/dissertation defense: January 8, 2016

AMERICAN UNIVERSITY OF BEIRUT

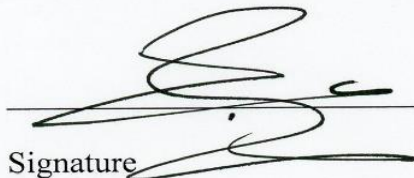
THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: AL Zoubi Obada Mohammad Yasser
Last First Middle

Master's Thesis Master's Project Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, **three years after the date of submitting my thesis, dissertation, or project**, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.


Signature

Jan 18, 2016
Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

ACKNOWLEDGMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Mariette Awad for the continuous support of my master study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Mohamad Adnan Alaoui and Prof. Nikola Kasabov for their insightful comments and encouragement.

I must also acknowledge my friend Nickolas Mitri for helping me in chapter VIII with designing the experimental part and the mobile application.

Last, but not the least, a special thanks to my family. Words cannot express how much grateful I am to my mother, and father for all of the sacrifices that you've made on my behalf. Your prayer for me was what sustained me this far. I would also like to thank my brother and sisters who supported me in writing, and incited me to strive towards my goal. Also, I would like express appreciation to my friends who inspired me through my study and life. I also place on record, my sense of gratitude to one and all, who directly or indirectly, helped me to accomplish my work.

AN ABSTRACT OF THE THESIS OF

Obada Mohammad Yasser Al Zoubi for Master of Engineering
Major: Electrical and Computer Engineering

Title: A Study on Liquid State Machine for Pattern Recognition

Spiking Neural Networks (SNNs) are a new promising approach for machine learning because they increase the ratio of biological realism, and thus the ability to capture complex data patterns. SNNs belong to the third generation of Artificial Neural Networks (ANNs). In contrast to the first and second generations of ANNs, SNNs deal with spatial-temporal information effectively. Liquid State Machine (LSM), introduced by Wolfgang Maass in 2003, is a randomly and sparsely recurrent SNNs that answers for the firmness in training SNNs and Recurrent Neural Networks (RNNs). One main advantage of using LSM is its ability to handle data streams from input to generate high-dimensional separable outputs, which makes LSM a suitable approach for dynamical systems pattern recognition.

Motivated by the compelling capabilities of LSM, this thesis explores different case studies in LSM. First, the work harnesses LSM for Emotion Recognition from EEG signals, where we use LSM as an anytime multi-purpose model for identifying Valence, Arousal and Liking. Second, we utilize LSM for Continuous Authentication from mobile devices and we show the benefits of LSM for such purposes. Third, we explore the possible deployments of LSM for the Feature Extraction task from raw data in comparison with Deep Belief Networks. Fourth, we introduce an Active Liquid States Selection method to effectively read from LSM and we show how this method can reduce the computational requirement while improving accuracy. Finally, because of the new trends in building hardware-based LSMs that are energy aware, we introduce an Inattentive Neurons Pruning method to rank and prune the uninformative neurons inside the LSM to reduce power consumption and computational requirements. This method have shown to be able to provide better accuracies in most of benchmarks, while reducing the number of neurons inside the LSM by up to 50%.

CONTENTS

ACKNOWLEDGMENTS	V
ABSTRACT	VI
ILLUSTRATIONS	XI
TABLES	XIII
Chapter	
I. INTRODUCTION.....	1
II. PRIMER ON SPIKING NEURAL NETWORK	5
A. SNN ARCHITECTURE	5
B. SPIKING NEURON MODELS.....	7
1. Hodgkin-Huxley Model.....	8
2. Integrate-and-fire models (I&F)	9
3. Leaky Integrate-and-fire Model (LIF)	10
4. Izhikevich Model.....	11
5. Thorpe’s model	11
6. Conductance-based Model (CbNeuron).....	13
C. INFORMATION CODING	14
D. SYNAPTIC TIME DEPENDENT PLASTICITY (STDP)	16
E. LEARNING IN SNN	18
III. PRIMER ON LIQUID STATE MACHINE	23
A. INTRODUCTION.....	23
B. LSM VS. TURING MACHINE- ORIENTED ML APPROACHES	24
C. LSM VS. DEEP LEARNING	25
D. HANDLING TIME IN LSM	27
E. LSM ARCHITECTURE	29
F. THEORY AND MATHEMATICAL FORMULATION BEHIND LSM	30
G. INPUT INTO SPIKES	32
1. Input to spikes conversion	33
a. Static Input	33
b. Timeseries Input.....	33
IV. LITERATURE REVIEW ON LQIUD STATE MACHINE.....	36
A. RELATED WORK ON APPLICATIONS	36
1. Image Recognition	36

2. Highly Variable Data Streams	36
3. Speech Recognition	37
4. Music Information Retrieval (MIR).....	39
5. Facial Expression Recognition	40
6. Robot’s Arm motion Prediction.....	42
7. Real time imitation learning.....	43
8. Movement Prediction from Videos.....	44
9. EEG Classification.....	45
10. Stochastic Behavior Modeling.....	46
B. LSM IMPROVEMENTS AND MODIFICATION	47
1. Readout	47
2. Structure and Processing Units	48
3. Conveying Information in the Liquid	51
V. LTTERATURE REVIEW ON EMOTION RECOGNITION FROM EEG SINGALS	54
A. INTRODUCTION ABOUT EMOTION IN HUMANS	54
B. AFFECTIVE STATE DEFINITION	55
C. DATASET FOR EMOTION RECOGNITION	57
D. LITERATURE REVIEW	58
1. Works that relies on different datasets	58
2. Works used DEAP dataset	62
VI . CASE STUDY FOR DEPLOYING LIQUID STATE MACHINE FOR FEATURE EXTRACTION FROM RAW DATA	72
A. LSM-BASED FEATURE EXTRACTION DESCRIPTION.....	73
1. LSM Configuration for Feature Extraction.....	73
2. DBN Configuration for Feature Extraction	76
B. RESULTS AND COMPARISON FOR FEATURE EXTRACTION USING LSM AND DBN	77
C. DISCUSSION.....	78
VII. LIQUID STATE MACHINE FOR EMOTION RECOGNITION FROM EEG.....	80
A. EXPERIMENTAL PROCEDURE FOR LSM-BASED EMOTION RECOGNITION MODEL	80
1. Experiment 1: Direct input to the LSM.....	83
a. Subject/ Video Independent	84
b. Subject Dependent	86
c. Video Dependent.....	87
d. Isolated-Subject.....	88
2. Results Comparison with other Machine Learning Approaches	89
a. IS scenario.....	89
b. LOSO scenario.....	90

c. LOVO.....	90
3. Results Comparison using other Spiking Network Architecture	90
4. Results Analysis and Discussion	92
5. Experiment 2: Spike Encoding	94
a. User/Video Independent	95
b. User Dependent.....	95
c. Video Dependent.....	96
d. Isolated-Subject	97
6. Discussion and Analysis	97
7. Experiment 3: Emotion Recognition Using Different Number of Channels	97
a. User/Video Independent	98
b. User Dependent.....	98
c. Video Dependent.....	99
d. Discussion and Analysis	99
B. CONCLUSION	100

VIII. A LIQUID STATE MACHINE -BASED FRAMEWROK FOR CONTINUOUS AUTHETICATION IN SMARTPHONES 101

A. MOBILE CONTINUOUS AUTHENTICATION	101
B. RELATED WORK.....	103
C. LSM-BASED CONTINUOUS AUTHENTICATION FRAMEWORK.....	107
1. Mobile Sensor Data.....	108
2. LSM-Based Framework.....	109
D. FEATURE SELECTION	112
1. Keystroke features	112
E. EXPERIMENTAL PROCEDURE	112
1. Application side configuration.....	112
2. LSM side configuration	115
F. ANOMALY DETECTION METHODS	116
1. Euclidean Detector.....	117
2. Normalized Minimum Distance Classifier	117
3. Manhattan method.....	117
4. Filtered Manhattan	118
5. Scaled Manhattan	118
6. Outlier-counting (z-score).....	118
7. One-Class SVM	118
G. LSM READOUT OPTIMIZATION FOR DISTANCE-BASED ANOMALY DETECTION METHOD.....	119
A. RESULTS OF TESTING CONTINUOUS AUTHENTICATION IN SMARTPHONES	119
1. Augmented Password Authentication	120
2. Long-Text Authentication.....	121
3. Gestures/strokes Authentication	121
B. RESULT DISCUSSION	122

IX . EFFIECNT SAMPLING TIME SLECTION FROM LIQUID STATE MACHINE	123
A. INTRODUCTION	123
B. ACTIVE STATES DETECTION METHOD.....	124
C. TESTING	127
D. RESULTS AND DISCUSSION.....	130
E. THE RELATION BETWEEN SAMPLING TIMES RANKS AND THE AFFECTIVE STATES OF THE BRAIN	131
1. Discussion	132
F. CONCLUSION	135
X. PRUNING INATTENTVIE NEURONS IN LIQUID STATE MACHINE.....	137
A. INATTENTIVE NEURONS PRUNING (INP) METHOD.....	138
1. Mathematical Representation.....	139
B. ILLUSTRATION EXAMPLE	142
C. TESTING AND RESULT.....	145
1. Experiment 1	147
2. Experiment 2.....	149
D. DISCUSSION	150
E. CONCLUSION	150
XI. CONCLUSION AND FEATURE WORK.....	151
A. CONCLUSION	151
A. FEATURE WORK	152
XII. BIBLIOGRAHPY.....	154

ILLUSTRATIONS

Figure	Page
Figure 1: Emitting a Spike Illustration.	7
Figure 2: Hodgkin-Huxley neuron model equivalent electrical circuit.	9
Figure 3: IF neuron model equivalent electrical circuit.	10
Figure 4: Different method for information encoding in Spiking Neural Networks.	15
Figure 5: Gaussian Receptive Fields for spike encoding [19].	16
Figure 6: Synaptic Time Dependent Plasticity weight updating [20].	17
Figure 7: Spiking Neural Network architecture [20].	18
Figure 8: SpikeProp pseudo code [21].	22
Figure 9: Restricted Boltzmann Machine.	26
Figure 10: LSM general architecture description.	30
Figure 11: BSA Algorithm pseudo code.	35
Figure 12: Russell's model to represent emotions.	56
Figure 13: LSM for feature extraction.	75
Figure 14: LSM for multi-purpose classification.	75
Figure 15: LSM for anytime feature extraction.	75
Figure 16: DBN for feature extraction architecture.	77
Figure 17: Topology for Experiment 1 for emotion recognition.	82
Figure 18: Main LSM architecture for continuous authentication.	110
Figure 19: LSM-Based Continuous Authentication Framework for Smartphones.	111
Figure 20: A screenshot of the application interface for Augmented Password Authentication.	113
Figure 21: A screenshot of the application interface for Long-text Authentication.	114
Figure 22: A screenshot of the application interface for Strokes Authentication.	115
Figure 23: Liquid state concatenation for improving distance-based anomaly detection. .	119
Figure 24: Active State Selection Method testing on Valence using Decision Trees.	128
Figure 25: Active State Selection Method testing on Valence using Linear Regression. ..	128
Figure 26: Active State Selection Method testing on Arousal using Decision Trees.	129
Figure 27: Active State Selection Method testing on Arousal using Linear Regression.	129
Figure 28: Active State Selection Method testing on Liking using Decision Trees.	130
Figure 29: Active State Selection Method testing on Valence using Linear Regression. ..	130
Figure 30: Valence Sampling Times Ranking for Users 1-8.	132
Figure 31: Arousal Sampling Times Ranking for Users 1-8.	132
Figure 32: The Sampling times ranking for one second from Subject one (Valence).	134
Figure 33: The Sampling times ranking for one second from Subject one (Arousal).	135
Figure 34: A Comparison for valence and arousal Courses from Subject 1.	135
Figure 35: Isolated Islands Illustration Example.	143
Figure 36: Binding Probabilities to Other Neurons from Neurons n1 and n6.	144
Figure 37: Binding Probabilities from Other Neurons to Neurons n1 and n6.	145

Figure 38: INP method testing on Augmented Password authentication using the Euclidean Detector anomaly detector.....	147
Figure 39: INP method testing on Augmented Password authentication using the Manhattan method anomaly detector.....	147
Figure 40: INP method testing on Augmented Password authentication using Filtered Manhattan anomaly detector.....	148
Figure 41: INP method testing on Augmented Password authentication using Scaled Manhattan anomaly detector.....	148
Figure 42: INP method testing on Augmented Password authentication using Outlier-Counting anomaly detector.....	149

TABLES

Table	Page
Table 1: DEAP dataset summarization.....	58
Table 2: Related work on Emotion Recognition summary 1.....	70
Table 3: Related work on Emotion Recognition summary 2.....	71
Table 4: LSM configurations for feature extraction.....	74
Table 5: Result of testing 343 extracted features using LSM and DBN.....	78
Table 6: LSM configuration for emotion recognition.....	84
Table 7: Testing results for different readouts.....	85
Table 8: Subject/Video independent scenario results using Decision Tress.....	86
Table 9: Subject/Video independent scenario results by using Linear Regression.....	86
Table 10: Subject dependent scenario results by using Decision Tress.....	87
Table 11: Subject dependent scenario results by using Linear Regression.....	87
Table 12: Video dependent scenario results by using Decision Tress.....	88
Table 13: Video dependent scenario results by using Linear Regression.....	88
Table 14: Isolated-Subject scenario results by using decision trees and linear regression...89	89
Table 15: Results Comparison with other Machine Learning Approaches for IS Scenario.89	89
Table 16: Results Comparison with other Machine Learning Approaches for LOSO Scenario.....	90
Table 17: Results Comparison with other Machine Learning Approaches for LOVO Scenario.....	90
Table 18: Results of Testing NeuCube on subject 1 from DEAP dataset.....	91
Table 19: Testing for non-linearity output from LSM (experiment 1).....	94
Table 20: Subject/Video Independent results for experiment 2.....	95
Table 21: LOSO results for experiment 2.....	95
Table 22: LOVO results for experiment 2.....	96
Table 23: Isolated-Subject scenario results by using decision trees (experiment 2).....	97
Table 24: Subject/Video Independent results for experiment 4.....	98
Table 25: LOSO results for experiment 3.....	98
Table 26: LOVO results for experiment 3.....	99
Table 27: Input description for LSM-based continuous authentication.....	110
Table 28: LSM 1 configuration.....	115
Table 29: LSM 2 configuration.....	116
Table 30: Augmented Password authentication (One vs. One scenario).....	120
Table 31: Augmented Password authentication (One vs. All scenario).....	120
Table 32: Long-Text authentication (One vs. One scenario).....	121
Table 33: Long-Text authentication (One vs. All scenario).....	121
Table 34: Gestures/Strokes authentication (One vs. One scenario).....	121
Table 35: Gestures/Strokes authentication (One vs. All scenario).....	122
Table 36: Clustering results for the outcome of the selection method.....	133
Table 37: INP testing results for experiment 2.....	149

Table 38: Neurons reduction rate for experiment 2.....	150
Table 1: DEAP dataset summarization.....	58
Table 2: Related work on Emotion Recognition summary 1.....	70
Table 3: Related work on Emotion Recognition summary 2.....	71
Table 4: LSM configurations for feature extraction.....	74
Table 5: Result of testing 343 extracted features using LSM and DBN.....	78
Table 6: LSM configuration for emotion recognition.....	84
Table 7: Testing results for different readouts.....	85
Table 8: Subject/Video independent scenario results using Decision Tress.....	86
Table 9: Subject/Video independent scenario results by using Linear Regression.....	86
Table 10: Subject dependent scenario results by using Decision Tress.....	87
Table 11: Subject dependent scenario results by using Linear Regression.....	87
Table 12: Video dependent scenario results by using Decision Tress.....	88
Table 13: Video dependent scenario results by using Linear Regression.....	88
Table 14: Isolated-Subject scenario results by using decision trees and linear regression...89	
Table 15: Results Comparison with other Machine Learning Approaches for IS Scenario.89	
Table 16: Results Comparison with other Machine Learning Approaches for LOSO Scenario.....	90
Table 17: Results Comparison with other Machine Learning Approaches for LOVO Scenario.....	90
Table 18: Results of Testing NeuCube on subject 1 from DEAP dataset.....	91
Table 19: Testing for non-linearity output from LSM (experiment 1).....	94
Table 20: Subject/Video Independent results for experiment 2.....	95
Table 21: LOSO results for experiment 2.....	95
Table 22: LOVO results for experiment 2.....	96
Table 23: Isolated-Subject scenario results by using decision trees (experiment 2).....	97
Table 24: Subject/Video Independent results for experiment 4.....	98
Table 25: LOSO results for experiment 3.....	98
Table 26: LOVO results for experiment 3.....	99
Table 27: Input description for LSM-based continuous authentication.....	110
Table 28: LSM 1 configuration.....	115
Table 29: LSM 2 configuration.....	116
Table 30: Augmented Password authentication (One vs. One scenario).....	120
Table 31: Augmented Password authentication (One vs. All scenario).....	120
Table 32: Long-Text authentication (One vs. One scenario).....	121
Table 33: Long-Text authentication (One vs. All scenario).....	121
Table 34: Gestures/Strokes authentication (One vs. One scenario).....	121
Table 35: Gestures/Strokes authentication (One vs. All scenario).....	122
Table 36: Clustering results for the outcome of the selection method.....	133
Table 37: INP testing results for experiment 2.....	149
Table 38: Neurons reduction rate for experiment 2.....	150

CHAPTER I

INTRODUCTION

Spiking Neural Networks [1-3] (SNNs) are a new promising approach for machine intelligence because they increase the ratio of biological realism, and hence the ability to capture complex data patterns. SNNs belong to the third generation of the Artificial Neural Networks (ANNs). In contrast to the previous generations of ANNs, SNNs are able to deal with the spatial-temporal information. In addition, SNNs differ from the second generation in using the spike timing in the learning phase. They have been used in many applications such as character recognition [4], image clustering [5], human behavior recognition [6], breast cancer classification [7], human localization in sensor networks [8] and a detector for IDSs [9]. However, SNNs are not as popular as the other methods of Machine Intelligence (MI). The main reason behind this is their high computational cost.

In this work, we aim to study and utilize SNNs in pattern recognition for dynamical systems.

By “dynamical systems”, we refer to systems that have an input of stream of data such as video streams, audio streams, audio-video streams, i.e. continuous signals. The work does not rely on training SNNs to adjust the learning model, but it uses SNNs as an operating model to build a Reservoir Computing (RC) model. RC is a new trend in machine

learning domain that solves the issues of training Recurrent Neural Networks (RNNs) and SNNs by training only a specific part of the network, the readout function.

Targeting RC is a reasonable choice due to many factors. First of all, for a model to be able to process a dynamical environment, it should have a memory. Whether it is a short term memory (STM) or long term memory (LTM), this memory cannot be achieved unless we have cyclic connections inside the model, and this lead to the Recurrent Neural Networks (RNNs) [10-12]. RNNs are second and third generation of ANN. The simpler form of RNNs, which depends on the second generation ANN, has yet to find a successful training approach that is robust to the vanishing and exploding gradient problems RNNs suffer from [13]. Moreover the gradual changes in the network parameters modify the dynamic of the networks drastically making the gradient information ill and reaching the convergence intractable. In addition, the computation of updating one parameter is expensive and heavily depends on mathematics [14].

Thus, training SNNs is a challenging problem: most of the error-functions in popular training algorithms, such as Backpropagation, handle time-continuous and real values, while SNNs propagate information as spikes [15]. In addition, the neuron model differs greatly from one model to other models, and available works on training SNNs are not general for all models. Thus, there is no general algorithm to train SNNs in order to build upon it a solid learning model.

The RC [14, 16, 17] appeared as a solution that alleviates and solves the challenges of training SNNs and RNNs, where training occurs on the connection found between the

network and the output. The weights and parameters of the network are randomly chosen under specific constraints. The RC has two main approaches according to the generation of the network; the Echo State Network (ESN) [16] which belongs to the second generation of ANN, and the Liquid State Machine (LSM) [17] which belongs to third generation of ANN.

We will focus on LSM since it incorporates the SNNs which advance its capabilities to handle pattern recognition in complex systems. Moreover, it includes unsupervised learning inside the network by using a time Hebbian rule learning, the Synaptic Time Dependent Plasticity (STDP) [18]. STDP allows LSM to refine the internal network, and thus capture the temporal patterns inside the network effectively.

This thesis is organized as follows; in chapter II, we survey the SNNs and neuron models, and we shed light on the available training algorithms in SNNs. Chapter III discusses LSM including its principles, architecture and mathematical representation. Chapter IV provides an extensive literature review on LSM including its applications and variants. In chapter V, we introduce the first application for this thesis and provide a literature review about emotion recognition from EEG by using LSM. Chapter VI discusses the benefits of using LSM for feature extraction from raw EEG data, where we compare its capabilities with Deep Belief Networks for the same purpose. In chapter VII, we provide an LSM implementation for emotion recognition and we draw some conclusion about human emotions. Chapter VIII introduces the second application for this thesis, where we use LSM for multi-stage continuous authentication in smartphones. Chapter IX is concerned with improving sampling time selection from LSM by introducing active states selection

method. Chapter X introduces a method to prune the LSM to reduce the computational overheads for hardware-based LSM implementations. Finally, chapter XI provides the conclusion and future work for this work.

\

CHAPTER II

PRIMER ON SPIKING NEURAL NETWORK

This chapter provides an extensive information and discussion about SNNs including their architecture, neuron models and learning algorithms. We also, review the difficulties found in training SNNs and justify our focus on Reservoir Computing (RC), which is considered to help in deploying SNNs for complex pattern recognition.

A. SNN Architecture

Few scientists and researchers in AI aim to build intelligent systems inspired by the human's brain. The best examples of these systems are ANNs. However, the earlier generations of ANNs namely, the first one was not biologically plausible, even though it captured some of the human's neurons. The second generation of ANNs has been considered more biologically plausible: it uses a continuous activation function for the neurons but discarded the temporal dimension data during learning procedure. On the other hand, SNNs have been introduced to make artificial networks able to deal with complex pattern recognition. To understand how SNNs differ from ANNs, we first have to understand the actual biological human's neuron architecture. These neurons use spikes to

transmit and learn the spectro and/or spatio-temporal data (SSTD). The human's neurons encode SSTD using the location of synapses for the spatial data and the spiking time activities for the temporal data. The model of the biological neuron is shown in the figure below:

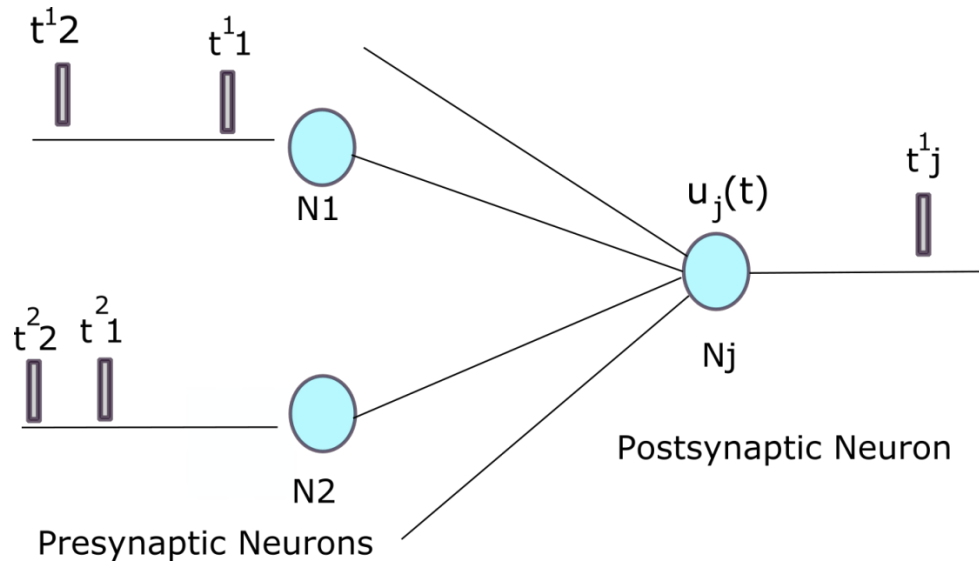


Figure 1: Presynaptic and Postsynaptic neurons Description.

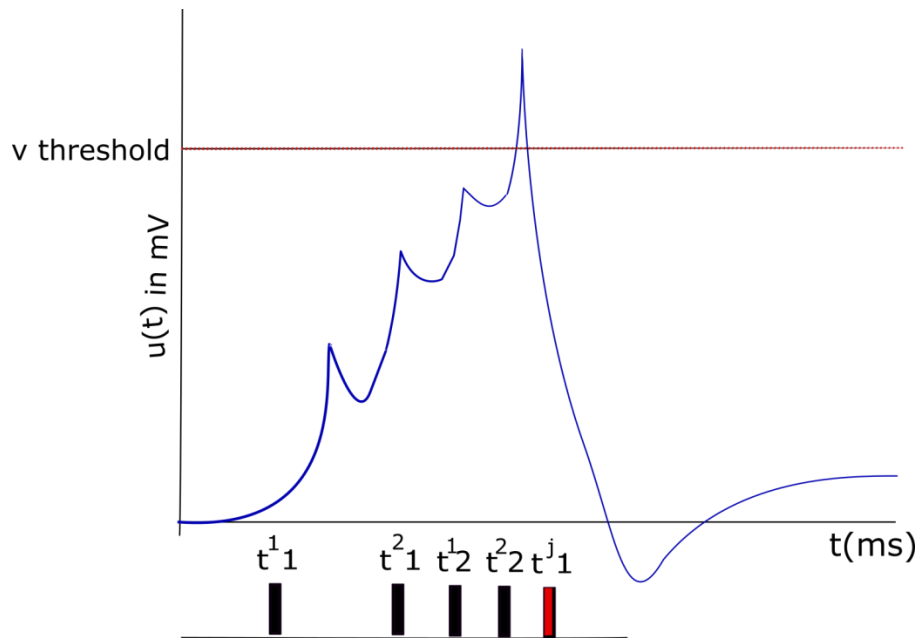


Figure 1: Emitting a Spike Illustration.

Neuron N_j , the presynaptic neurons, receives spikes from the presynaptic neurons N_1 and N_2 . These neurons generate the post-synaptic potentials (PSPs) which can be either excitatory PSPs (EPSPs) or Inhibitory PSPs (IPSPs). The neuron N_j generates a spike whenever the EPSP reaches a threshold value V . The t_i^n represent the time stamp for spike. In the next section, we will study the several neuron models that are used in the literature review.

B. Spiking Neuron Models

Several models of the neuron were proposed and studied. The models mentioned in this work are considered as the most common ones:

1. Hodgkin-Huxley Model

This model, shown in Figure 3, is the origin of all others. Hodgkin and Huxley modeled the electro-chemical information of the natural neurons by studying the giant axon of a squid. The resulting model from their study consists of four differential equations. The equations describe the change in electric charge on parts of the neuron's membrane capacitance as a function of the voltage and the current.

The parameters used to describe the model are:

C: is the capacitance of the membrane.

g_{Na} , g_K , g_L : the conductance parameters for the different ion channels (sodium Na, potassium L, etc.).

E_{Na} , E_K , E_L : the equilibrium potentials resulting for the different ions.

m , n and h : variables that governed three other differential equations.

$$C \frac{du}{dt} = -g_{Na}m^3h(u - E_{Na}) - g_Kn^4(u - E_K) - g_L(u - E_L) + I(t) \quad (2.1)$$

$$\tau_n \frac{dn}{dt} = -[n - n_0(u)], \quad \tau_m \frac{dm}{dt} = -[m - m_0(u)], \quad \tau_h \frac{dh}{dt} = -[h - h_0(u)] \quad (2.2)$$

Due to the complexity of these equations caused by the nonlinearity and the fourth dimensionality of the data, several simpler forms were proposed for the practical implementations, which are discussed in the coming subsections.

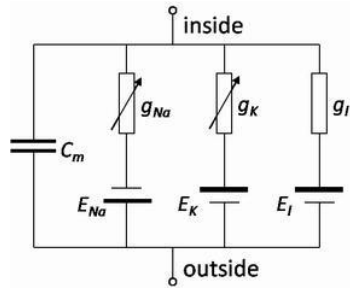


Figure 2: Hodgkin-Huxley neuron model equivalent electrical circuit.

2. Integrate-and-fire models (I&F)

I&F model shown in Figure 4, is derived from the original H&H model, but it neglects the shape of the potential actions. It assumes that all potential actions are uniform, but they differ in the time of occurrence. Due to this simplicity, most of the neuron models are based on this idea. The membrane capacitance and the postsynaptic potential (PSP) of this model are given by the following equations:

$$C \frac{du}{dt} = -\frac{1}{R}(u(t) - u_{rest}) + I(t) \quad (2.3)$$

$$u(t^{(f)}) = \vartheta \quad \text{with } u'(t^{(f)}) > 0 \quad (2.4)$$

Where:

u_{rest} : is the membrane potential of the neuron at the initial state.

ϑ : is the threshold value at which the neuron fires.

$t^{(f)}$: is the spike firing time.

$I(t)$: is the input current caused by the presynaptic potentials.

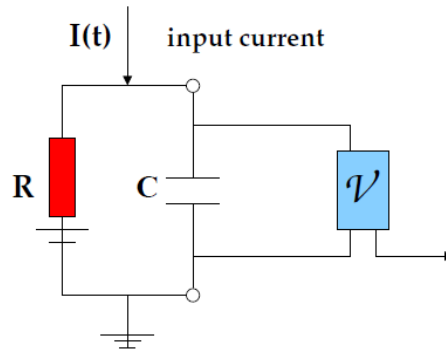


Figure 3: IF neuron model equivalent electrical circuit.

3. Leaky Integrate-and-fire Model (LIF)

This model is similar to the I&F model but with a small difference, the membrane potential of the neuron decays with time if no potentials arrive to the neuron. The work mechanism of this model is the following: when the membrane potential $u(t)$ of the neuron reaches a specific threshold ϑ at time t called the spiking time $t^{(f)}$ and $u(t)$ satisfies $u'(t^{(f)}) > 0$ condition, then the neuron emits a spike immediately. After that, the neuron goes under absolute refractoriness period u_{abs} . The refractoriness period lasts for a specific time d_{abs} and the membrane potential of the neuron during this period is:

$$u(t) = -u_{abs}, \text{ where } u_{abs} \text{ is the refractoriness potential.}$$

When d_{abs} expires, the membrane potential return to the urest case.

The membrane potential is given by the following equation:

$$\tau_m \frac{du}{dt} = u_{rest} - u(t) + RI(t) \quad (2.6)$$

Where:

τ_m : is the time constant of the neuron membrane.

$t^{(f)}$: is the spike firing time.

u_{rest} : is the membrane potential of the neuron at the initial state.

ϑ : is the threshold value at which the neuron fires.

$t^{(f)}$: is the spike firing time.

$I(t)$: is the input current caused by the presynaptic potentials.

4. Izhikevich Model

This model combines between the biological plausibility and the computational efficiency. It uses two differential equations to represent the activities of membrane potential. The model's equations are given below:

$$\frac{du}{dt} = 0.04 u(t)^2 + 5u(t) + 140 - w(t) + I(t) \quad (2.7)$$

$$\frac{dw}{dt} = a(bu(t) - w(t)) \quad (2.8)$$

The after-spiking action is described by the following term:

$$\text{if } u \geq \vartheta \text{ then } u \leftarrow c \text{ and } w \leftarrow w + d$$

5. Thorpe's model

This model is a simple model of integrate-and-fire models, but it takes into the consideration the order of the spikes that reaches the neuron. This gives it powerful capability. In addition, this model uses a simple mathematical representation, which makes

it suitable for many applications. The mathematical model of Thorpe's is given by the following equation:

$$PSP_i = \sum w_{ji} * mod^{order_j} \quad (2.9)$$

Where:

w_{ji} : is the weight or the efficiency of synapsis between neuron j and neuron i.

mod: is a modulation factor $\in [0,1]$.

$order_j$: is the firing order of the presynaptic neuron j where $j \in [1, n - 1]$ and n is the number of the presynaptic neurons that are connected to neuron i.

The weights in this model are updated according to the following equation:

$$\Delta w_{ji} = mod^{order_j} \quad (2.10)$$

This model makes stronger connections between the connected neurons that fire and reach the current neuron earlier. Spiking occurs whenever PSP_i reaches a threshold value PSP_{θ_i} . After the spiking, PSP_i is immediately set to zero.

$$PSP_i = \begin{cases} PSP_i + P_{ji} & \text{when } PSP_i < PSP_{\theta_i} \\ 0 & \text{when } PSP_i \geq PSP_{\theta_i} \end{cases} \quad (2.11)$$

6. Conductance-based Model (CbNeuron)

This model is equivalent to Hodgkin-Huxley model, but with a slightly different change in equation formulation. We emphasize on its model, since the following equation is the one used by the simulator that the thesis uses for SNNs.

$$C_m \frac{V_m}{dt} = -\frac{V_m - E_m}{R_m} - \sum_{c=1}^{N_c} g_c(t)(V_m - E_{rev}^c) + \sum_{s=1}^{N_s} I_s(t) + \sum_{s=1}^{G_s} g_s(t) (V_m - E_{rev}^{(s)}) + I_{inject} \quad (2.12)$$

With

- C_m : the membrane capacity (Farad).
- E_m : the reversal potential of the leak current (Volts).
- R_m : the membrane resistance (Ohm).
- N_c : the total number of channels (active + synaptic).
- $g_c(t)$: the current conductance of channels c (Siemens).
- E_{rev}^c : the reversal potential of channels c (volts).
- N_s : the total number of current supplying synapses.
- $I_s(t)$: the current supplied by synapses s (Ampere).
- G_s : the total number of the conductance based synapses.
- $g_s(t)$: the conductance supplied b synapses s (Siemens).
- $E_{rev}^{(s)}$: the reversal potential of synapses s (Volts).
- I_{inject} : the injected current (Ampere).

C. Information coding

Information coding in neuron has been a strong debate for a long time. The question was, is the information in the neuron encoded as a “rate coding” or as a “spike coding”? Recent studies have shown that information is encoded as a “spike coding” and the “rate coding” is poor in representing the neurons’ ability to rapidly process information. Both coding methods are discussed and compared.

Spike coding uses the time between spikes to encode the information. The recent studies have focused on “rank order coding” of information which extends the “spike coding” (temporal coding) methods. The differences among “rate coding”, “temporal coding” and “rank order coding” are explained by the following example. In this example, seven neurons respond to stimuli. Each neuron can fire at most one time in the next time window T . Suppose that the neurons A, B, C, D, E, and G emit a spike except the neuron F. The binary coding for this example is ‘1111101’. If “rate coding” is used to encode the information, then the maximum amount of the available information is $\log_2(8)$ since we have eight different events that can happen. The information coding capacity for this case is seven. If the temporal coding is used, then the amount of the available information for this case is $7 \cdot \log_2(T)$. The time window T specifies the precision of coding. For example, suppose that the precision is set to 1ms, then T is seven (the number of the spaces between the dotted vertical lines) and hence the amount of the available information becomes $7 \cdot \log_2(7)$. The rank order coding uses the order of emitting of the spikes from the corresponding neurons (the rank column). The amount of the available information for this example is $\log_2(7!)$ since there are $7!$ different combinations of neurons’ spiking orders.

As we can notice, rank order coding achieves a higher information capacity coding than the temporal coding and rate coding.

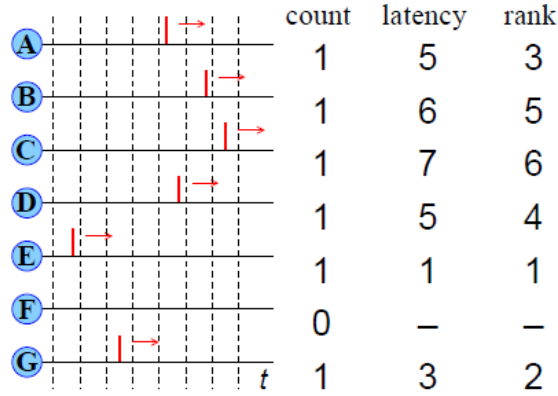


Figure 4: Different method for information encoding in Spiking Neural Networks.

As aforementioned, the rank coding is very efficient and can achieve the highest information coding capacity. The rank coding procedure is the following: first, it starts by converting the input values into a sequence of spikes using the Gaussian receptive fields. The Gaussian receptive fields consist of m receptive fields that are used to represent the input value n into spikes. Assume that n takes values from $[I_{\min}^n, I_{\max}^n]$ range, then the Gaussian receptive field of neuron I is given by its center u_i :

$$u_i = I_{\min}^n + \frac{2i-3}{2} * \frac{I_{\max}^n - I_{\min}^n}{M-2} \quad (2.13)$$

and width σ :

$$\sigma = \frac{1}{\beta} * \frac{I_{\max}^n - I_{\min}^n}{M-2} \quad (2.14)$$

β is a parameter that controls the width of the receptive field with $1 \leq \beta \leq 2$.

The following example shows the rank order coding for $n=0.75$ and $M=5$. The value of β was set to 2 and the range $[I_{\min}^n, I_{\max}^n]$ was set to $[-1.5, +1.5]$. The value of $n=0.75$ is converted to a sequence of spikes using the five neurons. The neuron with ID=3 is ranked first (rank=0) since it is the first neuron in firing, and the neuron with ID=2 is ranked second (rank=1) since it is the second neuron in firing and so on.

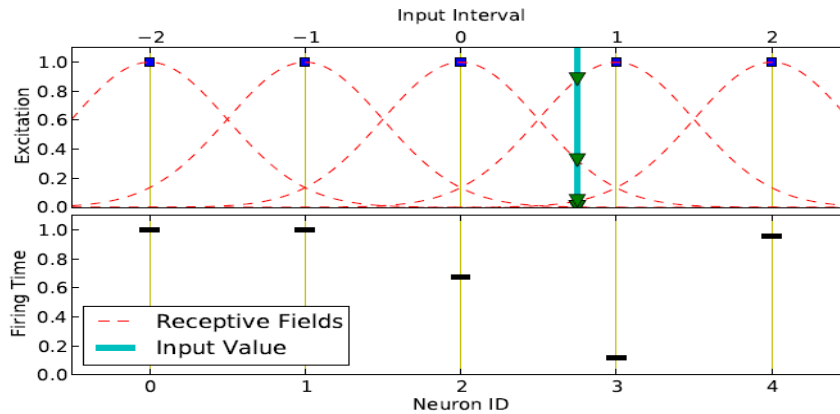


Figure 5: Gaussian Receptive Fields for spike encoding [19].

D. Synaptic Time Dependent Plasticity (STDP)

Like the second generation of ANN, SNNs adjust their weights between the neurons through the learning process. However, SNNs deploy a time-dependent mechanism to do this. SNNs use a variant of Hebbian's rule to emphasize the effect of the spikes timing unlike the common learning methods, which depend on the rate of spiking. The weights updating mechanism is based on the firing time between the presynaptic and the postsynaptic neurons; if the postsynaptic neuron fires directly just after the postsynaptic neuron fires, then the connection between these two neurons is strengthened.

if $\Delta t \geq 0$ then $w_{\text{new}} \leftarrow w_{\text{old}} + \Delta w$, where $\Delta t = t_{\text{post}} - t_{\text{pre}}$

While if the presynaptic neuron fires just after postsynaptic neuron fires, then the connection between two neurons is weakened.

if $\Delta t < 0$ then $w_{\text{new}} \leftarrow w_{\text{old}} - \Delta w$, where $\Delta t = t_{\text{post}} - t_{\text{pre}}$

The remaining case when the firing time of the postsynaptic neuron is apart from the firing time of the presynaptic neuron, then no weights updating occurs. The previous discussion was for the excitatory connection. The inhibitory connection uses a simple process since it does not take into account the firing time between the presynaptic and postsynaptic neurons. The weights between the two neurons are updated according to Hebbian's rule rather than the temporal Hebbian's one. The previous results have biologic backgrounds; however, we will not go through them.

The figure below explains the effects of Δt (the x-axis) on the Δw (the y-axis). The difference between the cases 1, 2 and 3 is that the effect of Δt on weight updating can be symmetric or asymmetric. Case 4 is for the inhibitory connection where the weights updating is independent from the Δt .

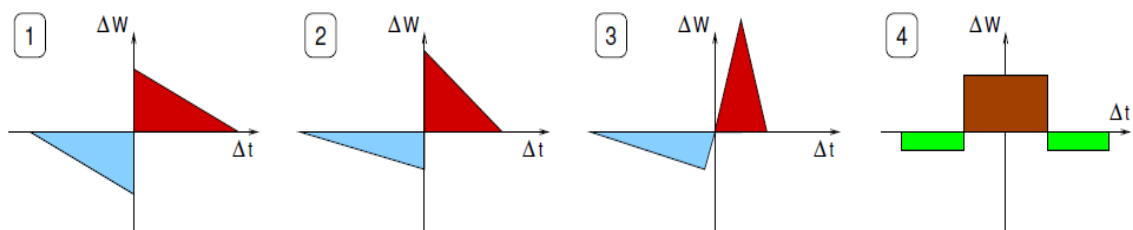


Figure 6: Synaptic Time Dependent Plasticity weight updating [20].

E. Learning in SNN

Before getting into the learning in SNN, we first explain the architecture of SNN and its main components. The SNN structure can be either feed-forward or backward architecture. The architecture depends on the learning method used. The architecture can be a multi-layer network with an output layer that contains several neurons. We assume that the values of input are converted into spikes using one of the available methods such as Gaussian receptive field. The input layer receives the sequences of spikes corresponding to the values of the input. Later, the output is computed by the network and the corresponding output neuron(s) for the input pattern spike(s). The network should be adjusted in order to produce the suitable spiking times at the output. The only way to achieve this goal is by adjusting the weight between neurons as in all other types of the neural networks.

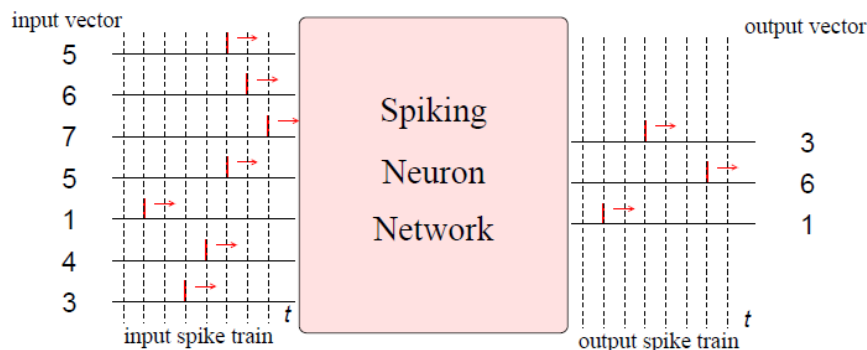


Figure 7: Spiking Neural Network architecture [20].

Number of works tried to design an efficient learning algorithm. The most popular algorithms are SpikeProp and Theta-learning rule. SpikeProp is similar to the Backpropagation algorithm that was designed to adjust the weights in the second generation of the neural networks. Theta learning rules is another learning algorithm that uses Quadratic Integrate and fire (QIF) neuron model. Both of these algorithms are very

sensitive to the parameters of the neuron model and sometimes these algorithms suffer from spike-loss problem. The spike-loss occurs when the neuron does not fire for any patterns and hence it will not be recovered by the gradient method. The other approach for training SNN is using the Evolutionary Strategies which do not suffer from the tuning sensitivity. However, they are very exhaustive and costly.

The figure below describes the SpikeProp algorithm steps for SNN with one input layer I, one hidden layer H and one output layer J. The neurons from each layer are represented by the lowercases i, j or h. The sets Γ_i and Γ^i are the neurons that are immediately preceding and succeeding the neuron i respectively. Each connection between two neurons in the adjacent layers consists of m subconnections. Each subconnection has its own delay factor d_k where $k \in \{1..m\}$ and its own weight w_{ij}^k . The variables t_i , t_j and t_h are the spiking time at each corresponding layers for the neuron i, and $(\hat{t}_i, \hat{t}_j, \hat{t}_h)$ are the actual spiking time at each corresponding layer for the neuron i.

The response function of the neuron i is given by the following equation:

$$y_i^k = \varepsilon(t - t_i - d_k)|_{t_i=\hat{t}_i} \quad (2.15)$$

Where:

$$\varepsilon(t) = \frac{t}{\tau} e^{1-\frac{t}{\tau}} \quad (2.16)$$

and τ : is the membrane constant.

Weights updating from the output layer to the hidden layer is given by the following equation:

$$\Delta w_{ij}^k = -\eta \cdot \delta_i \cdot y_j^k |_{t_i=\hat{t}_i, t_j=\hat{t}_j} \quad (2.17)$$

Where:

$$\delta_j = \left. \frac{\partial E}{\partial t_j} \right|_{t_j=\hat{t}_j} \cdot \left. \frac{\partial t_j}{\partial x_j} \right|_{x_j=\hat{x}_j} \quad (2.18)$$

$$= \frac{T_j - \hat{t}_j}{\sum_{i \in \Gamma_j} \sum_{l=1}^m w_{ij}^l \left. \frac{\partial}{\partial t} (y_i^l) \right|_{t_i=\hat{t}_i, t_j=\hat{t}_j}} \quad (2.19)$$

And updating weights from the hidden layers to the input layer is given by the following equation:

$$\Delta w_{hj}^k = -\eta \cdot \delta_i \cdot y_h^k |_{t_i=\hat{t}_i, t_h=\hat{t}_h} \quad (2.20)$$

Where:

$$\delta_j = \left. \frac{\partial E}{\partial t_i} \right|_{t_i=\hat{t}_i} \cdot \left. \frac{\partial t_i}{\partial x_i} \right|_{x_i=\hat{x}_i} \quad (2.21)$$

$$= \frac{\sum_{j \in \Gamma^i} \sum_{l=1}^m w_{ij}^l \left. \frac{\partial}{\partial t} (y_j^l) \right|_{t_i=\hat{t}_i, t_j=\hat{t}_j}}{\sum_{h \in \Gamma_i} \sum_{l=1}^m w_{hi}^l \left. \frac{\partial}{\partial t} (y_h^l) \right|_{t_i=\hat{t}_i, t_h=\hat{t}_h}} \quad (2.22)$$

The weaknesses of the SpikeProp algorithm can be summarized by the following points [21]:

- The membrane potential of neurons is calculated at fixed time-step intervals.
- There is no method for selecting the initial weights and the thresholds.
- The need for reference neuron that spikes at $t=0$.
- Failure to converge due to the insufficient spike response function.
- The figure below shows the SpikeProp algorithm.

Algorithm 1: The SpikeProp Algorithm

```
setup network;
initialize weights;
repeat
  foreach training pattern do
    // Set input spike times
    foreach input neuron  $p$  do
      | set  $p.spiked, t_p \leftarrow input_p$ ;
    end
    // Run network
     $t \leftarrow 0$ ;
    repeat
       $t \leftarrow t + time\_step$ ;
      for layer  $Q \leftarrow input+1$  to output do
        foreach neuron  $q$  do
          if  $q.spiked$  then continue;
           $x_q \leftarrow 0$ ;
          foreach neuron  $p$  in layer  $(Q-1)$  do
            if not  $p.spiked$  then continue;
            foreach subconnection  $k$  do
              |  $x_q \leftarrow x_q + w_{pq}^k y_p^k(t, t_p)$ ;
            end
          end
          if  $x_q \geq \vartheta$  then
            | set  $q.spiked, t_q \leftarrow t$ ;
          end
        end
      end
    until  $t = max\_steps$ ;
    // Calculate deltas backwards
    for layer  $Q \leftarrow output$  to  $input+1$  do
      foreach neuron  $q$  do
        | calculate  $\delta_q$ ;
      end
    end
    // Update weights
    for layer  $P \leftarrow input$  to  $output-1$  do
      foreach neuron  $p$  do
        foreach neuron  $q$  in layer  $(P+1)$  do
          foreach subconnection  $k$  do
            |  $w_{pq}^k \leftarrow w_{pq}^k - \Delta w_{pq}^k$ ;
          end
        end
      end
    end
    reset network;
  end
  // Calculate total MSE
   $totalMSE \leftarrow 0$ 
  foreach training pattern do
    | run network, calculate  $MSE$ ;
    |  $totalMSE \leftarrow totalMSE + MSE$ 
  end
until  $totalMSE \leq target\_error$ ;
```

Figure 8: SpikeProp pseudo code [21].

CHAPTER III

PRIMER ON LIQUID STATE MACHINE

In this chapter, we focus on the Liquid State Machine (LSM) since it provides an easy approach to cope difficulties in training SNNs and Recurrent Neural Networks (RNNs). RNN is an ANN with cyclic connections inside the network such that the network is able to maintain a memory about previous input. On the other hand, SNN as explained in chapter II is a network that is composed of spiking neurons. We also mentioned in chapter II that adding cyclic connection to SNNs is important to allow for some type of memory inside the network such that they are able to obtain a better understanding about the dependencies between inputs. These connections increase the complexity of training SNNs even more than we think. To overcome these difficulties, LSM trains only the output layer while using network as a dynamical kernel.

A. Introduction

LSM is a randomly and sparsely connected network of spiking neurons. It was introduced by Maass in 2002 [22, 23] to model the cortical microcircuits computations in the human brain. Unlike Turing machine-oriented approaches such as ANNs, SVM, KNN, etc., LSM is a dynamical system modeling approach i.e., the input to the system is a time-varying stream and the output are a congruent high dimensional time-varying output with the input. That is, LSM is capable to handle the arduous problems of time-varying prediction, pattern recognition in dynamic regimes and non-linear system recognition.

Moreover, LSM provides an adaptive scheme to learn from a limited number of samples and hence positions LSM as a superior ML approach as can be explained later.

LSM and Echo State Networks (ESN) [24] form a new trend in ML called Reservoir Computing [14, 25]. ESN was introduced relatively at the same time when Maass introduced LSM, but these two approaches were independently developed. ESN depends on the second generation of ANN, where it uses the non-spiking neuron models. In comparison with ESN, a model formed by LSM is able to encode larger information than a model built on ESN concepts, since amount of information that could be encoded by SNNs are larger than information that could be encoded by the second generation of ANN [20].

B. LSM vs. Turing machine- oriented ML approaches

Turing machine, the most common computational paradigm, is represented by a set of finite states that have an initial state, a goal state and a well-governed regime of transition between states. This definition of Turing machine shows how strict are the models that are built upon the Turing machine concept; the inputs should be available and known in ahead of time; the output is definite and the effect of the current output on the next input is absent. In contrast, systems in real life are more complex and sophisticated than this structure, for example, acquiring the knowledge in humans does not depend only on the current arriving information, but more generally depends on a previous knowledge and a previous memory including the long term memory and the short term memory.

To overcome the limitation of Turing-based models, the LSM incorporates the information not only from previous outputs, but also from the previous short memory and

hence LSM tends to be a more realistic representation of learning in humans. LSM plays a role of a dynamical kernel that maps the input into a high-dimensional separable output, which means that we can easily identify the corresponding inputs. Moreover, since the LSM does not impose restrictions on the input, then LSM is suitable for data stream input applications i.e., LSM is an intrinsic pattern recognition approach for data stream applications.

C. LSM vs. Deep Learning

Deep Learning vs. LSM

Deep Learning (DL) [26] came out as a solution for the problems that faced the shallow models of ANN such as vanishing gradient. These problems were solved by Hinton when he introduced a fast method to train DL using Deep Belief Networks (DBNs). Our concern is to understand the differences between DBN and LSM in training and functionality.

For DBN, training process is divided into two phases: first, the pre-training, which works in hierarchy to transform the input into abstracted representation using nonlinear transformations. DBN functionally is divided into Restricted Boltzmann Machines (RBMs). Each RBM consists of two layers; one is called the visible layer and the other is called the hidden layer. The idea is to represent input at the visible layer very well at the hidden layer. The hidden layer later becomes as a visible layer for the next stacked RBM. This process is repeated until reaching the last hidden layer in DBN, which is used as an output from DBN. It is important to say that this process is an unsupervised learning process, which means

that DBN learns features from the corresponding pattern at input without knowing the label of input. The second phase of training performs the fine-tuning for weights inside the network, where it is performed using the backpropagation algorithm. The second phase is a supervised process, where labels should be presented to training algorithm.

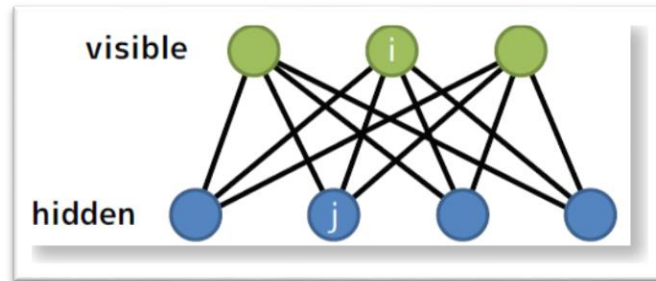


Figure 9: Restricted Boltzmann Machine.

Similarly, LSM has two learning phases to learn from input. In the first phase, LSM learns in unsupervised manner the activation patterns that correspond to input. This is done using STDP algorithms (Chapter II, section D). The second phase is training the readout function, which is done in a supervised learning process, where the label of input should be provided to the readout function.

However, DBN and LSM differ in the way of understanding the input. In DBN, the input is fed as a one static vector, which means that RBM in DBN does not know exactly the spatio/spectro temporal relationship within an input, i.e., DBN works by understanding the underlying structure layer by layer from input, and this understanding gets stronger as we go higher in the network. For example, DBN learns first the edges in pictures and then it begins to identify objects in upper layers. In contrast, learning in LSM using STDP understands the spatio/spectro temporal relationship in input, since learning relies on time and activation patterns within input.

In addition, LSM and DBN differ in the connectivity constraints between different neurons/units. LSM provides a flexible schema for connecting the network, where cyclic connections in the network are allowed. In contrast, connections in DBN are acyclic; connections are allowed from in one direction from visible layer to hidden layer in each RBM without any connections between units within a same layer.

Moreover, DBN has no internal memory to understand dependencies within same patterns, i.e., the time concept is absent in training DBN. On the contrary, the memory in LSM is achieved by the cyclic connections within the network, and time is the major component in training LSM.

From the perspective of reading from the model, LSM and DBN have different mechanisms for extracting the output of the model. In DBN, the far end layer is used as the output of the model, which means that we must wait until all previous layers are trained in order to obtain the final output of the model. Different from DBN, reading from LSM is achieved by reading the responses from the network to input, i.e., the output is LSM is the state of the network at different time steps.

In addition, the main purpose of LSM is to transform the input into a high dimensional output, which is the opposite function of DBN, i.e., DBN transforms the long input vector into a low dimensional output.

D. Handling Time in LSM

Time is a key factor in training and reading from LSM. LSM is a natural time handling model, where the time concept is present in training model using STDP, and

reading from LSM. In training, STDP uses spiking time information for presynaptic postsynaptic neurons in order to adjust weight of synapse to server information propagation within LSM, i.e., strengthening and weakening synapses such that we obtain a resilient regime inside the liquid. For example, LSM learns the underlying activation pattern over time from EEG signals such that a same flow of EEG delivers the same responses in the liquid. More importantly, a same flow of input should produce the same activation paths inside the liquid, where these paths are read and used as fingerprints for corresponding input.

On the other hand, time is present in reading from LSM, where the output is the sampled liquid states from LSM. This way of handling time ensures that the output reflexes the temporal integration of previous input flow along time. Although LSM has a memory, but this memory is a short term memory, and depends on the complexity of cyclic connections that live in LSM. Hence, reading from LSM must ensure that it is able to capture the dependencies within flow of input at the right moment, i.e., when to sample from LSM. In addition, handling time in LSM is an advanced paradigm of other techniques'. In such techniques, a signal is divided into segments and time is handled by sliding a window over this signal. However, windowing a signal does not sincerely capture the dependences between previous values of the same time series. Moreover, windowing cannot capture the dependencies from different time series, when input is composed of a different time series inputs. For example, when a model is supposed to handle EEG, which is composed of a number of channels. Thus, LSM provides a natural approach to handle time, not only from a single time series input, but from different times series inputs, where

it provides a dynamical segmenting and windowing approach from a combination of times series inputs.

Above all, LSM provides a relaxation method for time by projection input into high dimensional output. That is, transforming the input flow allows LSM to relax time into activation patterns of neurons, which are later used as an output for the readout function. In other words, the liquid forms a dynamical kernel for the input, where different neurons are used as different support vectors over time, in the contrary to SVM kernel concepts. Dynamically and relaxation permit LSM to deal with time effectively.

E. LSM Architecture

The LSM, in general, consists of three parts; the input, the liquid and the readout. The input(s) is/are data stream signal(s) that can be either continuous or discrete. The liquid consists of a number of spiking neurons (described in CHAPTER II), where they function together as a dynamical kernel. The readout is a memoryless function that receives the output, the liquid states, from the liquid to perform a recognition task. The Figure 10 bellow describes the general architecture of the LSM, where $I(s)$ is the input to the liquid and L^M is the liquid represented by a complex dynamical system equations in the high-dimensional space. $y(t)$ is the readout output after applying some filter operations to resolve the state of the liquid during specific periods.

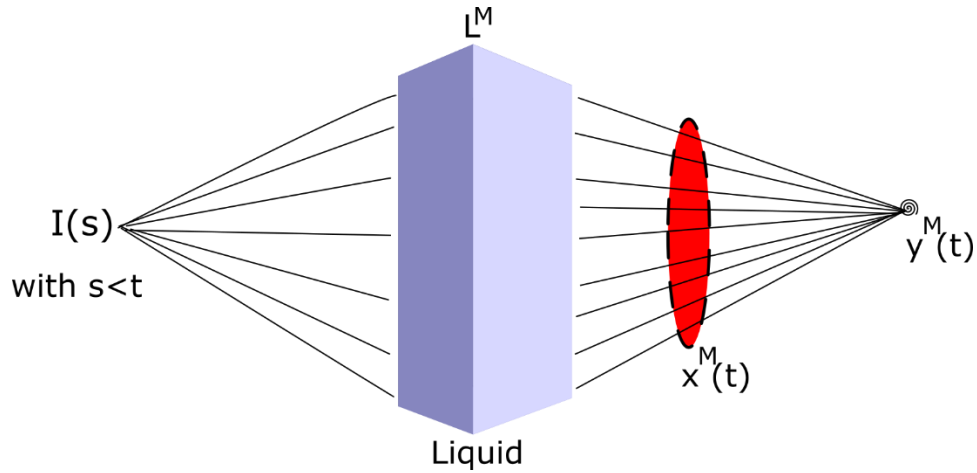


Figure 10: LSM general architecture description.

The LSM as a model can be described by the following simplified equation:

Let $I(s)$ be the input described until the time s with $s < t$, then:

$$\text{the liquid states: } x^M(t) = L^M(I(s)) \quad (3.1)$$

$$\text{the readout output: } y(t) = f^M(x^M(t)) \quad (3.2)$$

F. Theory and Mathematical formulation behind LSM

Let M be a computation machine that can deliver an online computations by performing a function F on some inputs, I . This function, F , is encoded as:

$$i: \mathbb{R} \rightarrow \mathbb{R}^n \quad (3.3)$$

We call the function F as a filter to describe its functionality since the term filter is more general than the term function. This filter, F , has a restrict regime that it is an input driven filter i.e., the output of the filter does not depend on any internal clock of the machine, but the input. Such filters are called time-invariant filters and satisfy that any

temporal shift of the input by some amount of time t_0 causes the output to shift by the same amount t_0 . We write this mathematically as follows:

$$Fi^{t_0}(t) = (Fi)(t + t_0), \text{ for all } t \text{ where } t_0 \in \mathbb{R} \quad (3.4)$$

Where

$$i^{t_0}(t) := i(t + t_0) \quad (3.5)$$

This indicates that to identify the characteristics of the filter, we need to observe the output at time $t=0$ while the input varies over the time. That is, the time invariant filter F is uniquely identified by:

$$y(0) = (Fi)(0) \quad (3.6)$$

At this level, we can replace the filter by a simpler mathematical representation or approximation. It is important to mention that the analog computations are greatly affected by the noise in the computation machine and we refer to this by the fading memory, which requires that for any input function $i(\cdot)$, the output $(Fi)(0)$ can be approximated by the outputs $(Fv)(0)$ for any other input functions that approximates i on a sufficiently long time interval $[-T, 0]$ in the past. We mathematically write the fading memory property as follows:

$F: I \rightarrow \mathbb{R}^N$ has a fading memory, if for every $i \in I$ and every $\varepsilon > 0$, there exist $\delta > 0$ and $T > 0$ such that $|(Fi)(0) - (Fv)(0)| < \varepsilon$ for all $v \in I$ with $\|i(t) - v(t)\| < \delta$ for all $t \in [-T, 0]$. This means that to say a filter F has a fading memory: it is enough for

its current output $(Fi)(0)$ to depend on the most significant bits of its input $i(\cdot)$ in some finite time interval $[-T, 0]$.

The universe of time-invariant fading memory filter is very large; it contains all the filters F that are characterized by Volterra series (a finite or infinite sum of integrals). The filter F needs to have, in addition to the fading memory property, the pointwise separation. That is, for two inputs $i(\cdot)$ and $v(\cdot)$ with $i(s) \neq v(s)$ and for some $s \leq t$:

$$Fi(t) \neq Fv(t) \quad (3.7)$$

Besides, the readout function, R , must have a universal approximation property i.e., any continuous function can be uniformly approximated by functions from R .

G. Input into spikes

Encoding input into spikes or into reservoir-understandable input is crucial step since the quality of conversion affects the performance of the reservoir. More importantly, each unique input should be able to generate unique responses in a reservoir. In addition, relatively close inputs should have relatively same responses. The reservoir with input encoding mechanism must not fall in or generate chaotic behaviors that are not correlated with inputs. This can be seen from the perspective of Butterfly Effect in Chaos Theory, where a small change in input may cause a tremendous change in output, therefor LSM implementation and configuration must ensure not to adopt any Butterfly Effect.

Methods of encoding or converting the input rely on the class of input. More specifically, input might be a time series such as EEG signals, voice signals, stochastic

prices over a period of time, etc. or a static input such as feature vector that specify iris flower type or simply any other examples that have features– label format.

Input might be encoded as a train of spikes or might be injected directly into the reservoir by mean of input neurons that are connected to reservoir’s spiking neurons and influence variables of neurons’ state e.g., an input is injected as current that affects the membrane potential in CbNeuron neuron model.

Interfacing the input with the reservoir raises conditions on the deployed methods; the input must excite the reservoir enough to capture the corresponding responses and must not oversaturate the reservoir such that the responses become uninformative. In the following, we review some of the works that use different methods of input conversion.

1. Input to spikes conversion

There are several methods to convert input into spikes where they utilize different mechanism and impose conditions on input class.

a. Static Input

The common method in such classes of input is to use Gaussian Receptive Fields where each feature in the input is converted into a number of spike trains (see CHAPTER II, section3).

b. Time Series Input

Here, we can identify two algorithms: BSA and HSA. Each channel in the input is converted into a train of spikes and this seems suitable for the reservoir because it mimics

the actual representation of information in biological neurons and allows the reservoir to handle time series input naturally.

- HSA: The idea behind this algorithm is to perform a reverse convolution of the stimulus. To mathematically explain this method, first, let us define some equations to explain the intuition behind HSA and BSA.

A stimulus can be estimated as follows:

$$S_{estim} = (x * h)(t) = \int_{-\infty}^{+\infty} x(t - \tau)h(\tau)d\tau = \sum_{k=1}^N h(t - t_k) \quad (3.8)$$

Where t_k is the neuron firing times, $h(t)$ is a linear filter impulse response and $x(t)$ is the spike train of the neuron. $x(t)$ can be represented by $\sum_{k=1}^N \delta(t - t_k)$.

The equation of S_{estim} turns into the following shape when it is filtered using a discrete Finite Impulse Response (FIR) filter that has M tabs.

$$o(t) = (x * h)(t) = \sum_{k=0}^M x(t - k)h(k) \quad (3.9)$$

HSA tries to reverse the value of $o(t)$ by comparing the shifted value of $h(t + \tau)$ with the matching value $s(t)$ for every time step τ , if the error from comparison is small then it subtracts $h(t + \tau)$ from $s(t)$ and it emits a spike.

- BSA uses the same process, but it utilizes two error metrics:

$$\sum_{k=0}^M abs(s(k + \tau) - h(k)) \quad (3.10)$$

$$\sum_{k=0}^M abs(s(k + \tau)) \quad (3.11)$$

If the first error in (3.10) is smaller than the second error (3.11) minus a certain threshold then it emits a spike and subtracts the filter from the input signal. The threshold in BSA is an experimental value and can be found by grid search for the optimal threshold that increases SNR. It has been shown that BSA is better than HSA since it produces

smother frequency and amplitude characteristics comparing with HSA [27]. BSA has been used in [28] for EEG classification with promising results. The goal was to identify (classify) the type of stimulus from RIKEN EEG dataset. The dataset contains four types of stimuli collected using 64 electrodes: Class 1 – Auditory stimulus; Class 2 – Visual stimulus; Class 3 – Mixed auditory and visual stimuli; Class – No stimulus. After preprocessing the data, 80% of data was used for training and 20% for testing. The parameters of BSA are as the following: the FIR filter has 20 taps and the threshold was chosen to be 0.955.

The pseudo code for BSA is shown below:

```

for i = 1 to size (input)
error1=0
error2=0
for j = 1 to size (filter)
if i+j-1 <= size(input)
error1+= abs (input (i+j-1)-filter (j))
error2+= abs (input (i+j-1))
end if
end for
if error1 <= (error2-threshold)
output (i) =1
for j = 1 to size (filter)
if i+j-1 <= size (input)
input (i+j-1) = input (i+j-1) – filter (j)
end if
end for
else
output (i)=0
end if
end for

```

Figure 11: BSA Algorithm pseudo code.

CHAPTER IV

LITERATURE REVIEW ON LIQUID STATE MACHINE

In this chapter, we survey the related work in LSM such that we cover the applications of LSM and the different suggested improvements on LSM. We divide this chapter into application and improvement levels related works.

A. Related work on Applications

1. Image Recognition

LSM has been successfully used for image recognition: for example in [29], the LSM was used to recognize predefined nine images. The architecture of LSM in this work consists of 100 input neurons that are randomly connected to 25 layers, where each layer has 24 neurons of Hodgkin- Huxley neuron model. ANN was used as a readout function from the 100 output neurons which they are connected to the 25 layers. The work claimed that the result are promising and supports the idea of using LSM for pattern recognition tasks.

2. Highly Variable Data Streams

One of the main powerful advantages of using LSM is the ability to deal effectively with times series problems which was investigated in [30]. This work explores different enhancements on LSM by studying the performance of LSM under different setting of firing thresholds and the synapses properties of the LSM's neurons. These settings are:

static firing thresholds with static synapses (Model A); static firing thresholds with dynamic synapses (Model B); dynamic firing thresholds with static synapses (Model C) and dynamic firing thresholds with dynamic synapses (Model D).

It was shown that Model A has the least sensitivity to weak stimuli but with highest spontaneous respond. Models B, C, and D showed a promising separation results for stimuli.

3. *Speech Recognition*

Speech recognition is one of the direct applications of LSM since it requires a temporal pattern recognition oriented methods. In [31], the work tested LSM on subset of T146 dataset, which is a common dataset for speech recognition. Specifically, [31] used a subset of 500 samples consist of ten utterances of the isolated numbers -from zero to nine, spoken by five different speakers. The samples were divided into 300 samples for training and 200 for testing. The reservoir in this work consists of LIF neurons varying in the number between 200 neurons to 1400 according to different scenarios. LIF neurons are identical with the following parameters: membrane time constant = 30 ms, firing threshold = 15 mV, reset voltage = 13.5 mV, absolute refractoriness for excitatory neurons =3 ms, and absolute refractoriness for inhibitory neurons =2 ms. The connections between neurons inside the reservoir are governed by the following stochastic equation:

$$P(Na, Nb) = C \cdot e^{\frac{-Dist^2(Na, Nb)}{\lambda^2}} \quad (4.1)$$

This equation describes the probability of connecting neuron Na with Nb , where:

- $\text{Dist}(Na, Nb)$: is the Euclidean distance between Na and Nb .
- C : is the connection weight; it depends on the type of Na and Nb ; whether the neuron is excitatory or inhibitory.
- λ : is a parameter that controls the average number of connections and the average distance between Na and Nb .

In this work, three methods of input-to-spikes encoding were tested: Hopfield-Borody method, MFCC method and Lyon Passive Ear method. Among these three methods, Lyon Passive Ear has shown to be the best method with average Word Error Rate (WER) of 2 % and best achieved WER of 0.5%.

In addition, the worked tested the robustness of LSM against different types of noise namely, the speech bubble, the white noise and the car interior noise added from NOISEX dataset. An LSM with 1232 neurons was first trained using pure training samples and then was tested according to three different SNR levels of 10 dB, 20 dB and 30 dB in testing samples. The result has shown that the LSM is robust for the noise and is able to recognize the patterns effectively even in noisy environments.

The same recognition task was tested in [32], but with different configuration for the LSM; the number of neurons inside the reservoir was varied from 100 to 300 neurons of LIF. The results has shown that as the number of neurons inside the reservoir increases, the WER decreases specifically, from 0.05% for 100 neurons to about 0.025% for 300 neurons. Moreover, the work studied the effect of choosing Booj synapse model instead of

the exponential one and it has been shown that Booi synapse model did not improve the WER.

4. Music Information Retrieval (MIR)

Music classification based on the content is another discipline of harnessing LSM in real life applications. In [33], an LSM was tested for two tasks: the recognition of music records from music and non-music records and classifying the music style into classical or ragtime music. The architecture of the LSM in this work consists of an input layer with 12×5 LIF neurons and ten layers of 12×5 LIF neurons. The input to LSM is a set of 60 spike trains that represent the encoded piano music data. The model of LIF neurons in LSM is given by the following parameter: the time membrane time-constant = 30 ms, reset voltage = 0 mV, input resistance = $1M\Omega$, I (inject) = 13.5nA, I (noise) = 1nA, absolute refractoriness for excitatory neurons = 3 ms, and absolute refractoriness for inhibitory neurons = 2 ms.

In the LSM, 80% of neurons were excitatory and the remaining 20% are inhibitory. The connectivity inside the LSM follows the common equation (Small World Connection) with $\lambda = 1.5$ and $C = 0.3, 0.2, 0.4$ and 0.2 for excitatory-excitatory connection, excitatory-inhibitory, inhibitory-excitatory and inhibitory-inhibitory respectively. The input layer was configured with different initialization; $\lambda = 5$ and with exception that it has no inhibitory neurons. The output from the LSM is acquired by mean of readout function; the linear regression.

$$P(Na, Nb) = C \cdot e^{\frac{-Dist^2(Na, Nb)}{\lambda^2}} \quad (4.2)$$

In the first task where the LSM is devised to recognize the music form non-music records, it showed a good capability with testing accuracy around 82%. The second task, where LSM role is to classify music style, asserts the efficiency of the LSM with best-achieved accuracy of 94.08%.

5. Facial Expression Recognition

Facial expression recognition is a common application in pattern recognition and has been tested using different ML approaches such as ANN, KNN, SVM, etc.

The LSM, and due to its successes, has been devised in this context. For example, in [34], the LSM was assessed and evaluated to recognize seven types of emotions namely, happiness, sadness, anger, fear, surprise, disgust and neutral from JAFFE database, which contains 213 images of female facial expression from 10 Japanese women. In this work, two methods were used to represent images: the Gabor representation and binary representation. Two topologies of LSM were utilized to perform the recognition phase depending on the representation method: for the Gabor representation, the LSM consists of an input layer of $1 \times 17 \times 12$ excitatory neurons, $3 \times 17 \times 12$ for the reservoir neurons and an output layer of $1 \times 17 \times 2$; for the binary representation, the LSM consists of an input layer of $1 \times 45 \times 10$ excitatory neurons, $3 \times 45 \times 10$ of reservoir neurons and an output layer of $1 \times 45 \times 10$ neurons.

In both topologies, the readout function was a two-layer perceptron network with 38 neurons in the hidden layer and seven neuron at output layer that represent the seven aforementioned facial expressions.

Under those circumstances, the Gabor and binary representations were tested and they reported average accuracies of 57.6% and 55.7%, respectively. The work claimed that since the binary representation is less computationally demanding than Gabor representation, the further experiments of the work were based on only binary representation. With this in mind, a different number of neuron models namely, integrate-and-fire, resonate-and-fire, Morris-Lecar, Hindmarsh-Rose, FitzHugh-Nagumo and Izhikevitch's models were implemented for LSM to evaluate the performance of the LSM accordingly. The highest average recognition rate was achieved by Morris-Lecar model (58%) and the lowest achieved by Izhikevitch's model (44%).

Furthermore, each neuron model has been shown to outperform other model for specific type of expressions; Morris-Lecar achieved a good rate of correct classification for happiness and fear while integrate and fire model performed better for disgust. In view of the variety of performances for each model according to the type of facial recognition expression, the work tested a merged classifier where the recognition is the majority voting from the six neuron models. It has been shown that the merged classifiers ameliorate the average recognition rate (82.4%) compared with previously achieved average recognition rate (57.6 %).

6. Robot's Arm motion Prediction

Robotic applications are not an exception for the LSM, in fact the LSM provides a potent mastery in action controlling and, more even, prediction in ahead of time, which is very important distinction.

With this intention, [35] proposed an Adaptive LSM (aLSM) where it exploits environment's parameters to adapt the parameters of LSM.

The experimental setup for [35] consisted of two robotic arms each with two degrees-of-freedom; one called the experimenter's arm and other is the robot's arm. The experimenter's arm has an access to a number of objects and, in the same time, the robot's arm has an access for some of those objects. The goal was to see if the robot's arm is able learn from the experimenter's arm to access these objects.

In the foreground, the aLSM in this work is composed of 150 Izhikevitch's neuron model with (60%) excitatory neurons. The number of input and output neurons is equal to the number of object in the environment.

Experiments tested two cases: first when the objects are static i.e., no changes occurred in the position of objects; second when objects are randomly positioned after training the robot's arm i.e., the robot's arm is first trained using the experimenter's arm information, then the robot's arm is tested on accessing the same objects after randomly positioning the objects in the environment.

For the static case without incorporating the adaptive learning, the average successful rate was (75.5%) for the environment with two objects and (42.85%) for four

objects, while these results increased by (30%) and (2%) respectively when using adaptive learning (aLSM). For the random position and without incorporating the adaptive learning, the average successful rate was (69.38%) for two objects and (34.69%) for four objects, while these results increased by (20%) and (19%) when using adaptive learning (aLSM).

7. Real time imitation learning

Some applications in real life require real time computations such controlling a vehicle or robot. For this purpose, an LSM was deployed to perform a real time computing in [36]. The work used a small robot called Khepera, which has a programmed controller, six infrared sensors and two motors. The controller allows the robot to avoid obstacles by turning around them. The purpose from using the LSM is to imitate the behavior of the controller such that it can do the same job under real time working. The work used an LSM with 54 LIF neurons to imitate two types of controllers; one with a linear reactive and the other with non-reactive. The training phase collected data from sensors and motors while the robot is controlled by one of the two types of controllers. After that, the data is put as segments, where each segment has one single occurrence of obstacles to allow the LSM to learn from the data. Two readouts were trained to control the left and right motors according to the recorded data from training. To that end, the LSM is now ready to perform the real time job; controlling the motors according to sensors data. The testing phase is done in real time, where the recorded commands from the controller are compared with the generated output by the LSM (the two readouts output). The results has shown that the LSM was able to imitate the job of the controller with 0.9334 and 0.9338 correlation coefficient for the left and right motors, respectively for the linear reactive controller. In the

non-linear reactive controller, correlation coefficients (0.8130 and 0.8263) were reported for the left and right motors respectively.

8. Movement Prediction from Videos

The capabilities of LSM were tested on real application for prediction purposes and it has been shown that LSM is suitable for such tasks. For example, in [37], an LSM was used to predict the movement of ball from a camera that is attached to a robot participated in RoboCup middle-size robotic scenario. The goal was to predict the position of the ball in ahead of time. The LSM in this work consists of $8 \times 6 \times 3$ LIF neurons with the following parameters: ($C_m=30\text{nF}$, $R_m=1 \text{ M}\Omega$, $V_{\text{thresh}}=15\text{mV}$, $V_{\text{resting}}=0\text{mV}$, $V_{\text{reset}} \sim \text{Uniform} [13.8 \text{ mV} - 14.5\text{mV}]$, $V_{\text{init}} \sim \text{Uniform}[13.5\text{mV} - 14.9\text{mV}]$, $T_{\text{refractoriness}}=3\text{ms}$ (for Excitatory neurons), 2m (for Inhibitory neuron)) and connection parameters as the following ($C= 0.3$ (EE) , 0.4 (EI), 0.2 (IE), 0.1 (II), $\lambda=[0.5 - 5.7]$). The readout is a linear regression function applied on the output layer. The output layer is composed of a number of neurons that are fully connected to LSM using static spiking synapses. The input layer is 8×6 input neurons that receive the activation sequences from robot's sensors. The activation represents the percentage of covered area by the ball to a sensory area of the sensor at each time of evaluation.

During the experiment, 674 video sequences were recorded, which have 50ms as a time step and differ in the length. Thereafter, the video sequences were divided randomly into 85% training and 15% validation. In the testing phase, the LSM was required to predict the position of the ball after 2 time steps (100ms), 4 time steps (200ms) and 6 time steps (300ms). The results were reported in terms of the mean absolute error and the correlation

coefficient for different values for parameter λ (0.5 – 5.7) and Ω (0.1 – 5.7). The parameter λ controls the average of connection inside the reservoir and Ω , the scaling factor, controls the strength of the connection between the neurons. The best achieved result for one time step prediction is when one of the parameters λ or Ω is low, for example ($\lambda=1$ and $\Omega=0.5$), and this was indicated by the low value of the mean absolute error and the high values of the correlation coefficient. The same conclusions were reached for two and four time steps prediction with highest correlation coefficient achieved of 0.7 for two time steps and 0.5 for four time steps prediction. The six time steps prediction has not been shown to be able to provide good results.

9. EEG Classification

BSA with LSM have been used in [28] for EEG classification with promising results. The goal was to identify (classify) the type of stimulus from RIKEN EEG dataset. The dataset contains four types of stimuli collected using 64 electrodes: Class 1 – Auditory stimulus; Class 2 – Visual stimulus; Class 3 – Mixed auditory and visual stimuli; Class – No stimulus. After preprocessing the data, 80% of data was used for training and 20% for testing. In this work, an LSM with 135 neurons was built and tested using four different types of neurons models: default LIF; Noisy Reset (NR) – the rest value of LIF is drawn from random distribution; Step-wise Noisy Threshold (ST) – the threshold value of LIF is drawn from random distribution, but during two consecutive spikes the threshold is fixed; Continuous Threshold (CT) - the threshold value of LIF is drawn from random distribution. The readout was acquired using the Naïve Bayes and MLP, where they reported results were compared with same methods but without using the reservoir. For Naïve Bayes, the

reservoir approach achieved 75% accuracy regardless of the neuron model compared with 66.9% without a reservoir. On the other hand, MLP achieved 75% using ST model compared with 64.87% without a reservoir. The study has shown that the root mean squared error (RMSE) is lowest when using ST model with Naïve Bayes although the accuracies were the same.

10. Stochastic Behavior Modeling

The LSM was used in [38] to design a neural controller that replicates the exploratory behaviors of cockroaches seeking for shelters. Particularly, this work used the same data that was used to introduce the Randomized Algorithm Mimicking Biased Lone Exploration in Roaches (RAMBLER) [39]. The LSM has to learn form information about the walls, shelters relative to the insect along with current velocity and angular velocity to produce probability distributions of the animal's velocity and angular velocity.

The deployed LSM in this work consists of 300 LIF neurons arranged in $5 \times 5 \times 12$ (3D architecture) with $\lambda = 2.1$ and 80% of the neurons are excitatory. The connection between neurons is as follows: $C_{EE} = 0.8$, $C_{EI} = 0.8$, $C_{IE} = 0.7$ and $C_{II} = 0.7$. Synapses between neurons implement short-term plasticity (STP).

The input data to the LSM is composed of a set of information as follows: a modeled wall sensors data that is composed as 12 distance sensors represents LIDAR-like sensors located on the head of the animal and five other sensors that represent the antenna-like sensors; a modeled shelter sensors data that is composed of the distance between the animal and the shelter and nine values that encodes the orientation of the animal relative to

the shelter. The readout function is a two -hidden-layer ANN with 300 sigmoidal neurons in each layer. The input data is sampled at 20 Hz from sensors data and then fed into the LSM. After that, the LSM estimates the probability distribution of the velocity and the angular velocity by using reinforcement learning to force the LSM to produce the same output.

For testing, 20 LSMs were randomly generated and trained accordingly and then the produced output was compared with the probability distributions of the velocity and the angular velocity. The results were reported in terms of the differences between the centroids of the produced probability distribution and the actual ones. The reported differences are 50 ± 130 mm/sec and 5 ± 13 rad/sec for velocity and angular velocity, respectively. Even though the mean of differences is not zero, the produced probability distributions have shown that the LSM tends to produce the actual values, but not to completely converge to them. In addition, the results have shown that the LSM, when compared with RAMBLER, is able to capture the complex behaviors of the insect without explicitly defining them.

B. LSM improvements and modification

We divide this section according to the targeted parts or mechanisms by the related works.

1. Readout

The readout part of LSM is a linear classifier without any memory as stated by Maass [22]. While this became as a principle for LSM, [40] studied the effect of adding a

memory for the readout. More specifically, the work converted the synapses between the LSM and the readout such that the emitted spikes have certain delays. In details, the work used Brody-Hopfield benchmark, which includes 500 samples of spoken digits from “0” to “9”, to identify to the correct labels of the numbers. In this work, two randomly generated LSMs with 135 neurons were used to perform this task after converting the audio into spikes. Here, 300 samples were used for training and the remaining samples, 200, for testing. In the first experiment, the work tested 5 ms, 10ms, 15ms, 20ms, 30ms and 50ms delays for 1, 2, 3, 5, 7 or 10 links of the outgoing synapses of each neuron that has a connection with the readout. The reported results have shown that the LSM performs better as the delay and the number of synapses increases, but the results have not shown to be steady and this led to the second experiment. The authors in the second experiment generated a dataset of 1500 with binary classes where 1000 samples were used for training. The second experiment didn’t proved that the performance improves steadily as the delay and the number of delayed synapses increase; however and generally speaking, the performance could be improved by adding delays to synapses.

2. Structure and Processing Units

In [41], the authors proposed a method to iteratively search for the best LSM that yields highest performance using GA. The work proposed an evaluation function that acts as a metric for the performance of the LSM. More specifically, the metric incorporates the liquid states means and variance in order to examine the separation of the data. The metric is deployed in context of Fisher Discriminant Ratio (FDR) for the separation between

classes at the input. To that end, the evaluation function, FDR, is ready to be optimized by GA to find the best architecture and neural model. The solution that GA needs to find is expressed as a chromosome that encodes three types of information: 1) the parameters of the neurons in the liquid and explicitly, the firing thresholds and the mean of Gaussian noise added to the neurons' output, 2) the architecture of the LSM and it has three options: input neurons are fully connected to the liquid; input neurons are partially connected to the liquid; and time varying input neurons are connected to different regions in the liquid while static ones are connected to all neurons, 3) the size of liquid and the average connections for each neuron in the liquid.

To assess the quality of the chosen metric, four different classification tasks were used and the resulting accuracies are compared with FDR metric. In the first task, an LSM with 125 neurons determines whether the input is over a specific rate specifically, over 5 Hz. The second task uses the LSM to classify two different types of motions by projecting the movement on 9×9 grid of receptive field neurons that are connected to an LSM with 63 neurons. In the third task, the LSM is required to classify three objects namely, a circle, a square and a hexagon. In the last task, the LSM has to decide whether the end point of a moving planar robotic arm is close or not to a given target location. After conducting the four tasks, the work reached a conclusion that FDR metric can be used to evaluate the performance of the LSM because it directly related to the separation property of the LSM and it does not depend on the deployed readout.

At the level of the processing units, the work [42] tested six neuron models namely, Integrate and Fire, Resonate and Fire, FitHugh-Nagumo, Morris-Lecar,

Hindmarch-Rose and Izhikevich model. For the architecture in this work, they used a mammalian visual system inspired architecture that consists of: a Retina or an input which consists of 2×2 neurons that are organized in a square 2D grid; LGN layer consists of $1 \times 8 \times 8$ neurons which is connected to the input; and five layers of $3 \times 8 \times 8$ neurons that are connected to each other. The connection follows the Small World Connection concept with the same configuration as in [31]. The work used two different randomly generated stimuli "S1" and "S2" to test the aforementioned models in order to test the distance between the responses in the liquid. The reported results are as follows: for integrate and fire model, the distance between stimuli was high when $\lambda=9$ and decreases as λ decreases and this means that as the connection density increases the separation property decreases. While for resonate and fire model, the separation property was not affected by the value λ .

In FitzHugh-Nagumo model, the distance rapidly grows for specific duration when $\lambda=9$ and then drops to zero which means that neurons got synchronized, whereas the distance value rapidly grows and reaches a maximum value and then falls down for $\lambda=3$.

Morris-Lecar model showed a good separation property for $\lambda=3$ and relatively high separation property at the beginning of the simulation and then drops down later for $\lambda=9$.

Regarding Hindmarch-Rose model, λ value has no influence on the separation property. Moreover, Izhikevich model showed a poor separation property.

When comparing the six models for different configurations for λ ; the best separation property achieved by Morris-Lecar for $\lambda=3$, while Hindmarch-Rose achieved the best for $\lambda=9$.

Regarding information entropy, the highest achieved entropy achieved by FitHugh-Nagumo, Hindmarsh-Rose. The other model showed unstable behavior and this indicates that these models do not encode information optimally inside the reservoir.

As a conclusion from this work, that FitHugh-Nagumo, Hindmarsh-Rose are suggested to be suitable choices for LSMs with respect to the discussed results.

3. Conveying Information in the Liquid

Information or the spikes transmission inside the liquid is governed by several factors such as the neuron model, the connectivity between neurons and the weights of synapses. The later depends on the initialization of the LSM; however, these weights can be updated through the unsupervised learning; the Hebbian learning. In [43], the work conducted two experiments to study the effects of using Hebbian learning, STDP, for updating the weights of synapses in the liquid. More specifically, the work observed how STDP deals with LSMs in two circumstances: when the LSM has pathological synchrony, which means that most of the neurons in the liquid are firing continuously regardless of the effect of the input due to the feedback loops connection; and when the LSM has over stratification and this occurs when groups of neurons become dead and do not produce spikes regardless of the input.

In the first experiment, a 100 iterations for a random input of 25 spikes over one second was introduced to four settings for LSM; an LSM with pathological synchrony and Hebbian learning, an LSM with pathological synchrony and random weights updating, an LSM with over stratification and Hebbian learning and an LSM with over stratification and random weights updating. The result has shown that the Hebbian learning preserves a

semi steady separation property for pathological synchrony and over stratification cases, while the separation property drops steadily after 10 iterations in the random weights updating for both cases.

Unlike the first experiment that uses a random input, the second one uses TIDGIT dataset which consists of audio files of spoken digits from “0” to “9”. After running the LSM for different permutation of training and testing samples, the authors have indicated that Hebbian learning can improve the separation property. However, the authors attested that the initial weights for the liquid with Hebbian learning affect the separation property and might lead to a low separation property.

In [44], the authors proposed a new metric for evaluation the liquid. This metric is based on Hebbian and reinforcement learning and it does not depend on the accuracies of the results i.e., the metric does not need the readout to be trained to evaluate the liquid. The metric is devised in an algorithm denoted by Separation Driven Synaptic Modification (SDSM), where it searches and updates the synaptic weights to produce a high separable liquid. SDSM takes into account two pieces of information about the status of the liquid and it updates the liquid accordingly. First information is the differentiating between classes and a variance within each class. In this case, SDSM tries to balance between these two pieces of information by strengthening the strong synapses and weakening the weak synapses even more. The second piece of information is the firing behavior of all neuron in the liquid where SDSM aims at thresholding the liquid to have only half of its neurons firing. This achieved by tracing all neurons firing activities and strengthening the excitatory synapses and weakening the inhibitory ones when less than half of the neurons fire. While if more than half of the neurons fire, SDSM reverse its work.

To evaluate SDSM, the authors carried out two experiments; one using artificial problem and the other using speech recognition. In the artificial problem, four input neurons that fire either at slow or fast rates were used. By encoding the slow rate as “0” and the fast one as “1”, the problem then changed to classify five chosen classes according to the state of the inputs. 50 randomly chosen liquid were run for 500 iterations and then the results were compared with an initial LSM. The results have shown that SDSM is able to improve the separation property for the LSM from 0.4 to about 0.55 and the accuracy from around 35% to around 78%. The works also shed the light on the capabilities of SDSM to reduce the number of iterations needed to find a good LSM; an LSM after 500 iterations is not able to compete with LSM and SDSM after 11 iterations.

In the speech recognition task, the author used TIMIT dataset to identify context independent phonemes. Among the 52 phonemes in the dataset, two simpler problems were chosen for further testing: a binary class problem to decide whether the phoneme is a constant or a vowel; and the second problem is to identify one of four distinctly chosen vowels. Again, the SDSM has proved that it can generate better LSMs in terms of the separation property and the accuracy.

The authors also tested performance of using LSM with SDSM for Transferring Learning and it has be shown that LSMs with SDSM have produced better models than the corresponding initial LSMs used in the SDSM.

CHAPTER V

LITERATURE REVIEW ON EMOTION RECOGNITION FROM EEG SIGNALS

This chapter is concerned with providing a solid introduction and literature review about emotion recognition from EEG. The reason behind our choice is to provide a challenging problem for the pattern recognition task, where it requires time series input capable-handling model. LSM seems suitable for such problems because we can examine the benefit of using LSM for processing EEG as time series signals. Moreover, we want to deploy LSM for analyzing and studying emotions in depth by showing how we can use LSM as an anytime classifier without significant changes in the learning model.

The chapter is organized as follows: in section A, we introduce the idea about emotions in humans and we explain in section B the different models used to describe emotions and the affective states of humans. Section C is concerned with introducing the dataset that we will use in this work. In section D, we survey the related work done in the context of emotion recognition from EEG.

A. Introduction about Emotion in Humans

Humans are not only pure-function creatures; humans associate with most of their actions or reactions emotions. For this reason, many researchers have studied human

emotions to make machines more aware of affective state of humans. However, there is still a gap between the humans and machines due to lack of methods in understanding these emotions. In order to achieve to the affective computing, many studies have tried to recognize human emotions by studying speech, visual appearance, audiovisual, facial expressions and body gestures. In addition, physical signals such as electrocardiogram (ECG), galvanic skin response (GSR), electromyogram (EMG) and electroencephalogram (EEG) have also been used to recognize human emotions, since they carry extra information about human state. More importantly, EEG data may reveal the actual human emotions since it represents brain activates, source of our actions or reactions in response to a stimuli.

B. Affective State Definition

The affective state defines our experience of feeling or emotion as a reaction for a stimulus or stimuli. The affective states are psycho-physiological components that can be measured using three principle dimensions, the valence, arousal and motivational intensity. The valence varies from negative-to-positive and measures the emotion's consequences, emotion eliciting circumstances or subjective feeling and attitudes. In our daily definition for emotions, valence is correlated with sad/ negative to happy/positive.

On the other hand, arousal measures the activation of the sympathetic nervous system; however, it does not necessarily imply an action. Arousal varies from bored /negative to excited/positive.

There are different models to represent the affective states such as the six basic emotions model [45], Dimensional scale of emotions model [46], the tree structure of emotions model [47] and the valence-arousal scale model [48]. In this chapter we will rely on the valence-arousal scale model since it is widely used by the related works. The following graph shows valence-arousal scale proposed by Russell, where the emotion or the state is described in 2D plane. The horizontal axis is the valence while the vertical one is the arousal. This model can describe most of the variation in an emotional state. The motivational intensity can be also included as a third axis in the model according to Russell.

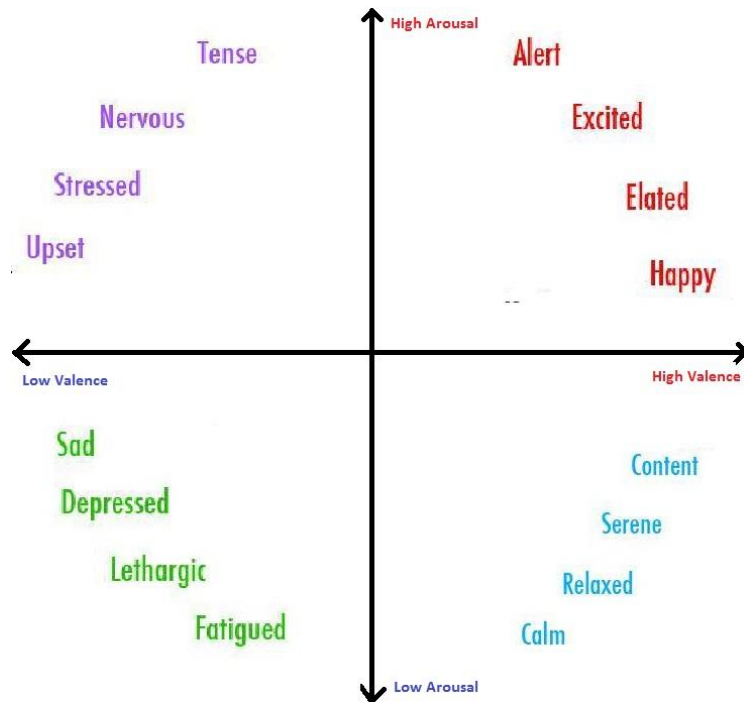


Figure 12: Russell's model to represent emotions.

The next section surveys the related work for emotion recognition from EEG, where the section first shed a light on some works that use different EEG datasets and then we

focus on the works that used the DEAP dataset [49] for EEG. By focusing on DEAP dataset, the work will be able to deliver consistent evaluation and analysis with other works in terms of the accuracies and testing approaches.

C. Dataset for Emotion Recognition

We chose in this work the DEAP dataset, since it is recently introduced and was used by many works. By choosing DEAP dataset, we ensure that our comparisons and evaluation are consistent with other works.

DEAP dataset consist of EEG data recoded from 32 subjects, while they were watching 40 prechosen videos. The 40 videos /stimulus in DEAP dataset were chosen among 120 initial YouTube videos, where half of the 120 were chosen manually while the remaining were chosen semi-automatically. For these 120 videos, a one-minute highlight part was determined and then all were presented to subjective assessments experiment to choose 40 videos.

In DEAP dataset, the 32 subjects are 50% female and 50% male, aged from 19 to 37 years with an average of 26.9. For each subject, each video was presented to him/her and then he/she was asked to fill a Self-assessment for his/her valence, arousal, liking and dominance. The valence scales from unhappy or sad to happy or joyful and corresponds directly to a number from 1-9 (1 represents sad and 9 represents happy). The arousal scales from calm or bored to stimulated or excited and directly corresponds a number from 1-9 (1 represents calm and 9 represents excited). The liking measures whether the subject likes the

video or not and corresponds to a number from 1-9, where 1 means that the subject did not like the video, while 9 means that the subject likes the video.

The EEG data were recoded according to 10-20 international system at the rate of 512 Hz. Afterwards the data was preprocessed and down-sampled to 128 Hz.

Table 1: DEAP dataset summarization.

Feature	Description
Number of Subjects	32
Number of Videos /Stimulus	40
Number of Channels	32
Labels	Valence, Arousal and Liking
Sampling Rate	128 Hz

D. Literature Review

This section is divided into two parts: in the first part, we survey the related work on emotion recognition from EEG done on datasets other than DEAP dataset; and in the second part, we survey related works done on DEAP dataset.

1. Works that relies on different datasets

In [50], five subjects were used to record the EEG data to measure four emotional states which are: joy, relax, sad and fear using a pre-chosen elicitation clips. For each participant, 62 types of active scalp site were recorded at a sampling rate of 1000 Hz, and then these signals were down-sampled to 200 Hz. Two types of features were extracted from the raw data: the time domain and the frequency domain features. In the time domain, six features from each of the 62 signals were calculated: the mean, the standard deviation, the mean of the absolute values of the first difference, the mean of the absolute values of the second difference, the means of the absolute values of the first differences of the

normalized signals and the means of the absolute values of the second difference of the normalized signals. For the frequency domain, five features of the log power spectrum were extracted from the delta, theta, alpha, beta and gamma frequency bands.

In the testing part, SVM with radial basis kernel function, K-NN and a three-hidden-layer MLP were used to classify the extracted features. The results have shown that SVM has the best performance with 43.39 % accuracy for time domain features and 66.51% accuracy for frequency domain features. It was also shown that frequency domain features deliver a better classification accuracy than those of the time domain. The study also showed that the frontal and parietal EEG signals have higher discriminative information about the emotional state than other EEG signals.

According to [51], EEG features can be divided into two domains; the time domain and frequency domain. In the time domain, the components that reflect the corresponding emotion are represented by the event-related potentials (ERPs). The ERP components are represented as three types according to their durations: P1 and N1 are the components that generated after the stimulus with a short latency, N2 and P2 are generated with a middle latency and P3 which is generated after a long latency. Besides the P3, the slow cortical potential (SCP) component might be taken into the account as a long latency component.

Several studies have shown that ERP components of short to middle latencies are correlated with valence, while the ERP components of middle to long latencies are correlated with arousal. ERP components have to be chosen carefully over several trials,

however a recent studies have shown that is applicable to get good ERP features from a single trial.

For the frequency domain, spectral power of various frequency bands has been used to infer the related emotional state. It has been shown that alpha power varies with valence state. In addition, the gamma power has a high correlation with the happiness and sadness. The theta power has been shown to be correlated with the transition between the emotional states. Other characteristics of the EEG signals such as phase synchronization and coherence have been also used to infer the emotional state, e.g., the beta oscillation has been shown to be correlated with high arousal stimuli.

In [52], 30 pictures from the International Affective Picture System (IAPS) that belongs to six different clusters of emotions were used as an elicitation for 20 subjects. Each picture was presented for the subject for 5 seconds with a gap of 5-12 seconds between each picture. To avoid the misinterpretation of the emotion, each subject was also asked to fill a Self-Assessment Mankind (SAM), which is used to rate the subject's valence and arousal in a 2D scale. Thereafter, the collected EEG data was screened for the users with low valence and arousal rate and then preprocessed to extract four types of features; the minimum value, maximum value, mean value and standard deviation. The final dataset is made up of 24 features from six channels plus the classes (positive or negative valence/arousal and neutral). Five machine intelligence techniques were applied on the final dataset namely, K-NN, Regression trees, Bayesian Network, SVM and ANN. In all experiments, SVM achieved the highest accuracy with 56.1%.

The work then tested the data to check if there is a problem with the generality by dividing it into three subsets, each has five subjects. The same machine intelligence techniques were again applied, and the reported results for this setup showed that SVM still outperforms the other techniques with average accuracy of 66.47% for the three subsets.

The issue of designing an automated method to detect the negative emotional states from EEG was studied in [53]. The method consists of three parts: model creation, threshold estimation and the detector. In model creation, the short term energy is computed for each channel by using a sliding window of 343.75 msec with 75% overlapping between consecutive frames. Then these short term energies are normalized to have zero mean and unit standard deviation. To that end, the data is ready for building a classifier model using SVM to classify samples into negative (“0”) state or others (“1”).

The second part is responsible for finding a threshold value form the independent data (development data) by computing the short term energy and the ratio of each state in the data.

After performing the first and the second part, the model is ready to predict the states from training data by first computing the short term energy and then evaluating each frame by a pattern recognition process to find the corresponding state. The final decision from the whole frames were conducted by calculating the ratio of negative states to the number of frames and then compare it to the threshold value. In the testing phase, the work selected 10 users out of the 32 users in DEAP dataset. From each user 8-9 samples were used for training data and 8-9 were used for development data. The reaming 22-24 samples

were used for testing. The highest achieved accuracy among all users was 73.9% while the lowest was 54.5% with average accuracy over all users of 62%.

2. Works used DEAP dataset

In [54], the work used the Deep learning Networks (DLN) to recognize emotions from DEAP dataset. The work used the power spectral of five frequency bands of EEG: delta (1-3 Hz), theta (4-7 Hz), alpha (8-13 Hz), beta (14-30 Hz) and gamma (14-30 Hz). For each of the 32 channels used in DEAP dataset, the power spectra of the five frequencies was extracted along with the differences in the power spectral between the 14 pairs of electrodes on the right and the left of the hemisphere. 230 features extracted from the previous dataset were fed into a stack of three auto-encoder and two softmax layers. The first softmax layer is used to estimate the three valence states (Negative, Neutral and Positive) and the second softmax layer is used to estimate three arousal states (Negative, Neutral and Positive). The experimental part used four setups: (1) used 100 neurons in each hidden layer in DLN; (2) used 50 neurons in each hidden layer; (3) used the PCA transformation on the input features to extract the most important 50 features and then fed the transformed data into a DLN with 50 neurons in each hidden layer; (4) used the Covariate Shift Adaptation of Principal Components (PCA+ CSA) and then fed the transformed data into a DLN with 50 neurons in each hidden layer. The testing used the subject dependent scenario where the reported results are as follows: DLN with 100 neurons provided 49.52% for valence and 46.03% for arousal, while these accuracies decreased to 47.87% and 45.50% for valence and arousal, respectively when using DLN

with 50 neurons. The PCA increased the accuracies by 3.01% for valence and 3.14% for arousal when used for DLN-50, where the best achieved results was when using PCA + CSA with DLN-50; the accuracies became 53.42% for valence and 52.03 %for arousal.

In addition to the previous experiments, the work tested the performance of an SVM-RBF classifier on the data after the auto-encoders. In this case, the SVM was tested according to three configurations: 230 input features, PCA transformed data and PCA+CSA transformed data. The highest achieved accuracy by SVM was when using pure input features without any transformation was 41.12% for valence and 39.02% for arousal.

In [55], the work proposed a three-step method to transform a pre-segmented EEG into feature vectors. In this work, each problem is considered as a binary classification problem, i.e. the valence, arousal, like, and dominance are divided into either low (class “0”) or high (class “1”). In the first step of the proposed method, the method finds the first 2K nearest neighbors in the segments that do not belong to the same response (EEG signals in each segment) after assuming that the responses are already labeled. Thereafter, the distances to neighbors in the two classes are computed and preserved to compute multinomial distribution vectors. Then, these vectors are fused to form response-level feature vectors. The work used four ways to perform the fusion: (1) NN voting histograms in which the features represent the relative number of segments that the nearest neighbors belong to the class “1”; (2) average segment-to-class distance which computes the average distance to the K nearest neighbors in the class”0” and K ones in the class “1”; (3) histograms which finds the B-bin normalized histograms for each dimension in segment-level probability vectors over all segments in the response; (4) generating Dirichlet

distribution where Dirichlet distribution parameters are computed using the moment matching approximation and then these parameters are used as features.

The final classification on the three fusion methods is carried out in three fashions: NB-NN classifier which compares the coordinates of average distance to the K nearest neighbors fusion; NN-voting classifier which compares the coordinates of NN voting histograms fusion with 0.5; and SVM with RBF kernel that uses the whole four fusions data combined in one vector to perform the classification.

In testing, the setup was to choose different window length specifically, 1s, 2s, 4s and 8s with 1s segment shift to generate the segments. From each segment, 230 features were extracted as in [54]. All classifications were conducted for user-dependent scenario and the results were averaged on all the 32 users and reported as means and standard deviations. In the first experiment, the segment level classification was tested after performing the kernel principle component analysis (K-PCA) and 1-NN classifier. The best results were as follows: valence: $64.4 \% \pm 5.2$ when segment size S is 2s, arousal: $60.6 \% \pm 8.1$ when S is 2s, dominance: $62.3\% \pm 10.4$ when S is 2s and $64.5 \% \pm 12.0$ for liking when S is 4s. In the second experiment, the mean, standard deviation, min, max range, mode, median, skewness and kurtosis were extracted for the whole segments features (230 features) $=9 \times 230$. After that, K-PCA with 1-NN was applied for the 2070 features, and the best reported results were as follows; valence: $54.9 \% \pm 8.2$ when S is 8s, arousal: $59.5 \% \pm 8.5$ when S is 2s, dominance: $61.3\% \pm 15$ when S=4s and $62.2 \% \pm 14.2$ for liking when S is 2s. The third experiment tested the performance of the proposed fusion methods. In this case, SVM with RBF delivered the highest accuracies compared with the two other fusions

where the accuracies are as follows; valence: $76.9\% \pm 6.4$ when S is 1s, arousal: $69.1\% \pm 10.5$ when S is 4s, dominance: $73.9\% \pm 11.1$ when S is 1s and liking $75.3\% \pm 10.6$ when S is 1s.

Later the work [56] introduced a robust method to transform segment-level features to response-level features by using two-part unsupervised generative model method. The first part is Gaussian Mixture Model (GMM) that is followed by, the second part, Generative models constraining GMM. The segment features are extracted as in [55] for segment size of 1s and 0.1s step size. The 230 dimensional segment-level feature vector is downsized to 140 dimensions using K-PCA. Thereafter, the proposed method is used to generate one 100 dimensional response-level vector. The final classification stage was conducted using SVM with best achieved accuracies of $70.9\% (\pm 11.4)$, $67.1\% (\pm 14.2)$, $70.9\% (\pm 12.8)$ and $70.5\% (\pm 17.1)$ for valence, arousal, dominance and liking, respectively.

The matter of low number of samples and choosing the critical channels in affective emotion was studied in [57]. The work proposed a new technique based on Deep Belief Networks (DBN) to handle these two issues. In details, the DBN was deployed to extract a low dimensional feature vector from input channels which include thousands of features. Then, these channels are evaluated in order to find the most critical channels that affect emotions. This is done by defining a zero-stimulus as an input before the DBN. Thereafter, the output of the DBN at the very end layer is evaluated by measuring how many units are activated in the last layer of DBN. If the ratio of the number of activated units to all units in this layer is closed to 0.5, then the related channel is irrelevant to the learning task. On the

other hand, if the ratio varies from 0.5, then this channel has an effect in the learning process. In this work, the top five critical channels were used to perform the next level that is the classification. To evaluate the new method, “like” and “dislike” were used for the classification task along with other five baseline methods for comparison purposes. The methods are as follows: (1) SVM, (2) SVM with PCA for features reduction, (3) SVM with Fisher Criterion for channel section, (4) SVM with PCA and Fisher Criterion, and (5) DBN for feature extraction with Fisher Criterion for features reduction.

The experimental part chose randomly 20 segments for training and 20 other for testing. The results were reported in area under the curve (AUC) form. For 28 out of 32 users in the dataset, the proposed method outperformed the other five baseline methods, while for the remaining four users, it achieved close results. In addition, the stability of the method for choosing critical channels is compared with Fisher Criterion and it has been shown that the proposed method tends to have stable choices among the channels.

In [58] which is an extension for the [57], the matter of choosing the critical channels was performed using a two-stage process. The first stage uses the unsupervised learning to evaluate the extracted features from RBM by measuring the constructed error from RBM. The second stage uses the supervised learning to refine the selection process of channels which resulted from the first stage by computing the significance of each channel to the classification task. More specifically, the second stage computes the distances between the extracted features for each channel and the mean of features' values for a specific class with respect to the distances between the extracted features to the mean of features in all classes.

In order to deal with low number of samples issue, the work suggested that the final layer in DBN can be trained on a combinations of supervised information that represent the labeled data, unsupervised information that represent the unlabeled data and produced data (generative data) by using RBM as a generative model. The whole process was called a semi-DLM method.

In addition, the work suggested that the proposed method can be used for data labeling by training the model on labeled and unlabeled samples. Afterwards, the unlabeled samples that have highest certainty to belong to a specific label are chosen and labeled accordingly (the method was called active-DLM).

In addition to the five baseline methods that were used in [58], the work also used Deep Learning Model (DLM) that uses only the discriminative and generative data in the final classification layer.

For all users in the dataset, the proposed method outperforms the other baseline methods except for two users with a maximum AUC of 0.852 and a minimum of 0.705.

In addition to the classification task, the work tested active-DLM for labeling the data. The work fixed 10 samples for each user for testing and 10 others out of the remaining 30 samples for initial training. The active-DLM chose best two samples from unlabeled data and then labeled them. This process was repeated five times. For DLM and semi-DLM, the same 10 samples were fixed for testing and 10 out of the remaining 30 samples were chosen randomly to train the model for 10 times. The results have shown

with active-DLM, the average AUC was 0.808 which is better than semi-DLM and DLM for data labeling with an average AUC of 0.767 and 0.789, respectively.

Other works took another approach to extract features from EEG. For example, in [59], the authors tested the band power for delta, alpha, beta and gamma frequencies as features for identifying emotions. Besides, the authors tested the power spectral density (PSD) by wavelet transformation as another method to extract the features. The work conducted the experiments according to two scenarios: first, when using all the 32 channels in DEAP dataset; second: when using the channels Fp1, Fp2, F3, F4, T7, T8, P3, P4 and O2 where these choices are according to work [60].

With the pervious setups, the work produced six datasets. D_10_1: where it uses the 10 channels and the band power of the four signals averaged over the one minute length (4 frequencies \times 10 channels features). D_10_5: where it uses five statistics for each frequency; (average, max, min, range and standard deviation). Hence, the feature vector for each sample is 4 frequencies \times 10 channels \times 5 statistics. D_32_1: which is the same as D_10_1, but it uses the 32 channels. D_32_5: is same as D_10_5, but using the 32 channels. D_10_1_WT: is similar to D_10_1, but using the PSD wavelet transformation. D_32_1_WT: is the same as D_32_1, but using the PSD wavelet transformation.

In all experiments, valence, arousal and liking were divided into binary classes that are positive and negative classes. The classification was performed using SVM with leave-one-trial-out (leave one video out of the 40 videos) or leave-one-subject-out (leave one subject out of the 32 subjects).

The results have shown that averaging over the one minute is better than the five statistics for valence, arousal and liking when using the band power case. In addition, the results have shown that using 10 channels gave better results than using 32 channels for valence and arousal, while the liking gave a slightly better result for the 32 channels case. Moreover, the results have indicated that there is no benefit from using the PSD of wavelet transformation when compared with the band power method.

In addition, the results have shown that the leave-one-trial-out is better than leave-one-subject-out for valence and arousal and slightly worse for liking.

The work [61] tested the recognition of the level of valence and arousal using the sample entropy method with SVM classifier. In this work, the first 23 seconds and the last 20 seconds of the EEG data were removed, and then the sample entropy was calculated for each channel in the DEAP dataset. Thereafter, the channels were screened according to the highest resultant sample entropy for each classification task. The result showed that channels

F3, CP5, FP2, FZ and FC2 are informative for differentiating between High Arousal – High Valence (HAHV) and High Arousal – Low Valence (HALV), while channels FP1, T7 and AF4 are informative to differentiate between Low Arousal – Low Valence (LHLV) and High Arousal – High Valence.

The results of testing the quality of the extracted features were conducted in two cases: 3-fold cross validation and leave-one-user-out (LOSO). In 3-fold cross validation, the average accuracy was 80% for recognition between HAHV and HALV and 79% for recognition between LALV and HALV. While for LOSO, the average accuracy was 71%

for recognition between HAHV and HALV and 64% for recognition between LALV and HALV.

Table 2: Related work on Emotion Recognition summary 1.

Work	Features	Classes	Classifier	Accuracies	Criticism
[55]	Band power of delta, alpha, beta, gamma frequencies. -power Spectral density by wavelet transformation for alpha, beta, gamma and delta - followed by GMM and Generative model constraining GMM.	Valence(high/Low) Arousal (high/Low) Liking (yes/no) Dominance(high/Low)	SVM	Valence 70.9% (11.4) Arousal 67.1%(14.2) Liking 70.5(17.1) Dominance 70.9(12.8)	Per-user
[60]	Sample entropy	High Arousal High Valence /High Arousal Low Valence (HAHV/HALW) Low Arousal Low Valence /High Arousal Low Valence (LALV/HALV)	SVM	3-fold cross validation (HAHV/HALW) 80% (LALV/HALV) 79% LOSO (HAHV/HALW) 71% (LALV/HALV) 64%	Classes are questionable Time of EEG signal was not all used
[53]	PCA + CSA +DBN	Valence (LOW/Neutral/high) Arousal (LOW/Neutral/high)	DBN	LOSO Valence 53.42 +- 9.64 Arousal 52.02 +-9.74	
[57]	DBN Extracted Features + Critical channels Selection	Liking (yes/no)	SVM	0.705-0.852 (AUC)	The work focuses on solving low number of samples problem, but not classification

Table 3: Related work on Emotion Recognition summary 2.

Work	Features	Classes	Classifier	Accuracies	Criticism
[54]	<p>- after dividing the EEG into segments, the power spectral for theta (4-8) Hz, slow alpha (8-10 Hz), alpha(8-12 Hz), beta (12-30 Hz), gamma (30 + Hz) and spectral power differences between symmetric channels for the same bands; 5×(32 + 14) features.</p> <p>- A method to convert segment level features to response level features.</p>	<p>Valence(high/Low)</p> <p>Arousal (high/Low)</p> <p>Liking (yes/no)</p> <p>Dominance(high/Low)</p>	K-PCA+S2R+SVM	<p>Valence 76.9% (6.4)</p> <p>Arousal 69.1%(10.5)</p> <p>Liking 75.3(10.6)</p> <p>Dominance 73.9(11.1)</p>	Accuracies are using per-user testing
[56]	DBN Extracted Features + Critical channels Selection	Liking (yes/no)	DBN	0.67-0.87 (AUC)	The work focuses on solving low number of samples problem, but not classification
[58]	<p>- Band power of delta, alpha, beta, gamma frequencies.</p> <p>-power Spectral density by wavelet transformation for alpha, beta, gamma and delta</p>	<p>Valence (high/Low)</p> <p>Arousal(high/Low)</p> <p>Liking (yes/no)</p>	SVM	<p>Val 64.9% (LOVO)</p> <p>Arousal 64.9%(LOVO)</p> <p>Liking 67.3%(LOSO)</p>	

CHAPTER VI

CASE STUDY FOR DEPLOYING LIQUID STATE MACHINE FOR FEATURE EXTRACTION FROM RAW DATA

This chapter studies a novel deployment, for LSM, which is the feature extraction. Feature extraction is a challenging problem in ML and AI, since the quality of features strongly affects classification or regression tasks based on these features. In this chapter, we study harnessing LSM for feature extraction from raw data. More specifically, this chapter is concerned with studying automatic feature extraction from data streams of EEG.

For comparison purposes, we use one of the most successful approaches for feature extraction that is DBN. Deep Learning (DL) is a variant for the shallow architectures of Artificial Neural Networks (ANNs). DL consists of hierarchical many layers that transfer information from the lower layers to upper layers through a non-linear conversion while performing error minimization between consecutive layers. The upper layers preserve the most abstract representation for the data, where it can be used as a feature extraction layer. DBN [26], a DL network with fast training algorithm, has been deployed in many works for feature extraction, where it showed powerful capabilities. For example, DBN has shown to be able to produce high informative features than Met-Frequency Cepstral Coefficients (MFCC) for music audio feature extraction [62]. For EEG raw data, the extracted features by DBN in [58, 63] have produced a high classification accuracy than those extracted by applying time and frequency domain methods.

A. LSM-based Feature Extraction Description

The LSM receives inputs as data streams from one or multiple inputs. We will use direct information feeding to LSM by mean of an analog neuron to provide the LSM with raw data. For this purpose, we will use CbNeuron neuron model to build the LSM. CbNeuron neuron model, which was described in section 2 of CHAPTER II, can receive an analog input from an analog input neuron. We will use CSIM simulator [64], since it provides an easy environment to simulate SNNs.

1. LSM Configuration for Feature Extraction

The architecture of LSM consists of $7 \times 7 \times 7$ neurons with 32 inputs corresponding to the 32 channels of DEAP dataset channels [65]. To make EEG suitable for CSIM simulator and Experiment 1, we first rescale EEG data for each channel in DEAP dataset into some ranges between 0.1 and 10. Scaling ensures that EEG data are suitable for analog input neuron and CbNeuron in CSIM simulator. In addition, scaling ensures that data are consistent among channels from different subjects' EEG signals.

Analog input neurons, implemented in CSIM simulator, interfaces EEG signal with CbNeuron neurons in LSM, where we configured it to update its internal state at 128 Hz. This implies that LSM updates its state at the same rate. Analog input neurons in experiment 1 are connected to the LSM with a probability of $w_{input} = 0.15$ and connection scaling of $C_{scaling}=0.2$, i.e., the probability that an analog input neuron is connected to a neuron in LSM is 0.15 with scaling of this connection with 0.2.

Neurons in LSM are connected with “average distance” synaptic connections $\lambda = 2$. 80% of neurons inside the LSM are excitatory neuron and the remaining 20% are inhibition ones. Reading from LSM is achieved by sampling the liquid states, and more specifically reading the spiking activities from LSM. For this purpose, CSIM simulator records the spiking time activates from each neuron in the LSM along the simulation time, which was configured to 59s. To obtain the liquid states from LSM, we read the recorded spiking activities by CSIM using an exponential filter with time constant $\tau = 0.5$. Sampling from LSM is performed every 0.4s starting from 0.5s until the end of simulation time at 59s. All neurons in LSM are used for reading, which produces a feature vector of $7 \times 7 \times 7 = 343$ values from each sample.

Having obtained the liquid states from LSM, these liquid states are then preprocessed to normalize data in each column to get zero mean and unit variance. Next, different readouts are used to identify valence, arousal. The table describes the configuration and architecture of the LSM.

Table 4: LSM configurations for feature extraction.

Neuron Configuration	LSM Connectivity and Architecture	Readout
Model: CbNeuron $V_{\text{thresh}} = -0.045$ (V) $V_{\text{reset}} = 0$ (V) $T_{\text{refract}} = 0.003$ (sec) $C_m = 3e-08$ (F) $R_m = 1e+06$ (Ohm) $E_m : 1.0395e-314$ (V) $V_{\text{resting}} = -0.06$ (V) $V_{\text{init}} = -0.06$ (V) $V_m : 1.0395e-314$ (V)	Architecture = $7 \times 7 \times 7$ neurons $\lambda = 2$ $w_{\text{input}} = 0.15$ $C_{\text{scaling}} = 0.2$	Sampling =[0.5:0.4:59] (147 samples per each input) Filter = exponential filter with $\tau = 0.5$

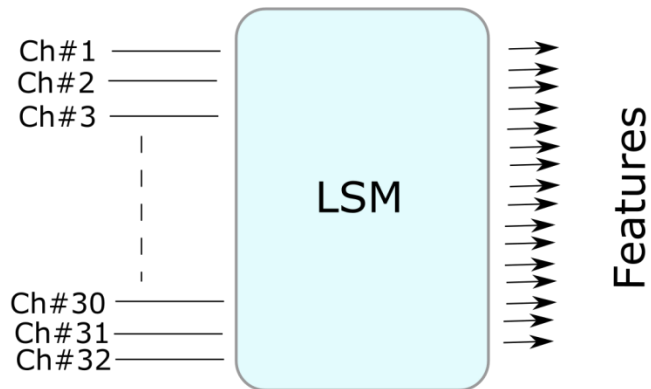


Figure 13: LSM for feature extraction.

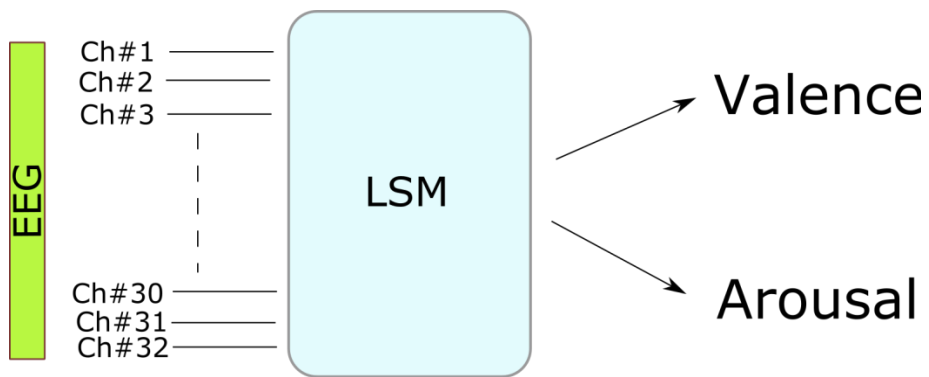


Figure 14: LSM for multi-purpose classification.

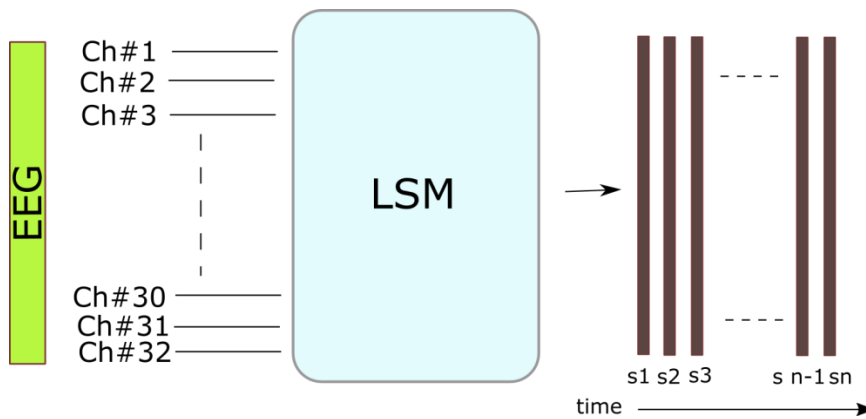


Figure 15: LSM for anytime feature extraction.

2. DBN Configuration for Feature Extraction

As mentioned previously, we will use DBN for feature extraction from EEG, and we will follow the same methods used in the literature review. The first method uses augmented channels information as an input to DBN. The second method uses each channel as independent input to the DBN, where the information in channels is used to train the DBN to generate features at the far end layer.

For the first method, we configure the network to have three layers stacked in a hierarchical structure as follows: the first layer has 768 neurons which they are connected directly to the $32 \times 128 \times 60$ augmented channels values; 512 neurons in the second layer; and 343 neurons in the third layer. We read from the third layer the information to use them later for classification. The number 343 neurons is the same number of neurons that was used in LSM such that we compare the results fairly. All the experiments are conducted using 10-fold cross validation for valence and arousal recognition tasks. We use Decision Trees and Linear Regression classifiers for the outputs from the LSM and the DBN.

For the second method, we use information from each channel as an independent sample of the input. Here, we have different configuration for the network: the first layer has 7680; second layer has 768; third layer 512 and the last layer (the feature layer) has 343 neurons.

We refer to the first configuration by augmented channels DBN and for the second configuration by independent channels.

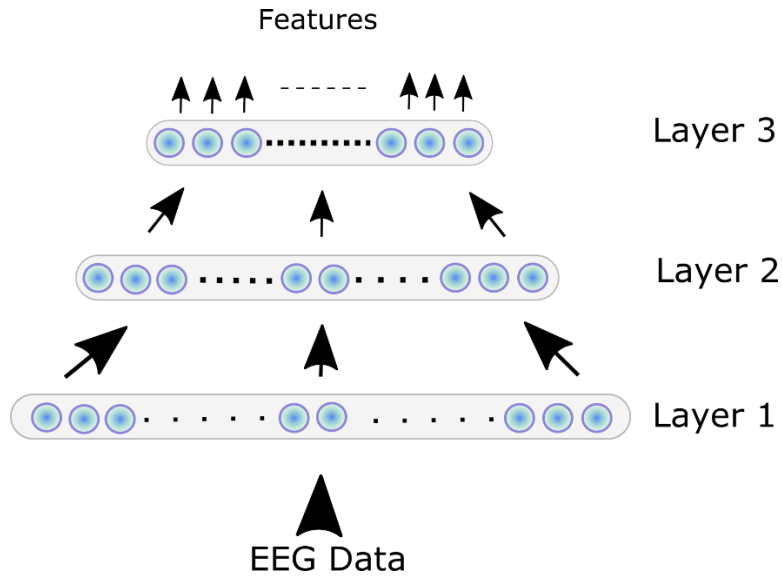


Figure 16: DBN for feature extraction architecture.

B. Results and Comparison for Feature Extraction Using LSM and DBN

In testing, we compare the accuracy of testing the resultant feature vector from LSM in comparison with DBN. In addition, we compare the generalization capability of DBN, since LSM requires one model for valence and arousal; LSM is first trained using unsupervised learning, STDP, on EEG signals, and then we train different readout functions to recognize the valence and arousal. We test DBN extracted features for valence to test recognition the arousal and vice versa. All the experiments are done in 10-fold cross validation.

Table 5: Result of testing 343 extracted features using LSM and DBN.

	Execution time	Valence		Arousal	
		Decision Trees (%)	Linear Regression (%)	Decision Trees (%)	Linear Regression (%)
DBN (Augmented channels)	690712 sec			52.58	57.34
DBN (Independent channels)	432000 Sec	64.76	70.88	64.87	72.43
LSM	50200 Sec	94.12	64.64	95.15	59.63

C. Discussion

As can be seen from section 6.B, LSM provides better accuracies than DBN with DT classification tasks and worst when using the linear regression readout. In addition to the good performance, the LSM learns to generate features from EEG as an accumulative information learning produced from all channels, i.e., LSM samples the states of the liquid, which are induced by the current and previous streams of EEG channels values. That is, LSM learns the model as one structure rather than independent inputs. For the generalization, we need to train one LSM in order to extract different information from a common domain. That is, we train LSM on EEG data to identify valence and arousal in contrast to DBN, which needs different networks for each independent task, even though the domain is common for these tasks. Moreover, the LSM is able to provide anytime feature extraction without waiting until all inputs to be provided. In other word, through sampling at a desired time, we can extract the features from LSM, in contrast to the DBN, which needs all inputs to be present ahead of time (static input) to perform the process. This is important when some inputs are missing or cannot be provided, e.g., some channels of

EEG cannot be accessed. In addition to all previously mentioned benefits from using LSM, the LSM is less computational demanding than DBN; to train the DBN on the DEAP dataset, it took 6 days of heavy simulation on a powerful PC, while LSM required 80% less time to achieve extract features from data. In conclusion, LSM produced more informative features with less costs in comparison with DBN.

CHAPTER VII

LIQUID STATE MACHINE FOR EMOTION RECOGNITION FROM EEG

In chapter V, we surveyed the related work on emotion recognition from EEG, and in chapter VI, we examined the capabilities of using LSM for feature extraction from EEG. In addition, we explained in chapter VI how we can use LSM for anytime multi-purpose learning model from EEG signals. To that end, we built a robust assumption for how to build a universal LSM-based emotion recognition model from EEG. In this chapter, we use LSM, extensively, for emotion recognition and then test the model using different scenarios and configurations. Moreover, we draw some conclusions about emotion in humans by building upon the results.

This chapter is organized as follows: In section A, we describe the experimental procedure that we will follow in all experiments in this section. This section includes three tests for different aspect of emotion recognition. Section B concludes all experiment done in this chapter.

A. Experimental Procedure for LSM-Based Emotion Recognition Model

We will use LSM to recognize valence, arousal and liking from DEAP dataset. More specifically and following the literature review, we divide the task of recognizing

valence, arousal and liking into binary classification problems, i.e., High /Low Valence, High/low arousal and like/do not like.

This chapter uses four different scenarios for testing emotion recognition that was used in the literature review:

- Subject/ Video Independent.
- Leave-one-subject-out (LOSO).
- Leave-one-video-out (LOVO).
- Isolated-User Classification (IS).

The results are reported in each experiment as follows: For Subject/ Video Independent and IS, we report the results as an average of 10-fold cross validation testing accuracies. In LOSO, we report the results as an average of testing accuracies of the 32 subjects from DEAP dataset. For LOVO, we report the results as an average of testing accuracies of the 40 videos from DEAP dataset.

For each scenario, we test the classification task at different intervals (by taking advantage of anytime feature extraction described in chapter VI); i.e., at the first 10s, 20s, 30s, 40s, 50s and on the entire length of the signal.

In addition, we test two methods to feed EEG into the LSM; using Bens Spike Algorithm (BSA) [27] and direct input using analog neurons used in chapter VI.

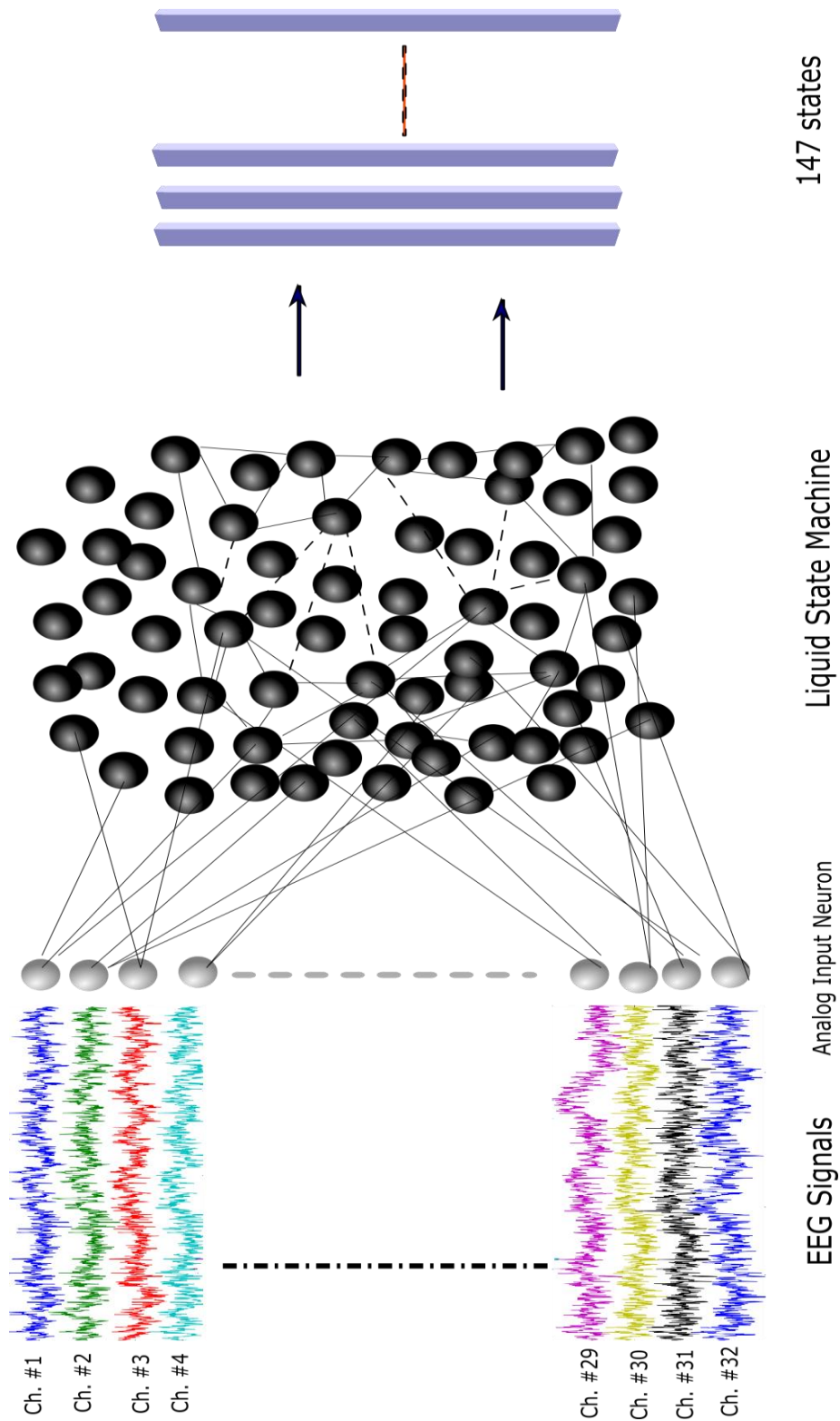


Figure 17: Topology for Experiment 1 for emotion recognition.

1. Experiment 1: Direct input to the LSM

This experiment tests the performance of emotion recognition by directly feeding the LSM with EEG signals by mean of input analog neurons. Input neurons receive signals from EEG channels, where they propagate voltages to the connected neurons inside the liquid. The LSM learns and generates the corresponding responses. The liquid in this experiment is made up of “CbNeuron” neuron model described in section 2, chapter II.

To make EEG suitable for CSIM simulator and Experiment 1, we first rescale EEG data for each channel in DEAP dataset to range between 0.1 to 10. Scaling ensures that EEG data are suitable for analog input neuron and CbNeuron in CSIM simulator. In addition, scaling ensures that data are consistent among channels from different subjects’ EEG signals.

Analog input neurons, implemented in CSIM simulator, interfaces EEG signal with CbNeuron neurons in LSM, where we configured it to update its internal state at 128 Hz. This implies that LSM updates its state at the same rate. Analog input neurons in experiment 1 are connected to the LSM with a probability of $w_{input} = 0.15$ and connection scaling of $C_{scaling}=0.2$, i.e., the probability that an analog input neuron is connected to a neuron in LSM is 0.15 with scaling of this connection with 0.2.

Neurons in LSM are connected with “average distance” synaptic connections $\lambda = 2$. 80% of neurons inside the LSM are excitatory neuron and the remaining 20% are inhibition ones. Reading from LSM is achieved by sampling the liquid states, and more specifically reading the spiking activities from LSM. For this purpose, CSIM simulator records the spiking time activates from each neuron in the LSM along the simulation time, which was

configured to 59s. To obtain the liquid states from LSM, we read the recorded spiking activities by CSIM using an exponential filter with time constant $\tau = 0.5$. Sampling from LSM is performed every 0.4s starting from 0.5s until the end desired time. Different configuration of the desired time were tested namely, 10s, 20s, 30s, 40s, 50s, and 59s. All neurons in LSM are used for reading, which produces a feature vector of $7 \times 7 \times 7 = 343$ values from each sample.

Having obtained the liquid states from LSM, these liquid states are then preprocessed to normalize data in each column to get zero mean and unit variance. Next, different readouts are used to identify valence, arousal and liking.

Table 6: LSM configuration for emotion recognition.

Neuron Configuration	LSM Connectivity and Architecture	Reading from LSM
Model: CbNeuron Vthresh = -0.045 (V) Vreset = 0 (V) Trefract = 0.003 (sec) Cm = 3e-08 (F) Rm = 1e+06 (Ohm) Em : 1.0395e-314 (V) Vresting = -0.06 (V) Vinit = -0.06 (V) Vm : 1.0395e-314 (V)	Architecture : $7 \times 7 \times 7$ neurons $\lambda = 2$ $w_{input} = 0.15$ $C_{scaling} = 0.2$	Sampling: [0.5:0.4:59] (147 samples per each input) Filter : exponential filter with $\tau = 0.5$

a. Subject/ Video Independent

This method tests the performance of classification task using 10-fold classification, where the model is trained randomly by choosing 90% of samples for training, and then is tested on the remaining 10% of samples. The final accuracy is the average over the 10 testing accuracies from the 10 folds. First, we want to choose a suitable classifier for the

model. In the table below, we show testing results of different readouts on entire length of EEG signals (59s).

Table 7: Testing results for different readouts.

Classifier	Valence	Arousal	Liking
ANN	57.82%	60.96%	67.11%
SVM	74.74%	75.13%	77.13%
K-NN	66.25%	68.22%	70.20%
Random Forests	94.03%	94.58%	94.87%
Linear Regression	64.64%	59.63%	59.57%
Decision Trees	94.12%	95.15%	95.25%

Among different types of readouts, we choose Decision Trees and Linear Regression for our further analyses. Decision Trees achieves very good results with minimum efforts of time and resources in comparison with other types of readouts. For example, SVM requires about 540 minutes for 10-fold cross validation in comparison with about 5 minutes for Decision Trees and Linear Regression. Moreover, ANN requires much memory than Decision Trees and Linear Regression without improving the accuracies. We keep using Linear Regression in all tests even though it does not improve the accuracies. This is because we want to test the produced data from LSM against linearity, i.e., we use Decision Trees as a nonlinear readout, while using Linear Regression as a linear readout.

Table 8: Subject/Video independent scenario results using Decision Tress.

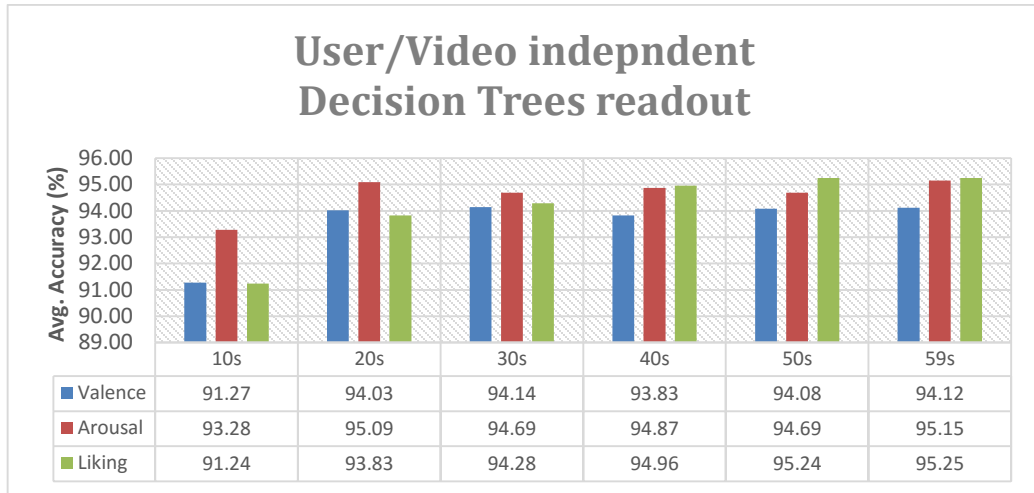
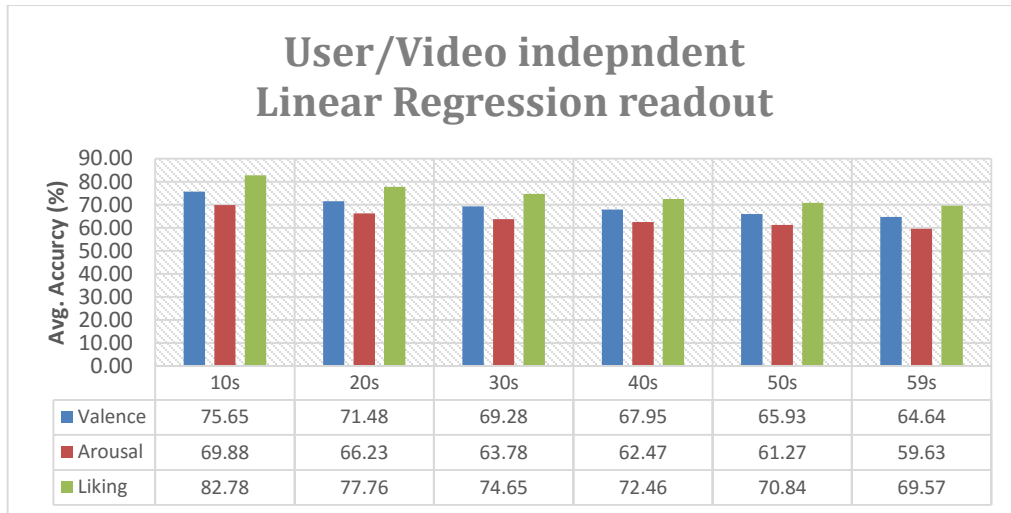


Table 9: Subject/Video independent scenario results by using Linear Regression.



b. Subject Dependent

This method trains the model on the data from 31 subject and tests on the remaining subject; Leave-One-Subject-Out (LOSO). The reported results in the figure below are the average of resultant accuracies from the 32 subjects using Decision Trees and Linear Regression Classifiers.

Table 10: Subject dependent scenario results by using Decision Trees.

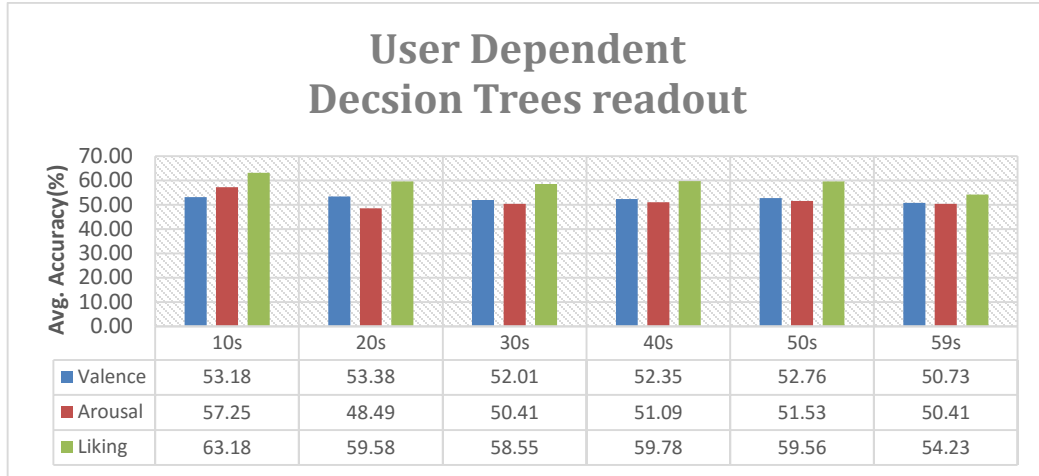
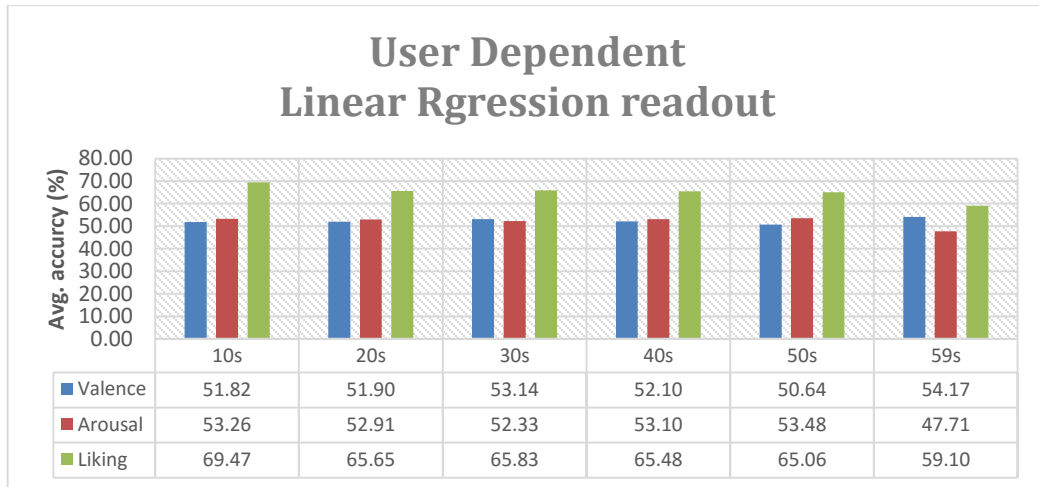


Table 11: Subject dependent scenario results by using Linear Regression.



c. Video Dependent

This method trains the model on the data from 39 videos and tests on the remaining videos; Leave-One-Video-Out (LOVO). We report the result by using Decision Trees and Linear Regression Classifier as we did in LOSO.

Table 12: Video dependent scenario results by using Decision Tress.

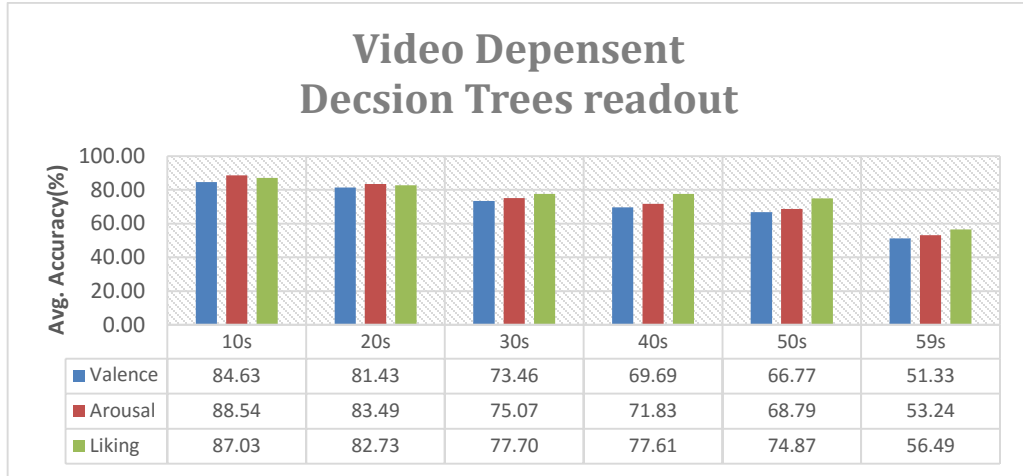
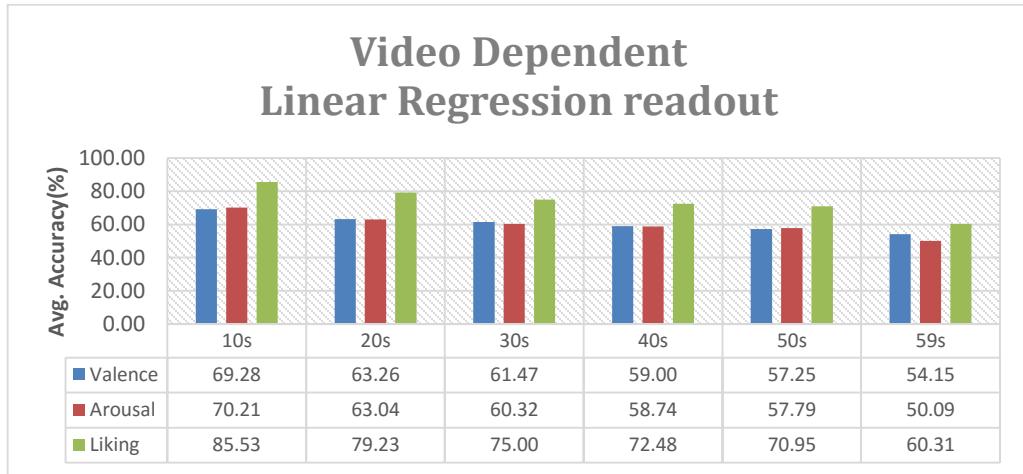


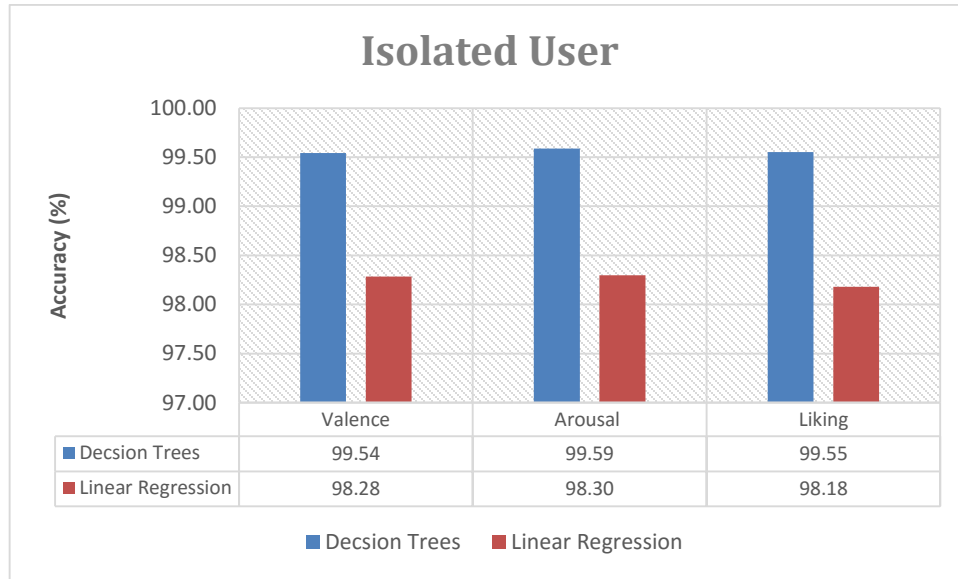
Table 13: Video dependent scenario results by using Linear Regression.



d. Isolated-Subject

This method performs 10-fold cross validation on the each subject data alone.

Table 14: Isolated-Subject scenario results by using decision trees and linear regression.



2. Results Comparison with other Machine Learning Approaches

This section is concerned without evaluate the obtained model from experiment 1 with other machine learning approaches. More specifically, we will compare our results with best reported results in literature review. We divide the comparison according to the deployed scenario in the literature review. Moreover, we provide information about type of features and classifiers used in literature review.

a. IS scenario

Table 15: Results Comparison with other Machine Learning Approaches for IS Scenario.

Work	Features	Classifier	Valence	Arousal	Liking
[65]	Spectral Power from EEG	Naive Bayes	57.6%	62.0%	55.4%
[55]	Spectral Power from EEG	RBF-SVM	76.9%	68.4%	75.3%
[56]	Spectral Power from EEG	RBF-SVM	70.9%	67.1%	70.5%
	Spectral Power from EEG	Naive Bayes	65.1%	56.3%	63.0%
Experiment1	Spiking activities from neurons in LSM	Decision Trees	<u>99.54%</u>	<u>99.59%</u>	<u>99.55%</u>

b. LOSO scenario

Table 16: Results Comparison with other Machine Learning Approaches for LOSO Scenario.

Work	Features	Classifier	Valence	Arousal	Liking
[59]	Spectral Power from EEG(from 10 channels)	SVM	51.1%	52.9%	<u>67.0%</u>
[54]	Spectral Power from EEG	DL	<u>53.42%</u>	52.03%	-
Experiment 1	Spiking activities from neurons in LSM	Decision Trees	53.38%	<u>57.25%</u>	63.18%

c. LOVO

Table 17: Results Comparison with other Machine Learning Approaches for LOVO Scenario.

Work	Features	Classifier	Valence	Arousal	Liking
[59]	Spectral Power from 10 channels of EEG	SVM	64.9%	64.9%	66.8%
Experiment 1	Spiking activities from neurons in LSM	Decision Trees	<u>84.63%</u>	<u>88.54%</u>	<u>87.03%</u>

3. Results Comparison using other Spiking Network Architecture

This section provides a comparison with other SNN architecture. We chose to compare results with NeuCube [66-71]. The NeuCube is a framework for learning Spatio/Spectro – Temporal Data (STBD) such as EEG and fMRI. The NeuCube model consists of input data encoding module; a 3D (cube) network of LIF; an eSNN classifier; and an optimization module. The framework uses STDP to learn STDB from input.

Besides, NeuCube provides a very rich environment to visualize learning progress inside the 3D network. In addition, it provides a semi-brain simulation environment, where locations of EEG channels when recoding from a brain can be mapped spatially to same locations in the 3D network.

Due to license constraints, we were only able to run NeuCube on one subject data namely, subject 1 from DEAP dataset, which is was not enough to train the NeuCube for reasonably acceptable results. Here, we provide testing results of NeuCube on subject 1 (3-fold cross validation).

Table 18: Results of Testing NeuCube on subject 1 from DEAP dataset.

	Valence	Arousal	Liking
Decision Trees	99.52%	99.76%	99.88%
Linear Regression	99.93%	99.95%	99.97%
NeuCube (BSA: threshold=0.995; number of neurons =343)	60%	51.25%	95%
NeuCube(BSA: threshold=0.5; ; number of neurons =343)	55%	52%	95%
Neucube (Thresholding Representation: threshold =0.5; number of neurons =343)	42.25%	66.25%	95%
NeuCube (BSA: threshold=0.995; number of neurons =1000)	65.25%	48.57%	95%

The low accuracies achieved by NeuCube are due to lack samples to train the model.

However, NeuCube provides a unique opportunity to understand information propagation inside the 3D network of NeuCube, and more importantly the relation between different channels with respect to emotions. A more realistic approach to study emotion could be performed using NeuCube by mapping channels of DEAP dataset into the relative locations in the 3D network of NeuCube, where we can identify the interactions between different

channels for the affective state of humans. In addition, we can use NeuCube to study Spike communications and information routes for the affective state of humans.

4. Results Analysis and Discussion

For 10-fold cross validation: experiment 1 shows that valence and arousal can be determined effectively after the first 20 seconds of a continuous stimulus where the accuracies of determining the affective state are around 94% and 95% for valence and arousal, respectively. Regardless of the duration of the stimulus, the accuracies remain slightly around these values when the duration of a stimulus is more than 20 seconds. The accuracies of the first 10 seconds were about 91% for valence and 93%, and this indicates again that to accurately determining the affective states when need at least more than 10 seconds of a stimulus. On the other hand, the accuracies of determining “liking” are improved as the duration of a stimulus increases (from 91.24% for 10 seconds to 95.25% for 59 seconds).

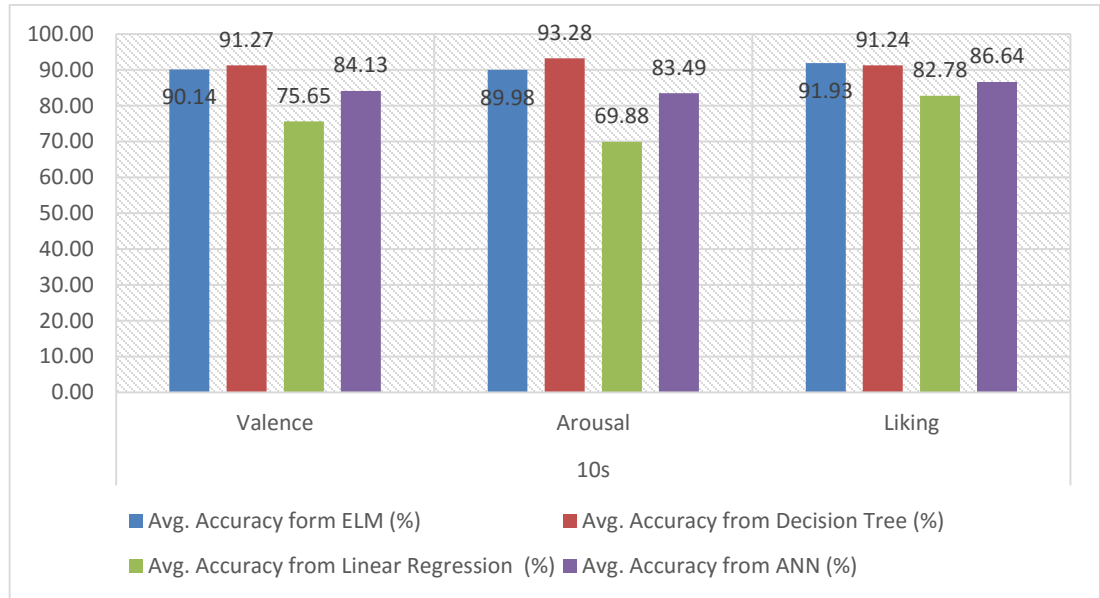
The reported results for Subject/Video independent scenario show that decision tree outperforms linear regression classifier in all cases. This is a strong indication that valence and arousal recognition are a nonlinear problem. More specifically, valence, arousal and liking accuracies drop steadily as the duration of the stimulus increases. In details, the valence accuracy drops from 75.65% for 10 seconds to 64.64% for 59 seconds; the arousal drops from to 68.88% for 10 seconds to 59.63% for 59 seconds; the liking drops from 82.78% for 10 seconds to 69.57% for 10 seconds.

In addition, the valence tends to have a more non-linear relation with a stimulus than arousal. Also, liking has the strongest linear relation with a stimulus.

For valence, arousal and liking, the relation between stimulus and the affective states becomes more non-linear as the duration increases.

To test for the non-linearity in the data, we used an Extreme Learning Machine (ELM) [72] Classifier and ANN as additional methods for readout. ELM produces nonlinear decision boundaries to separate the different classes. Hence, ELM can be used to judge whether the data are non-linear or the decision trees classifier happened to work well for emotion recognition. In addition, we configured ANN to have two hidden layers each with 10 neurons such that it allows the decision function to have non-linear boundaries. Due to the heavy computational requirements of ELM and ANN, we conducted the classification task for the first 10 seconds of the EEG signal. The following graph shows 10-fold cross validation for user/video dependent scenario for the four readouts; decision trees, linear regression, ANN and ELM. The testing shows that ELM and ANN work better than linear regression which is a strong indication that the data (the data produced from the LSM) are non-linear. Moreover, ELM works better than ANN because the number of hidden neurons is relatively small which does not allow for high non-linear relationship in the decision boundaries. The table below shows the results of testing the four ML approaches. Linear Regression delivered 75.65%, 69.88% and 82.78 for valence, arousal and liking, respectively, which are below the scored accuracies reported by ELM, ANN and Decision Trees.

Table 19: Testing for non-linearity output from LSM (experiment 1).



5. Experiment 2: Spike Encoding

In this experiment, we use the BSA algorithm to encode EEG signals into a train of spikes. The purpose is to test whether BSA is able to provide LSM with better information about EEG than using direct feeding of information into the liquid as used in Experiment 1. Here, we replace the analog input neurons and CbNeuron neurons used in Experiment 1 with spiking input neurons and LIF neurons, respectively. Spiking input neurons accept only spike timing from input, which means that we have to encode input/EEG as spikes. For this purpose, we use BSA algorithm with two configurations: *Configuration A* produces about half spiking activity in comparison with *Configuration B*, i.e., number of spikes that are produced by *Configuration A* for each input/EEG channel are approximately half of those that were produced by *Configuration B*. Parameters of BSA for the two configurations are as follows:

- Configuration A: Filter Tabs = 32, Threshold = 0.995.
- Configuration B: Filter Tabs = 64, Threshold = 0.955.

a. User/Video Independent

Table 20: Subject/Video Independent results for experiment 2.

	Duration	Configuration A (%)	Configuration B (%)
Valence	10s	55.89	58.87
	20s	65.14	66.97
	30s	73.52	72.60
	40s	77.08	73.20
	50s	78.45	72.15
	59s	79.55	72.26
Arousal	10s	58.71	58.38
	20s	63.17	61.69
	30s	74.62	67.04
	40s	75.09	62.49
	50s	79.26	63.33
	59s	71.86	62.82
Liking	10s	63.17	61.19
	20s	70.68	66.52
	30s	76.33	69.47
	40s	81.22	71.87
	50s	80.94	71.02
	59s	80.76	68.65

b. User Dependent

Table 21: LOSO results for experiment 2.

		Configuration A (%)	Configuration B (%)
Valence	10s	50.95	51.17
	20s	50.41	51.15
	30s	50.84	50.78
	40s	50.44	50.79
	50s	50.45	51.12

	59s	50.32	50.63
Arousal	10s	51.60	51.67
	20s	51.18	51.25
	30s	51.29	51.33
	40s	51.24	51.66
	50s	51.41	51.49
	59s	51.57	51.36
Liking	10s	50.03	55.20
	20s	51.32	55.66
	30s	52.69	55.29
	40s	52.11	54.83
	50s	51.57	55.57
	59s	51.30	55.18

c. Video Dependent

Table 22: LOVO results for experiment 2.

		Configuration A (%)	Configuration B (%)
Valence	<i>10s</i>	50.08	50.02
	<i>20s</i>	49.89	49.87
	<i>30s</i>	50.23	50.19
	<i>40s</i>	50.19	50.82
	<i>50s</i>	50.76	50.54
	<i>59s</i>	50.42	50.24
Arousal	<i>10s</i>	51.20	51.58
	<i>20s</i>	51.73	51.28
	<i>30s</i>	51.30	51.41
	<i>40s</i>	51.52	51.52
	<i>50s</i>	51.93	51.90
	<i>59s</i>	51.73	51.25
Liking	<i>10s</i>	55.48	55.47
	<i>20s</i>	55.05	55.43
	<i>30s</i>	55.71	54.86
	<i>40s</i>	55.56	54.86
	<i>50s</i>	55.42	54.90
	<i>59s</i>	55.13	54.87

d. Isolated-Subject

Table 23: Isolated-Subject scenario results by using decision trees (experiment 2).

		Configuration A (%)	Configuration B (%)
Valence	59s	84.81	87.81
Arousal	59s	86.43	88.14
Liking	59s	86.16	88.69

6. Discussion and Analysis

The results show that BSA algorithm does not provide an effective method to encode EEG into spikes. In all tested scenarios, the reported results for this experiment are below of those results scored by the direct feeding in experiment 1. For example, the best reported accuracies for 10-fold cross validation were 79.55%, 71.86% and 80.76% for valence, arousal and liking, respectively in comparison with 94.12%, 95.14% and 95.25% for the same scenario in experiment 1. This could happen due to two reasons:

- Parameters are not well selected, since there is no definite way to choose parameters values.
- The method by itself is not efficient and this opens the door for a future research.

7. Experiment 3: Emotion Recognition Using Different Number of Channels

In this experiment, we use the same configuration that was used in experiment 1, but using 10 specific channels. These channels are: Fp1, Fp2, F3, F4, T7, T8, P3, P4, and O2, which are chosen according to work [60]. The work claimed that these specific channels deliver a better performance than using the 32 channels.

a. User/Video Independent

Table 24: Subject/Video Independent results for experiment 4.

	Duration	Accuracies (%)
Valence	10s	77.29688
	20s	70.87969
	30s	67.96771
	40s	66.16016
	50s	63.71563
	59s	62.20823
Arousal	10s	87.17813
	20s	89.81094
	30s	90.97917
	40s	91.25156
	50s	92.1525
	59s	92.02434
Liking	10s	86.87813
	20s	90.275
	30s	90.91354
	40s	92.09453
	50s	91.72688
	59s	91.7905

b. User Dependent

Table 25: LOSO results for experiment 3.

	Duration	Configuration A
Valence	10s	49.66875
	20s	49.47656
	30s	49.70521
	40s	50.15781
	50s	50.04063
	59s	50.60055
Arousal	10s	51.60938
	20s	51.62813
	30s	52.33958
	40s	50.88594
	50s	51.1525
	59s	51.50563
Liking	10s	55.13125
	20s	55.06563
	30s	55.15833

	40s	55.03594
	50s	55.19
	59s	55.40179

c. Video Dependent

Table 26: LOVO results for experiment 3.

	Duration	Configuration A (%)
Valence	10s	77.29688
	20s	70.87969
	30s	67.96771
	40s	66.16016
	50s	63.71563
	59s	62.20823
Arousal	10s	87.17813
	20s	89.81094
	30s	90.97917
	40s	91.25156
	50s	92.1525
	59s	92.02434
Liking	10s	86.87813
	20s	90.275
	30s	90.91354
	40s	92.09453
	50s	91.72688
	59s	91.7905

d. Discussion and Analysis

The results indicate that using specific channels does not improve the accuracy of a classification task. LSM requires the entire 32 channels in order to deliver good accuracies. However, the experiment demonstrates that LSM works with acceptable performance when

some of the features are missing (in terms of EEG, the LSM can deliver the classification under lack of input information). The best reported results are 62.21%, 92.02% and 91.79 for valence, arousal and liking, respectively when testing 10-fold cross validation. In LOSO, the reported results are between 49.67% and 55.4%. The accuracies improve when using LOVO to 77.3%, 92.15% and 92.09% for valence, arousal and liking, respectively.

B. Conclusion

This chapter provided an extensive study for different task in emotion recognition from EEG. In the first experiment, we showed how LSM can deliver a good accuracy in identifying the affective state. Moreover, we showed how we can deploy LSM as a multi-purpose classifier, where we tried to identify the affective state, including valence, arousal and liking from one single trained LSM. In addition, we suggested that LSM can be used for anytime classification for online learning purposes. In experiment 2, we tested spike encoding algorithm, BSA, for identifying the affective state where we showed that this method of encoding needs more study in order to make it suitable for spike encoding for LSM's input. Experiment3, tested the affective state recognition in the lack of information, where we used 10 channels out the 32 channels for affective state recognition. The experiment did not improve the performance. However, it showed that one LSM can work and deliver the classification task when some features are missing.

CHAPTER VIII

A LIQUID STATE MACHINE -BASED FRAMEWROK FOR CONTINUOUS AUTHETICATION IN SMARTPHONES

This chapter introduces an LSM-Based Continuous Authentication Framework for smartphones. The motivation behind this chapter is to use the flexibility of LSM in handling input and anytime classification to design a framework for continuous authentication in smartphones. This is because, such a framework requires a learning model with exact capabilities that LSM provides, which are anytime pattern recognition, working in lack of information and time series handling.

This chapter is organized as follows: section 1 describes the motivation behind continuous authentication in smartphones. Section 2 surveys the related work in this context. In section 3, we introduced how we harnessed LSM for this purpose, and we describe the methodology and the suggested framework. Section 4 tests the framework for augmented password authentication, long-text authentication and gestures/strokes authentication.

A. Mobile Continuous Authentication

Smartphones have become ubiquitous devices that offer increasingly high computing power in small affordable packages. Market analysis predicts that in 2015 there will be 1.5 billion smartphones and 640 million tablets in use worldwide [73]. Their increased popularity and growing capabilities have transformed them from simple

communication devices into powerful information hubs that process and store a tremendous amount of private and personal data. Consequently, with their small size and high mobility, smartphones are under constant threat of unauthorized access. The need for robust authentication methods is therefore at an all-time high.

Authentication utilities offered with current smartphones are typically static. Users are prompted to provide proof of identity at login through means of patterns (numerical or graphical) or physiological prints (iris, fingerprints, facial recognition, voice ID, etc.). The first class is vulnerable to spoofing and hacks while the second, while more secure, demands too much user attention. With both means of static authentication, only login credentials are checked, and no session-wide security is provided, making the device and the sensitive information it holds susceptible to all kinds of post login exploits. For these reasons, there is growing demand for a robust continuous authentication system (CAS), which aims to authenticate the user from the initial stages of login till logout; thus checking the identity of the user throughout a session.

Of special interest are Continuous Behavioral Biometric Authentication systems (CBBAS) that have garnered a great deal of recent attention as an emerging research area. Instead of employing physiological methods that explicitly prompt a user for re-authentication on a periodic basis, a CBBAS transparently monitors the activity of the user throughout the session in order to continuously confirm their identity. Only on the occasion that the user's activity is construed as atypical or anomalous, the system demands re-authentication.

Several behavioral biometrics have been proposed in recent years and evaluated for their performance and usability. In what follows, we shed some light on a few stand out efforts before proposing a transparent continuous authentication framework.

B. Related Work

Continuous authentication research on mobile devices is a novel field with a relatively limited number of studies. Relevant literature reveals two popular continuous behavioral authentication techniques: typing dynamics and touch/gesture analysis. Additionally, miscellaneous behavioral metrics, both bio-centric and otherwise, like gait analysis, communication monitoring, location tracking, and others have been proposed to authenticate or supplement more robust authentication metrics. We provide a brief account of these works in what follows with special focus on touch.

Starting with keystroke dynamics, continuous seamless authentication based on the user's typing characteristics was proposed in [74]. Two typical handset interactions are tackled, dialing telephone numbers and typing text messages. In comparison with a kernel-based (RBF) neural network (13.6%) and a generalized regression neural network (GRNN) (13.3%), a feed forward multi-layered perceptron (FF-MLP) achieved an equal error rate (EER) of 12.8% proving the robustness of the implemented system. However, Clarke et al. note the extra computational power required for such a network and the subsequent restrictiveness on its implementation on current mobile handsets. In [73], Feng et al. studied Typing Authentication and Protection (TAP) for two stages, login and post-login. Three user studies, which compare authentication performance under different virtual key typing settings, were evaluated: character input time data (including pressing and flight

time), time and pressure data without haptic feedback; and time and pressure data with haptic feedback. Decision trees, Random Forest, and Bayes net classifiers were used. Leveraging a large number of features, TAP achieved best performance of 1.0% False Acceptance Rate (FAR) and 1.0% False Rejection Rate (FRR) at 40 character long inputs using Random Forest algorithm.

As for touch analysis, the increasing popularity of smartphones coupled with touch being the primary form of interaction has led to recent Continuous Biometric Authentication systems (CBAS) research investigating the discriminative qualities of gestures/strokes and their applicability to continuous authentication. Frank et al. [75] designed a CAS using up to 30 features extracted from users' strokes. Features included are: start and end point coordinates, gesture direction, area covered by finger, etc. Two classifiers, SVM and KNN, were employed to distinguish the client after initial login. The number of strokes was noted to affect the EER recorded where 13% EER was achieved with single strokes versus an EER between 2% and 3% for intersession authentication and between 0% and 4% for inter-week authentication when 11 or 12 strokes were aggregated. The work also concluded that SVM achieved lower EER in comparison with k-NN for all scenarios. Using the public database made available by [75], Govindarajan et al. [76] proposed a novel framework for outsourcing the continuous authentication (CA) of smartphones using secure privacy-preserving protocols. Touch events from 41 users were used to create user profiles on an Android platform. 90% of the data was used for training while the remainder was used for testing. Biometric verification was performed by matching user templates with test templates using scaled Euclidean and scaled Manhattan verifiers. A feature selection phase and an outlier filtering phase were performed to find the

optimally performing feature subset and discard atypical samples respectively. Equal error rates (EER) were computed for both verifiers and while the Manhattan verifier was found to deliver best performance, the EER recorded at 20+% was significantly worse than that reported in [75]. Leveraging touch events similarly while incorporating multi-touch gestures, Zhao et al. [77] proposed Graphic Touch Gesture Feature (GTGF) to extract the identity traits from the touch traces. Six commonly used touch gestures were taken into consideration (flick up/down, flick right/left, zoom in/out). Features extracted consisted of the time duration, the length of touch traces, the directions and speeds of finger movements, and the tactile pressures. Extracted traces were further filtered into one of the six predefined gestures. The proposed method was evaluated for multiple scenarios and achieved best performance with 2.62% and 4.31% EER for combined gestures and single tip gestures respectively, thus joining the above reviewed work in making a solid argument for touch based continuous authentication. Lastly, providing a counterpoint to these approaches, Serwadda et al. [78] aimed to show that a simple Lego robotic arm driven by input gathered from general population swiping statistics can generate forgeries that achieve alarmingly high penetration rates against touch-based authentication systems. The performance evaluation used in the user studies they conducted revolved around a zero-effort threat model in which the adversary is assumed to be unable to pull off a sophisticated forgery. 28 features related to touch positions, touch area, pressure, and others were extracted and fed to a Support Vector Machine (SVM) and a k-Nearest Neighbors (k-NN) classifier. The EER calculated based on classifier output was shown to increase by as much 1009% when user data was tested against robotic arm generated forgeries.

Apart from single module CAS, significant work has been done on multimodal systems. Multi-modal systems have the benefit of recruiting multiple sensors and data sources making them tolerant to hardware failures. Additionally, aggregated features incorporating multiple sources can increase confidence in user identity by resolving ambiguity. The information coming for individual modules can be integrated at four levels: sensor level, feature extraction level, matching score level, and decision level. With data typically hidden at sensor and feature levels, most approaches resort to simple rules (max, sum, and product) that combine matching scores and produce a compound trust or confidence metric. In [79], a DARPA sponsored CAS on a desktop computer was tested with behavior-metrics collected from 99 users over the course of 10 weeks. The metrics included keystroke dynamics, mouse movements, CPU and RAM usage, and processes and applications used. Deutshmann et al. proposed a novel trust metric that controls changes in trust levels based on user scores. These scores are based on user profiles created using fuzzy sets that are later used by a Bayesian network to compare against test data. Experiments on mouse and keyboard data, both individual and combined, were conducted leading the authors to conclude that keyboard dynamics performed best by never falsely rejecting correct users and recognizing imposters in as little as 38 interactions. Azzini et al. [80] followed the fuzzy route as well with a multimodal CAS leveraging face recognition with asynchronous finger print recognition. A fuzzy controller with custom rules was implemented to fuse metrics at the decision level using the match scores of the biometric sub-systems as inputs. Experiments were conducted by monitoring the activity of 100 users during an hour-long session. Multiple membership functions were evaluated with the best choice demanding an average of 8.68 fingerprint scans per hour. Finally, Crawford et al. in

[81] provided a framework for transparent user authentication using a combination of Keystroke Dynamics (KD) and Voice Verification (SV) to calculate the confidence in user identity according to which the client is allowed to take control of the device at three level of confidence. For example, changing the PIN of the device requires a higher confidence level than taking a photo. The authors conducted tests on different devices (iPhone 1,2,3,4 & iPod 1,2,3,4) using Naïve Bayes, Decision Trees, and 5-NN as classifiers and concluded that 5-NN achieved the best EER median with 19.5% for KD and 28.54% for SV. Crawford also showed the results of testing multimodal system using either Naïve Method or Posterior Probability Method (PPM) to combine KD and SV reporting that PPM is the best method with 32.84% EER median, but the single biometric system using KD is still competing multi-biometric and single SV with 29.52 EER median for overall classifiers.

C. LSM-based Continuous Authentication Framework

In chapter VI, we showed how we can use LSM for feature extraction and online feature extraction from EEG raw data For continuous authentication for smartphones, the environment is similar for two reasons; first, LSM can be used for feature extraction directly from the sensors' output; second features will be available automatically as long as there is data collected from the sensors. We try in this framework to reduce the preprocessing of data from sensors such that we obtain a real anytime authentication framework. In the next section, we describe the type of information that we can obtain from mobile sensors for keystrokes and gestures/strokes. In our design, the framework provides authentication for three components in smartphone; first, augmented password authentication when a user enters his/her password; second, when a user enters a long text

such as SMS, email, MMS, etc.; finally; strokes/gestures authentication when a user swipes his/her finger(s) on touchscreen. The first and second components share the same features, but differ in the length, while the third component has different features.

1. Mobile Sensor Data

We target Android based devices for our framework and built our own application to collect user's data interactions and logs. For keystroke, we collect the following information for each user:

- The amount of pressure on the touch screen when pressing each character.
- The area of the finger when pressing each character.
- The time of pressing each character.
- The time of releasing each character.

While for strokes/gestures component, we collect the following features:

- Time stamps for stroke/gesture (multiple recordings per one continuous swiping).
- X-coordinates stroke/gesture (multiple recordings per one continuous swiping).
- Y-coordinates stroke/gesture (multiple recordings per one continuous swiping).
- Velocity on X-coordinates (multiple recordings per one continuous swiping).
- Velocity on Y-coordinates (multiple recordings per one continuous swiping).
- The amount of pressure on the touchscreen when swiping (multiple recordings per one continuous swiping).
- The area of the finger when swiping on touchscreen (multiple recordings per one continuous swiping).

2. LSM-Based Framework

In this section, we introduce our framework base on the previous discussion. As previously mentioned, we want to reduce the amount of preprocessing information before the LSM. It can be noticed that the timing of pressing and releasing characters for keystroke can be fed directly as a spike into the LSM as spikes. On the other hand, we can feed other information such as the pressure and size of a finger on the touchscreen using direct feeding as we have done in experiment 1, chapter VII. However, due to some constraints on the simulator that we will use, csim, we have to build two LSM and connect them with each other (CbNeuron model accepts direct input, while LIF accepts spike input). The first LSM, LSM 1, consists of LIF neurons, which accepts spike encoding. The second LSM, LSM2, consists of CbNeuron model neurons and accepts direct information feeding by mean of an analog input neuron. We connect the two LSMs with each other and read from LSM 1. That is, we use LSM 1 as an output for the framework, where this output holds an accumulative information from input 1 to input 7. The information flow is depicted in Figure 17.

Since the keystroke does not include information about velocity and coordinates, we need to have before the main LSM, i.e., the LSM that includes LSM 1 and LSM 2, an input interfacing component such that we map the input for the corresponding point inside the main LSM. The main LSM has 7 inputs as follows:

Table 27: Input description for LSM-based continuous authentication.

Input	Function	Type	Connected to	Used by
Input #1	Deliver pressing time as spikes.	Spiking	LSM 1	Keystrokes and gestures
Input #2	Deliver releasing time as spikes.	Spiking	LSM 1	Keystrokes and gestures
Input #3	Deliver average pressure on the touchscreen.	Analog	LSM 2	Keystrokes and gestures
Input #4	Deliver average finger size on the touchscreen.	Analog	LSM 2	Keystrokes and gestures
Input #5	Deliver Y-coordinates of gestures/strokes on the touchscreen	Analog	LSM 2	Gestures only
Input #6	Deliver Y-coordinates of gestures/strokes on the touchscreen	Analog	LSM 2	Gestures only
Input #7	Deliver velocity of gestures/strokes on the touchscreen	Analog	LSM 2	Gestures only

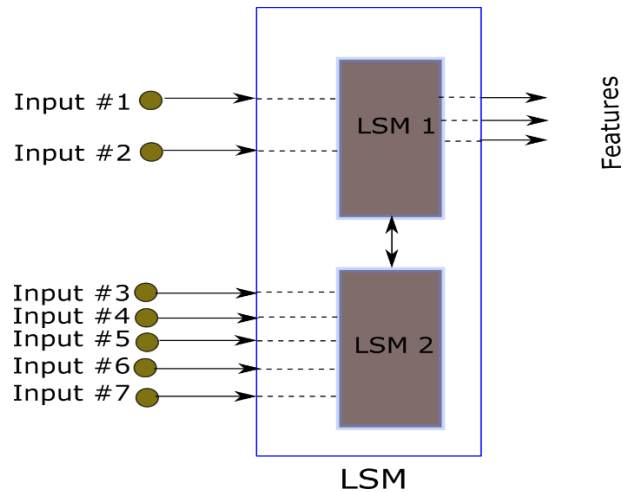


Figure 18: Main LSM architecture for continuous authentication.

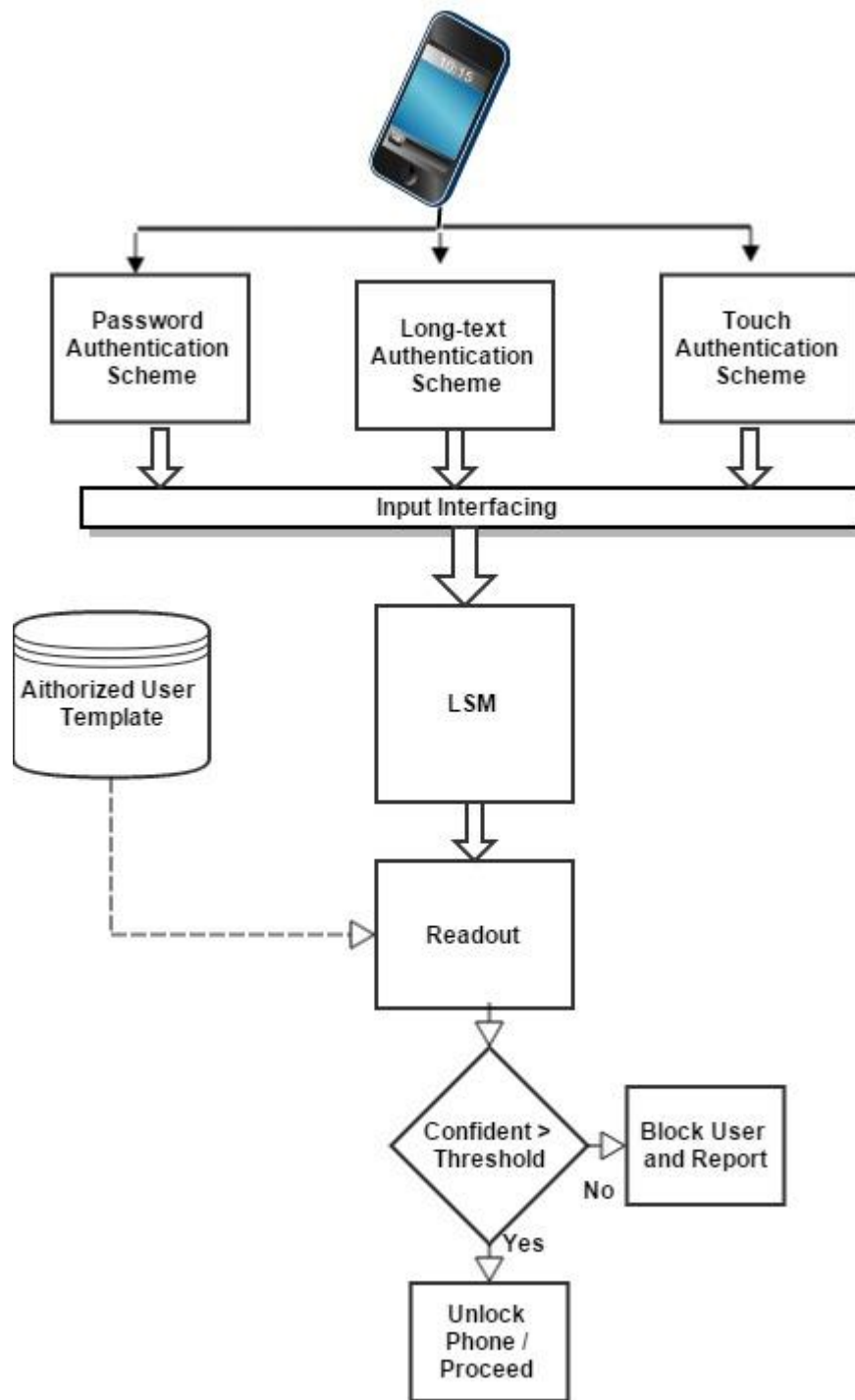


Figure 19: LSM-Based Continuous Authentication Framework for Smartphones.

D. Feature Selection

Having reviewed the related work in continuous mobile authentication, we choose from the related work the best reported feature vector for keystroke and gestures/strokes. More specifically, we will use the same feature vector information in related work for comparison purposes with the features extracted by the suggested architecture using LSM.

1. Keystroke features

According to the works [82, 83], a heterogeneous vector of the following components would deliver the highest performance: p_i , the average pressure of character i on touchscreen; s_i , the average size of finger on touchscreen for character i ; h_i , the hold time for character i ; f_i , the flight time, which is the difference between pressing character $i + 1$ and releasing character $i + 1$; pr_i , the relative pressing time for character i ; and r_i , the releasing time for character i . The feature vector is represented as follows:

$$v = \{p_1, p_2, \dots, p_N, s_1, s_2, \dots, s_N, h_1, h_2, \dots, h_N, f_1, f_2, \dots, f_N, pr_1, pr_2, \dots, pr_N, r_1, r_2, \dots, r_N\} \quad (8.1)$$

E. Experimental Procedure

1. Application side configuration

To evaluate the quality of features from LSM, we decided to collect our own dataset to ensure the accessibility to data from mobile logs. To do this, first we designed an Android Application that collects and logs user interactions with a smartphone. The

application was installed on Samsung GT-i9100 smartphone and we asked 22 users to participate in data collection. All users are students between 20 to 28 years with 40% of them are males. Each user was asked to perform a three-step data collection.

In the first step, the user was asked to enter a fixed password for 15 times. We chose “.tie5Roanl” as a password to compare our results with the available works on augmented passwords [84].

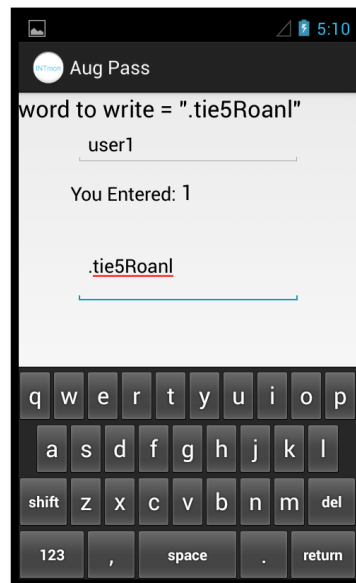


Figure 20: A screenshot of the application interface for Augmented Password Authentication.

In the second step, the user was asked to enter the following text using the smartphone keyboard.

“”

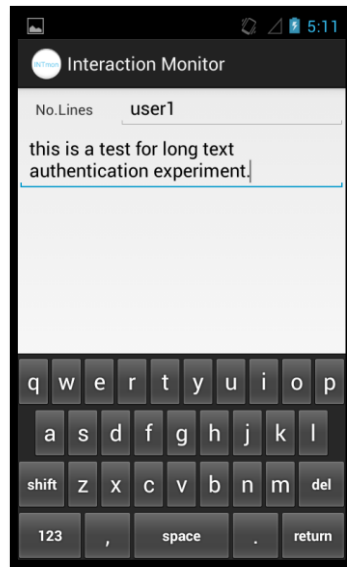


Figure 21: A screenshot of the application interface for Long-text Authentication.

The third step includes data collection for strokes/gestures, where the user was asked to swipe anywhere on the screen in specific direction according to a label that appears on the smartphone touchscreen. In total, we have 120 gestures/strokes with different direction; up, down, left and right.

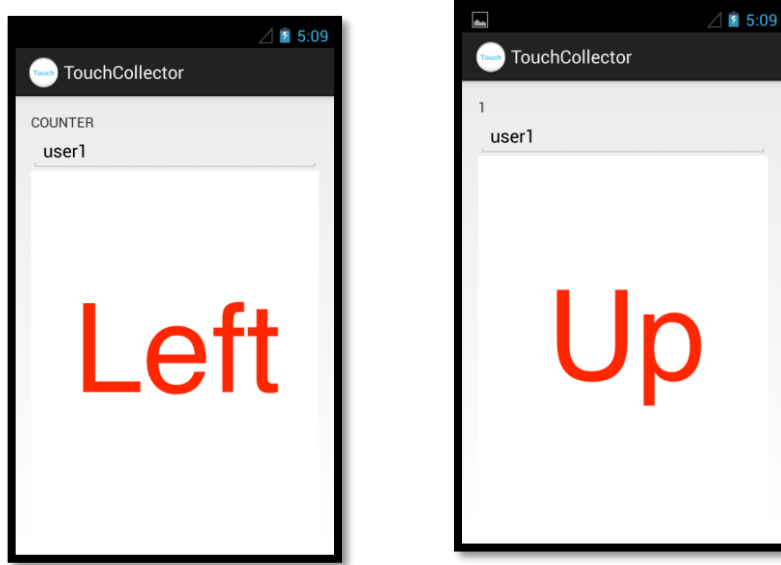


Figure 22: A screenshot of the application interface for Strokes Authentication.

2. LSM side configuration

The following tables show the configuration for LSM 1 and LSM 2.

Neuron Configuration	LSM Connectivity and Architecture	Readout
Model: LIF V_{thresh} = 0.015 (V) V_{reset} = 0.01447 (V) T_{refract} = 0.002 (sec) C_m = 3e-08 (F) V_{resting} = 0 (V) V_{init} = .0146533 (V) V_m : 0 (V)	Architecture : 6×6×6 $\lambda = 2$ $w_{\text{input}} = 0.15$ $C_{\text{scaling}} = 1$	Sampling: [0.01:0.025:T] With T=2 sec for Augmented password authentication, 200 seconds for long-text authentication and 0.5 second for Gestures/strokes authentication. Filter : exponential filter with $\tau = 0.03$

Table 28: LSM 1 configuration.

Table 29: LSM 2 configuration.

Neuron Configuration	LSM Connectivity and Architecture	Readout
Model: CbNeuron $V_{\text{thresh}} = -0.045$ (V) $V_{\text{reset}} = 0$ (V) $T_{\text{refract}} = 0.003$ (sec) $C_m = 3e-08$ (F) $R_m = 1e+06$ (Ohm) $E_m : 1.0395e-314$ (V) $V_{\text{resting}} = -0.06$ (V) $V_{\text{init}} = -0.06$ (V) $V_m : 1.0395e-314$ (V)	Architecture : $5 \times 5 \times 5$ neurons $\lambda=2$ $w_{\text{input}}=0.4$ $C_{\text{scaling}}=1$	No readout applied

F. Anomaly Detection methods

The most intuitive way to evaluate continuous authentication in smartphones is to use anomaly detection approach. Differently from the published works that deploy classification to evaluate continuous authentication, anomaly detection is a more natural approach to address this problem because it is not reasonable to build a database for all possible non authorized users in a supervised environment. That is, classification requires the security system to specify the genuine user's data and other data from non-genuine users and hence the classification model will be limited and not generalized for other yet to be see data from non-genuine users. This is an anomaly detection problem.

In this section, we review the available methods to perform anomaly detection. Most of the methods were taken from [84].

Let us first assume the following variables to ease the explanation. Let m be the mean of the training data (the genuine user data), x_i is a sample from training data, y_i is a sample from testing data and sd is the standard deviation of the training data.

1. Euclidean Detector

The idea here is simple and it relies on the Euclidean distance. Specifically, this method calculates the mean of training data m and then calculates the scores of the testing data by finding the Euclidean distance d_i between the mean and each sample in the testing data.

$$score_i = d_i \quad (8.2)$$

2. Normalized Minimum Distance Classifier

This method is similar to the Euclidean one, however it differs in the way of calculating the testing score. The score of each test sample is calculated by normalizing the Euclidean distance between testing sample y_i and the mean m by the norm of the mean m and y_i .

$$score_i = \frac{d_i}{\|m\| * \|y_i\|} \quad (8.3)$$

3. Manhattan method

This method is also similar to the Euclidean method; however, it uses the Manhattan distance instead to calculate the distances between the mean of training data m and samples of testing data.

$$score_i = d_i \quad (8.4)$$

With d_i is the Manhattan distance between y_i and m .

4. Filtered Manhattan

Here, the method is similar to the Manhattan method, but it tries to repair the training data by ignoring the samples that are three or more standard deviations from the mean. After finding those samples in the training data, the method then drops them and recalculates the new mean. The scores of testing samples are calculated according to the new mean using Manhattan distance.

5. Scaled Manhattan

This method uses the same methodology as in Manhattan method, but it scales each feature in the testing sample by the mean absolute deviation of each feature s_p , where s_p is calculated in the training phase by finding the mean of the differences between each feature in training samples and the mean m .

6. Outlier-counting (z-score)

This method calculates the mean and the standard deviation of each feature in the training phase. In the testing phase, the method calculates the z-score for each feature in the testing sample. After that, the score of testing sample is calculated by finding how many features exceeded a predefined threshold. The threshold here is experimental value and can be tuned by trial.

7. One-Class SVM

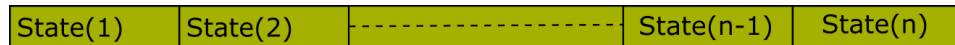
This method aims at finding a decision boundary around the training data. One-Class SVM uses the same concept of the kernel method where it maps the data into high

dimensional data. The scores here are computed by finding the distance between the decision boundary and testing samples.

G. LSM Readout Optimization for Distance-based Anomaly Detection Method

Since anomaly detection methods depend on evaluating the sampling according to their distances to the mean of training data in general, then improving distances such that the samples from the same user become closer and samples from different users become farther.

The suggested solution is to concatenate resultant states after LSM in one vector. That is, we merge states horizontally after LSM to form a long vector. Even though this idea seems simple; however, it showed great improvements in accuracy for anomaly detection methods.



Concatenated Feature Vector

Figure 23: Liquid state concatenation for improving distance-based anomaly detection.

A. Results of testing continuous authentication in smartphones

In this section, we test the proposed framework for continuous authentication in smartphones. We test each component in the framework independently, i.e., we report the result of testing augmented password authentication, long-text authentication and gestures/strokes authentication. For augmented password authentication, we report the results of testing anomaly detection on best reported feature vector form literature review

(see section D), using LSM feature vector and using a concatenated liquid state feature vector described in section G. All results are reported for one vs. one scenario and one vs. all scenarios for each type of feature vector.

For long-text and gestures/strokes authentication, we report the results for liquid states feature vector and a concatenated liquid states feature vector, since they need further works that are out of the scope of this thesis.

In all tests, the results are reported in Equal Error Rate (EER) form after averaging over all 22 users participated in the data collection. The values of EER are in %, i.e., 50 is 50%.

1. Augmented Password Authentication

Table 30: Augmented Password authentication error rate (One vs. One scenario).

	Literature review feature vector	LSM feature vector	Concatenated feature vector
Euclidean Detector	<u>1.71055</u>	46.50597	3.05931
Normalized Minimum Distance	<u>1.417414</u>	49.65097	4.015016
Manhattan method	<u>1.586273</u>	45.12553	2.668668
Filtered Manhattan	1.673142	41.55802	<u>0.058583</u>
Scaled Manhattan	<u>0</u>	15.9788	<u>0</u>
Outlier-counting (z-score)	5.625655	35.08592	<u>0.03243</u>

Table 31: Augmented Password authentication error rate (One vs. All scenarios).

	Literature review feature vector	LSM feature vector	Concatenated feature vector
Euclidean Detector	13.36383	49.50233	<u>12.78469</u>
Normalized Minimum Distance	13.92221	49.82517	16.20212
Manhattan method	13.68596	48.76225	<u>5.519532</u>
Filtered Manhattan	13.26725	44.59739	<u>0</u>
Scaled Manhattan	0	17.07342	<u>0</u>
Outlier-counting (z-score)	14.7841	34.68849	<u>0</u>

2. Long-Text Authentication

Table 32: Long-Text authentication error rate (One vs. One scenarios).

	LSM feature vector	Concatenated feature vector
Euclidean Detector	47.70873	<u>0.585811</u>
Normalized Minimum Distance	50	<u>0.855847</u>
Manhattan method	47.2924	<u>0.752246</u>
Filtered Manhattan	45.24488	<u>0.00086</u>
Scaled Manhattan	36.03566	<u>0.329986</u>
Outlier-counting (z-score)	45.55512	<u>0.52056</u>

Table 33: Long-Text authentication error rate (One vs. All scenarios).

	LSM feature vector	Concatenated feature vector
Euclidean Detector	50.53207	<u>1.59801</u>
Normalized Minimum Distance	50	<u>2.021999</u>
Manhattan method	49.97733	<u>2.08626</u>
Filtered Manhattan	47.90583	<u>0.0168</u>
Scaled Manhattan	37.43904	<u>1.214364</u>
Outlier-counting (z-score)	45.34701	<u>0.6828</u>

3. Gestures/strokes Authentication

Table 34: Gestures/Strokes authentication error rate (One vs. One scenario).

	LSM feature vector	Concatenated feature vector
Euclidean Detector	40.64681	<u>2.75268</u>
Normalized Minimum Distance	50	<u>2.862201</u>
Manhattan method	40.32027	<u>2.265007</u>
Filtered Manhattan	33.84939	<u>0.741661</u>
Scaled Manhattan	9.409299	<u>0.006753</u>
Outlier-counting (z-score)	31.38148	<u>0.528848</u>

Table 35: Gestures/Strokes authentication error rate (One vs. All scenarios).

	LSM feature vector	Concatenated feature vector
Euclidean Detector	49.28858	<u>8.127847</u>
Normalized Minimum Distance	50	<u>8.319069</u>
Manhattan method	46.08879	<u>6.674691</u>
Filtered Manhattan	38.27089	<u>2.543448</u>
Scaled Manhattan	11.25073	<u>0.058676</u>
Outlier-counting (z-score)	34.24807	<u>8.127847</u>

B. Results Discussion

The results show that LSM provides a better discriminative feature vector than the best reported feature vector in the literature review except for some methods. For Filtered Manhattan, Scaled Manhattan and Outlier-counting (z-score), a concatenated states feature vector always performs better than literature review feature vector (for augmented password authentication). For long-text and stroke authentication, the concatenated feature vector produces a very low EER.

In addition, LSM provides an online method and natural environment to handle continuous authentication via continuous feature extraction. Moreover, we showed that LSM can be considered as a universal-model to handle signals from different sources of signals, keystrokes and gestures. If we are to build the same model using techniques other than LSM, then we have to build a different model for each different component in the framework.

CHAPTER IX

EFFICIENT SAMPLING TIME SELECTION FROM LIQUID STATE MACHINE

A. Introduction

LSM works by sampling states of a liquid during the course of its work. These states are used later by different readout functions to identify the corresponding patterns, i.e., the extracted states from the same input pattern are assigned to the same label or class accordingly. These states hold information about the liquid state at each sampling time. However, LSM at some periods or at specific sampling times may hold more or less information about input. That is, some states are more informative than others with respect to when it is sampled. In this chapter, we introduce a method to choose the most informative sampling times to improve the classification task. This chapter also uses the DEAP dataset for testing purposes and specifically it will use the Subject/Video independent scenario. In addition, this chapter uses the proposed method to study how our brain acts with respect to the valence and arousal.

This chapter is organized as follows: in section B, we introduce the proposed method mathematically. Section C provides testing and results for applying this method. In section D, we devise the method to study the course of valence and arousal in humans, where we show the benefit from using this method not only for states selection, but also in evaluating states for further analyses.

B. Active States Detection Method

In this section, we will use some statistics to evaluate states after the liquid to rank them accordingly. Then, we will choose the top ranked states to see how they will improve the classification task.

Let $s = \{s_1, s_2, s_3 \dots s_n\}$ be liquid states values, where n is number of sampling times and $t = \{t_1, t_2, t_3 \dots t_n\}$ are moments of sampling states. Among t , we want to choose the k highest sampling periods such that we maintain from s the k top informative liquid states. To achieve this goal, we will use t-test method as an approach to perform this goal. This idea is inspired from work [85] which detects some unique markers molecules that are found in the sera of the ovarian cancer patients to use them in nearest neighbor classifier (Adapting the method for LSM is a novel idea and has not been used for LSM before). This is a challenging problem since molecules differ significantly from a person to another; however, the method presents a statistical way to choose the k -best discriminative markers molecules to differentiate between patient and healthy persons. The method has been shown to be very effective for this problem with 100% discrimination.

Choosing the effective sampling time can be seen from the same perspective, where each sampling time can be considered as a marker molecule. In the next section, we reformulate the method such that can be applied on sampling time selection. The problem formulation follows the same mathematical derivation in [86].

The goal of this method is to test for the zero differences between $(u_1 - u_2)$ where u_1 is the mean of the values of the liquid state at $t_j, j = 1, 2, \dots, N$, for the class ω_1 and u_2

is the mean for the same feature when it is taking for the class ω_2 . Let us now assume x_i , $i=1, 2, \dots, N$, be the state values of the period t_j in class ω_1 with mean u_1 . Similarly, let y_i , $i=1, 2, \dots, N$, be the state values of the same period, t_j , in class ω_2 with mean u_2 . For the variance, we assume that the variance for $x_i = \sigma_1^2$ and the variance for $y_i = \sigma_2^2$ are equal; $\sigma_1^2 = \sigma_2^2 = \sigma^2$.

Let us build the hypothesis that test the closeness between the two mean as follows:

$$H_1: \Delta u = u_1 - u_2 \neq 0 \quad (9.1)$$

$$H_0: \Delta u = u_1 - u_2 = 0 \quad (9.2)$$

Now, let us assume that $z = x - y$

Where x, y denote the random variable corresponding to the values of the liquid states in the two classes ω_1 and ω_2 , respectively.

Under the assumption of the statistical independence between x and y , we can write:

$$E[z] = u_1 - u_2 \text{ with } \sigma_z^2 = 2\sigma^2 \quad (9.3)$$

Then the estimation of the mean of the random variable z is:

$$\tilde{z} = \frac{1}{N} \sum_{i=1}^N (x_i - y_i) = \tilde{x} - \tilde{y} \quad (9.4)$$

Here, we assume that the variance and means are known and that: $\tilde{z} \sim N(u_1 - u_2, \frac{2\sigma^2}{N})$. However, if the variance is not known (the mean should not be a problem because \tilde{z} is consistent and unbiased estimator), then the test statistics is required as follows:

$$q = \frac{(\tilde{x} - \tilde{y}) - (u_1 - u_2)}{s_z \sqrt{\frac{2}{N}}} \quad (9.5)$$

Where

$$s_z^2 = \frac{1}{2N-2} (\sum_{i=1}^N (x_i - \tilde{x})^2 + \sum_{i=1}^N (y_i - \tilde{y})^2) \quad (9.6)$$

It can be shown that $\frac{s_z^2(2N-2)}{\sigma^2}$ follows a Chi-square distribution with $2N - 2$ degree of freedom.

For the time being, we assume the availability of the variance and mean in the further steps. The test statistic value for the most important periods or sampling times will be formulated as follows:

$$r_i = \frac{(x_i - y_i)}{\sqrt{\frac{\sigma_{x_i}^2}{N_{x_i}} + \frac{\sigma_{y_i}^2}{N_{y_i}}}} \quad (9.7)$$

Where r_i is the ranking for sampling period t_i , $i=1,2, \dots, N$, and N_{x_i} is the number of values from liquid state s_i and belongs to class ω_1 . N_{y_i} is the number of values from liquid state s_i and belongs to class ω_2 . x_i, y_i and $\sigma_{x_i}^2, \sigma_{y_i}^2$ are the means and variances for liquid state s_i values that correspond to class ω_1 and ω_2 , respectively.

We will use this metric to rank the sampling periods s and then choose the top k sampling periods.

C. Testing

As aforementioned, the experiment will use Subject/Video independent scenario to assess the method. Moreover, the method later will be used to study the valence and arousal courses of human brain.

For the configuration, the same sampling time that was used in Emotion Recognition in Chapter VI is used in this experiment; [0.5: 0.4:59], which means that our sampling starts after the 0.5 second and we sample every 0.5 seconds until the moment the 59 seconds. The total number of the produced states for each input pattern (the EEG for each different video) is $N = 147$ states.

In testing, we compare the results in two cases for different configurations of k . The first case uses the selection method to choose top ranked k states. The second case uses a random selection for k states. To increase the reliability of the second case, we repeat the random selection for 20 times, and then we average the results accordingly. We choose $k = [5\ 10\ 20\ 40\ 60\ 80\ 100\ 120]$. The readout functions are Decision Trees and Linear Regression. All experiments are done in 10-fold cross validation form.

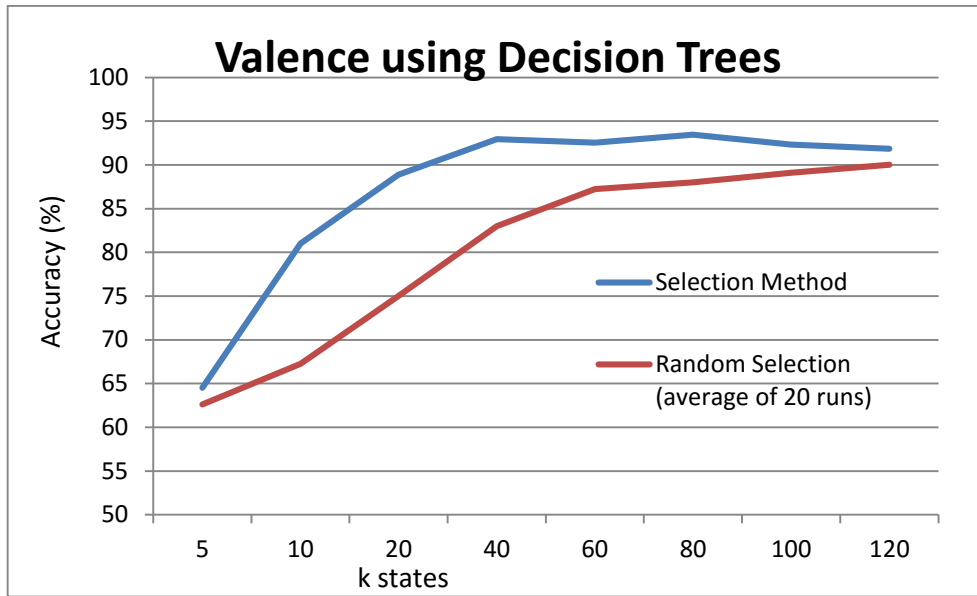


Figure 24: Active State Selection Method testing on Valence using Decision Trees.

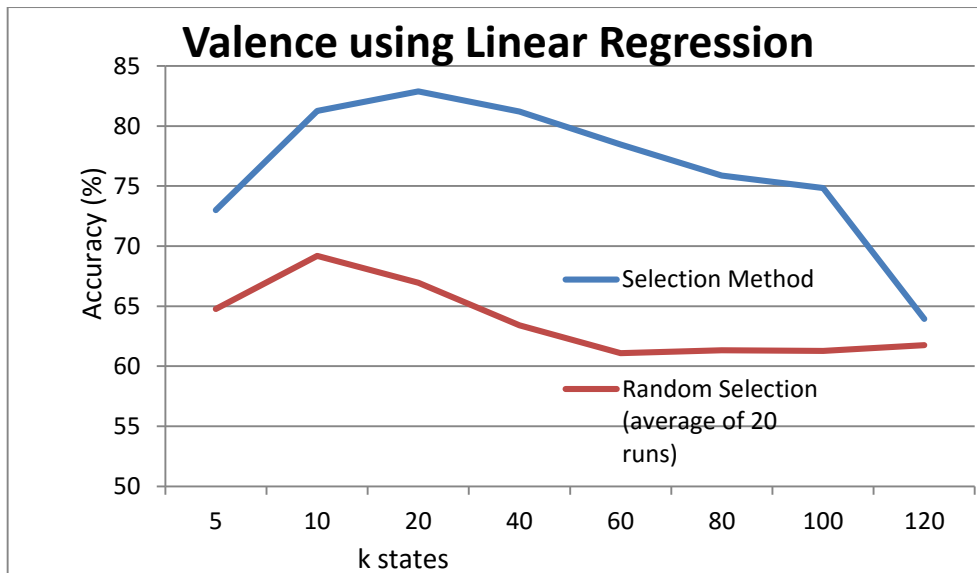


Figure 25: Active State Selection Method testing on Valence using Linear Regression.

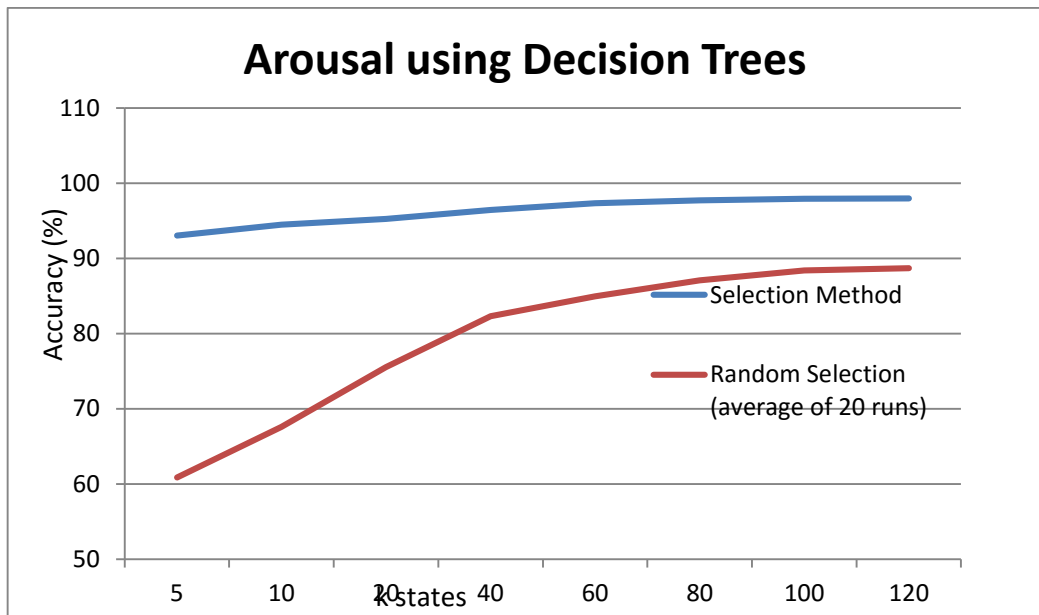


Figure 26: Active State Selection Method testing on Arousal using Decision Trees.

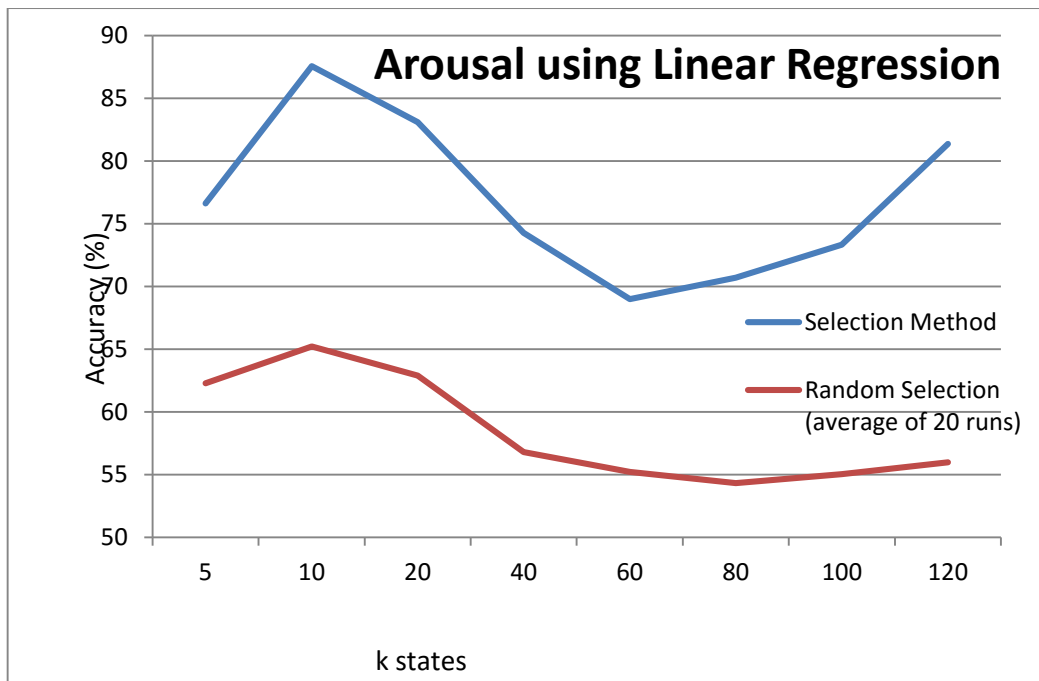


Figure 27: Active State Selection Method testing on Arousal using Linear Regression.

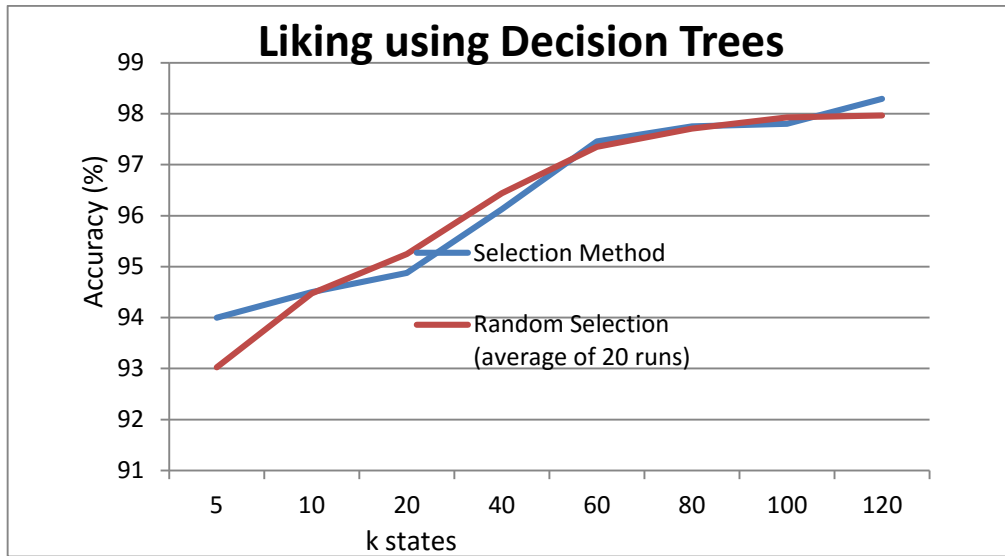


Figure 28: Active State Selection Method testing on Liking using Decision Trees.

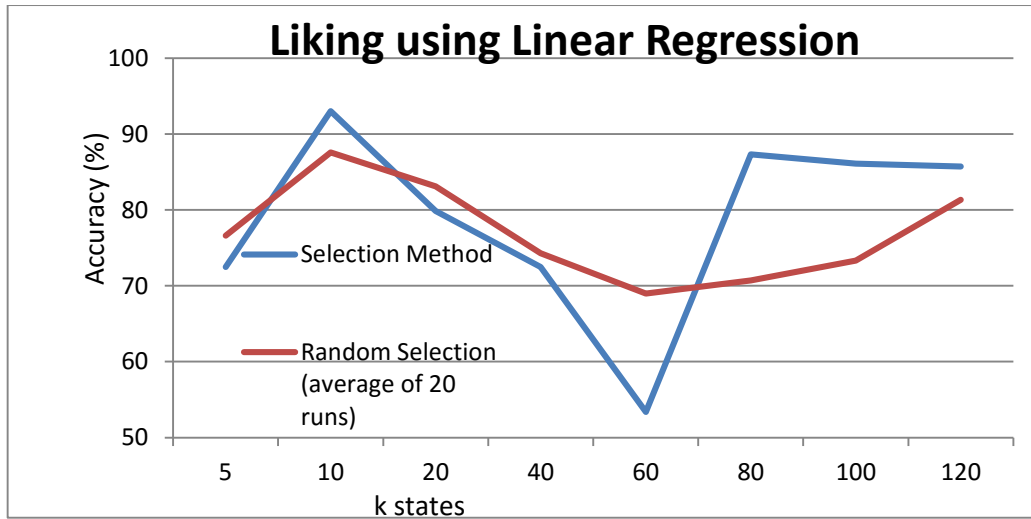


Figure 29: Active State Selection Method testing on Valence using Linear Regression.

D. Results and Discussion

The results show that the selection performs better than any random selection for the states except for liking classification case with Linear Regression readout, where it generated relatively lower accuracies than random selection for some configurations for k.

In addition, the method reduces the computational cost of processing extra and useless liquid states from LSM. For example, For example, with about 10% of the whole extracted liquid states from LSM, we were able to deliver the better accuracy for valence and arousal than using the entire liquid states in readout.

E. The Relation between Sampling Times Ranks and the Affective States of the Brain

This section examines the proposed method to study the valence and arousal courses during a stimulus by studying the ranks of each sampling time. That is, we want to see how the valence and arousal vary during one stimulus with respect to the targeted classification task. In the following figures, we show the ranks of the sampling times for the valence from the users 1-8 of DEAP dataset. The Figures for the remaining users (9-32) are in the Appendix of this chapter.

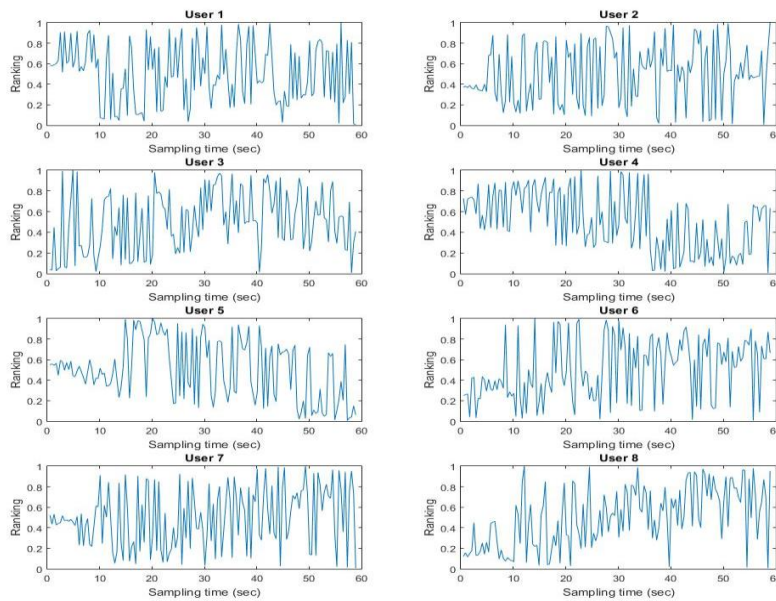


Figure 30: Valence Sampling Times Ranking for Users 1-8.

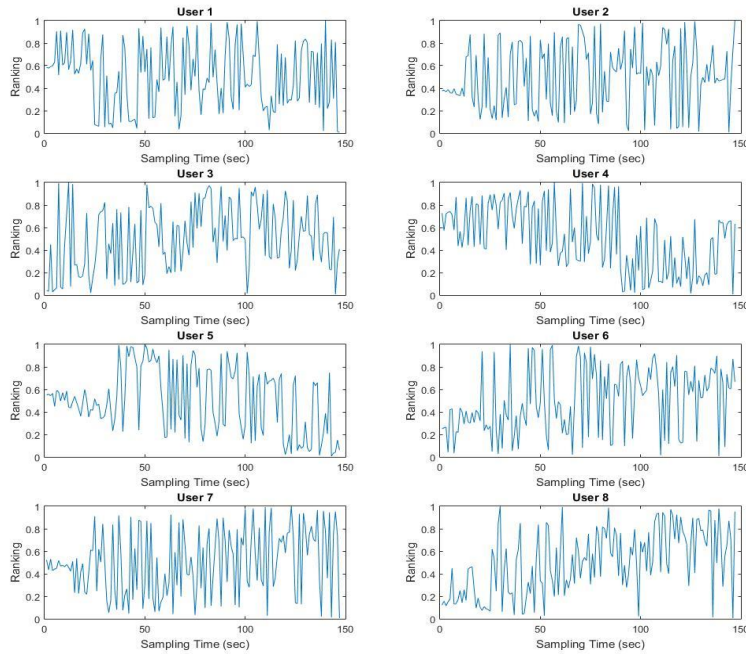


Figure 31: Arousal Sampling Times Ranking for Users 1-8.

1. Discussion

In this section we discuss the outcome of the selection method in order to draw some conclusions about the valence and arousal courses during a stimulus.

At first glance, we can notice some specific patterns of valence for some users. For example, user 4, user 22 and user 30 have the same valence course, where the high ranking states for valence are concentrated before the first 33 seconds. Those users, in fact, are female subjects in DEAP dataset, according to DEAP dataset manual.

Secondly, we cluster the ranking information from each user into two subsets using k-means algorithm. The clustering information shows that valence course between female

differ from those of males; out of the 32 users, 22 users were categorized correctly into either male or female. The following table shows the clustering result of the 32 users.

Table 36: Clustering results for the outcome of the selection method.

	Predicted	Actual
User 1	Male	Male
User 2	Female	Female
User 3	<u>Male</u>	<u>Female</u>
User 4	Female	Female
User 5	Male	Male
User 6	Male	Male
User 7	Male	Male
User 8	Female	Female
User 9	Female	Female
User 10	Female	Female
User 11	Female	Female
User 12	Male	Male
User 13	Female	Female
User 14	Female	Female
User 15	<u>Male</u>	<u>Female</u>
User 16	Male	Male
User 17	<u>Female</u>	<u>Male</u>
User 18	<u>Female</u>	<u>Male</u>
User 19	Male	Male
User 20	<u>Female</u>	<u>Male</u>
User 21	<u>Female</u>	<u>Male</u>
User 22	<u>Male</u>	<u>Female</u>
User 23	Male	Male
User 24	<u>Male</u>	<u>Female</u>
User 25	Female	Female
User 26	Male	Male
User 27	Male	Male
User 28	<u>Female</u>	<u>Male</u>
User 29	Male	Male
User 30	Male	Male
User 31	Female	Female
User 32	<u>Male</u>	<u>Female</u>

Testing arousal did not show interesting result with respect to the gender.

It can be seen from plotting of the sampling times ranking for valence and arousal (Figure 30 and 31) that valence and arousal are not steady nor have flat responses; actually they fluctuate over time. That is, the consecutive periods of time include high/ low activeness of valence and arousal courses. To examine this fact, we conducted an independent experiment, where we increased the sampling times to the sampling time of the EEG signals in the DEAP dataset, 128 Hz. This produced 7552 sampling periods. We were able to perform this experiment for only Subject one, since it includes high computational efforts.

It can be seen from the Figure 32 that active valence periods (the dark blue vertical lines) happen as bursts. Figure 33 shows that arousal course within the same period (one second) for Subject one. It can be noticed that the active periods of valence and arousal courses happen at different sampling times except for some periods as depicted in Figure 34.

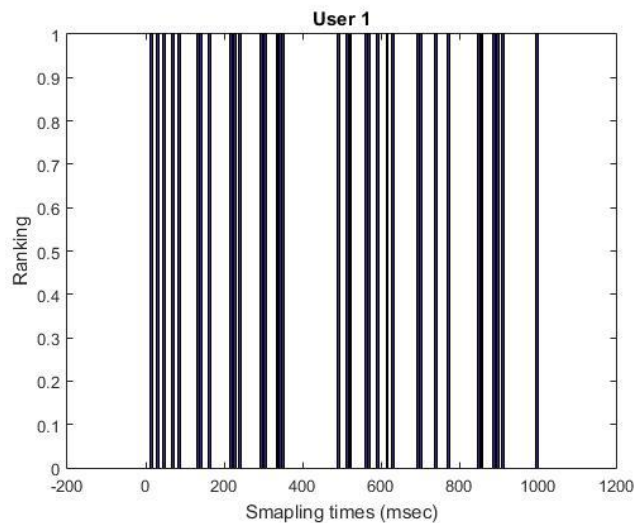


Figure 32: The Sampling times ranking for one second from Subject one (Valence).

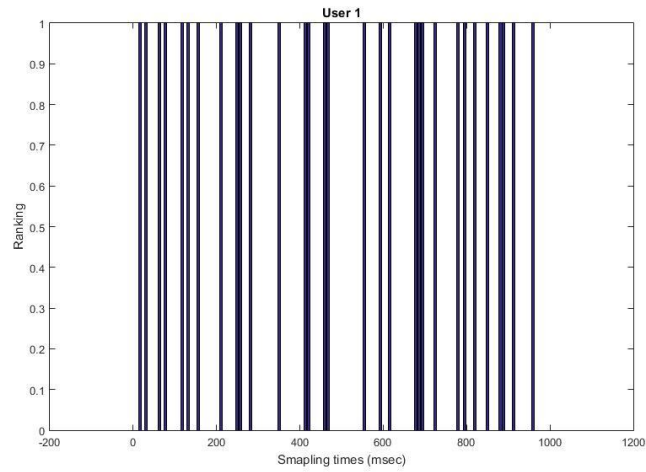


Figure 33: The Sampling times ranking for one second from Subject one (Arousal).

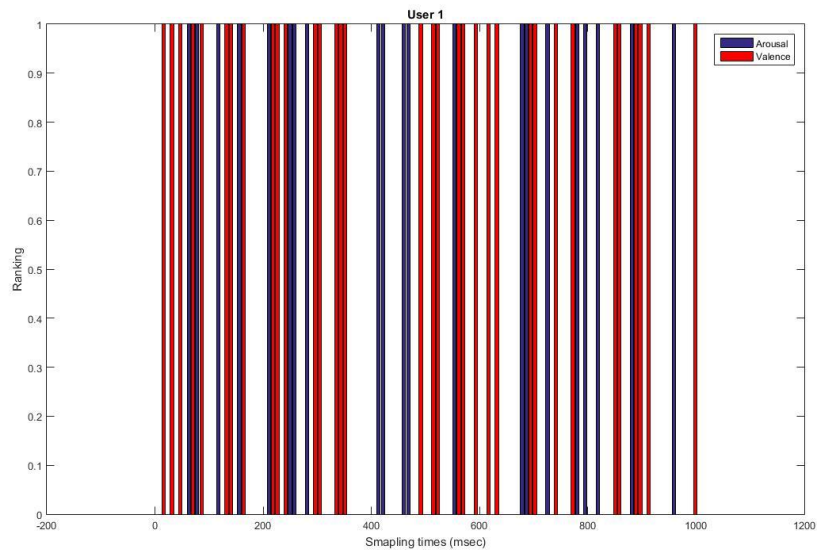


Figure 34: A Comparison for valence and arousal Courses from Subject 1.

F. Conclusion

In this chapter, we introduced a method to rank the sampling times in a way to maintain the top informative sampling times. We showed in the experimental part how this method can improve the performance of a classification task. This is important not only for

the performance of a classifier, but also for reducing the power consumption for a hardware-based implementation of LSM. The power saving is a result of reducing the number of extracted liquid states/ samples from LSM.

In addition, we used the proposed method to study the valence and arousal courses in brain, where we drew some conclusions about their working course. Other analyses could be done based on the outcome of the selection method, where we open this for future work.

CHAPTER X

PRUNING INATTENTIVE NEURONS IN LIQUID STATE MACHINE

This chapter is concerned to look into an energy aware computing LSM architecture. LSM consists of 3D architecture of randomly connected neurons that follows the small world connectivity described in chapter III. During information propagation inside LSM some neurons become lazy, while others become more informative with respect to the targeted task. This chapter presents a method to evaluate neurons during information propagation by studying the probability of binding information between them. Some neurons in LSM fire in harmony with irrespective to an input, because they form isolated islands inside the liquid. We refer to this phenomenon as “isolated islands” and the neurons that participate in this phenomenon as an “inattentive neurons”. The isolated islands could happen because of pathological paths or stratification phenomenon in LSM. Pruning such inattentive neurons increases the performance in both classification and anomaly detection tasks, since each neuron represents a feature in terms of pattern recognition. Moreover, removing such neurons is important to reduce the power consumption, since the recent trends in RC target building some hardware-based LSMs [87-89].

This chapter is organized as follows: in section A, we introduce a mathematical formulation for pruning inattentive neurons. Section B tests the suggested method on

continuous authentication in smartphone data and on five datasets from UCI. In section C, we provide our discussion and analysis of the results. Finally, we deliver a conclusion of this chapter in section D.

A. Inattentive Neurons Pruning (INP) Method

This section introduces the INP method. First, we put the requirements and goals form such a method. The INP method should meet the following requirements:

- The method should be able to capture the similarity in functioning between the neurons.
- The method should be able to understand the connectivity between neurons and finds a way to define the isolated neurons (inactive neurons).
- The method should be able to provide a consistent ranking for the inattentiveness of each neuron, such that any later evaluation for the LSM will be based on a systemic procedure.

Based on the previous requirements, we suggest a method to evaluate the inattentiveness of the neurons inside the LSM based on Stochastic Outlier Selection Algorithm [90]. Even though this method has been used mainly for outlier selection, we slightly modify this method to rank neurons according their behavior in propagating the information.

1. Mathematical Representation

Let $R = [r_1, r_2, r_3, \dots, r_n]$ be the responses from neurons in LSM along simulation time, where n the number of neurons inside the liquid. Let D be the dissimilarity between R , where the dissimilarity between response r_i and r_j is computed as follows:

$$d_{ij} = \sqrt{\sum_{k=1}^m (r_{jk} - r_{ik})^2} \quad (10.1)$$

With r_{ik} is the k^{th} liquid state value of the i^{th} response.

After computing the dissimilarity between neurons responses, we compute the affinity between a neuron's responses to another neuron's response. The affinity that neuron i response has with neuron j response given d_{ij} is:

$$a_{ij} = \begin{cases} e^{\left(\frac{-d_{ij}}{2\sigma_i^2}\right)} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \quad (10.2)$$

Where σ_i^2 is the variance associated with r_i .

To that end, we computed the affinity between neurons responses. Now we define the perplexity parameter, h , which defines the number of the affected neurons when considering the current neuron. The perplexity parameter is similar to the parameter k in k -nearest neighbor algorithm; however, it plays a smoothing role since the affinity is a relative relationship. That is, the affinity has an exponential relationship with respect to the dissimilarity and variance and hence "being a neighbor" is a relative relationship.

After determining h , we want each neuron to have the same number of affected neurons around it. This can be done by controlling the variance of neuron's response such that the variance yields the same number of affected neurons. The variance for each neuron is found by the binary search.

Now, we use graph theory to define the Stochastic Neighbor Graph based on the affinity measure, where the vertices are the responses of neurons. Generating a direct relation between vertices depends on the binding probability concept. The later, depends on the functional relationship between neurons. The binding probability, b_{ij} , between two vertices v_i and v_j is a proportional to the affinity between r_j and r_i .

$$b_{ij} = p(i \rightarrow j) \propto a_{ij} \quad (10.3)$$

and can be written as

$$b_{ij} = \frac{a_{ij}}{\sum_{k=1}^n a_{ik}} \quad (10.4)$$

The binding probabilities for vertex v_i form the binding distribution b_i .

$$b_i = [b_{i1}, b_{i2}, b_{i3}, \dots, b_{in}]$$

We denote the matrix of binding distributions from different neurons by B .

Now, we define a stochastic process as follows: let a stochastic Neighbor Graph (SNG) be $G = (V, \varepsilon_g)$ where V is a set of vertices and ε_g is a set of directed edges. Let $i \rightarrow j$ denote the direct edge from vertex v_i to vertex v_j . If the vertex v_i binds to vertex v_j , then

we add the directed edge $i \rightarrow j$ to ε_g . As a consequent result, we say that that neuron that has the response r_i is a “virtually functional neighbor” to a neuron that has a response r_j .

Then we define that the neuron n_i is an inattentive with respect to other neurons or in other words the neuron n_i belongs to isolated islands N_0 if and only if it is corresponding vertex v_i has no inbound connection (has zero edges).

$$N_0|G = \{r_i \in R | \deg(v_i) = 0\} \quad (10.5)$$

The previous discussion was for one generated SNG, G . However, G is been generated stochastically, and hence the neurons are selected randomly. For this reason, we generate $(n - 1)^n$ SNGs such that we cover all the possible connections between vertices in neurons responses. Since some of binding probabilities are not uniformly distributed among all responses, some edges between vertices are more probable to be generated than others. To put the previous discussion in a mathematical way, let us assume that ζ is the all possible sampled SNGs.

$$p(G) = \prod_{i \rightarrow j \in \delta_G} b_{ij} \quad (10.5)$$

This means that generating a SNG depends only on the binding probabilities. We denote $G \sim P(\zeta)$ as sampling the SNG G from the probability distribution $P(\zeta)$. The probability that a neuron is an inattentive neuron is computed by sampling the SNG as follows:

$$p(r_i \in N_0) = 1 - \left(\lim_{S \rightarrow \infty} \frac{1}{S} \sum_{s=1}^S \mathbf{1}(r_i \in N_0 | G^s) \right) \quad (10.6)$$

With \aleph takes 1 if r_i belongs to N_0 and 0 if not. $G^S \sim P(\zeta)$

We notice that we can compute the exact probability that a neuron is an inattentive neuron by marginalizing out the stochastic graph G , because one SNG is more probable than others.

However, we can compute that a neuron belongs to inattentive neurons without generating any SNG directly from the binding probabilities of neurons [90].

That is, a neuron belongs to inattentive neurons “Isolated islands” if its responses from strong Neighborhood relationships with other responses from other neurons. This can be computed as:

$$rank_i = 1 - \prod_{i \neq j} (1 - p_{ji}) \quad (10.7)$$

We will use equation (10.7) as a ranking for neurons in the liquid, and we say that a neuron belongs to inattentive neurons its ranking exceeded a specific threshold:

$$if \ rank_i > \ threshold, \ then \ neuron_i \in \ inattentive \ neurons \quad (10.8)$$

B. Illustration Example

In this section, we provide an illustration example for using the pruning method, where we use it to rank and identify some neurons in artificial example. Let us assume in the figure below a part of a network inside a LSM. In this example, neurons labeled with n1 to n5 form a semi-isolated network within the liquid network. This could happen due to the stochastic process that connects neurons inside the LSM. Another reason for this problem

could happen because of the unsupervised learning in LSM using STDP, where some connections might be weakening or strengthening unsuitably. To overcome such problems, we use the proposed ranking method to evaluate neurons. First, we need to find responses associated with each neuron in LSM. Let us denote r_1 to r_6 as the responses from n_1 to n_6 , respectively. The response could be the spiking activities or the internal state of voltage for a neuron. We use in our evaluation and experiments the spiking activities from a neuron as a response. We obtain responses from neuron for different input patterns such that we cover different activation forms inside the liquid. This is important because some input patterns may form such isolated islands for only specific cases, which is not a global case. Pruning algorithm is concerned with semi-isolated islands that allow inattentiveness with irrespective to the input.

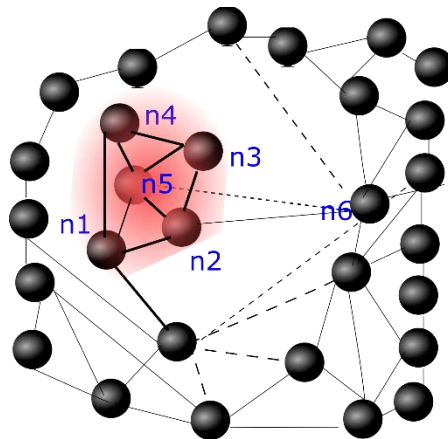


Figure 35: Isolated Islands Illustration Example.

Having we find responses from neurons, we are ready to find binding probabilities using equations 10.1, 10.2 and 10.3. In the artificial example below, neuron n_1 associated with response r_1 binds information to n_2 to n_6 . However, information propagation among neurons n_1 to n_5 is much stronger than any other information propagation with other

neurons inside the liquid. For example, n6 forms a weak connection with n2 to n6. Binding probabilities from n1 to n2, n3, n4, n5 and n6 are shown in the figure below. n1 binding probability to n6 is the lowest in comparison with binding probabilities to n2, n3, n4, n5 and n6. This is because the responses from n2 to n5 are closer “more similar” for n1 than the response from n6, i.e., when n1 fires, other n2, n3, n4 and n5 most probably will fire. From the perspective of n6, binding probabilities to n2, n3, n4 and n5 are somehow equal, since responses from those neurons lay at equal distances from it, i.e., when n6 fires, there are no strong connections that allow any of neurons in n2, n3, n4, n5 to fire.

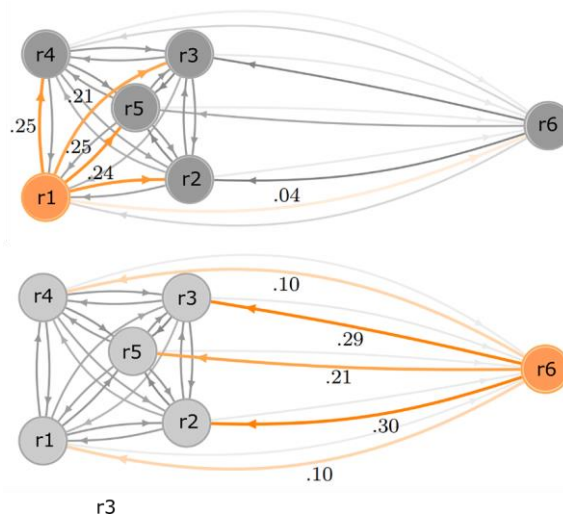


Figure 36: Binding Probabilities to Other Neurons from Neurons n1 and n6.

To find the ranking for a neuron, we find the joint probabilities that other neurons do not bind to it, and then we substitute ranking from 1 (see equation 10.8). The figure below shows binding probabilities that other neurons bind to neuron n1 and n6. We can notice that other n1, n2, n3, n4 and n5 binds to n6 with very low probability values. On the other hand n2, n3, n4 and n5 bind with high probabilities. This indicates that n1, n2, n3, n4

and n5 form a semi-isolated island. This applies for n2, n3, n4 and n5 when computing that other neurons bind to them. Compute ranking values for n1 and n6 is shown below:

$$rank_1 = 1 - [(1 - 0.24) \times (1 - 0.22) \times (1 - 0.23) \times (1 - 0.18) \times (1 - 0.1)] = 0.663$$

$$rank_6 = 1 - [(1 - 0.04) \times (1 - 0.05) \times (1 - 0.04) \times (1 - 0.05) \times (1 - 0.04)] = 0.201$$

As can be noticed, $rank_1 > rank_6$. And when choosing a proper threshold, we can identify neurons that form isolated islands.

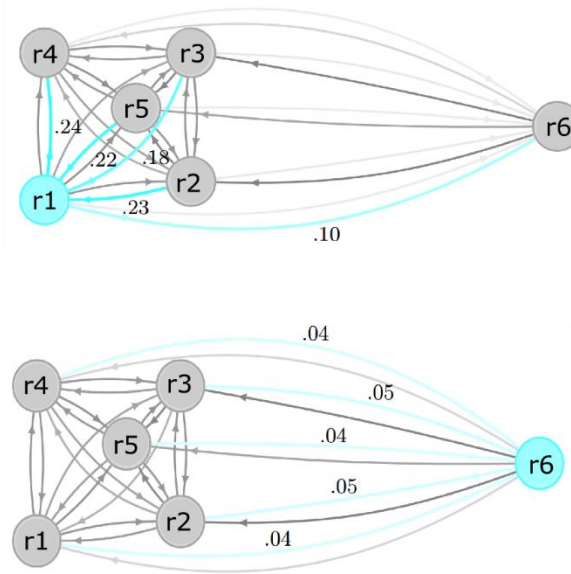


Figure 37: Binding Probabilities from Other Neurons to Neurons n1 and n6.

C. Testing and Result

The output of the method generates for each neuron in the liquid the probabilities of being an inattentive neuron. We test INP method for anomaly detection tasks as well as for classification tasks as follows:

Experiment 1: an anomaly detection for augmented password authentication task (One vs. One scenario) described in chapter VIII.

Experiment 2: a classification for five datasets downloaded from UCI namely, Fisher Iris, Pima, Sonar, Parkinson's and Ecoli.

For anomaly detection task, we use the five distance-based anomaly detectors described in chapter VIII namely, Euclidean, Euclidean (normed), Manhattan, Manhattan (filtered), Manhattan (scaled) and Z-Score. We compare results between three cases in this experiment: 1) when using all neurons for readout function, 2) after using INP method, 3) we repeat the second case for 50 times (average of 50 runs) using the same number neurons resulted from case 2, but we choose neurons randomly. For distance-based anomaly detectors, we report the performance in terms of Equal Error Rate (EER) as in chapter VIII. We vary the threshold from 0.05 to 0.75.

For classification task, we use Decision Trees classifier on the five datasets mentioned previously, where we set the threshold to be 0.3.

In all experiments, the perplexity parameter of INP was chosen to be 4.5 so that it is inline with the available literature.

1. Experiment 1: (Anomaly Detection on Augmented Password Authentication)

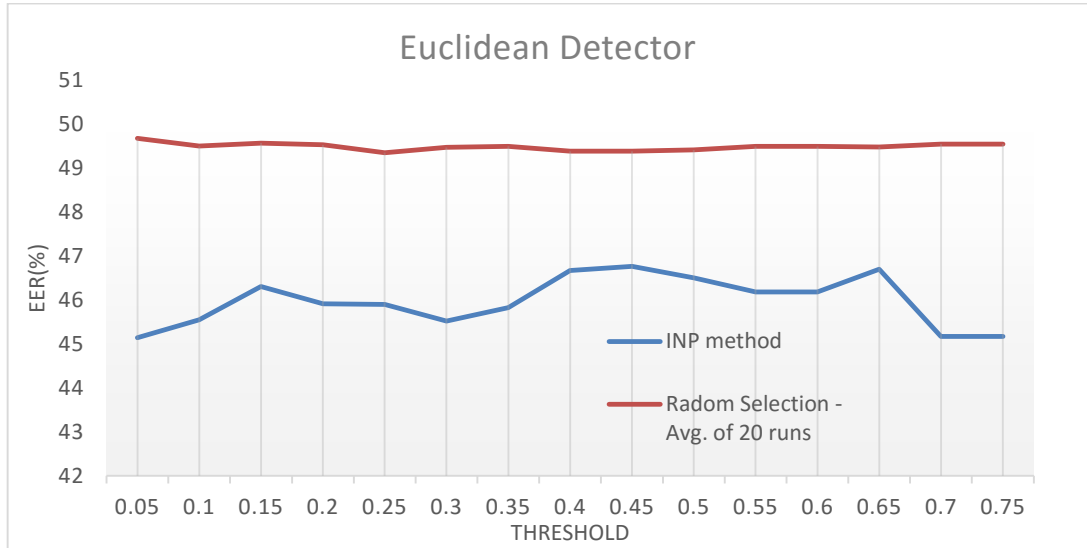


Figure 38: INP method testing on Augmented Password authentication using the Euclidean Detector anomaly detector.

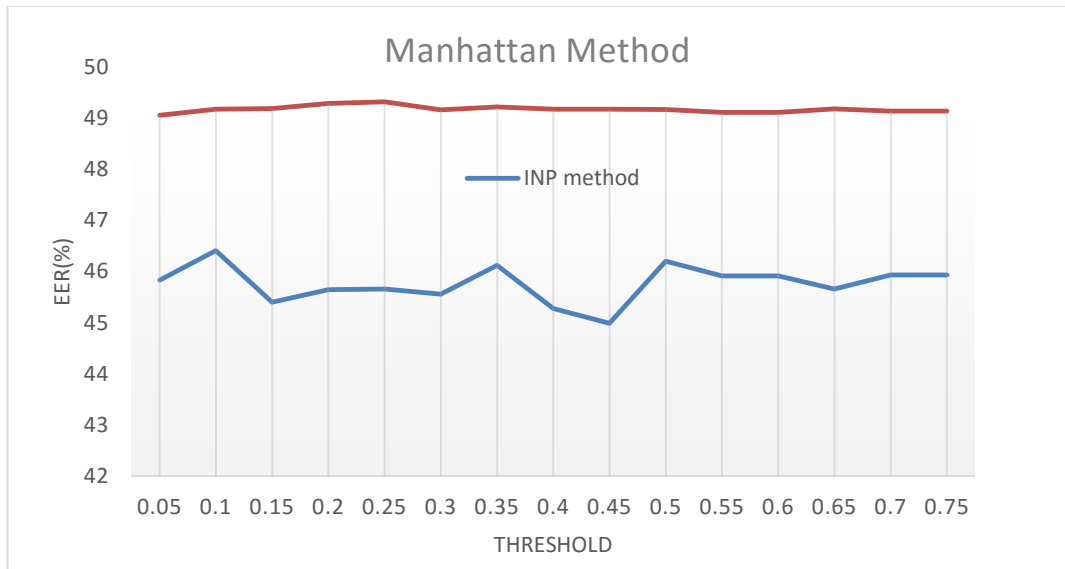


Figure 39: INP method testing on Augmented Password authentication using the Manhattan method anomaly detector.

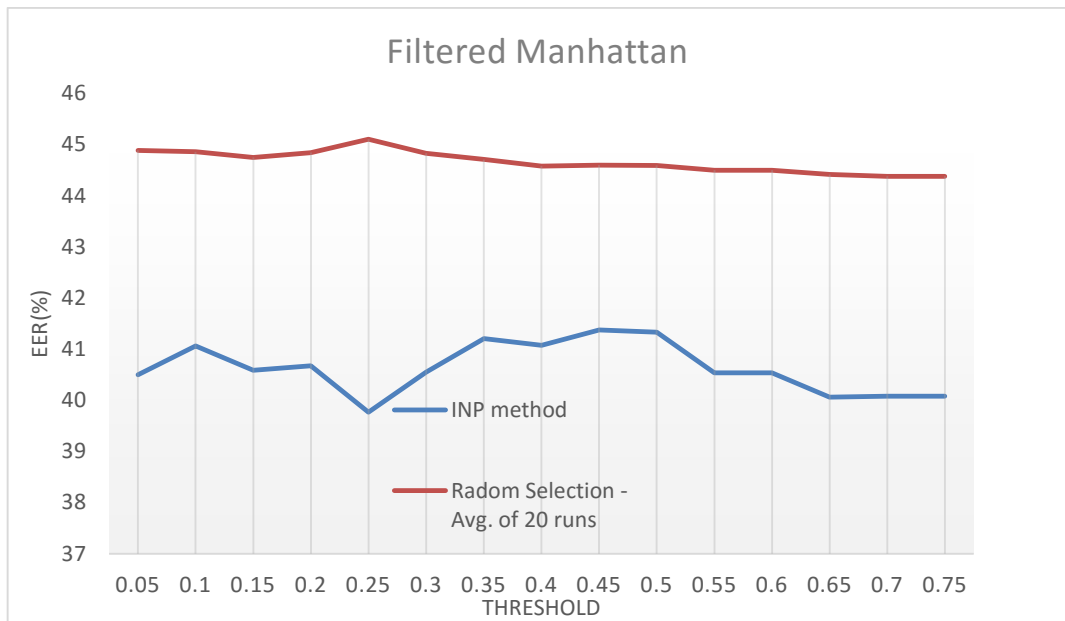


Figure 40: INP method testing on Augmented Password authentication using Filtered Manhattan anomaly detector.

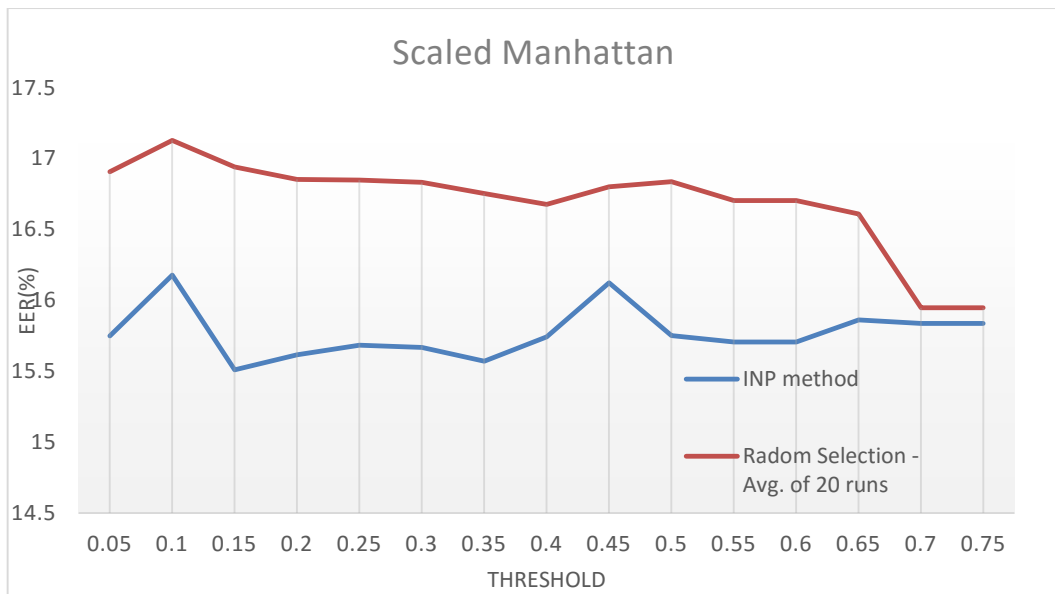


Figure 41: INP method testing on Augmented Password authentication using Scaled Manhattan anomaly detector.

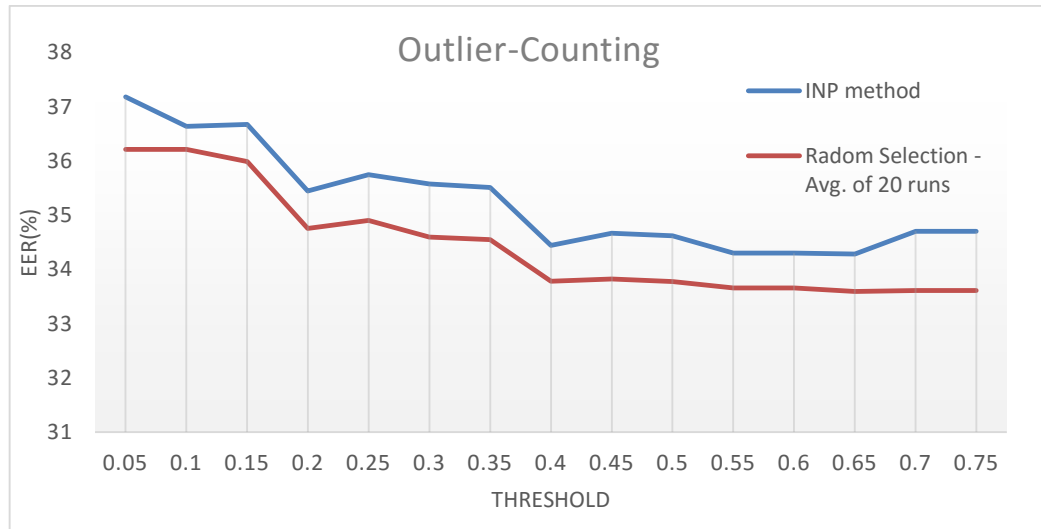


Figure 42: INP method testing on Augmented Password authentication using Outlier-Counting anomaly detector.

2. Experiment 2: (Classification of UCI datasets)

In this experiment, we used 10-fold cross validation. The random selection results are an average of 50 runs with the same number of neurons resulted after INP method (with threshold=0.3).

Table 37: INP accuracy results for experiment 2.

Decision Tree				
	Without LSM	With LSM	LSM with INP	Random Selection
Fisher Iris	95.01	95.73	<u>96.14</u>	93.41
Pima	70.49	75.35	<u>77.59</u>	70.63
Sonar	<u>71.72</u>	69.17	68.82	67.42
Parkinson's	86.33	92.30	<u>93.00</u>	90.19
Ecoli	81.32	83.77	<u>84.08</u>	81.04
Linear Regression				
Fisher Iris	<u>97.33</u>	91.72	87.26	79.85
Pima	74.10	72.74	<u>74.29</u>	66.43
Sonar	75.81	83.44	<u>84.41</u>	77.47
Parkinson's	88.76	87.16	<u>90.72</u>	85.45
Ecoli	62.02	70.08	<u>72.13</u>	69.27

Table 38: Neurons reduction rate for experiment 2.

Number of Pruned Neurons out of 216 neurons	
Fisher Iris	84
Pima	88
Sonar	145
Parkinson's	137
Ecoli	91

D. Discussion

As can be seen from experiment 1, the INP method maintains the performance while reducing the number of neurons inside the liquid. In experiment 1, the INP method always generates lower EER and always better than any randomly selected neurons (when using the same number of neurons that were generated after INP). For experiment 2, INP improved the accuracy in four out of the five datasets for Decision Trees as well as for Linear Regression. The reduction rates of the number of neurons are 53.33%, 55.32%, 32.87%, 36.28 and 53.57% for Fisher Iris, Pima, Sonar, Parkinson's and Ecoli, respectively.

E. Conclusion

We introduced in this chapter a method to prune the LSM in such a way that we maintained the informative neurons, and hence ensured that the readout is robust and less subject to overfitting. The method depends on graph theory and information binding probabilities, which makes it more suitable for LSM architecture.

CHAPTER XI

CONCLUSION AND FUTURE WORK

A. Conclusion

In this work, we introduced LSM as a universal machine learning approach to handle pattern recognition in complex systems. By using two different applications, we showed that LSM is suitable for such environments. In the first application, emotion recognition from EEG, we presented LSM as an anytime multi-purpose model to handle

inputs from a stream of signals and used LSM to analyze and study emotions in humans. In the second application, continuous authentication in smartphones, we showed how LSM can be used in real life applications. Applying LSM for real life applications is important, since most of the research on LSM were as experimental approaches.

The work also introduced two methods to improve on LSM. In the first method, active states selection, we introduced a mechanism to sample an LSM at informative states such that we reduce the overhead of oversampling, unnecessary information and thus power consumption.

In the second method, which is applying Inattentive neurons pruning, we introduced a graph theory based approach to prune uninformative neurons inside LSM. This method provides a systematic methodology to rank neurons inside LSM for later analysis and pruning.

A. Future work

The work includes enormous opportunities for future work, since it covered important aspects and topics in LSM. We can summarize the future work as follows:

- In chapter VI, we used LSM for feature extraction from EEG raw data. The results were good. However, the CbNeuron neuron model used in this experiment needs more study in order to build a universal model for feature extraction.
- Chapter VII provided an extensive study for deploying LSM for emotion recognition. Among the tested scenarios, LOSO didn't perform well. Which suggest that LSM need to be fine-tuned in order to make it suitable for such scenarios.

- Spike encoding in this work, BSA algorithm, didn't achieve good results, where the accuracies of testing BSA with LSM are worse than those resulted from direct input feeding. This suggests that a further analysis and improvement for spike encoding is needed to improve the performance of such methods.

- Chapter VIII introduced a framework for continuous authentication in smartphones. The results have shown to be promising. However, more data collection is needed, since we were able to collect for 22 users. In addition, the procedure of data collection was an offline procedure, i.e., the data are first collected from smartphone and then transferred and processed on PCs. The suggestion is to use interfacing between mobile device and PC such that the process is done online. Moreover, a friendly implementation of LSM for smartphones is recommended in order to test the framework in a real scenario.

- In chapter IX, we introduced the active states selection method for choosing the most informative sampling times for LSM. In this method, each state is ranked independently from other states. The suggestion is to change the method such that it takes into consideration the dependencies between states. Moreover, the number of sampling times is choosing arbitrarily and this needs a further study to find the effective number of sampling times for an LSM.

- In chapter X, we introduced a method to prune LSM. This method works by ranking neurons in LSM. Firstly, the threshold at which the method prunes LSM is chosen arbitrarily. This suggests that we need a procedure to choose the threshold for this method. Secondly, the ranking of neurons can be used to build a more robust LSM

by using the method to select the architecture of LSM. Thirdly, the method uses the binding information between neurons. However, this binding information is computed virtually, i.e., the algorithm does not take into account the actual connectivity inside the liquid. A more robust version of this method would take into consideration the connectivity inside the liquid such that the method will generate a subjective ranking for neurons inside the liquid, functionally and architecturally.

BIBLIOGRAPHY

- [1] C. Christodoulou, G. Bugmann and T. G. Clarkson, "A spiking neuron model: applications and learning," *Neural Networks*, vol. 15, pp. 891-908, 2002.
- [2] J. Vreeken, "Spiking neural networks, an introduction," *Institute for Information and Computing Sciences, Utrecht University Technical Report UU-CS-2003-008*, 2002.
- [3] H. Paugam-Moisy and S. Bohte, "Computing with spiking neuron networks," in *Handbook of Natural Computing* Anonymous Springer, 2012, pp. 335-376.
- [4] A. Gupta and L. N. Long, "Character recognition using spiking neural networks," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference On*, 2007, pp. 53-58.
- [5] B. Meftah, A. Benyettou, O. Lezoray and W. Q. Xiang, "Image clustering with spiking neuron network," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference On*, 2008, pp. 681-685.
- [6] Y. Meng, Y. Jin and J. Yin, "Modeling activity-dependent plasticity in BCM spiking neural networks with application to human behavior recognition," *Neural Networks, IEEE Transactions On*, vol. 22, pp. 1952-1966, 2011.

- [7] M. O'Halloran, B. McGinley, R. C. Conceicao, F. Morgan, E. Jones and M. Glavin, "Spiking neural networks for breast cancer classification in a dielectrically heterogeneous breast," *Progress in Electromagnetics Research*, vol. 113, pp. 413-428, 2011.
- [8] T. Obo, N. Kubota, K. Taniguchi and T. Sawayama, "Human localization based on spiking neural network in intelligent sensor networks," in *Robotic Intelligence in Informationally Structured Space (RiiSS), 2011 IEEE Workshop On*, 2011, pp. 125-130.
- [9] G. Budjade, *Intrusion Detection using Spiking Neural Networks*, 2014.
- [10] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks*, vol. 1, pp. 339-356, 1988.
- [11] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive Modeling*, vol. 5, pp. 3, 1988.
- [12] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179-211, 1990.
- [13] R. Pascanu, T. Mikolov and Y. Bengio, "On the difficulty of training recurrent neural networks," *arXiv Preprint arXiv:1211.5063*, 2012.
- [14] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, pp. 127-149, 2009.
- [15] A. Grüning and S. M. Bohte, "Spiking Neural Networks: Principles and Challenges," 2014.
- [16] H. Jaeger, "Echo state network," *Scholarpedia*, vol. 2, pp. 2330, 2007.
- [17] W. Maass, "Liquid state machines: motivation, theory, and applications," in *Computability in Context: Computation and Logic in the Real World*, pp. 275-296, 2010.
- [18] L. F. Abbott and S. B. Nelson, "Synaptic plasticity: taming the beast," *Nat. Neurosci.*, vol. 3, pp. 1178-1183, 2000.
- [19] S. Schliebs and N. Kasabov, "Evolving spiking neural network—a survey," *Evolving Systems*, vol. 4, pp. 87-98, 2013.
- [20] H. Paugam-Moisy and S. Bohte, "Computing with spiking neuron networks," in *Handbook of Natural Computing* Anonymous Springer, 2012, pp. 335-376.
- [21] V. Thiruvardhelvan, J. W. Crane and T. Bossomaier, "Analysis of spikeprop convergence with alternative spike response functions," in *Foundations of Computational Intelligence (FOCI), 2013 IEEE Symposium On*, 2013, pp. 98-105.

- [22] W. Maass, T. Natschläger and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Comput.*, vol. 14, pp. 2531-2560, 2002.
- [23] Löwe, S Barry Cooper Benedikt and A. Sorbi, "Computation and Logic in the Real World," .
- [24] H. Jaeger, "Echo state network," *Scholarpedia*, vol. 2, pp. 2330, 2007.
- [25] B. Schrauwen, D. Verstraeten and J. Van Campenhout, "An overview of reservoir computing: Theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks*, 2007, .
- [26] G. E. Hinton, S. Osindero and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527-1554, 2006.
- [27] B. Schrauwen and J. Van Campenhout, "BSA, a fast and accurate spike train encoding scheme," in *Proceedings of the International Joint Conference on Neural Networks*, 2003, pp. 2825-2830.
- [28] N. Nuntalid, K. Dhoble and N. Kasabov, "EEG classification with BSA spike encoding algorithm and evolving probabilistic spiking neural network," in *Neural Information Processing*, 2011, pp. 451-460.
- [29] W. A. Kamiński and G. M. Wójcik, "Liquid state machine built of Hodgkin-Huxley neurons-pattern recognition and informational entropy," *Annales UMCS Sectio AI Informatica*, vol. 1, pp. 1-7, 2015.
- [30] S. Schliebs, M. Fiasché and N. Kasabov, "Constructing robust liquid state machines to process highly variable data streams," in *Artificial Neural Networks and Machine Learning–ICANN 2012* Anonymous Springer, 2012, pp. 604-611.
- [31] D. Verstraeten, B. Schrauwen, D. Stroobandt and J. Van Campenhout, "Isolated word recognition with the liquid state machine: a case study," *Information Processing Letters*, vol. 95, pp. 521-528, 2005.
- [32] D. Verstraeten, B. Schrauwen, M. d'Haene and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, pp. 391-403, 2007.
- [33] H. Ju, J. Xu and A. M. VanDongen, "Classification of musical styles using liquid state machines," in *Neural Networks (IJCNN), the 2010 International Joint Conference On*, 2010, pp. 1-7.

- [34] B. J. Grzyb, E. Chinellato, G. M. Wojcik and W. Kaminski, "Facial expression recognition based on liquid state machines built of alternative neuron models," in *Neural Networks, 2009. IJCNN 2009. International Joint Conference On*, 2009, pp. 1011-1017.
- [35] J. Baraglia, Y. Nagai and M. Asada, "Action understanding using an adaptive liquid state machine based on environmental ambiguity," in *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference On*, 2013, pp. 1-6.
- [36] H. Burgsteiner, "Imitation learning with spiking neural networks and real-world devices," *Eng Appl Artif Intell*, vol. 19, pp. 741-752, 2006.
- [37] H. Burgsteiner, M. Kröll, A. Leopold and G. Steinbauer, "Movement prediction from real-world images using a liquid state machine," *Appl. Intell.*, vol. 26, pp. 99-109, 2007.
- [38] A. Lonsberry, K. Daltorio and R. D. Quinn, "Capturing stochastic insect movements with liquid state machines," in *Biomimetic and Biohybrid Systems* Anonymous Springer, 2014, pp. 190-201.
- [39] K. Daltorio, B. R. Tietz, J. A. Bender, V. Webster, N. S. Szczecinski, M. S. Branicky, R. E. Ritzmann and R. D. Quinn, "A stochastic algorithm for explorative goal seeking extracted from cockroach walking data," in *Robotics and Automation (ICRA), 2012 IEEE International Conference On*, 2012, pp. 2261-2268.
- [40] P. Jakimovski and H. R. Schmidtke, "Delayed synapses: An LSM model for studying aspects of temporal context in memory," in *Modeling and using Context* Anonymous Springer, 2011, pp. 138-144.
- [41] E. Hourdakis and P. Trahanias, "Improving the classification performance of liquid state machines based on the separation property," in *Engineering Applications of Neural Networks* Anonymous Springer, 2011, pp. 52-62.
- [42] B. J. Grzyb, E. Chinellato, G. M. Wojcik and W. Kaminski, "Which model to use for the liquid state machine?" in *Neural Networks, 2009. IJCNN 2009. International Joint Conference On*, 2009, pp. 1018-1024.
- [43] D. Norton and D. A. Ventura, "Preparing more effective liquid state machines using hebbian learning," 2006.
- [44] D. Norton and D. Ventura, "Improving liquid state machines through iterative refinement of the reservoir," *Neurocomputing*, vol. 73, pp. 2893-2904, 2010.
- [45] P. Ekman, W. V. Friesen, M. O'Sullivan, A. Chan, I. Diacoyanni-Tarlatzis, K. Heider, R. Krause, W. A. LeCompte, T. Pitcairn and P. E. Ricci-Bitti, "Universals and cultural

differences in the judgments of facial expressions of emotion." *J. Pers. Soc. Psychol.*, vol. 53, pp. 712, 1987.

[46] A. Ben-Zeev, "The nature of emotions," *Philosophical Studies*, vol. 52, pp. 393-409, 1987.

[47] W. G. Parrott, *Emotions in Social Psychology: Essential Readings*. Psychology Press, 2001.

[48] J. A. Russell, "A circumplex model of affect." *J. Pers. Soc. Psychol.*, vol. 39, pp. 1161, 1980.

[49] S. Koelstra, C. Mühl, M. Soleymani, J. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt and I. Patras, "Deap: A database for emotion analysis; using physiological signals," *Affective Computing, IEEE Transactions On*, vol. 3, pp. 18-31, 2012.

[50] X. Wang, D. Nie and B. Lu, "EEG-based emotion recognition using frequency domain features and support vector machines," in *Neural Information Processing*, 2011, pp. 734-743.

[51] M. K. Kim, M. Kim, E. Oh and S. P. Kim, "A review on the computational methods for emotional state estimation from the human EEG," *Comput. Math. Methods Med.*, vol. 2013, pp. 573734, 2013.

[52] A. T. Sohaib, S. Qureshi, J. Hagelbäck, O. Hilborn and P. Jerčić, "Evaluating classifiers for emotion recognition using EEG," in *Foundations of Augmented Cognition* Anonymous Springer, 2013, pp. 492-501.

[53] F. N. Feradov and T. D. Ganchev, "Detection of Negative Emotional States from Electroencephalographic (EEG) Signals," .

[54] S. Jirayucharoensak, S. Pan-Ngum and P. Israsena, "EEG-based emotion recognition using deep learning network with principal component based covariate shift adaptation," *The Scientific World Journal*, vol. 2014, 2014.

[55] V. Rozgic, S. N. Vitaladevuni and R. Prasad, "Robust EEG emotion classification using segment level decision fusion," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference On*, 2013, pp. 1286-1290.

[56] X. Zhuang, V. Rozgic and M. Crystal, "Compact unsupervised EEG response representation for emotion recognition," in *Biomedical and Health Informatics (BHI), 2014 IEEE-EMBS International Conference On*, 2014, pp. 736-739.

- [57] K. Li, X. Li, Y. Zhang and A. Zhang, "Affective state recognition from EEG with deep belief networks," in *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference On*, 2013, pp. 305-310.
- [58] X. Jia, K. Li, X. Li and A. Zhang, "A novel semi-supervised deep learning framework for affective state recognition on EEG signals," in *Bioinformatics and Bioengineering (BIBE), 2014 IEEE International Conference On*, 2014, pp. 30-37.
- [59] I. Wichakam and P. Vateekul, "An evaluation of feature extraction in EEG-based emotion prediction with support vector machines," in *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference On*, 2014, pp. 106-110.
- [60] R. Cabredo, R. S. Legaspi, P. S. Inventado and M. Numao, "Discovering Emotion-Inducing Music Features Using EEG Signals." *Jaciii*, vol. 17, pp. 362-370, 2013.
- [61] X. Jie, R. Cao and L. Li, "Emotion recognition based on the sample entropy of EEG," *Biomed. Mater. Eng.*, vol. 24, pp. 1185-1192, 2014.
- [62] P. Hamel and D. Eck, "Learning features from music audio with deep belief networks." in *Ismir*, 2010, pp. 339-344.
- [63] M. Långkvist, L. Karlsson and A. Loutfi, "Sleep stage classification using unsupervised feature learning," *Advances in Artificial Neural Systems*, vol. 2012, pp. 5, 2012.
- [64] T. Natschläger and W. Maass, "CSIM: a neural Circuit SIMulator," *User Manual. Institute for Theoretical Computer Science, Graz University of Technology*. Available in <Http://Www.Lsm.Tugraz.at/Csim/Index.Html>, 2006.
- [65] S. Koelstra, C. Muhl, M. Soleymani, J. Lee, A. Yazdani, T. Ebrahimi, T. Pun, A. Nijholt and I. Patras, "Deap: A database for emotion analysis; using physiological signals," *Affective Computing, IEEE Transactions On*, vol. 3, pp. 18-31, 2012.
- [66] E. Capecci, N. Kasabov and G. Y. Wang, "Analysis of connectivity in NeuCube spiking neural network models trained on EEG data for the understanding of functional changes in the brain: A case study on opiate dependence treatment," *Neural Networks*, vol. 68, pp. 62-77, 2015.
- [67] N. Kasabov and et al, "Design methodology and selected applications of evolving spatio-temporal data machines in the NeuCube neuromorphic framework, *Neural Networks*," 2016.

[68] M. Doborjeh, G. Wang and N. Kasabov, "A Neucube Spiking Neural Network Model for the Study of Dynamic Brain Activities during a GO/NO_GO Task: A Case Study on Using EEG Data of Healthy Vs Addiction vs Treated Subjects," 2015.

[69] N. Kasabov and E. Capecchi, "Spiking neural network methodology for modelling, classification and understanding of EEG spatio-temporal data measuring cognitive processes," *Inf. Sci.*, vol. 294, pp. 565-575, 2015.

[70] N. K. Kasabov, "NeuCube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62-76, 2014.

[71] N. Kasabov, V. Feigin, Z. Hou, Y. Chen, L. Liang, R. Krishnamurthi, M. Othman and P. Parmar, "Evolving spiking neural networks for personalized modelling, classification and prediction of spatio-temporal patterns with a case study on stroke," *Neurocomputing*, vol. 134, pp. 269-279, 2014.

[72] G. Huang, Q. Zhu and C. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.

[73] T. Feng, X. Zhao, B. Carbanar and W. Shi, "Continuous mobile authentication using virtual key typing biometrics," in *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference On*, 2013, pp. 1547-1552.

[74] N. L. Clarke and S. Furnell, "Authenticating mobile phone users using keystroke analysis," *International Journal of Information Security*, vol. 6, pp. 1-14, 2007.

[75] M. Frank, R. Biedert, E. Ma, I. Martinovic and D. Song, "Touchalytics: On the applicability of touchscreen input as a behavioral biometric for continuous authentication," *Information Forensics and Security, IEEE Transactions On*, vol. 8, pp. 136-148, 2013.

[76] S. Govindarajan, P. Gasti and K. S. Balagani, "Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference On*, 2013, pp. 1-8.

[77] X. Zhao, T. Feng and W. Shi, "Continuous mobile authentication using a novel graphic touch gesture feature," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference On*, 2013, pp. 1-6.

[78] A. Serwadda and V. V. Phoha, "When kids' toys breach mobile phone security," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, 2013, pp. 599-610.

- [79] I. Deutschmann, P. Nordstrom and L. Nilsson, "Continuous authentication using behavioral biometrics," *IT Professional*, vol. 15, pp. 12-15, 2013.
- [80] A. Azzini, S. Marrara, R. Sassi and F. Scotti, "A fuzzy approach to multimodal biometric continuous authentication," *Fuzzy Optimization and Decision Making*, vol. 7, pp. 243-256, 2008.
- [81] H. Crawford, K. Renaud and T. Storer, "A framework for continuous, transparent mobile device authentication," *Comput. Secur.*, vol. 39, pp. 127-136, 2013.
- [82] Anonymous "An augmented computer user login authentication using classifying regions of keystroke density neural network." PST-15418/36, 2005.
- [83] K. S. Balagani, V. V. Phoha, A. Ray and S. Phoha, "On the discriminability of keystroke feature vectors used in fixed text keystroke authentication," *Pattern Recog. Lett.*, vol. 32, pp. 1070-1080, 2011.
- [84] K. S. Killourhy and R. Maxion, "Comparing anomaly-detection algorithms for keystroke dynamics," in *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP International Conference On*, 2009, pp. 125-134.
- [85] W. Zhu, X. Wang, Y. Ma, M. Rao, J. Glimm and J. S. Kovach, "Detection of cancer-specific markers amid massive mass spectral data," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 100, pp. 14666-14671, Dec 9, 2003.
- [86] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. ELSEVIER, October 2008.
- [87] K. Vandoorne, W. Dierckx, B. Schrauwen, D. Verstraeten, R. Baets, P. Bienstman and J. Van Campenhout, "Toward optical signal processing using photonic reservoir computing," *Optics Express*, vol. 16, pp. 11182-11192, 2008.
- [88] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso and I. Fischer, "Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing," *Optics Express*, vol. 20, pp. 3241-3249, 2012.
- [89] Y. Zhang, P. Li, Y. Jin and Y. Choe, "A Digital Liquid State Machine With Biologically Inspired Learning and Its Application to Speech Recognition," 2015.
- [90] J. Janssens, F. Huszár, E. Postma and H. van den Herik, "*Stochastic outlier selection*," *Stochastic Outlier Selection*, 2012.