Implementation of Memory for Cognitive Agents Using Biologically Plausible Associative Pulsing Neurons

A dissertation presented to

the faculty of

the Russ College of Engineering and Technology of Ohio University

In partial fulfillment

of the requirements for the degree

Doctor of Philosophy

Basawaraj

August 2019

This dissertation titled

Implementation of Memory for Cognitive Agents Using Biologically Plausible Associative Pulsing Neurons

by

BASAWARAJ

has been approved for

the School of Electrical Engineering and Computer Science

and the Russ College of Engineering and Technology by

Wojciech M. Jadwisienczak

Associate Professor of Electrical Engineering and Computer Science

Dennis Irwin

Dean, Russ College of Engineering and Technology

ABSTRACT

Basawaraj, Ph.D., May 2019, Electrical Engineering and Computer Science

Implementation of Memory for Cognitive Agents Using Biologically Plausible Associative Pulsing Neurons

Director of Dissertation: Wojciech M. Jadwisienczak

Artificial intelligence (AI) is being widely applied to various practical problems, and researchers are working to address numerous issues facing the field. The organizational structure and learning mechanism of the memory is one such issue. A cognitive agent builds a representation of its environment and remembers its experiences to interpret its inputs and implements its goals through its actions. By doing so it demonstrates its intelligence (if any), and it is its learning mechanism, value system and sensory motor coordination that makes all this possible. Memory in a cognitive agent stores its knowledge, knowledge gained over a life-time of experiences in a specific environment. That is, memory includes the "facts", the relationships between them, and the mechanism used to learn, recognize, and recall based on the agent's interaction with the world/environment. It remembers events that the agent experienced reflecting important actions and observations. It motivates the agent to do anything by providing assessment of the state of the environment and its own state. It allows it to plan and anticipate. And finally, it allows the agent to reflect on itself as an independent being. Hence, memory is critical for intelligence, for it is the memory that determines a cognitive agent's abilities and learning skills.

Research has shown that while memory in humans can be classified into different types, based on factors such as their longevity and cognitive mechanisms used to create and retrieve them, they all are achieved using a similar underlying structure. The focus of this dissertation was on using this principle, *i.e.* different memories created using the same underlying structure, to implement memory for cognitive agents using a biologically plausible model of neuron. This work was an attempt to demonstrate the feasibility of implementing self-organizing memory structures capable of performing the various memory related tasks necessary for a cognitive agent using a computationally feasible model of biologically inspired neuron. For the purpose of this work the memory capabilities were limited to those necessary for a cognitive agent capable of solving some simple problem, say satisfying its hunger, and in the process creating some high-level abstract needs such as increasing food supply and learning to address them, eventually creating a set of abstract pains and goals.

Intelligence requires the ability to learn, and a cognitive agent can demonstrate its intelligence through learning from its experiences and observations to solve problems. In this work it was assumed that the following capabilities: ability to recognize objects, ability to recognize sequences, form relationships between information in the memory, make predictions/anticipate, and demonstrate creativity were necessary for a cognitive agent to learn to solve a problem and demonstrate its intelligence. Subsequent to providing proof that these abilities were necessary for a cognitive agent capable of learning to solve a problem that required creation of a set of abstract pains and goals, a biologically plausible associative pulsing neuron model was introduced. This neuron

model was subsequently used to implement various memory structures and learning mechanisms to demonstrate the above mentioned memory capabilities. The declarative memory implemented demonstrated the ability to create semantic relationships and demonstrated creativity. The ability to recognize sequences was demonstrated by both the episodic memory and the structure resulting from the lumped minicolumn associative knowledge graph (LUMAKG) algorithm. LUMAKG structure also demonstrated the ability to create semantic relationships and make predictions. Finally an associative pulsing neural network (APNN) structure created using an associative adaption process demonstrated its ability to recognize objects.

The implemented memory structures were tested on various datasets, *e.g.* MNIST, children's book test, and yeast, and their performances compared to various state of the art techniques. The results demonstrated that the memory structures implemented in this work had performance better than or comparable to other techniques. Finally, the results of this dissertation demonstrate the feasibility of building memory for a cognitive agent based upon a single computational unit, *i.e.* neuron model, and simple learning mechanisms.

DEDICATION

Dedicated to my parents and my wife.

ACKNOWLEDGMENTS

I would like to thank Dr. Janusz A. Starzyk for his guidance and constant encouragement that has helped me complete this dissertation research. Additionally, I would like to thank my advisor Dr. Wojciech M. Jadwisienczak for patiently working with and helping me complete my dissertation.

I would also like to thank Dr. Savas Kaya, Dr. Maarten Uijt de Haag, Dr. Jundong Liu, Dr. Jeffrey Dill, Dr. Ronaldo Vigo, and Dr. Annie Shen for their support and encouragement, and for serving on my committee. I would also like to thank Dr. Adrian Horzyk for his assistance with software implementations.

Finally, I would like to thank my friends and family for their constant support and encouragement throughout all these years.

TABLE OF CONTENTS

Page

Abstract 3
Dedication 6
Acknowledgments 7
List of Tables 11
List of Figures 12
List of Abbreviations 14
1. Introduction 15
1.1 Introduction 16
1.2 Research Overview 19
1.3 Research Objectives and Scope 22
1.4 Dissertation Organization 26
2 A Simplified Model of Cognitive Agent 28
2.1 Introduction 28
2.2 Embodied Intelligence and Motivated Learning 33
2.3 Cognitive Agent Model 36
2.3.1 Background 36
2.3.2 Model and Assumptions 38
2.4 Associative Memory – Using Symbolic Inputs 43
2.4.1 Simulation Results 45
2.5 Conclusion 49
3 Associative Pulsing Neuron Model 50
3.1 Introduction 50
3.2 Associative Neuron 54
3.2.1 Sample Network Structure 63
3.3 Neural Mechanisms 67
3.3.1 Threshold Increase 67
3.3.2 Axon Growth 70
3.3.3 Synaptic Fatigue 70
3.4 Conclusion 73
4 Declarative Memories 74

4.1 Introduction 74
4.2 Semantic Memory 75
4.2.1 Structural Organization of Semantic Memory 76
4.2.2 Testing Semantic Memory 79
4.3 Episodic Memory 80
4.3.1 Structural Organization of Episodic Memory 80
4.3.2 Algorithm for Episodic Memory Retrieval 82
4.3.3 Testing Episodic Memory 83
4.4 Emergent Creativity of Declarative Memories 86
4.5 Conclusions 88
5 Lumped minicolumn associative knowledge graph 90
5.1 Introduction 90
5.2 Organization of LUMAKG 93
5.2.1 Minicolumn Organization of the Associative Memory 93
5.2.2 Organizing Principles of LUMAKG 96
5.2.3 The LUMAKG Algorithm 97
5.3 LUMAKG Design Example 98
5.3.1 Finding Non-overlapping Sequences 99
5.3.2 Establishing a Sequence of Linked PA Neurons 103
5.3.3 Design Example 104
5.4 Comparative Tests of LUMAKG 108
5.4.1 Test Preparation 108
5.4.2 Network Response Quality Measures 109
5.5 Conclusion 121
6 Handwritten Digit Recognition using Associative Adaptation 122
6.1 Introduction 122
6.2 Background 124
6.3 Associative Adaptation 126
6.4 Results 130
6.5 Conclusion 132
7 Multi-class Classification using Associative Adaptation 133
7.1 Introduction 133
7.2 Related Work 134

7.3 Experimental Results 135
7.3.1 Datasets 135
7.3.2 Evaluation Metrics 137
7.3.3 Performance Comparison 138
7.4 Conclusion 140
8 Conclusions and Future Work 141
References 144

LIST OF TABLES

Page

Table 2-1. Some meaningful pain-motor pairs and their effects on resources. 38
Table 2-2. Associations for mental saccade. 46
Table 4-1. Semantic memory answers to various initial contexts. 80
Table 4-2. Declarative memory answers to various initial contexts. 88
Table 6-1. Comparison of handwritten digit recognition. 131
Table 7-1. Database statistics. 136
Table 7-2. Results of 10-fold cross-validation (mean ± std. dev). 139
Table 7-3. Simulation times for 10-fold cross-validation. 140

LIST OF FIGURES

Page

Figure 2-1. Simplified computational model of cognitive agent. 41

Figure 2-2. Neural network structure of the memory. 45

Figure 2-3. Result of visual saccade. 46

Figure 2-4. Primitive (top) and abstract (bottom) pain levels. 48

Figure 3-1. Model of a neuron. 55

Figure 3-2. States of the associative model of neurons. 59

Figure 3-3. Steps of a sample ANAKG structure developed according to the associative process. 66

Figure 3-4. When a neuron grows its soma gets larger and requires more charges to be activated. 68

Figure 3-5. Synaptic vesicles represented by small circles on the top and postsynaptic receptors shown in postsynaptic neuron at the bottom. 71

Figure 3-6. Changes in neuron sensitivity due to a fatigue factor. 72

Figure 4-1. A sample neuronal structure formed during associative processes for training sequences. 78

Figure 4-2. A model of LTM cell based on associative neurons. 81

Figure 4-3. Activation levels of “winning neuron” in LTM cells 1 and 2. 85

Figure 5-1. Activated minicolumns with the existing synaptic connections. 99

Figure 5-2. Linked episodic neurons of the previously learned sequences of symbols. 101

Figure 5-3. Merging of three overlapping PGMs. 102

Figure 5-4. A longer sequence of activated minicolumns. 103

Figure 5-5. A new connection between the UA minicolumn L and a PA neuron in the minicolumn M. 105

Figure 5-6. A new connection between UA minicolumn K and a PA neuron in minicolumn L. 106

Figure 5-7. New connections between the PA neuron in Q and a selected MNOC neuron in the minicolumn R. Additional connections are established between the PA neurons in R and A and between I and a new PA neuron in J. 107

Figure 5-8. Modified synaptic connections for the input sequence. 107

Figure 5-9. Plot of mean Levenshtein distances for the LSTM, the ANAKG network and the LUMAKG networks of different column sizes as a function of the number of symbols. 111

Figure 5-10. Plot of mean Levenshtein distances for the LSTM, the ANAKG network and the LUMAKG networks of different column sizes as a function of the number of unique symbols. 112

Figure 5-11. Plot of mean RWP for the LSTM, the ANAKG network and the LUMAKG networks of different column sizes as a function of the number of symbols. 115

Figure 5-12. Plot of the mean (a) Levenshtein distances and (b) RWP, for the LSTM, the ANAKG networks and the LUMAKG networks of different column sizes as a function of the number of symbols. The reference line at recall quality threshold of 70% was added. 118

Figure 5-13. Learning times of the LSTM, ANAKG network and the LUMAKG network of different column sizes as a function of the number of symbols. 119

LIST OF ABBREVIATIONS

AGI	Artificial General Intelligence
AI	Artificial Intelligence
ANAKG	Active Neuro-Associative Knowledge Graph
ANN	Artificial Neural Networks
APNN	Associative Pulsing Neural Networks
EI	Embodied Intelligence
HTM	Hierarchical Temporal Memory
LSTM	Long Short-Term Memory
LTM	Long-Term Memory
LUMAKG	LUmped Minicolumn Associative Knowledge Graph
ML	Motivated Learning
NN	Neural Network
RNN	Recurrent Neural Networks

1. INTRODUCTION

Artificial intelligence (AI) as a formal research field has existed since the 1950s and is generally believed to have been founded at the Dartmouth Summer Research Project on Artificial Intelligence in 1956 [1]. The long term goal of the field is general intelligence or human-like intelligence. Initially it was believed that a machine capable of succeeding at the Turing test will be intelligent. Alan Turing proposed an imitation game, better known as "Turing test", as a test for machine intelligence [2]: if a human judge is unable to correctly distinguish whether conversation is with a human or a machine then the machine is considered intelligent. While Turing test can be helpful to determine intelligence it has a few limitations, the major one is that it is a test for human intelligence of a specific kind. It is limited to demonstration of verbally expressed knowledge, *i.e.* ability to use language and grammar, and does not require learning. But according to some researchers, learning is the key ingredient of intelligence [3], [4]. Thus, Turing test may not actually be an efficient test of intelligence or may not test intelligence at all.

Early attempts of AI tried to develop advanced abilities of human mind like theorem proving, games, logical reasoning and other tasks where the knowledge and reasoning skills were built-into the programs. Some of the earliest attempts were the symbolic integration program, analogy program, Eliza, SIR, semantic nets *etc*. While promising and providing impetus to later research the developments where either too simple to be theorems (or laws) of AI like those in the hard sciences or very complex yet performed very poorly to be considered like laws in biological sciences [5]. After some

early failures and years of unfulfilled bold predictions of intelligence without a learning agent, research direction changed from general intelligence to solving specific problems. Today AI has grown into a vast research field focused on analyzing and solving everyday problems such as speech, image, face, and handwriting recognition, natural language processing, prediction, decision making, autonomous control, robotics, *etc*.

1.1 Introduction

Developments in AI over the last 60 years, in combination with other technological advancements, have made AI ubiquitous. Indeed applications of AI include the gamut from enterprise uses such as data mining (*e.g.* credit approval and content ID) to personal devices (*e.g.* Google Assistant or Apple's Face ID). But even with access to cutting edge technologies and large computational power, it is observed that problems that are regularly handled by humans are too complicated and difficult to solve using existing AI systems. Some examples of this are the widely covered limitations/challenges of self-driving cars or autonomous vehicles, *e.g.* challenges making left-turns at intersections, failure to detect stationary objects, and city-driving in general. While there might be, and indeed are, engineering and managerial decisions (such as need to avoid liability and possible negative publicity) underlying these issues, there also exist some fundamental problems. A major issue is that the "intelligent" systems in these machines are not based on understandings of the working of the human (or animal) brain, or even understanding of the observed environment.

In pursuit of human-level artificial intelligence, known as artificial general intelligence (AGI), researchers have for long time worked on developing cognitive

architectures, grounded in understandings of the brain, that can provide a framework (aka computational model) to support the development of intelligent agents. ("Agent" refers to the system under consideration and can be implemented either as a software in a virtual world, or hardware in a real world.) A cognitive architecture is a cognitive/psychological model that includes the general structure of a machine brain, essential modules/subsystems and the relationship between them, and other invariant aspects necessary for its computational implementation. A cognitive architecture not only can be important in understanding the human brain, it can also provide the necessary computational framework for further modeling and exploration of the brain. While cognitive architectures and intelligent software systems share some features (*e.g.* memory storage, and input/output devices), they are different. Cognitive architectures are designed to be models that can form the basis for agents that understands its environment, can learn, plans its actions to solve a variety of problems in dynamic environments and consequently is capable of development, unlike intelligent software systems (*e.g.* Watson [6]) that have fixed computational models [7] and may or may not be able to learn depending on how its knowledge is entered into the system.

Over the last 40 odd years researchers have proposed about 300 cognitive architectures, about 50 of which are being actively developed [8]. The lack of a general or "grand unified" theory of cognition has meant that researchers have based their architectures on different presumptions and assumptions, including various competencies and behaviors necessary to define intelligence. But implementing even a limited set of abilities in an architecture requires extensive work. Consequently, only a few cognitive

architectures explicitly pursue AGI. While Soar [7], ACT-R [9], NARS [10], LIDA [11] are examples of the more "established" or "mature" cognitive architectures, SiMA (formerly ARS) [12], Sigma [13] and CogPrime [14] examples of the more recent cognitive architectures that explicitly pursue AGI. In fact, many cognitive architectures focus on specific cognitive aspects, such as attention (ARCADIA [15], STAR [16]), emotion (CELTS [17]), perception of symmetry (Cognitive Symmetry Engine [18]) or problem solving (FORR [19], PRODIGY [19]). While a few are more narrowly focused and designed for specific applications, such as visual inspection of surfaces (ARDIS [20]) or music comprehension and generation (MusiCog [21]).

A good understanding of the organization and inner working of the human brain is necessary for developing cognitive architectures that can be used to build machines capable of human level intelligence. But understanding human brain and consequently human cognition is a very challenging and difficult task. Hence researches either tend to focus on specific behavioral aspects such as memory, recognition, recall, decision making, and navigation or study and model abilities of insects or animals. In addition, research in the fields of neuroscience, physiology, medical diagnostics, scanning and imaging techniques, and psychology has provided beneficial information about working and organization of human brain that is aiding in development of intelligent machines. Techniques such as functional magnetic resonance imaging (fMRI), electroencephalography (EEG), and near-infrared spectroscopy (NIRS) have proven very useful for studying brain responses to various stimuli. These techniques provide information about activation in brain areas and can support or interpret studies in

anatomy, neurobiology, or psychology. Hopefully, these and future techniques combined with research in related fields will provide a more accurate, if not complete, understanding of the human brain and this can be used to design and build efficient cognitive agents.

1.2 Research Overview

The development of cognitive agents capable of solving real-world problems necessitates both a suitable cognitive architecture and a means of implementing the same. While the various cognitive architectures differ in both their underlying assumptions and structure, they all share a few common units or building blocks, one of which is memory. Memory is central to intelligence because it enables an AI agent to interpret its inputs and respond, based on its experiences and the knowledge acquired [22]. It should be noted that the term memory, as used in AI and related domains, generally includes both the storage and the learning mechanisms. That is, memory includes the "facts", the relationships between them, and the mechanism used to learn, recognize, and recall based on the agent's interaction with the world/environment. Hence, memory is critical for intelligence, for it is the memory that determines an AI agent's abilities. And this research is an exploration of the implementation of memory systems for use in autonomous cognitive AI agents.

The ability to create, and subsequently modify, memories is crucial to the development and survival of a cognitive agent. This is because memory is not a single predefined thing but a dynamic concept, it is a function that enables animals to retrieve information that has been previously acquired and retained and use it to determine

response to the present situation [23]. And while researchers have postulated to and/or evidenced the existence of multiple types (or systems) of memories [24] in both human [25] and non-human animals [26], the core underlying function of the various types of memories is the same: use knowledge of previous experiences to help solve the various problems confronting the agent in its environment. That is, memory is necessary for learning. Thus, it can be postulated that cognitive agents require the ability to not only acquire, store, and recall previous experiences but also make associations, draw meaningful conclusions, and apply the acquired knowledge to novel situations. This ability to draw meaningful conclusions and apply the knowledge to novel situations is essential for all cognitive agents and becomes more critical for learning in a complex environment. An agent's abilities to draw complex, logical conclusions are directly related to its cognitive abilities, the better its cognitive abilities the better its chances of survival and growth.

A challenge in implementing a memory structure is the choice of the underlying architecture, *i.e.* the building blocks and the learning mechanism. Broadly speaking there exist two major choices: make use of tables (or other related mechanisms) as in expert systems or look towards the animal kingdom. In this work the memory architecture is inspired by observations and understandings from the animal brain. There are a variety of reasons for this, two of which are: it is well suited for parallel operation, and the organization is efficient and parsimonious. Many animals, including simple organisms, have been observed showing "intelligence" that can be considered to be a result of some form of memory. There exists a vast range of behavior that can be considered intelligent

among the various non-human animals. For example, apes have been known to manipulate objects and use tools, while bees and desert ants [27] have been known to learn how to find a source of food or navigate over long distances. Different animals have different abilities to behave intelligently. Such variations result from the biological differences between their nervous systems. The nervous system of vertebrates such as apes is more complex than that of bees which in-turn is more advanced than that of invertebrates like molluscs, whose learning abilities can be considered rudimentary at best. Thus, it can be said that the complexity of the memory structure, and its signaling mechanism, is related to the resulting intelligence and the ability to survive in more hostile environments. And, if a cognitive agent is to survive and thrive in an environment its memory abilities should, at a minimum, be "sufficient" for that environment.

It should be noted that most memory models as understood and used in cognitive agents, including in this work, are traditionally based on the idea of connections between cells (or more specifically neurons). But this is not necessary, work in systems biology is being used to model learning in single celled organisms [28]. Consequently, while the sophistication of the underlying model might not limit learning in a cognitive agent per se, it can complicate the design of its memory. Hence, while it is possible to implement learning using only chemical kinetics, it is relatively less efficient compared to using synaptic strengths between neurons, even though most synapses use chemical signaling to communicate. In this work, for efficiency and simplicity of design, a neural network model based on biologically inspired neuron and associated learning mechanism will be used.

A well designed memory architecture is one that can, by suitably changing the learning mechanisms, be used for creating various types of memory, namely long-term, short-term, episodic, semantic, *etc*. That is, the underlying structure across the various types of an agent's memory would remain the same but changes in the learning and recall mechanism would lead to different types of memories. Such memory architectures are widely observed in the animal kingdom. For example, while anatomically or structurally different regions of the human cortex are similar [29] they differ in the inputs they receive and the role they play in the cognitive process they handle [30]. Thus, it is not only possible to develop a memory architecture that uses similar underlying structure to achieve different types of memories but is, probably, highly robust, economical, and efficient. Following this principle, the memory structures used in this work will be based on a single model of the neuron.

1.3 Research Objectives and Scope

Motivated by insight about human cognition, this work is an attempt to design an associative neuron based memory system for a cognitively plausible motivated learning agent. This work is based on a few core design principles. Firstly, memory is central to intelligence. This is obvious: an intelligent agent is able to learn and learning requires memory. This can be as simple as the ability to change weights in a feed-forward neural network or a system with multiple memory modules, such as sensory, semantic, and episodic memories, interacting with each other. Learning can take various forms, supervised, unsupervised or reinforcement learning, each with their own set of challenges and all should be considered in building intelligent systems. Overfitting, building a very

complex model to approximate input data, is an issue in supervised learning and should be avoided [31]. Similarly, lack of measure of success or unique challenge and exploitation-exploration are challenges to be managed in unsupervised [32] and reinforcement learning respectively [33].

Secondly, memory is hierarchical and self-organizing in nature. Agents with simple and non-hierarchical memories, while intelligent, have limited abilities and are not capable of developing complex purposeful behavior. Hierarchical memories, in addition to pattern storage and passive information processing abilities, are suitable for continuous and active learning. Thus such agents are able to adapt to new environments (or changes in environment) and learn new skills [34].

Thirdly, it is assumed that actions of an intelligent agent are goal driven. This principle suggests that an agent acts in order to achieve a goal. The goal can be fulfilling some basic or innate need, such as hunger/energy, or a higher level abstract need, such as money for food. This principle also simplifies the design of the system by limiting actions to those that can be understood at the agent's cognitive level of development. That is, for example a designer need not predefine system states on account of actions taken by agents in fulfillment of their high-level cognitive needs such as ambition, respect from peers *etc*. A related principle is the requirement that the agent have a built-in mechanism to create goals based on motivated learning principles. Though an agent has some basic or built-in goals, creation of additional goals should be a result of the agent's interaction with its environment [3], [35].

In this work an arbitrary and functional definition of intelligence related to the concept of embodied intelligence (EI) and memory is used. The EI agent as envisioned here would learn predominantly in a goal-oriented manner. It would be able to build memory structures representing its knowledge and experiences and to use its memory in order to achieve its goals in an unknown environment. It is required that the agent should be capable of developing hierarchical memory structure while learning to solve its goals. The agent would be able to learn to resolve pains, create necessary goals and reach them for a problem similar to the one described in [3]. The agent would have at least one primitive pain that it needs to address and in this process it can form abstract pains. The number of such abstract pains and their hierarchy is dependent on the complexity of the environment. To solve the pains the agent would be capable of interacting with and recognizing its environment. An agent as envisioned in [3] would have complex sensory and motor processing abilities, similar to that in animals. But for the purpose of this work it is assumed that the memory receives symbolic inputs, processes them, and returns symbolic outputs. This assumption simplifies the process of testing the memory structure envisioned here by separating the sensor/motor aspects of the cognitive agent from its memory.

The development of an EI agent's memory structure can be simplified if the capabilities of the agent are first determined in terms of the abilities that the agent would need to successfully solve its pains and goals. The most basic capability of such an agent would be to recognize objects, this is necessary for the agent to recognize objects needed to solve its pains. The means of sensing objects is not important, only the ability to sense

the objects is required. The second capability of the agent's memory would be to form relationships between objects/chunks/bits of information in the memory and the agent's needs and actions. This relationship can be as simple as their separation in time, that is to specify their relative position in a sequence. A related capability is the ability to recognize sequences. An agent would need to recognize the sequence of operations it previously performed, this recognition would help the agent either repeat operations that were helpful or avoid those that were hindering. The fourth capability of an agent's memory would be to make predictions or anticipate based on its past experiences. This ability would help the agent to prepare for the future. It also informs the agent about unexpected, new observations that may be a subject for its learning. The capability to recognize sequences, and anticipate subsequent elements of the sequence are also critical for an agent to navigate in its environment. The capability to be creative helps the agent generate novel responses to present circumstances based on its knowledge and recent experiences and is the fifth capability of its memory. Thus, the five major capabilities of an EI agent's memory are:

1) recognize objects,
2) create semantic relationships,
3) recognize sequences,
4) make predictions and detect novelty,
5) demonstrate creativity.

The objective of this dissertation is to show that a biologically plausible model of the neuron can be used to implement memory structures that are able to demonstrate these capabilities.

Note that the preceding discussion on *object recognition*, the term "object" was not defined. From a cognitive agent's perspective, *object* can refer to any abstract concept, and is generally the result of its observations during interactions with the environment. For cognitive agent an object can also be a category of objects (like fruits) or an abstract concept (like a race). In an embodied cognitive agent its sensory inputs would activate lower levels of a hierarchy of neurons, providing objects recognition and categorization known as the symbol grounding process [36]. A full treatment of this process, object representation and symbol grounding, necessitates addressing issues of sensorimotor selection and interactions between them. This is beyond the scope of this work, and in further discussions symbolic representation is assumed.

1.4 Dissertation Organization

The overall organization of this dissertation is as follows. Following the discussion about the background, objectives and scope of the research problem in Chapter 1, proof of the sufficiency of memory capabilities suggested as necessary for a cognitive agent designed to address a well-defined application is provided in Chapter 2. A short discussion of the simplified cognitive agent model is also included in Chapter 2.

A detailed discussion of the neuron model chosen as the building block for neural networks implementing the various memory capabilities is provided in Chapter 3. This is followed by implementation of semantic, episodic, and declarative memories in Chapter 4. Chapter 4 also shows how creativity results from declarative memories. Chapter 5 describes the extension of the semantic memory described in Chapter 4 to a columnar

structure. Results of Chapter 5 demonstrate the ability of the memory structure constructed to recognize sequences, and to make predictions.

Chapters 6 and 7 describe the implementation of object recognition and multi-label classification using neural network structures and a recent learning mechanism. This is followed by a summary of the whole dissertation, restatement of the novelty and original contributions of this work and a discussion of future research in Chapter 8.

2 A SIMPLIFIED MODEL OF COGNITIVE AGENT

In the previous chapter, Introduction, the research problem, the dissertation objectives and scope were clarified. While the various memory capabilities to be implemented and tested were addressed, there remains a critical question. Are these objectives sufficient for a cognitive agent, albeit one designed with a specific task in mind? This chapter attempts to answer this question. In this chapter, following a short introduction and brief discussion on a few well-known cognitive architectures in *Section 2.1*, embodied intelligence and motivated learning will be discussed in *Section 2.2*. In *Section 2.3* a simplified model for cognitive agent, based on principles of motivated learning and embodied intelligence, will be discussed. Subsequently, in *Section 2.4* the results of tests on this cognitive agent will be discussed, with conclusion in *Section 2.5*. The results in *Section 2.4* are from a work we published [37].

2.1 Introduction

Though AI as a research field has existed for over 60 years, the goal of achieving human-like intelligence, *i.e.* true artificial general intelligence (AGI) seems elusive. Humans can effortlessly draw upon their various cognitive capabilities to successfully complete any given task, and can interrupt tasks while still being able to return to complete them later. For example, we can interrupt watching a movie to receive a call, complete the conversation, and easily return back to continue watching the movie while still remembering the plot. In addition, when faced with challenges humans draw upon their past experiences, or find some help, and can generally find an acceptable solution. Learning from past experiences humans can become experts at diverse tasks.

Research in AI has led to development of systems that outperform humans on various tasks such as playing games (Go [38], Jeopardy [6]), or analyzing vast troves of data to detect anomalies or patterns. However, most AI systems are designed to solve specific types of problems, at which they are extremely good, and they cannot integrate the many capabilities generally associated with human intelligence. One research area in AI that has strived to overcome the problem specific limitations has been the development of cognitive architectures, *i.e.* computational models or systems that attempt to create blueprints for creation of systems with cognitive abilities similar to animals (or even humans). A cognitive architecture is a general purpose, generic computational cognitive model that captures the critical structure and process of the brain for use in multi-domain analysis of behavior, and includes the general structures, divisions of modules and required relations between, fundamental representations and algorithms within modules, and other aspects of mind [39]. While a cognitive architecture provides the building blocks for creating intelligent systems it is not an algorithm for solving a problem but is a well-defined "description" of the structure and mechanisms that lead to cognition in animals.

In addition to their use in development of AI systems, cognitive models play an important role in many areas of cognitive science. Cognitive architectures are a useful tool in understanding the working of the animal brain because while psychological studies and neuroscience experiments can help, at present it seems unlikely that they are sufficient for this objective. Cognitive architectures provide researchers with means to simulate behavior based on their theories (or models) of cognitive behavior, providing

valuable insights into their assumptions and helping to improve design of animal studies. Simultaneously, cognitive architectures force researchers to specify their models of psychological and cognitive mechanisms and processes in great detail leading to clearer and consistent theories [40]. Consequently, it is understandable that there exists a great interest in the development of cognitive architectures, and over the last few decades as many as 300 cognitive architectures have been proposed [8].

Cognitive architectures differ not only in their goals, designing AGI vs. specific ability, but also in their position on various characteristics of the mind such as computational operation, embodiment, perception, memory, planning, cognition, *etc*. Contrasting between models on various distinct characteristics, Vernon *et al.* [41] group cognitive architectures into three paradigms: cognitivist, emergent, and hybrid. Cognitivist approaches, also known as information processing or symbol manipulation approach, focus on information processing using high-level symbols and are implemented as a set of if-then-rules. Emergent approaches model cognition as the process through which a system, via a process of self-organization, becomes viable and effective in its environment. Hybrid approaches are various combinations of these approaches. A brief description of a few architectures is provided below. See [8], [42] for a review of cognitive architectures and [43] for a discussion of the various issues that designers of cognitive architectures need to address.

SOAR (State, Operator And Result) [7] architecture is an example of the cognitivist paradigm and is designed to model AGI. Following the symbolic information processing approach all knowledge in SOAR is represented and stored as production

rules, organized according to the set of states representing the task. The primary learning mechanism in SOAR, called chunking, is based on psychological ideas of memory and problem solving. That is, when sequences of operations useful to solving problems are detected they are stored as chunks. SOAR architecture has evolved over time to include other learning mechanisms like reinforcement, semantic, and episodic learning. Finally, SOAR architecture has been applied in a variety of applications successfully demonstrating high-level cognitive functions such as planning, problem solving, and natural language understanding.

HTM (Hierarchical Temporal Memory) is an emergent architecture grounded in neuroscientific understanding of the human neocortex structure and organization [34]. Motivated by the hierarchical organization of the growing size of cortical receptive fields where the perceptual stimuli is diffused from bottom up, HTM is built as a hierarchical structure of network nodes implementing learning and memory functions, with each node connected with others, in its own and adjacent layers, implementing similar cognitive functions. Higher layer nodes, representing secondary and higher-level associations, receive stimuli from a large number of nodes in lower levels. HTM addressed the temporal aspects of perception, with each layer of the network trained separately to learn the spatio-temporal objects.

ACT-R (Adaptive Control of Thought – Rational) [9] is a hybrid cognitive architecture that aims to build a system capable of performing every cognitive task that a human can perform and is based on developments in cognitive neuroscience. Perceptual-motor modules and memory modules are the primary components of ACT-R. While the

perceptual-motor modules act as the interface between the system and the environment, memory modules are a store of knowledge. There are two types of memory modules in ACT-R: declarative and procedural. While declarative memory is a store of factual knowledge, procedural memory is a store of skills (how to do things). A symbolic-connectionist structure is used to store the knowledge, with the symbolic level representing the facts or procedures and the connectionist structure representing the sub-symbolic information about past usage that controls its operations. All modules in ACT-R can only be accessed through buffers that serve as temporary storage. ACT-R has been successfully used in a large number of psychological studies, especially on memory and attention.

CLARION (Connectionist Learning with Adaptive Rule Induction ON-line) is a hybrid architecture that distinguishes, and captures the interactions, between explicit (symbolic) and implicit (subsymbolic) processes [48]-[50] and is designed to aid in both development of artificial agents and understanding human learning and reasoning processes. It has four subsystems, each with a dual representation (explicit and implicit): a) action-centered subsystem (ACS), to regulate the agent's actions; b) non-action-centered subsystem (NCS), to maintain the general knowledge; c) motivational subsystem (MS), to provide motivation for perception, action, and cognition; and d) metacognitive subsystem (MCS), to monitor, direct, and alter the operations of the other subsystems. Learning in CLARION is achieved through different methods, implicit knowledge is acquired through reinforcement learning (Q-learning) or supervised approach (back-propagation) and this implicit knowledge is then used to create the

explicit knowledge through bottom-up learning. Similarly, top level rules (precoded or fixed by the designer) can guide the actions while allowing for accumulation of bottom-level knowledge providing for top-down learning. CLARION has successfully been used to account for psychological data from a variety of tasks.

2.2 Embodied Intelligence and Motivated Learning

A question that has bothered philosophers, and consequently researchers in AI, is: does cognition require embodiment? For long the dominant view in AI was that of dualism, a theory that mind and body are separate, and cognitive mechanisms are independent of the physical embodiment. But, over the last few decades there has been growing acceptance of the view that cognition requires embodiment because cognition process in the real-world quite often involves sensory perception and motor action in the fulfillment of explicit goals [44], [45]. The embodied cognition paradigm is heavily influenced by the works of researchers such as Rodney Brooks [46], [47], Andy Clark [48], [49], Rolf Pfeifer [50], Ester Thelen and Linda Smith [51] and holds that intelligence is the result of an agent's interaction, via sensorimotor activity, with its environment. Thus, cognition is considered an embodied or situated activity. The core principle of embodied cognition paradigm is that embodiment, ability of an agent to sense its environment and subsequently act/modify the same via its actions, is required for development of cognitive abilities. This is because the agent's higher level cognitive functions, such as reasoning and planning, that determine its actions are influenced by its surroundings.

While there is no single accepted definition of what constitutes cognition, it is generally accepted that cognition involves some specific capabilities such as ability to reason, plan, solve problems, *etc*. It is also accepted that cognitive agents have the ability to build internal models or representations of their environments supported by large associative memories conducive to efficient learning, prediction, planning *etc*. In addition, it is understood that a system's behavior (no matter how sophisticated) should not be used as a measure of its intelligence and therefore used as guidance for building intelligent machines [34]. Note that the terms intelligence and cognition are frequently used to describe the same thing. The term intelligence is generally used when discussing learning within a steady state structure of cognition, whereas the term cognition is used when discussing the changes in the underlying structure and includes all the mental processes in the brain.

Research grounded in embodied cognition paradigm, embodiment and the ability to interact with the environment is essential for intelligence [50], [52], [53], has led to the multidisciplinary field of embodied intelligence (EI). In general, EI is defined as a mechanism that learns to survive in a hostile environment [3]. Implicit in this definition is that EI agent interacts with the environment using motors and sensors. This definition is general enough to encompass all forms of EI agents, biological, mechanical or virtual. Also implicit is the hostility of the environment, it is the hostility (pains) of the environment that causes the agent to learn and develop intelligence. The hostility can be external (bad weather or predators) or internal (low energy, hunger or boredom)

perceived as pains or needs, and in trying to resolve these pains the EI agent learns to observe, act and develops.

An intelligent agent is motivated to learn and develop by satisfying its needs. A well designed agent would have few basic inbuilt needs, *e.g.* need for survival, but by developing new needs would be able to develop complex behavior and considerable knowledge. In general, an agent's goals are to learn how to resolve its pains, where a pain signal is generated when the agent's need is not satisfied. In addition, the agent must have the ability to not only solve pains related to its basic needs, but also to create new needs and related pains based on the existing needs. For example, if the agent would need to learn to resolve pain related to its energy level ("hunger"), then is should be able to create a new need (and the "abstract pain") as the resources needed to resolve its energy pain go down. Similarly, the agent can create abstract needs or abstract pains, *e.g.* need for money to buy food to satisfy hunger pains. Thus, theoretically this concept of abstract needs (and related pains) can be extended ad infinitum, though for a practical agent these are limited by its environment and agent's ability to explore it and find ways to realize its needs. Since they are driven by basic pains, the resulting hierarchy of needs depends on the agent and its environment. A motivated learning (ML) agent can accomplish this using a goal creation system (GCS). In addition, an ML agent would also need a well-organized memory for storing knowledge acquired, ability to recognize objects, make predictions, plan, *etc*.

All embodied motivated learning agents share a few core abilities, such as the ability to sense and act, build memories, plan and execute actions, handle pains, create

goals *etc*. But the ability to create, and subsequently modify, memories of past actions is crucial to the development and survival of the agent because it is memory that enables animals to retrieve information that has been previously acquired and retained [23]. The structure of the memory system for the cognitive agent in this work is based on understanding of memory in humans.

2.3 Cognitive Agent Model

2.3.1 Background

A variety of systems/architectures for memory in cognitive agents have been proposed in the literature. Some of the earliest approaches had agents with one type of associative memory, namely semantic memory, but differentiated the memories in terms of the duration for which those associations were maintained, that is long-term vs. short-term. With increased understanding of the structure and working of the animal brain, especially human, researchers started experimenting with agents having other types of memories, especially episodic memory and working memory. Over a period of time more complex learning mechanisms, with units such as planning, reasoning, attention switching, motivation, and goal creation, were proposed and studied in addition to developing various types of memories. The understanding being that to solve complex problems, *e.g.* navigation in autonomous vehicles, it is necessary to look at the animal kingdom for answers, and that animals have a very complex learning mechanism with various interlinked subsystems working in parallel to solve these problems.

Researchers have used various approaches in their attempts to solve problems using knowledge of animal cognition. While various cognitive architectures [4], [7], [54],

[55] that can be used to build AI systems have been proposed, these need neural network models that can implement the requisite functionality. Consequently research into neural networks has developed and various models of neurons have been proposed and tested [56]. It has been found that models with behavior similar to biological neurons are computationally expensive for use in large networks [56]. Over time neural networks, from simple single layer feedforward networks to highly complex networks with multiple hidden layers and feedback have been proposed [57]. While there have been incremental enhancements to neural networks that have improved their performance, such networks have not been able to achieve true human intelligence. The lack of success on more challenging ML tasks combined with increased availability of computational power has led to interest in deep learning [58] and theories such as hierarchical temporal memory (HTM) [59] based on cognitive neuroscience.

There are various algorithms that attempt to explain how learning occurs. The Hebbian learning rule, based on Hebbian theory of dynamics of biological systems [60], is one of the oldest algorithm and states that: synaptic connection between two neurons is strengthened if their activations are correlated and weakened if uncorrelated [61]. Hebbian learning and its variations or extensions are still widely used. Spike-timing-dependent plasticity (STDP), another widely used algorithm, attempts to account for the relative timing of a neurons output and input activities: an input to a neuron is made stronger if a spike on that input, on average, occurs immediately before that neuron's output spike, whereas it is made weaker if that neuron's output tends to spike, before its input spikes. Both STDP and Hebbian learning rule has been demonstrated and used in

neural networks [62]. Self-organizing maps [63], recurrent neural networks [64], long-short term memory (LSTM) [65] are some widely used neural network based approaches. In addition there exist a variety of statistical learning [66] and reinforcement learning [33] algorithms that have been developed and used in applications of memory systems.

2.3.2 Model and Assumptions

Using understanding of human cognition as basis, this work attempts to build a memory system for an embodied intelligent (EI) agent capable of learning to survive and develop in a hostile environment. The EI agent envisioned here, would be capable of resolving its pains, create additional pains/goals based on its experiences and learn to navigate in a structured but unknown environment.

Consider an agent with hunger as its primary need that is required to survive in the city, similar to the agent described in [3]. In addition, some of the resources needed by the agent that exist in the environment can run low and the agent will need to learn how to restore them. The agent has a specific number of pains (based on sensory inputs) and motors skills. Only a few of the pain-motor pairs (goals) lead to valid and useful results. Table 2-1 shows some useful goals for each pain and their effect on the environment and resources.

Table 2-1. Some meaningful pain-motor pairs and their effects on resources.

Pain	**Goal**	**Increase**	**Decrease**
Hunger	Eat Food	Energy level	Food supplies
Low Food Supply	Buy Food	Food supplies	Money in hand
Less money at hand	Withdraw from bank	Money in hand	Money in bank
Low bank balance	Find work	Money in bank	

Note that only a few of the possible pain-motor pairs have been shown. Other pairs do not advance agent goals and were omitted, however uneducated agent can try to use them in various situations wasting both resources and time to perform useless actions. This scenario is based on a simple agent with only one primary pain: hunger. All other pains, low food supply, less money at hand *etc*. are abstract pains that the agent creates. The idea here is that, over a period of time the agent uses its energy and starts to feel hungry. To handle this pain the agent tries various actions and learns that eating food satisfies its hunger. After a few iterations of eating to satisfy its hunger pain, the agent observes that its food supply is starting to run low. The agent then creates an abstract pain for its food supply. Now, in addition to trying to satisfy its hunger it tries to find food and learns to buy food from the store. This action (buying food in store), while restoring its food supply and resolving "low food supply" abstract pain causes a new abstract pain, "less money at hand", to be created. This process of the agent learning to resolve a pain by performing certain actions and using its resources thus creating abstract pains, continues to the extent made possible by the complexity of the environment. The agent described here is called a motivated learning agent, *i.e.* the agent is motivated to learn the goals to resolve its pains, and is capable of creating new motivations and abstract goals. The scenario described here is a simple case scenario and it is used only for demonstration purpose. In addition, the goal of this work is not to design such a motivated learning agent (this has previously been proposed and implemented [3], [37], [4], [67]), but to design a suitable memory system for such an agent. This system will have the five major capabilities, discussed previously, for such a memory: 1) recognize

objects, 2) create semantic relationships, 3) recognize sequences, 4) make predictions, and 5) demonstrate creativity.

A simplified computational model of the cognitive agent, based on [68], is shown in Figure 2-1. The cognitive model of the agent consists of a central executive with attention switching, planning, action monitoring and motivation & goal selection units. Attention switching between objects, tasks or pains *etc*. is handled by the attention switching unit, while the actual selection of goals and motivations on which the agent acts is performed by the motivation & goal selection unit. The planning is performed by the planning unit whereas monitoring of actions is tracked by the action monitoring unit. The semantic memory acts as a store of all the knowledge that the agent has acquired and is used by other units to select the most appropriate task based on the agent's experience. The rewards & subconscious processing unit keeps track of the various pains, rewards received and performs subconscious actions, such as reflexive actions. The sensory and motor processing units are responsible for sensing and motor actions. For the purpose of this work we will concentrate on design of semantic memory, and episodic memory. Implementations of other units, where needed, will be taken from author's previous work [37].

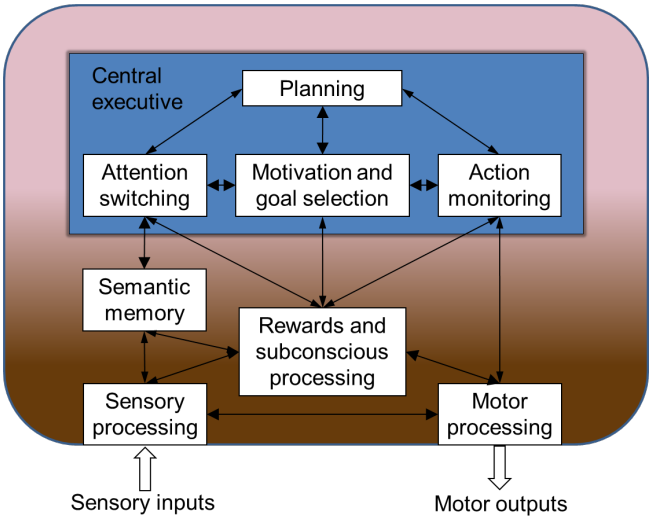

Figure 2-1. Simplified computational model of cognitive agent.

In general, the semantic memory provides an agent with the ability to acquire, store and relate factual information. This relationship between facts, known as semantic knowledge, provides an agent with the ability to think about objects and plan motor actions that can be performed on those objects. Note that an agent detects an object through processing of its sensory inputs based on its existing knowledge. That is, an object is represented as a set of sensed features that the agent has "found" to be relevant to *its* concept of that object. A cognitive agent can use such feature representations to form higher-order concepts. For example, the concept of *food* might consist of objects that can relieve the agent's *hunger*. This concept formation can be extended ad infinitum.

The semantic memory unit in this agent plays a similar role, it stores information about the facts and relationships it has learned. Here the facts could be objects the agent encounted in the environment, sensorimotor pairs learned, and learned relations between them. This knowledge forms the basis for an agent's planning and action formulation,

and is updated to reflect new experiences. Here the semantic memory is implemented as a hierarchical self-organizing structure.

The episodic memory is a store of temporal events or episodes, and the spatio-temporal relations between those events. Episodic memory is built on the semantic memory, *i.e.* it uses the concepts or knowledge in the semantic memory to represent its underlying components. In humans it is understood that the act of retrieving information from the episodic memory can act as a type of input to the episodic memory and thus change it [69]. Thus episodes may be recalled, strengthen, and fully or partially forgotten. This feature will not be implemented in this work, *i.e.* retrieval of information will not change the episodic memory. In humans there exist sub-categories of memories, especially episodic memory, and are generally referred to as short-term and long-term memories. The underlying assumption is that overtime links representing connections that are not used tend to get weaker, eventually dying, *i.e.* lead to forgetting. In this work forgetting will not be explicitly implemented, *i.e.* links once created will not be deleted, but overtime the relative weights of links not used can go down leading to weakening of relationships.

In this proposed work, we are interested in developing suitable memory structures that will enable an embodied cognitive agent to perform tasks, *i.e.* reach it goals, via learning in an unknown environment. The requirement to learn in an unknown environment assumes that the agent is capable of sensing the changes in its environment and is able to build a mental map of its environment. The memory unit, in addition to being able to store such a map, should be able to recognize scenes or landmarks in the

environment by comparing its sensory inputs to its memory. In [37] such an agent was implemented and shown to learn to reach its goals, *i.e.* solve/manage its pains. Additional details of the memory structure used in this work are provided in *Section 2.4*. For simplicity, it is assumed that the visual inputs are used for this purpose. It is also assumed that the task of recognizing scenes or landmarks can be simplified to the task of object recognition, a reasonable assumption because any scene can be partitioned into sub-regions and the sequence of sub-regions (ordered according to a fixed logic) can be used to represent the complete scene. For the purpose of this work we will use handwritten digit recognition on the MNIST dataset [70] as a representation of the object recognition task.

2.4 Associative Memory – Using Symbolic Inputs

In [37] the memory implementation is based on a neural-network approach and uses weighted links to store associations, as shown in Figure 2-2. The nodes C_i represent the concepts learned by the agent. These concepts can be actual objects in the environment, low-level sensory inputs, abstract concepts, models or representations. For example, the concept nodes can represent objects such as “apple” and “banana”, or the abstract concept of “fruit” that encompasses a variety of actual fruits. The nodes A_i represent the motor actions that an agent is capable of performing on objects, and P_i nodes refer to pains that an agent has to learn to for survival and/or growth. Action-object pairs are called goals, useful action-object pairs help the agent in handling one (or more) of its pains. For example, “Eat Banana” is the action “Eat” performed on an object

"Banana" and can help the agent in handling "Hunger" pains. The agent learns the useful action-object pairs based on changes in its pain levels.

The weights w_{PA}, w_{AC} are employed to describe the association between nodes. The weight w_{PA} shows how much a particular action could help in reducing the associated pain. The greater the weight, the more acute effect an action has on the pain. Similarly the weight w_{AC} tells how useful an action is for a given concept. For example the weight between "Hunger" pain and "Eat" action would be considerably higher than the weight between "Hunger" and "Sleep" and similarly the weight between "Eat" and "Banana" (or the concept of "Fruit" or "Food") would be considerably higher than between "Eat" and "Mattresses". Note that complex relationships, such as "Wake your mom from her sleep and ask for something to eat", are possible but for simplicity neither shown nor discussed here.

Initially, weights of the links are set randomly in the interval of 0-α_A and 0-α_C respectively, where α_A and α_C are weight upper limits guaranteeing that all weights are appropriately normalized. Each time the agent acts the inputs to the memory may change. An action that leads to a decrease of a pain causes an increase in the associated weights. However, if the action performed does not lead to any decrease in pain then the weights are decreased.

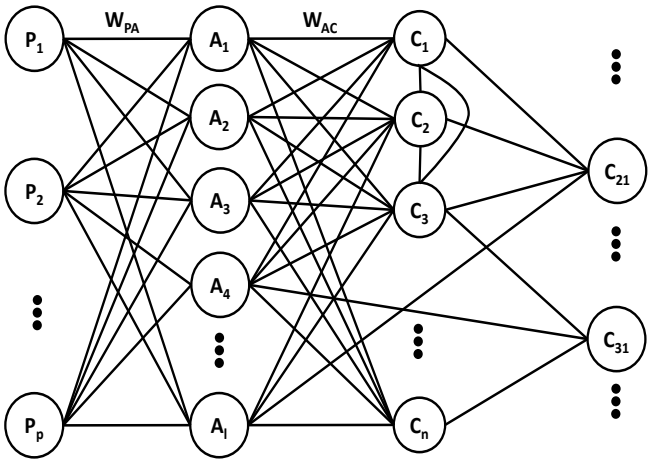

Figure 2-2. Neural network structure of the memory.

2.4.1 Simulation Results

This concept of memory as a neural network with adjustable weights was successfully implemented in [37]. Figure 2-3 shows a MATLAB GUI [37] that illustrates the operation of the agent's memory, and shows the agent performing a visual saccade. Visual saccades, generally known as saccades, are the rapid orienting movements of the eyes which change the focus point of a visual target. The amplitude of visual saccades can range from small movements made while reading to the much larger movements made while looking around [71]. Starzyk and Prasad [68] proposed a modification to the visual saccades and call it "mental saccades". [68] defines mental saccades as a processes similar to visual saccades but instead of changing the visual point of fixation mental saccades refers to changing mental point of fixation, *i.e.* for example changing the attention focus between "pain", "action", "concept", *etc*. that the agent is thinking about is defined as mental saccades.

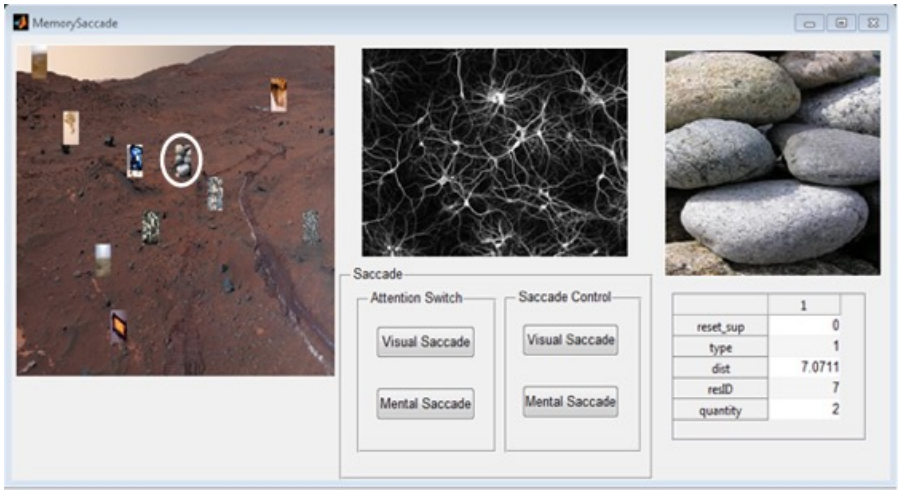

Figure 2-3. Result of visual saccade.

Table 2-2. Associations for mental saccade.

Mental Associations	**Definition**
Pain	Low Supply – Small Rock
Goal	Break Medium Rock
Motor Action	Break
resID	Small Rock
resID	Energy Pill
Motor Action	Go to Base

In Figure 2-3 the agent's environment is shown on the left and the object under attention is highlighted, by a white circle around it, and is shown on the right (medium size rocks). A partial output of the memory is displayed near the lower right corner. The output consists of: type of saccade performed (type = 1, which represents visual saccade);

distance of the object from the agent (dist); name of the object (resID = 7, which represents medium size rocks); and quantity of the object present. Figure 2-3, shows a snapshot taken when the agent running low on its supply of small rocks comes across medium rocks. Table 2-2 shows the various pains, goals, objects, *etc*. that are associated with the object currently under attention spotlight (medium rocks). If the agent performs mental saccade, *i.e.* searches its associative memory, for associations related to the concept/object medium rocks it knows that the medium rocks can be broken into small rocks and this action-object pair, break-medium rocks, would provide it with a supply of small rock and consequently decrease the pain, due to low supply of small rocks. Continuing this process of mental saccades, the agent knows that it will need "energy pills" to break medium rock and that it can find them at its "base". When the agent performs a mental saccade on the object under the attention spotlight, it can either saccade only through associated objects (or actions, *etc*.) or saccade through other objects (or actions) associated with the object in focus as in the example shown above.

In the memory building approach discussed previously the agent was successfully able to learn to handle its pains and resources. As shown in Figure 2-4, the agent is able to successfully control both primitive and abstract pains, displayed separately for readability. We can observe that the agent is able to form high-level abstract pains, such as the "Low Camera Memory", the "Lack of Pills", the "Lack of Small Rocks" and the "Lack of Shelter" (Bottom in Figure 2-4) and manage them effectively.

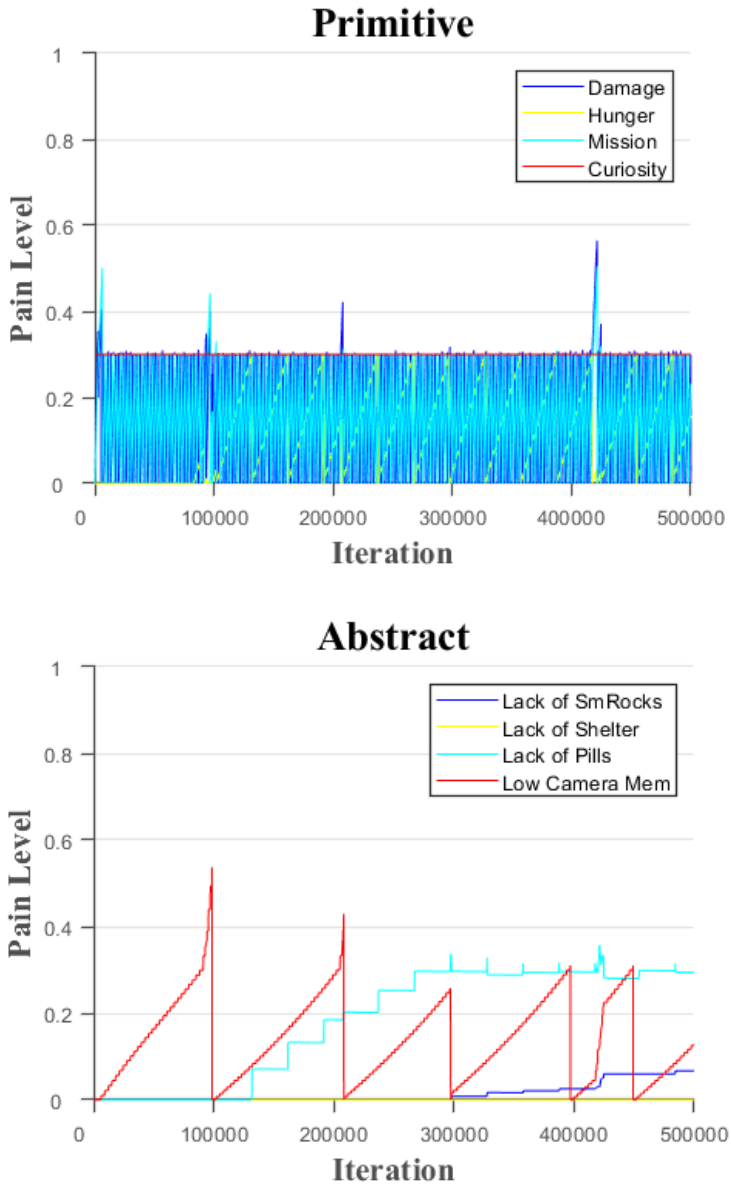

Figure 2-4. Primitive (top) and abstract (bottom) pain levels.

The associative memory implemented for this agent show the following capabilities: recognize objects, create semantic relationships, recognize sequences, make predictions, and show creativity. While the implementation of the first three of these capabilities is obvious, ability to recognize objects, *e.g.* medium rocks, in the

environment and learning the relationships between them, break medium rocks to obtain small rocks, the last two can be inferred. If an intelligent agent has learned that it can break medium rocks to obtain small rocks it can be inferred that the agent can make some predictions based on its experiences. Demonstrating agent's creativity is difficult. For example, during the learning process the agent tries various actions before it learns the useful sensorimotor pairs. Can this be considered demonstration of creativity? In the above example the agent can solve its pains of *lack of small rock* and *lack of medium rocks* by breaking *medium* and *large rocks* respectively. Assume the agent first learns that breaking *medium rocks* provides it with *small rocks*. Subsequently, if when faced with *lack of medium rocks* the agent attempts breaking *large rocks*, it might be considered a demonstration of creativity, which is defined as being imaginative or showing novel understanding [72]. Another example of creativity is when an agent associates similar meanings or categories of objects and applies properties learned for one type of objects to those that are similar to the objects it previously learned.

2.5 Conclusion

In this chapter, a simplified model of a cognitive agent based on principles of embodied intelligence and motivated learning was implemented. It was shown that such an agent with the following memory capabilities: recognize objects, create semantic relationships, recognize sequences, make predictions, and exhibiting creativity, can successfully learn to achieve its goals. These memory capabilities are the same capabilities whose implementation is the objective of this dissertation.

3 ASSOCIATIVE PULSING NEURON MODEL

In the previous chapter, a cognitive agent model was shown to successfully learn to reach its goals. It was also shown that the agent's memory capabilities were the same as the capabilities of memories whose implementation is the objective of this dissertation work. While in Chapter 2 the memory used symbolic inputs, a practical implementation requires the ability to recognize sensory inputs. In this work, this is implemented using neural networks. But what model of neuron to use? Following a brief introduction to the developments in the field of neuron modeling in *Section 3.1*, the implementation of the neuron model chosen for this work is discussed in *Section 3.2*. Subsequently in *Section 3.3* neural mechanisms, namely threshold growth, axon growth and synaptic fatigue, are discussed. Finally in *Section 3.4* conclusions are discussed. The work neuron model and its mechanisms described in this chapter has been published by us in [73].

3.1 Introduction

Memory is key to the growth, development, and survival of a cognitive agent. This is because memory in cognitive agents is not only a storage of facts and rules (like in expert systems), but is also a storage of associations between the facts, associations that change overtime as new information is acquired or the agent's needs, emotions, and environment changes. Hence the memory in a cognitive agent is the storage of not just information but of knowledge about the environment, past experiences, and associated concepts, objects and events. But how should such a memory be implemented in cognitive agents? While one solution is to use look-up-tables, indexes, and other

techniques developed for use in conventional digital computers, another solution is to look towards neurophysiological research, and this is the approach taken in this work.

There has been great interest among researchers in trying to understand the learning process that takes place in animal brains. The discovery during the late 19th century [74], that the nervous system is composed of discrete individual cells, called neurons, led researchers to trying to model the said neuron based on experimental observations of animals or animal tissue. The earliest models, like the integrate-and-fire [75] and McCulloch and Pitts neuron [76] (although crude and have various limitations that have been addressed by other researchers,) are still widely used. The Hodgkin-Huxley model [77] based on the squid giant axon was the first true model of a biological neuron, but due to its computational complexity [56], [78] is not suitable for use in large neural networks. Researchers have also created biologically inspired models, models with characteristics like, but not the same as, biological neurons, making them computationally feasible for large networks. Rosenblatt's perceptron [79] is probably the most widely used biologically inspired model of artificial neural network. Spiking neuron models [56], [80], [81] are a widely researched class of biologically inspired models that are more realistic, compared to the traditional perceptron models, and incorporate timing and membrane potential ideas from biological neurons. And such models have been widely used in artificial neural networks in various applications.

In human brain the knowledge is stored as a combination of neurons and synaptic connections between them, the strength of the synaptic connections depends on both the frequency and sequence of activations of the neurons that they connect. This knowledge

forms automatically, and is developed, expanded, and changed over an individual's life-time. When the human brain receives inputs from the environment, it activates neurons representing the various details of the environment that is sensed and subsequently these neurons trigger contextual associations defined by the present needs, emotions, previous activations, *etc.*, recalling previous experiences and knowledge to help determine a response. Changes in the input received change both the recalled associations and the data enabling learning, adaption, and intelligent behavior. While the human brain recalls the most similar facts and associations, it is capable of recalling and applying any and all facts and associations previously learned that are even partially similar or related to the present inputs. Consequently, the triggered associations can provide a means for exhibiting creative behavior. Memory implementation in this work is, broadly speaking, based on mechanisms grounded in this understanding of the working of human brains.

The availability of a great many models of neurons, including many that are biologically inspired, meant that it was possible to select and make use of an existing model rather than develop and test a new model from scratch. In this work the selection of the model was based on two considerations: a) biologically inspired, and b) computationally feasible. A strength of biological nervous systems is their ability to generalize their knowledge, including facts and associations, and apply it to novel situations. That is, biological nervous systems are capable of creativity. And though artificial neural networks are capable of generalization and creativity they generally are not comparable to human-like abilities. This is due to the limitations of artificial neural networks. They generally use simple models of neurons and at most account for elements

of biological systems such as receptors, glial cells, and interneuronal space to a limited extent [81]. The following are some of the limitations generally present in many of popular artificial neurons and neural networks based on them [81]. First, the use of continuous activations functions such as sigmoidal or radial for neurons is an approximation that prevents representation of groups of combinations and does not account for the relaxation and refraction after excitation and activation respectively. Second, they do not account for the timing influences generally present in biological neurons. That is, the artificial neurons synchronize the influences of all input signals while in biological neurons the differences in activation times are very important. In addition, differences in the moment the input signals are processed are important due to their possible influence on subsequent activations of connected neurons. Third, they do not account for all possible means of interaction between biological neurons. For example, biological neurons can interact through biochemical means, in the form of hormone transmission, via cells in the interneuronal space.

A biologically plausible and computationally less demanding model of the neuron that addressed the limitations of artificial neuron models described above was proposed by Dr. Horzyk in [81]. In [73] we modified this associative neuron model by changing the neuron's activation threshold to reflect frequency of its use, adopting self-organization of neurons to requirements of episodic memory, considering influence of fan-out on neuron activation, and introducing synaptic fatigue. The modified model from [73] is described in *Sections 3.2* and *3.3* and is used to implement the memory structures in this work.

3.2 Associative Neuron

Artificial neural networks (ANN), often referred to as “neural networks”, are motivated by the recognition that the human brain is a highly complex, nonlinear, and parallel information-processing system. The brain’s structural organization allows it to accomplish certain complex tasks (*e.g.* object recognition, perception, and sensorimotor control) orders of magnitude faster than is possible with digital computers [61]. In the human brain this is made possible by the nervous system, consisting of simple processing units like neurons that support learning via adaptation to an individual’s experiences. ANNs attempt to approximate this process mainly through the use of artificial neurons (or specifically models of neurons) and modification of synaptic strengths (*i.e.* synaptic or connection weights) between them. While various models of neurons have been proposed in literature, they can all generally be approximated to that shown in Figure 3-1 and consist of three basic elements: 1) synapses or connection links characterized by individual weight or strength; 2) an adder for summing the weighted input signals; and 3) an activation function that limits the neuron’s output to some finite range. While the model in Figure 3-1 includes an external signal bias signal, b_k, this signal is not always present, *i.e.* $b_k = 0$. The bias can, depending on whether it is positive or negative, increase or decrease the input of the activation function, respectively.

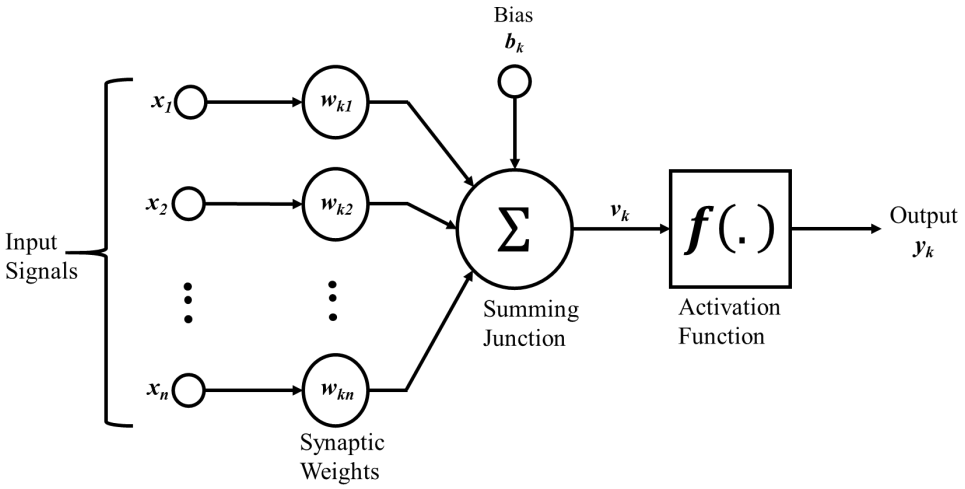

Figure 3-1. Model of a neuron.

Mathematically, this neuron can be described by the following equations.

$$v_k = \sum_{i=1}^{n} w_{ki} x_i + b_k \tag{3-1}$$

$$y_k = f(v_k) \tag{3-2}$$

where x_1, x_2, …, x_n are the input signals; w_{k1}, w_{k2}, …, w_{kn} are the synaptic weights; v_k is the linear combination of the weighted input signals and the bias; $f(.)$ is the activation function; and y_k is the output signal of the neuron. Based on the "activation function", models of neurons can be classified into three generations [82]. The first generation models, sometimes called threshold gates or circuits, can only give digital outputs, while the second generation models are capable of producing a continuous set of possible output values due to the use of activation functions such as sigmoidal function [82]. Experimental evidence shows that biological neural systems can use the timing of single "spikes" (*i.e.* action potentials) to encode information [83], [84], and while the second generation models are biologically more realistic, they do not account for time coding.

Consequently, this let to study of the third generation models, called spiking neurons, that can encode information by using the time difference between pulses [82]. Spiking neurons are more realistic, and better model biological neurons, but are seldom used in computational intelligence because of the difficulties involved in training and modeling them efficiently [80].

It has long been understood that neural functions are a result of activation of groups of neurons [85], and such interaction between neurons is the basis of all neural network models. This leads to a fundamental question, what impact should a neuron have on its surroundings? Here surroundings are defined by the connected elements and other neurons that are close in space. *Plasticity* permits the human brain to develop and adapt to its environment [85], and may be accounted for either by the creation of new synaptic connections or modifications of existing synaptic connections [61]. While there exist no general rules in ANN that define plasticity of neurons, provide new neurons or new connections and change the structure or parameters of already created neurons and synapses, most methods adjust a structure experimentally, adding new elements when the result of training are insufficient or removing them when a network overfits [61]. In this work, we use a biologically plausible mechanism which is defined by activations of neurons and the time that elapses between these activations. This *associative pulsing model* of neuron focuses on properties of biological neurons that enable cooperation in representation of frequent combinations of input signals. That is, the focus of this model is on the reproduction of functional aspects of biological neurons, such as association, plasticity, and neurogenesis, that enable development and adaptation of neuronal

structure from scratch, with the internal processes managed and processed using Internal Process Queues (IPQ) and a Global Event Queue (GEQ) [86]. IPQ is a sequence of changes to a neuron's internal state dependent on its external stimuli and previous internal states, with the internal states of the neurons updated only at the end of internal processes that are supervised by the GEQ. This is different from spiking models where the focus is on the accurate reproduction of the biological neuron and processes in its membranes, such as electrical potentials, and complex mathematical functions are used to define the internal processes. Note that, for simplicity future references to this model will use the term *associative neuron*.

A neuron's input signals can have an external (sensory) or internal (neuronal) origin. In this model, each neuron represents the combinations of input signals that activate it. The combinations of input signals can be spread over time, *i.e.* input signals are successively added over time. An automatic recovery processes that relaxes or refracts neurons over time helps discharge neurons and balances the accumulation of input activations. If a neuron reaches its spiking threshold it is "activated" or "fires" and subsequently starts a refraction process that, temporarily, makes the neuron be not or less sensitive to input signals, enabling other competing neurons to activate. But, if the neuron does not reach its threshold it starts to relax and gradually returns to its resting state. This gradual return means that context of previous input signals decreases according to the time elapsed since the neurons excitation. In addition, neurons in excited states can not only be activated more quickly, if other input excitations are received before they return to their resting state, but such neurons highlight potential temporal context.

Plasticity enables biological neurons to change their activation frequency. As the frequency of a neuron's activation is increased, *e.g.* due to its role in multiple combinations of input stimuli, its sensitivity to input signals decreases. In this model, the neuron's activation threshold model's sensitivity of the neuron to various combinations of input stimuli, allowing each neuron to specialize and be reactive to only those combinations of input stimuli which it represents and reacts to. Variable and conditional updating of sensitivity thresholds allows for controlling the set of input combinations that are represented by the neuron. Changing of this threshold enables a neuron to specialize its function and role in the network. Thus, the main idea of this associative model of neurons is to enable neurons to represent various input combinations, specialize neurons in their representation and connect neurons to emphasize their spatio-temporal associations.

While biological neurons work and update their states concurrently and asynchronously in time [87], [88], in contemporary models of neurons computations are sequentially processed in discrete steps during which the weights are updated [61]. Moreover, many of internal neuron processes in biological neurons are temporal, enabling them to take into account context of previous stimulations for events associated in time that should have an impact on parameters and plastic changes in neurons. Further, while artificial neurons use various artificial rules to connect neurons or construct a neural network, biological neurons can, due to their plasticity, update their structures and parameters according to their activities and their frequencies in time [87], [88] to represent processed data. This new associative model of neurons incorporates this natural

ability of biological neurons and hence is conditionally plastic, works and updates its synaptic connections concurrently and in real time. In this model, each associative neuron is in one of six states (Figure 3-2): resting, charging, relaxing, activation, absolute refracting, or relative refracting, which are decided by its internal excitation level and possible external stimulations.

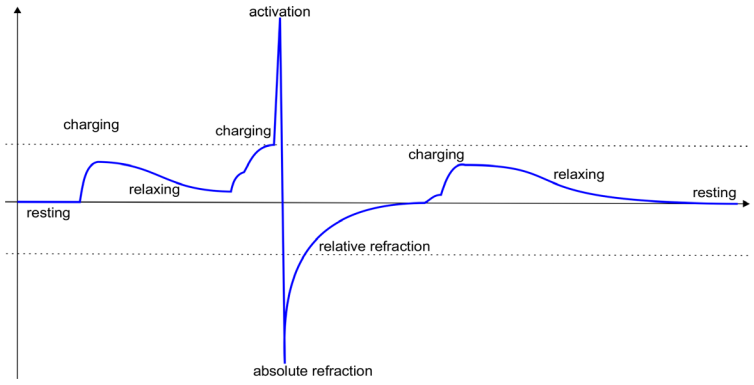

Figure 3-2. States of the associative model of neurons.

Equations (3-3), (3-4), and (3-5) evaluate an associative neuron's excitation levels during charging, relaxing, and refraction periods respectively,

$$X_{N_i}^{t+\Delta t} = X_{N_i}^{t} + \left[\sum_{N_m \to N_i}\left(X_{N_m}^{t} \cdot w_{N_m,N_i}\right)\right] \cdot sin\left(\frac{\pi \cdot \Delta t}{2 \cdot \Delta t^{C}}\right) \tag{3-3}$$

where t is the time when the presynaptic stimulation started to influence postsynaptic neuron N_i, Δt – is the interval from time t when neuron N_i started its charging $\mathrm{t} < \Delta\mathrm{t} \leq \mathrm{t} + \Delta\mathrm{t}^{\mathrm{C}}$, Δt^{C} is the period of time necessary to charge and activate postsynaptic neuron N_i after stimulating synapse between neurons N_m and N_i (here $\Delta t^{C} = 20$ ms, w_{N_m,N_i} is the synaptic permeability – a component of the synaptic weight,

$$X_{N_i}^{t+\Delta t} = X_{N_i}^{t} \cdot \frac{1}{2} \cdot \left(1 + cos\left(\frac{\pi \cdot \Delta t}{X_{N_i}^{t} \cdot \Delta t^{R}}\right)\right) \tag{3-4}$$

where $t < \Delta t \leq t + \Delta t^R$, Δt^R is the maximum period of time during which postsynaptic neuron N_i relaxes and returns to its resting state after its charging that was not strong enough to activate it (here $\Delta t^R = 300\ ms$),

$$X_{N_i}^{t+\Delta t} = X_{N_i}^{t} \cdot \frac{1}{2} \cdot \left(1 + cos\left(\frac{\pi \cdot \Delta t}{\left|X_{N_i}^{t}\right| \cdot \Delta t^F}\right)\right) \tag{3-5}$$

where $t < \Delta t \leq t + \Delta t^F$, Δt^F is the maximum period of time during which postsynaptic neuron N_i finishes its refraction after activation and returns to its resting state (here $\Delta t^F = 60\ ms$).

The synapse model described here distinguishes between presynaptic and postsynaptic neuron influences that determine a final synaptic weight:

$$w = b\ c\ m \tag{3-6}$$

where

- b is the behavior factor that determines how the synapse influences the postsynaptic neuron ($b=1$ when this influence is excitatory and $b=-1$ when is inhibitory),
- c is the synaptic permeability that specifies how strongly the input stimulation influences the postsynaptic neuron considering elapsed time between activations of pre- and postsynaptic neurons,
- m is the multiplication factor that determines how strongly this stimulation should influence the postsynaptic activity due to the frequency and importance of the association defined by training sequences and their repetitions.

The presynaptic influence is determined by the synaptic efficiency δ_{N_m,N_i} of a synapse between neurons $N_m \rightarrow N_i$ which is defined as:

$$\delta_{N_m,N_i} = \sum_{\{(N_m,N_i)\in S^n \in \mathbb{S}\}} \left(\frac{1}{1+\frac{\Delta t^A - min(\Delta t^C, \Delta t^A)}{\Delta t^R}} \right)^{\gamma} \tag{3-7}$$

where

Δt^A is the period of time that lapsed between stimulation of synapse between N_m and N_i neurons and activation of postsynaptic neuron N_i during training,

Δt^C is the period of time necessary to charge and activate postsynaptic neuron N_i after stimulating synapse between N_m and N_i neurons (here $\Delta t^C = 20$ms),

Δt^R is the maximum period of time during which postsynaptic neuron N_i relaxes and returns to its resting state (here $\Delta t^R = 300$ms),

γ is a context influence factor changing the influence of the previously activated and connected neurons on the postsynaptic neuron N_i (here equal to 4),

S^n is a training sequence during which activation of presynaptic neuron N_m and postsynaptic neuron N_i were observed,

$\mathbb{S}$ is the set of all training sequences used for adaptation.

Using (3-7) the synaptic permeabilities are computed for all outgoing synapses by one of the following methods:

linear permeability formula

$$c = \frac{\eta}{2\eta - \delta} \tag{3-8}$$

square root permeability formula

$$c = \frac{\sqrt{\eta\delta}}{\sqrt{\eta\delta}+\eta-\delta} \tag{3-9}$$

quadratic permeability formula

$$c = \frac{\eta\delta}{\eta\delta+\eta^2-\delta^2} \tag{3-10}$$

proportional permeability formula

$$c = \frac{\delta}{\eta} \tag{3-11}$$

power permeability formula

$$c = \left(\frac{\delta}{\eta}\right)^{\frac{1}{k}} \tag{3-12}$$

where η is a number of activations of a presynaptic neuron N_m during training, δ is a synaptic efficiency computed for this synapse (3-7), and $k > 1$ is an integer.

If the context of presynaptic neurons activity is unique and represents a full subsequence of any training sequence (Figure 3-3) it should be able to activate the neuron representing the next element in that sequence. If not, it means that the existing connections are too weak and should be increased. Thus, synapses between neurons are multiplied and strengthen by the postsynaptic neurons that were supposed to be activated but were not. A simple rule is used: If the neuron is not activated by previously activated neurons that represent the first part of a sequence ($S_1 \to \cdots \to S_L$), then the synaptic weights between all activated presynaptic neurons $N_{S_1}, \ldots, N_{S_L}$ (representing the context) and this neuron $N_{S_{L+1}}$ should be increased. In order to correctly compute necessary multiplication of synaptic connections between presynaptic neurons and the postsynaptic neuron we have to compute a postsynaptic neuron total excitation $X_{N_i}^{S_1+\cdots+S_L}$.

The multiplication factors are computed using:

$$m = \frac{\theta_{N_i}^t}{X_{N_i}^{S_1+\cdots+S_L}} - \frac{x^{LAST}}{2} \tag{3-13}$$

and if the computed multiplication factor (3-13) is bigger than the threshold of the postsynaptic neuron it is reduced to it,

$$m \leq \theta_{N_i}^t \tag{3-14}$$

where

$\theta_{N_i^n}$ is the activation threshold of postsynaptic neuron N_i (here $\theta_{N_i^n} = 1$),

x^{LAST} is the last postsynaptic stimulation made by activated presynaptic neurons to the postsynaptic neurons.

3.2.1 Sample Network Structure

The semantic memory investigated in this work is based on the idea of active neuro-associative knowledge graph (ANAKG) that can represent and associate sequences of objects or classes of objects [81]. The knowledge graphs are obtained dynamically by adding associative neurons and changing their synaptic connections based on the input sequences and activation levels of presynaptic and postsynaptic neurons. If the updated synaptic weights provide incorrect activations of postsynaptic neurons, then inhibitive connections are created between the previously activated neurons and the incorrectly activated neurons. The gradual activation and relaxation of ANAKG neurons enable them to represent elements (objects) in terms of the previous elements of the sequence and neuron activations [81].

ANAKG networks are built from associative pulsing neurons, with receptors sensitive to input stimuli and effectors transforming neuronal stimuli into output data [81], [89]. The associative neurons can quickly adapt to represent any given set of training sequences of elements in a neural graph structure which integrates and associates

them. ANAKG networks require only a single presentation of each training sequence to create the neuronal structure. ANAKGs training process is much faster than traditional ANN approaches, and its computation much easier than spiking NNs. ANAKG training process uses synaptic efficacy (3-7), computed using the time elapsed between activities of the pre- and postsynaptic associative neurons that were activated in temporal proximity. The higher the temporal proximity, the higher the impact on synaptic efficacy is. Synaptic efficacy is also dependent on the frequency of synaptic stimulation to the postsynaptic neuron. Consequently, the adaptation process in ANAKG network is significantly simpler in comparison to networks based on other neuron models, both spiking and those using nonlinear activation functions [90].

This section describes the creation of a simple ANAKG network structure using the associative neuron model described earlier. The following sequence set is used is create the network: *I have a monkey. My monkey is very small. It is very lovely.* The developed ANAKG structure is shown in Figure 3-3. Here, each associative neuron represents a single word, and the numbers under their names represent their number of activations (η) during the training phase. These numbers are also equal to the number of occurrences of each word in all training sequences. The small circles represent the postsynaptic elements of the synapses, and a red dot inside them means that the value of the synaptic weight is equal to the activation threshold of the postsynaptic neuron (θ), *i.e.* the activation of this synapse is sufficient to activate the postsynaptic neuron. Whereas numbers in the postsynaptic elements (small circles) represent synaptic permeability

values (w) as a percentage of the threshold value (θ). Similarly, the crescent shape denotes the presynaptic elements and shows the direction from which the stimuli come.

Figure 3-3(a) shows the neural network following the presentation of the first sentence: *I have a monkey.* Following the process described in [81], for each word in the sentence, we first check if there exists an associative neuron that reacts to the presented word. If none of the existing associative neurons are activated, a new associative neuron is created. This process is repeated for all words in this sentence. If a word is repeated, the same neuron represents it. The efficiencies of synaptic connections following the stimulation of presynaptic associative neurons were computed using (3-7). The connection weights were computed according to (3-6). Because this is the first training sentence, and there are no repetitions of words, the activation of any associative neuron is sufficient to activate the postsynaptic associative neuron representing the next word in the sentence. Besides, associative neurons representing subsequent words in the sequence are also stimulated by the associative neurons representing previous words of the trained sequence, establishing the context of the following words. In Figure 3-3(a), this context represented by the additional connections (small circles with numbers) does not influence the result of stimulation significantly, but these connections play a substantial role when subsequent sentences partially composed from the same words will be represented by this structure (Figure 3-3(b-c)).

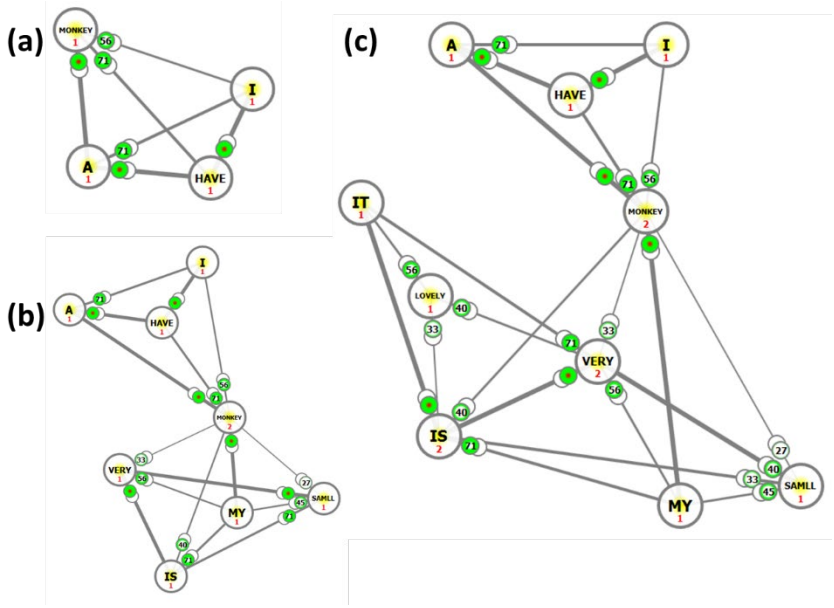

Figure 3-3. Steps of a sample ANAKG structure developed according to the associative process.

Figure 3-3(b) shows the ANAKG structure after the presentation of the second sentence: *my monkey is very small.* Four new as-neurons representing the new words {*my, is, very, small*} are created, and the synaptic connections to all their predecessors in the sentence are added. The synaptic efficiencies and connection weights are calculated according to (3-7) and (3-6) respectively. Note the aggregation of the representation of the word *monkey* occurring in both of the currently trained sentences. When a word is shared between a few training sentences, the neuron representing this word does not stimulate neurons representing subsequent words sufficiently to activate them. For instance, the word *monkey* cannot activate the neuron *is* from the second sentence by itself, but stimulation of the neuron representing even earlier word *my* (the context) is

required. Similarly, following the training with the third sentence: *it is very lovely*, the synaptic efficiency and connection weights between *very* and *small* change due to the occurrence of *lovely* following the word *very*. The activation of the pre-synaptic associative neuron *very* is not sufficient to activate the postsynaptic neuron *small* anymore (see Figure 3-3(c)). It is necessary to use the context of previously activated neurons {*my, monkey, is}* or {*it, is*} to adequately stimulate the neurons *small* and *lovely* to activate the right one according to its context.

3.3 Neural Mechanisms

3.3.1 Threshold Increase

The development, maturation, and growth of cerebral cortical interneurons were studied in [91]. The morphological study revealed that large interneurons had significantly more branching material in the postnatal brains than their prenatal neurons (Figure 3-4). These increases of dendritic span and branching provide larger receptive areas which may improve the development of connections in functional intracortical columns. Increase in the neuron size and its dendritic span corresponds to a larger number of ions that must be delivered to a neuron to activate it. In addition, a neuron that is more frequently activated grows, while the one that is not activated shrinks reducing a minimum number of ions required for its activation. Thus, we can reasonably assume that the size of the soma can grow as more connections are made to a neuron.

Thus we propose an associative neuron model which increases its activation threshold when the neuron is more frequently activated. This changes the sensitivity of neurons to input stimulations with larger charge needed to activate a neuron again.

Subsequently, only combinations of stronger or more frequent stimuli will activate a neuron. This leads to specialization of such neurons and limits input combinations that can activate them. Such a process is not destructive because specialization of neurons enables them to be more specific, and react more adequately to a situation. Rejected frequent combinations by already specialized neurons are represented by smaller neurons and thus automatically an input data space is represented more precisely by a larger number of neurons.

The existence of neurons with various sizes of soma and number of dendrites and axonal terminals is proven by evidence from the fields of neurobiology and neuroscience [87], [88]. But satisfactory functional explanation for these differences, including neurons sensitivity to combinations of input stimuli, ability to specialize, and creation of multiple connections between the same neurons to strengthen associations between them, is lacking [81], [92].

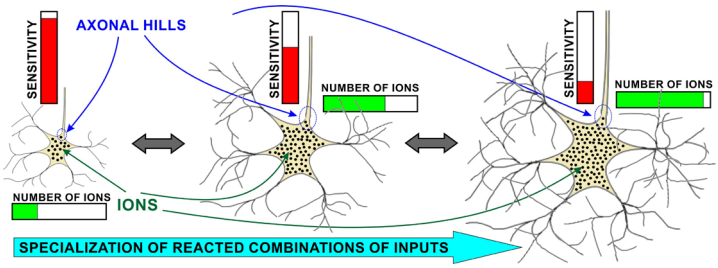

Figure 3-4. When a neuron grows its soma gets larger and requires more charges to be activated.

In associative neurons this translates to increase of the activation threshold. More frequently activated neurons have usually also more connections.

In this chapter we model some of these functional aspects of neurons and use them in later in machine learning algorithms to adapt active associative neural graphs. Biological neuron bodies have different shapes and sizes [88] so the bigger neuronal bodies need to be stronger or more frequently stimulated to achieve activation thresholds (Figure 3-4). Moreover, bigger neuron bodies have larger surfaces that can have more built in ion channels which can accelerate ion flux processes. Thus bigger neurons charge, relax and refract usually faster than smaller ones [88], [93].

Neurons should be activated in a proper sequence to the activations of other neurons to represent the subsequent elements of the trained sequence. Thus only a full previously activated context for each element should activate the neuron representing this element. Hence, we need to adapt the neuron sensitivity to make its activation possible only when stimulations from all presynaptic neurons representing this context come. Therefore,

$$X_{N_i}^{S_1+\cdots+S_{L-1}} \le \theta_{N_i}^t \le X_{N_i}^{S_1+\cdots+S_L} \tag{3-15}$$

where $X_{N_i}^{S_1+\cdots+S_L}$ represents the excitation achieved for the total previous context represented by stimulations $S_1, \ldots, S_L$ coming from neurons $N_{S_1}, \ldots, N_{S_L}$; $X_{N_i}^{S_1+\cdots+S_{L-1}}$ represents the excitation achieved for the total previous context without the last element of sequence that should charge the neuron above its threshold $\theta_{N_i}^t$. Thus, when postsynaptic neuron $N_{S_{L+1}}$ representing the S_{L+1} sequence element is activated too early its threshold should be increased to exceed its current excitation level:

$$\theta_{N_i}^t = X_{N_i}^{S_1+\cdots+S_K} + \varepsilon \tag{3-16}$$

where ε is a small number, *e.g.* $\varepsilon = \theta_{N_i}^t / K^2$.

3.3.2 Axon Growth

A growing axon connects to an increasing number of postsynaptic neurons that it needs to activate. Hence, a growing axon requires more charges. From an electrical signal point of view this can be compared to the fan-out issue in logic gates, larger the fan-out the longer it takes for a gate to charge its output.

While normalization of synaptic permeabilities is a simple means of introducing the fan-out effect, it has a drawback. Normalization of synaptic permeabilities would increase the importance of weak synapses. To avoid this, the synaptic strength is scaled by the norm of all synaptic permeabilities of the presynaptic neuron. Thus, the associative neuron excitation level during charging is evaluated using activities of all presynaptic neurons as follows:

$$X_{N_i}^{t+\Delta t} = X_{N_i}^{t} + \left[\sum_{N_m \to N_i}\left(X_{N_m}^{t} \cdot \frac{\left(w_{N_m,N_i}\right)^2}{\|w_{N_m}\|}\right)\right] \cdot sin\left(\frac{\pi \cdot \Delta t}{2 \cdot \Delta t^C}\right) \quad (3\text{-}17)$$

Modification of the associative neuron excitation level to (3-17) is needed in episodic memory to distinguish sequences that begin with the same subsequence and is discussed in Chapter 4.

3.3.3 Synaptic Fatigue

Synaptic fatigue [91], [94] is a form of short change in the synaptic plasticity that lowers the firing activities of a postsynaptic neuron. Synaptic fatigue is a result of frequent stimulation of the same sensory neuron, resulting in habituation and lowering of the neuron's response. Acting as a negative feedback, synaptic fatigue, can physiologically control neuron's activity. A presynaptic neuron sends an activation signal to a postsynaptic neuron through neurotransmitters stored in synaptic vesicles. These

neurotransmitters propagate the signal to the postsynaptic neuron before, eventually returning back to the presynaptic neuron for reuse. The neurotransmitters take between 1-40 seconds to complete their journey to the postsynaptic neuron and return. But, if the neurotransmitters are released faster than they are replenished, it leads to a depletion of the synaptic vesicles. This temporary depletion causes synaptic fatigue Figure 3-5 shows a typical central nervous system synapse.

Figure 3-5. Synaptic vesicles represented by small circles on the top and postsynaptic receptors shown in postsynaptic neuron at the bottom.

Source https://en.wikipedia.org/wiki/Synaptic_fatigue

Although the existence of synaptic fatigue is widely accepted, and it can affect synapses of many different types of neurons [95], the exact mechanisms underlying this phenomenon are not well understood. Introducing a similar mechanism in our associative neuron model provides us with a better tool to model operation of declarative memory. In this model synaptic fatigue is simulated by modifying the resistance of associative neurons to activation during charging as follows:

$$X_{N_i}^{t+\Delta t} = X_{N_i}^{t} + \left[\sum_{N_m \to N_i}\left(X_{N_m}^{t} \cdot \frac{\left(w_{N_m,N_i}\right)^2}{\|w_{N_m}\|}\right)\right] \cdot S_{N_i}^{t+\Delta t} \cdot sin\left(\frac{\pi \cdot \Delta t}{2 \cdot \Delta t^C}\right) \quad (3\text{-}18)$$

where the neuron sensitivity factor $S_{N_i}^{t+\Delta t}$ is updated as:

$$S_{N_i}^{t+\Delta t} = 1 - \left(1 - S_{N_i}^{t}\right) \cdot e^{\frac{-\Delta t}{\Delta t^F}} \quad (3\text{-}19)$$

and Δt^F is the fatigue relaxing time constant during which postsynaptic neuron N_i recovers from the fatigue and approaches its full sensitivity to stimuli (here $\Delta t^F = 20\ s$).

Equation (3-18) describes an automatic process of gradual recovery from the fatigue, however each time a neuron is activated its activation sensitivity $S_{N_i}^{t+}$ is lowered using

$$S_{N_i}^{t+} = \frac{1}{\frac{1}{S_{N_i}^{t-}} + \Delta F} \quad (3\text{-}20)$$

where $S_{N_i}^{t-}$ is neuron's N_i sensitivity before activation, $S_{N_i}^{t+}$ is neuron's N_i sensitivity after firing, ΔF is a fatigue factor (here we use $\Delta F = 0.03$).

Figure 3-6 shows changes in neuron sensitivity due to fatigue after it was frequently activated.

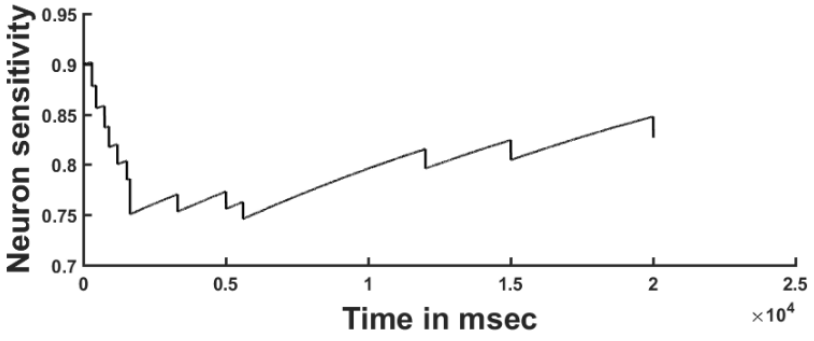

Figure 3-6. Changes in neuron sensitivity due to a fatigue factor.

Here the neuron's initial sensitivity was set to 0.9 and activation was in discrete time moments equal to [305, 450, 750, 900, 1200, 1530, 1640, 1630, 3300, 5000, 5600, 12000, 15000, 20000] ms. Frequent activation of the neuron during the first 1630 ms results in decreasing sensitivity, with a gradual recovery eventually.

3.4 Conclusion

Associative neurons originally presented in [81], and shown to be capable of associating information into knowledge in a means that is conducive to contextual recalling [81] were modified by us to accommodate growth and sensitivity of neurons and their strength of connections [73]. This modification resulted in automatic threshold changes and specialization of neurons. In the presented model the same neurons can be activated by various combinations of input stimuli that represent various contexts of the elements of training sequences. In addition, we introduced neuron's synaptic fatigue.

The associative neuron model described here is biologically inspired, computationally feasible, and address the limitations of other associative neuron models. The model accounts for the timing behavior of biological neurons and plastic processes of biological neurons, *i.e.* automatically adapt and connect contextually to other neurons.

4 DECLARATIVE MEMORIES

In the previous chapter a biologically plausible associative pulsing neuron model [73]that is used in this work to demonstrate the five memory capabilities from *Section 1.3* was described. In this chapter the model will be used to implement memory structures capable of a) creating semantic relationships, b) recognizing sequences, and c) demonstrating creativity. Following a brief introduction to the various types of memory in *Section 4.1*, details of the implementation and testing of the semantic and episodic memories is provided in *Sections 4.2* and *4.3* respectively. *Section 4.4* will demonstrates emergent creativity of declarative memories. Finally in *Section 4.5* conclusions are discussed. The work described in this chapter has been published by us in [73].

4.1 Introduction

Declarative memory, a type of explicit memory, is a long-term memory of past experiences and acquired knowledge and its presence in humans can be cognitively explained [96]. Declarative memories play an important role in all aspects of learning such as recognition, understanding, planning, and motor action, and are hence a necessary system for any cognitive agent. Declarative memories can be classified into two complementary memory systems: episodic and semantic memories. While semantic memory is a structured record of facts, concepts, and knowledge about the world acquired over the lifetime, episodic memory is a representation of personal experiences and specific events (time, place, emotions and other contextual knowledge) that can be explicitly stated. Episodic memory supports learning in the semantic memory by providing a recollection of past events. While the organization and internal dynamics are

different, both semantic and episodic memory require the storage of sequential information and, in this work, are based on principles of self-organization and use the associative neuron model described in Chapter 3. It is believed that in humans the neocortex and hippocampus, parts of the brain with structurally different organization of neurons, are responsible for semantic and episodic memories respectively [97].

Declarative memories are built upon personal experiences, that is they are built upon episodic memories. While humans can create and retain memories of many personal experiences, most episodic memories will ultimately contribute to development and modification of general knowledge about the world and be “lost”. That is, overtime episodic memories sharing some elements are linked together, lose their unique information, and form semantic memory. In return, the semantic memory provides a wider context to the observed scenes through triggering of learned associations potentially helping suggest alternative solutions to a problem. Thus, semantic and episodic memories complement each other in both their operations and concept formation. Work presented in this chapter assumes that neurons represent symbolic concepts, *i.e.* words, that are stored and associated with each other based on observations.

4.2 Semantic Memory

The semantic memory in a cognitive system is a store for knowledge about the environment that an agent has acquired and can provide solutions to novel situations, *i.e.* exhibit emergent creativity. The semantic memory investigated in this chapter uses ANAKG, described earlier in *Section 3.2*, and can represent and associate training sequences of objects or classes of objects [81]. The ANAKG memory adaptation process

binds objects that appear in close proximity in the input sequences, providing time domain or spatial associations. The created synaptic connections are weighted, shorter the distance stronger the association, so each association has its own importance. The model demonstrates that memories can be changed if new data is processed or old data is repeated.

4.2.1 Structural Organization of Semantic Memory

Humans learn to perceive the relationship between perception and action via complex sensorimotor functions, providing them with knowledge about objects in the environment that have utility for them [98]. Using similar principles, focusing on features of the observed episode can improve the predictive power of the episodic memory. The semantic memory plays an important role in this, it provides a mechanism to focus on the characteristic features of the observed episode.

The ANAKG algorithm is applied to the following set of training sequences: "*I have a monkey. My monkey is very small. It is very lovely. It likes to sit on my head. It can jump very quickly. It is also very clever. It learns quickly. My monkey is lovely. My son has a small dog. His dog is white and sweet. My daughter has a black cat. Her cat is small and clever.*" The resulting semantic memory from the process of adding neurons and connections according to the neuron model described in Chapter 3 is illustrated in Figure 4-1. Nodes in Figure 4-1 represent words (concepts) while edges represent spatio-temporal associations. As described earlier, each associative neuron represents a single word, and the numbers under their names represent their number of activations during the training phase. The small circles represent the postsynaptic elements of the synapses, and

a red dot inside them means that the value of the synaptic weight is equal to the activation threshold of the postsynaptic neuron. Numbers in the postsynaptic elements (small circles) represent synaptic permeability values as a percentage of the threshold value. Similarly, the crescent shape denotes the presynaptic elements and shows the direction from which the stimuli come.

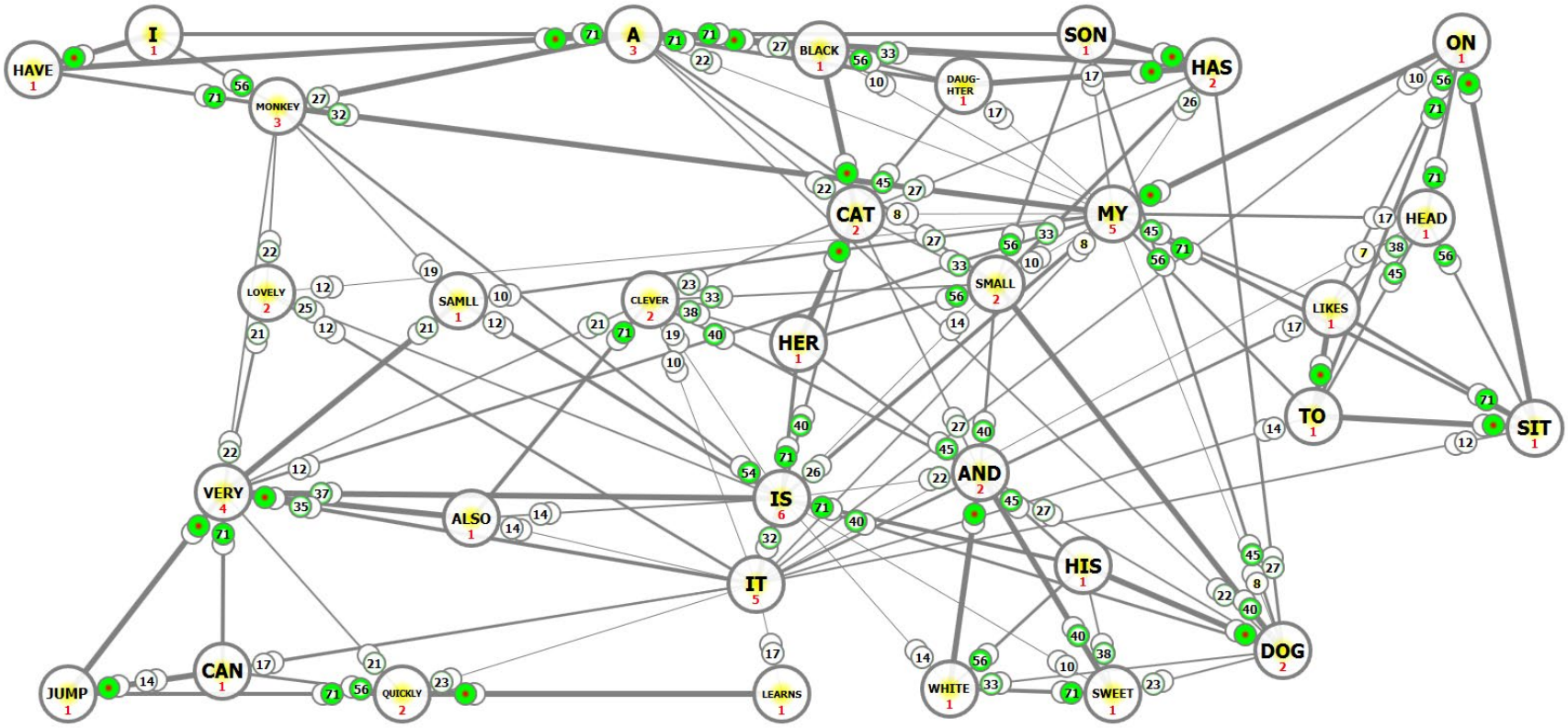

Figure 4-1. A sample neuronal structure formed during associative processes for training sequences.

"*I have a monkey. My monkey is very small. It is very lovely. It likes to sit on my head. It can jump very quickly. It is also very clever. It learns quickly. My monkey is lovely. My son has a small dog. His dog is white and sweet. My daughter has a black cat. Her cat is small and clever.*"

4.2.2 Testing Semantic Memory

Semantic memories are stores of facts, concepts, and knowledge about the environment acquired over the lifetime. Because they store associations about various objects, dependencies between them, and actions performed on them, semantic memory enables retrieval of contextual information about observed objects making generalization and creativity possible.

The semantic memory model implemented here can learn the training sequences, the strength of the connections between elements dependent on the spatio-temporal distance between them and the frequency of their activation. The semantic memory model can also both recall generalized sequences and create new ones. That is, the semantic memory displays generalization and creativity. Table 4-1 shows the results of simulating the network from Figure 4-1 with various initial contexts. The results show that when the initial context is unique, that is occurs only once in the training sequences, the memory can recall the training sequence even with a single presentation. The memory provides new or generalized answers when the initial context is new or not unique the memory provides new or generalized answers. With non-unique contexts, repetition forces the memory to recall the most frequent subsequences matching the context from the training data. Note, the generalized and new answers provided by the network are bolded to highlight them. For example, when the initial context is not unique (*my monkey* and *monkey is*), repeating the context results in the network providing generalized answers (*my monkey is small very lovely* and *monkey is very small lovely* respectively).

Table 4-1. Semantic memory answers to various initial contexts.

Initial context / Question	Network Answer
I have (a unique context)	I have a monkey
Her (a unique context)	Her cat is small and clever
His (a unique context)	His dog is white and sweet
My (no unique context)	My
My monkey (repeated 5 times)	My monkey is **small very lovely**
Monkey is (repeated 3 times)	Monkey is very small **lovely**
Cat (repeated 5 times)	Cat is small
Dog (repeated 3 times)	Dog is white
It (repeated 6 times)	It is very lovely
My son and his dog	My son has a small dog and his is white sweet dog
Can I	Can I **have a monkey**

4.3 Episodic Memory

Episodic memory is a store of specific personal experiences and events, with events stored as sequences of spatio-temporal elements that can be used for recollection of observed events. The strength and durability of the memory of events are strongly influenced by their significance to the cognitive agent, it is easier to recall significant events. Episodic memory gives us time perspective and provides continuity in everyday activities. Note that due to the complementary nature of semantic and episodic memories, an event observed by different individuals can be remembered differently by them.

4.3.1 Structural Organization of Episodic Memory

Episodic memory plays a critical role in cognitive agents. Consequently, in recent years several structural models have been proposed [99]–[102]. The various models differ in their structural organization, storage and retrieval mechanism and properties such as forgetting, anticipation, and novelty detection.

The model presented in [92] used a flexible matching mechanism that measured the similarity between the learned and tested sequences. The model is robust and could tolerate errors, distortions, and varying time delay. In this work a simplified version of this memory structure, using associative neurons described in Chapter 3, has been used. The ability of these associative neurons and their semantic connections, *i.e.* learning mechanism, to formulate episodic memories is shown.

The structure of the episodic memory, shown in Figure 4-2, is based on a self-organizing structure of long-term memory (LTM) cells. It is built upon the concepts stored in the semantic memory. During training, the primary neurons P_i that reside in the semantic memory are first activated. This sequence of semantic memory neuron activations $\{P_1, P_2, P_3, \dots, P_{n-1}, P_n,\}$ stimulate the corresponding secondary neurons $\{S_1, S_2, S_3, \dots, S_{n-1}, S_n,\}$ in the episodic memory. The primary neurons represent concepts and the associations between them, that is, they represent objects, activities, motivations, or goals and semantic associations between them.

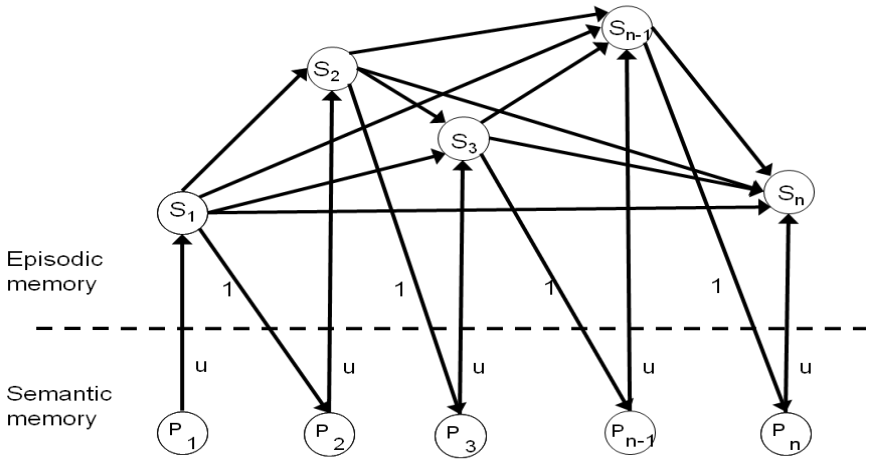

Figure 4-2. A model of LTM cell based on associative neurons.

The primary neurons implement symbol grounding [36] and can thus provide understanding of the observed scene. While each LTM cell stores a different event, their structure and strength of synaptic connections is fixed. Consequently no learning of synaptic strengths inside the LTM is needed. Individual LTM cells differ in their connections to primary neurons, and these connections need to be learned. While the activation of primary neurons activate secondary neurons in the LTM cells, these activations of secondary neurons can be used to predict the next element of the input sequence.

While the primary neurons are linked to the corresponding secondary neurons with weights equal to u, set to 1 in this work but could be normalized to limit the maximum activation of the secondary neurons, the secondary neurons are connected to the primary neurons representing the next element of the episode/sequence using prediction links with weights equal to 1. The secondary neurons are also connected to all the subsequent elements of the episode sequence using weights obtained from any one of (3-8) - (3-12). The activations of the secondary neurons are computed using (3-18) – (3-20).

4.3.2 Algorithm for Episodic Memory Retrieval

Episodic memory, once trained, can be activated and retrieved during any sequence recognition process. The retrieval process is simple and occurs in three stages: event detection, episode recognition, and episode recall. Activation of any primary neuron in the semantic memory, directly or through associations with activated neurons, that is associated with an LTM cell triggers event detection. Activation of primary

neurons leads to activation of secondary neurons in the LTM cells, this activation of LTM cell neurons triggers episode recognition. If secondary neurons in multiple LTM cells were activated, the first LTM cell to activate is considered to be the most likely representative of the observed episode and is recalled. Episode recall is a means of predicting or anticipating the expected elements of the sequence, and plays an important role in learning. No learning takes place if the prediction was correct.

The memory retrieval algorithm works as follows:

- Activate primary neurons P_j^t and the corresponding secondary neurons S_j^t in all LTM cells.
- Compute the activation level of the secondary neurons S_j^{t+i} using (3-18) - (3-20).
- Use the secondary neuron S_j^{t+i} with the strongest activation to find the winning LTM.
- Secondary neuron S_j^{t+i} of the winning LTM cell predicts the next episode.

4.3.3 Testing Episodic Memory

The ability to recall past events based on present context is a very useful feature in memory. As the episodic memory is built upon the semantic memory it uses the contextual information provided by the semantic memory to help in recalling previous episodes. In this simulation, the episodic memory consists of LTM cells, with each cell representing an episode and the individual secondary neurons S_j, representing elements of it. These secondary neurons are triggered by primary neurons P_j in the semantic memory. The primary neurons in the semantic memory are triggered by external inputs. The

semantic and episodic memories were first created from training sequences using associative neurons described in Chapter 3. Subsequently the ability of the episodic memory to recall the correct sequence and differentiate between similar sequences has been tested.

4.3.3.1 Example 1

For simplicity consider a training file consisting of the following sequences:

1. A B C D; 2. A B C; 3. B A C E; 4. P Y E J.

First, the semantic memory is created through consolidation of the training sequences using the ANAKG approach described earlier. Each neuron in the resulting semantic memory structure represents one element of the training sequence. As the semantic memory structure is context dependent, each element of a sequence simulates its successors, *i.e.* subsequently learned elements of the sequence, through links with weights that reflect their occurrence, frequency and distance (number of elements separating them), in the training sequences. A similar process was used to create the LTM cells in the episodic memory, but with one difference: each LTM cell was created using only one sequence. Thus, in this example the episodic memory consisted of four LTM cells.

The memories once created were ready for testing. Here the results of testing with one sequence, "A B C D", are described. Following the testing protocol discussed earlier this test sequence was used as an external stimulation to semantic memory, and consequently activation of its neurons (primary neurons). The strength of the resulting activations of these primary neurons was treated as the output from the semantic memory

and was provided as an input to the LTM cells in the episodic memory. That is, test sequence activated the primary neurons {A, B, C, D} in the semantic memory that in turn activated their corresponding secondary neurons in the LTM cells.

In this example only three of the four LTM cells (“A B C D”, “A B C”, and “B A C E”) contain elements of the test sequence (“A B C D”). Among these three, the first two (“A B C D” and “A B C”) are of particular interest because not only are they very similar to each other, but one of them is the same as the testing sequence and the other is its subsequence.

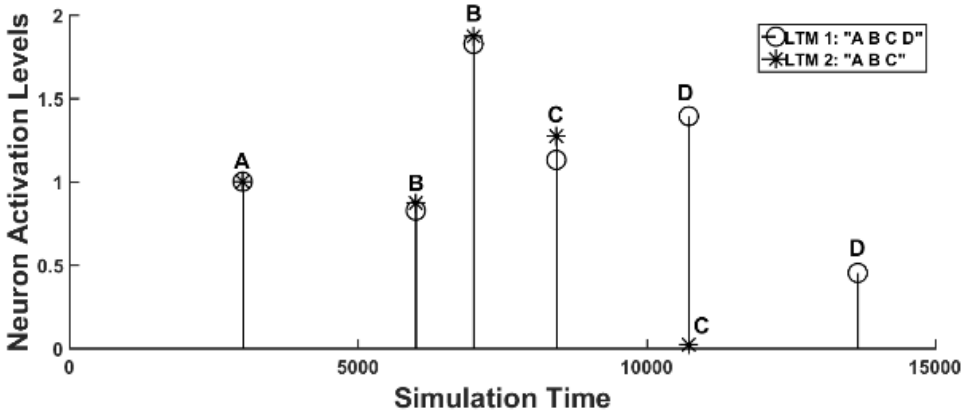

Figure 4-3. Activation levels of “winning neuron” in LTM cells 1 and 2.

Figure 4-3 shows a comparison of the activation levels of the “winning neuron” in these two LTM cells with symbol o marking LTM1 and * marking LTM2. As a neuron goes through various states: resting, charging, relaxing, activation, absolute refracting, or relative refracting, its activation level changes. In addition, while the time periods for the six states are the same for all neurons, the activation levels of individual neurons differ

because the presynaptic neurons can have different levels of activations, can be connected to different number of post-synaptic neurons, or can be in different states.

When two LTM sequences having the same initial elements are stimulated with the shared elements we expect that the shorter LTM sequence will have higher activation initially, as expected from (3-17). This was observed in the simulation and is very clearly visible around the 8000 ms mark. Figure 4-3 shows that at this point in time the activation levels of the winning neuron in LTM 2 is higher than the activation level of the winning neuron in LTM 1. Only towards the end, when neuron 'D' is activated does the winning neuron in LTM 1 have a higher activation level than that from LTM 2.

4.4 Emergent Creativity of Declarative Memories

In humans declarative memories, also known as explicit memory, are a type of long-term memory that can be explicitly (or consciously) be recalled and can be subdivided into episodic and semantic memories. Due to their importance in cognition episodic and semantic memories play an important role in various cognitive architectures. For, example declarative memories, obtained through integration of episodic and semantic memories, are an essential functional component of the motivated learning cognitive architecture (MLECOG) [4], an early version of which was used as the basis for the cognitive agent discussed in Chapter 2. This section illustrates the ability of declarative memories based on associative neuron model to create new categories, generalize, and answer new questions.

Table 4-1 demonstrates some ability of the declarative memory to answer questions with initial contexts not used in training sequences. The answers obtained while

different from the training sequence still reflected the knowledge obtained from the training sequences. While repetition of the "question" was needed to obtain an answer if multiple options existed in the training sequence, this is normal when people are trying to answer difficult questions. Hence, this behavior of the memory can be considered similar.

The results show also that change in context (Table 4-1) may, based on the training set, result in a change in the answer. This is a reflection of the generalization and creativity of the network that can be obtained from such declarative memories and is determined by the dynamic modification of knowledge gained during. As a network develops it changes the way it processes the input data, reflecting its stage of development and its ability to generate context based answers.

A declarative memory, obtained through integration of semantic and episodic memories, was created and tested for this work. This declarative memory had the semantic memory system receiving the external stimulations/inputs and the output of the semantic memory was the input to the episodic memory. The output of the episodic memory, the winning LTM neuron, is considered the output of the declarative memory. Both the semantic and episodic memories were trained with the sample data from Figure 4-1. Table 4-2 shows the answers provided by this declarative memory to the same initial context as in Table 4-1. Note that as the individual neurons start their relaxing or refraction phase the winning LTM cells can change. In Table 4-2 only the first winning LTM cell is specified. The major advantage of the declarative memory is observed in the response to the last five initial context/questions shown in Table 4-2. The declarative memory was able to generate meaningful responses without necessitating repetitions of

inputs, this is because any activation in the semantic memory will stimulate the LTM cells of the episodic memory thus potentially enabling the declarative memory to generate a response.

Table 4-2. Declarative memory answers to various initial contexts.

Initial context / Question	Declarative Memory Answers
I have	I have a monkey
Her	Her cat is small and clever
His	His dog is white and sweet
My	My monkey is lovely.
My monkey[1]	My monkey is lovely.
Monkey is[1]	My monkey is lovely.
Cat[1]	Her cat is small and clever.
Dog[1]	His dog is white and sweet.
It[1]	It learns quickly.

[1] Note, unlike in Table I, no repetitions are required or used here

4.5 Conclusions

A structural organization of declarative memories using the model of associative neuron from Chapter 3 was developed and tested. In the presented memory structure the same neurons can be activated by various combinations of input stimuli representing different contexts of the elements of the training sequences.

We tested sequence recognition and associative properties of the episodic and semantic memories obtained following a self-organizing process were tested. Symbolic approach, with each neuron representing an object, action, or idea, was used.

Emergent creativity in the developed memory structures was demonstrated. To test the memories we first organized them based on a set of training sequences, and

subsequently submitted a various questions. The questions included those with unique context in the training sequences, and also those with new or non-unique context. We demonstrated that the memory could, using the stored knowledge, respond within the associative context of the question asked. Thanks to the strengthened associations between neurons, which reflect the frequency of training subsequences, we achieve the ability of the network to generalize.

Future work includes further studies on neuron models developing their ability to adapt to training subsequences that include other sequences. Networks could be enriched with autonomous motivational signals and mechanisms of their automatic associations with symbols and actions represented in declarative memories. Declarative memory structure could be enhanced by adding a feedback from the LTM cells in the episodic memory to the semantic memory, this can generate new or generalized relationships.

5 LUMPED MINICOLUMN ASSOCIATIVE KNOWLEDGE GRAPH

In the previous chapter the associative pulsing neuron model [73] was used to implement declarative memories and the results demonstrated the memory structures to be capable of capable of a) creating semantic relationships, b) recognizing sequences, and c) demonstrating creativity. This chapter will demonstrate memory structures that are capable of a) creating semantic relationships, b) recognizing sequences, and c) prediction. The memory structure implemented in this chapter, lumped minicolumn associative knowledge graph (LUMAKG), is a generalization of the active neural associative knowledge graph (ANAKG) used in Chapter 4 to implement the semantic memory to its minicolumn form. Following a brief introduction in *Section 5.1*, details of the LUMAKG organization are presented in *Section 5.2*. In *Section 5.3* the LUMAKG organization process is explained using an example. Test results are discussed in *Section 5.4* and finally conclusions are discussed in *Section 5.5*. The LUMAKG algorithm described in this chapter was first introduced by us in [103]. A later work incorporating the results presented here are undergoing peer review.

5.1 Introduction

Memory plays an important role in cognitive systems, providing it with the knowledge about its environment and how to deal with it. Its structure self-organizes as a result of the past observations, actions and their consequences [104]. The learning process includes changes in the long-term memory cells and the synaptic connections between neurons. Associations between neurons reflect contexts for the learning and representation building process [105].

There are several artificial neural network models used to simulate semantic and episodic memories. Associative networks are content addressable and are able to retrieve stored data based on only a part of what was stored [57]. They are resistant to noise and can detect missing data and sensory failures [106]. Models of associative networks with feedback loops, called recurrent neural networks (RNN), can be trained to predict the next output symbol after reading a stream of input symbols. In [65] gradient-based RNNs were used to retrieve the memories of the stored input sequences. In [107] an unsupervised algorithm that, using RNNs, learns fixed-length feature representations of sentences, paragraphs, and documents was proposed. The Penn Corpus and Switchboard, with about 1 and 4 million words respectively, were used in [107]. Similar to RNNs, this algorithm is trained to predict words in a document given an input context. Memory networks [108] are a new class of learning models that combine the input content with the dynamic knowledge base stored in the long-term memory to predict the output. Memory networks represent the input information in the form of features and are capable of generalization to produce the desired response.

In response to demand for services based on speech recognition and large knowledge bases like Wikipedia, researchers in recent years have focused on contextual question answering (QA). Direct approaches to QA like string matching are ineffective [109], and solutions that include recursive neural networks like QANTA [109] are becoming popular. Neural Turing machines (NTM) combine the concept of neural network learning and classical Turing machines to retrieve context-based input information [110]. NTMs can learn simple algorithms from input and output examples

and use them to generalize. Compared to RNN long short-term memory[65], NTMs show better accuracy over longer sequences in recall and copy tasks [110].

Semantic knowledge and short-term memory must cooperate to provide a context-based scene understanding and recall of the useful operations that the system performed in an open environment. Semantic memory aggregates representation of the training data and forms a context searchable knowledge base. This memory is obtained by binding the semantic contexts for all trained objects and linking their neuronal representations together. Semantic memory can be built using an active neural associative knowledge graph (ANAKG) that uses the associative spiking neuron model presented in [81].

Recurrent neural networks that use artificial minicolumns have much larger storage capacity than ordinary networks in which each neuron represent a single concept as discussed in [111]. Minicolumn refers to a group of one or more neurons that function as a unit. This property of artificial minicolumns is used to increase the memory capacity and improve the quality of knowledge representation in ANAKG memories.

ANAKG networks are robust to distortions in the input signal, and to some degree resemble the effectiveness of the long short-term memories (LSTM) [65] that can store short-term sequential information over longer periods of time through its gating system. LSTMs are a kind of RNNs that are capable of learning long-term dependencies. While LSTMs and RNNs have a similar form, a recurrent hidden layer consisting of a chain of recurrently connected neural network modules, the neural network modules themselves are different. The neural network modules in RNNs have a very simple structure, for example, a single memory cell, whereas the neural network modules in LSTM have

multiple neural network layers, *e.g.* one or more memory cells and gates to control the flow of information. In contrast to LSTM, ANAKG networks are easy to train and do not require supervised learning, which makes them a better choice for natural learning in an open environment.

In this chapter, a generalization of ANAKGs to their minicolumn form known as lumped minicolumn associated knowledge graph (LUMAKG) memory is presented. LUMAKG organization, and a learning process to establish spatiotemporal associative connections between neurons are described. In LUMAKG, each symbol is represented several times following the idea of minicolumn organization presented in [111]. LUMAKG uses the same pulsing neuron model as ANAKG and similar self-organization principles. The most significant difference between the two memory structures is that LUMAKG uses columnar organization and uses a new mechanism for selection of synaptic connections between neurons. While the columnar organization increases the memory capacity, the new mechanism for synaptic connections improves the resolution of context-based sequence recognition.

5.2 Organization of LUMAKG

5.2.1 Minicolumn Organization of the Associative Memory

A columnar organization of the associative memory was proposed by J. Hawkins *et al*. [111] where the authors introduced cortical learning algorithms in which minicolumns were used to store sequential information in structures known as hierarchical temporal memory (HTM). Since then HTMs were further developed, and their properties were analyzed and tested. In [112], the authors show that HTM is able to

continuously learn a large number of temporal sequences using an unsupervised learning neural network model. HTM was shown to have similar accuracy as another state of the art sequence learning algorithms like echo state networks [113] or long short-term memory [112]. However, they also show some drawbacks like larger sensitivity to temporal noise than long short-term memory [112]. ANAKG memories do not have this drawback of HTM networks because as-neurons use the time delay of the input signal to make associations, and ANAKG associates not only the individual inputs but also their sequences spread over time, greatly minimizing the effect of temporal noise (or any single event). The gradual change in sensitivity, activation threshold, synaptic weights, and connections to other neurons and sensors that results from the model parameters also helps to minimize the effect of temporal noise. Thus, improving ANAKG by introducing a minicolumn structure to its architecture provides a better associative memory capable of storing spatiotemporal relations between data. Continuous learning from input data and rapid adaptation to changing environmental conditions are desired properties of machine learning [112], so it is important that the algorithm can recognize and learn new patterns quickly. ANAKG memories have this property.

Both in HTM and ANAKG, neurons do not perform a simple weighted sum of their inputs as in most neural network models ([58], [114], [115]), but integrate them over time. This is similar to spiking neuron networks [82]. Spiking neurons are biologically motivated and produce patterns similar to biological neurons. Several computationally efficient models of spiking neurons have been developed [116]. Networks of spiking neurons spontaneously self-organize into groups and generate polychronic patterns of

activity, and this property is believed to be necessary for cognitive neural computations, symbol grounding, attention, and consciousness [117]. ANAKG achieves similar properties to spiking neurons by using a much simpler spiking neuron model and self-organization principles to capture the spatiotemporal relationship between data [90]. LUMAKG maintains these properties of ANAKG while increasing its recall quality, memory capacity, and resolution.

Following HTM organization, each neuron in ANAKG is replaced with a minicolumn of several neurons, where all the minicolumn neurons represent one unique symbol (*e.g.* a single word). Individual neurons in the active columns represent information regarding the learned temporal context, and they may be activated by different learned temporal contexts. Like in HTM the neurons in LUMAKG receive three types of inputs. The input from the lower layer network carries the sensory information and is used to recognize the learned sequences, the input from the higher layer represents feedback prediction, and the inputs from the same layer represent context-based prediction and lateral inhibition used to create self-organizing maps.

While the neurons in a minicolumn are duplicates of each other, their inputs, outputs, and synaptic connections (weights) are not. Using the design principles from HTMs [111], the inputs and outputs are distributed across all the minicolumn neurons so that multiple sequences can be represented using the same set of minicolumns. Individual neurons in each minicolumn use the ANAKG algorithm to establish associative connections and their synaptic weights. Like ANAKG neurons, LUMAKG neurons modify their thresholds to stimulate learning by various minicolumns and their neurons.

Like in HTM, LUMAKG minicolumns have three output states, active from feed-forward input (can be input from the sensor), active from lateral input (representing a prediction), and inactive. Thus, LUMAKG neurons can fire even without sensory input stimulation. In the predictive mode, neuron's activation from the lateral input is used to complete the sequence. During learning of new sequences, prediction and input activation should match for the learning (changing the synaptic weights) to take place.

5.2.2 Organizing Principles of LUMAKG

The design of LUMAKG memory is illustrated by using sequences of words as its input. Each minicolumn in the developed structure represents a different word. Although a minicolumn based memory is capable of handling raw sensory data to obtain its symbolic representations [118], this simplified approach where the input signals are the sequences of symbols used. This is done in order to have a simple interpretation of the neurons' activities and to use simple measures to compare test results with ANAKG or other neural networks that use symbolic inputs. A distributed version of the minicolumn associative knowledge graph (DIMAKG) is currently under development.

The LUMAKG graph structure is obtained dynamically. New minicolumns and synaptic connections are added each time a new input sequence is provided to the network. Specifically, if a new symbol is observed, a new minicolumn is added, and at least one of its neurons is linked to other minicolumns establishing new synaptic connections. The organizing principles of LUMAKG are as follows:

A. Duplicate each symbol m times to form an individual symbol minicolumn.

B. If a neuron in a minicolumn is activated above its threshold from the associative connections, it is called to be in a **predictive mode**.

C. A sensory input activates either all the neurons in a given minicolumn that are in the predictive mode or the whole minicolumn if no neuron is in a predictive mode.

D. Activated neurons that were in a predictive mode are in **predicted activation** (PA).

E. An activated minicolumn without any neuron in a predictive mode has all the neurons in **unpredicted activation** (UA).

F. Synaptic weights of connections between activated neurons in the predecessor and the successor minicolumns are changed according to (5-2).

The number of neurons in each minicolumn *m* is set arbitrarily. In the NuPIC software that implements HTM memory, *m* is set to 32 [118], [119]. In the human cortex, the number of neurons in each minicolumn is between 80 and 120 (with twice this number in the visual cortex area) [120]. In the approach used here it is demonstrated that the memory and its resolution depend on this number, m is an important network design parameter. It is hypothesized that larger memory networks need a larger number of neurons in their minicolumns for their optimum performance.

5.2.3 The LUMAKG Algorithm

According to the described organizing principles, the LUMAKG algorithm can be organized as follows. The individual steps of this algorithm are explained and illustrated using a design example.

The LUMAKG algorithm:

I. Read the consecutive elements of the input sequence to activate the corresponding minicolumns.
   1. Check if the symbol from the input sequence is represented by a minicolumn.
   2. If it is not, add a new minicolumn.
   3. Put all neurons of this new minicolumn in the state of unpredicted activation.

II. Establish the predecessor-successor neurons in all the minicolumns activated by the input sequence.
   4. Find the non-overlapping sequences of the previously stored episodes.
   5. Establish a sequence of linked PA neurons in all the activated minicolumns.

III. Update the synaptic weights in the synaptic connections between all predecessor-successor neurons.

Update all the synaptic weights between all the PA neurons in the predecessor and successor minicolumns according to the rules developed for ANAKG [81].

5.3 LUMAKG Design Example

Since the LUMAKG algorithm has a convoluted process for modification of synaptic connections, an example is used to illustrate how the algorithm works. First, we will illustrate how to find non-overlapping sequences of the previously stored episodes in all the minicolumns activated by the input sequence (point II.4 of the LUMAKG

algorithm). In this example, we assume for simplicity of the graphical illustration that the number of neurons in each minicolumn is equal to 5. This, however, should be optimized depending on the desired memory size.

For each consecutive activated minicolumn, activate all the neurons in the minicolumn that correspond to the input symbol according to point C of the organizing principles. Typically, the first activated minicolumn has no PA neurons, unless it is considered in the broader context of associative learning and was a part of the previously stored episode. Thus, typically all neurons in the first activated minicolumn are in the state of an unpredicted activation.

5.3.1 Finding Non-overlapping Sequences

To better explain point II.4 of the LUMAKG algorithm, let us illustrate it with an example of a sequence of activated minicolumns. Let us assume that the sequence "A, B, C, D, E, F, G, H, I, J" was inputted to the LUMAKG memory and activated the corresponding minicolumns as shown in Figure 5-1. This sequence could represent a number of sentences with all different words like the following sentence: *I didn't really know how to cook these green plantains*.

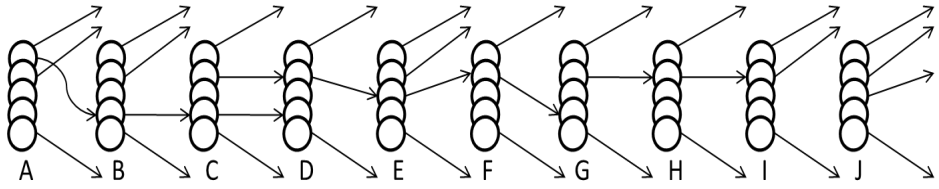

Figure 5-1. Activated minicolumns with the existing synaptic connections.

If the previously obtained inputs contained sequences that used some of these words, then there will be synaptic connections between possibly different neurons in the corresponding minicolumns. For instance, if the previous inputs to LUMAKG memory contained the following sentences:

I didn't really know this. Nuns really know how to cook oysters. Don't cook these green mushrooms.

Then the corresponding synaptic connections could start at various locations in their minicolumns – as can be observed in Figure 5-1.

In order to modify the existing synaptic connections or to introduce new connections while preserving the episodic storage, we need to find a sequence of activated neurons in these minicolumns that preserves the most significant episodes. This is accomplished by following the existing associative links to specific locations within each minicolumn.

We first identify which subsequences of the newly activated minicolumns were parts of the stored episodic memories. In Figure 5-2, we show these neurons in the newly activated minicolumns that already have synaptic connections to other consecutive activated minicolumns and were parts of previously learned sequences. If activated, they will predict activations of the corresponding postsynaptic neurons in the previously learned sequences. We call these neurons **linked episodic neurons** (LEN). In Figure 5-2, we mark these LEN neurons using a darker shade.

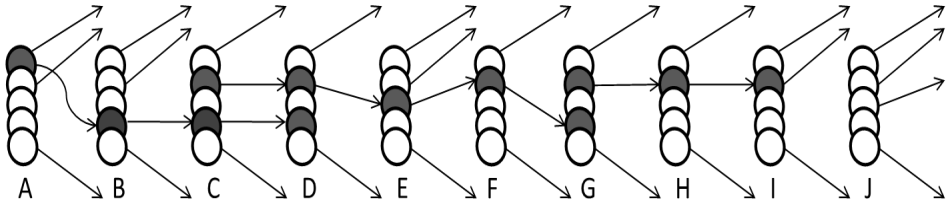

Figure 5-2. Linked episodic neurons of the previously learned sequences of symbols.

Following the directed links from each LEN neuron, we can find related **predictive graphs of minicolumns** (PGM). For instance, in Figure 5-2, we can observe three PGMs: A, B, C, D and C, D, E, F, G and G, H, I.

First, we find the PGM with the maximum number of minicolumns and declare all its linked neurons as PA neurons. Thus all linked neurons in PGM that contains minicolumns C, D, E, F, G are PA neurons. In this way, we take advantage of the previously stored episodic fragments, strengthening their joint probability of activations.

If a smaller PGM has minicolumns that overlap with the larger PGM, then its overlapping minicolumns are removed from the PGM. For instance, PGM composed of A, B, C, D has minicolumns C and D that are also a part of a larger PGM, and thus this PGM is reduced to two neurons A, B. If after reduction a PGM has less than 2 neurons, it is trivial and does not define any PA neurons. When PGMs overlap we need means to determine where the new synaptic connections are added to represent the new contextual relationship observed. The process of removing overlapping minicolumns from smaller PGMs helps to find the non-overlapping sequences of previously stored episodes and to determine where the new synaptic connections are added. While we could find the non-overlapping sequences of previous episodes by removing the overlapping minicolumns

from the larger PGM, doing so can lead to fragmentation of stored episodes. Note that the removal of overlapping minicolumns from smaller PGM does not remove existing synaptic connections.

Similarly, minicolumn G in the PGM graph G, H, I overlaps with the larger PGM graph C, D, E, F, G. Hence, this PGM graph (G, H, I) can only define two PA neurons in columns H and I. Such overlapping PGMs can be merged by removing the overlapping neurons from the smaller PGMs and adding new synaptic connections from the predecessor PGM to a successor PGM as illustrated in Figure 5-3, where a predecessor PGM is the one that has minicolumns whose activation precedes activation of minicolumns in the successor PGM.

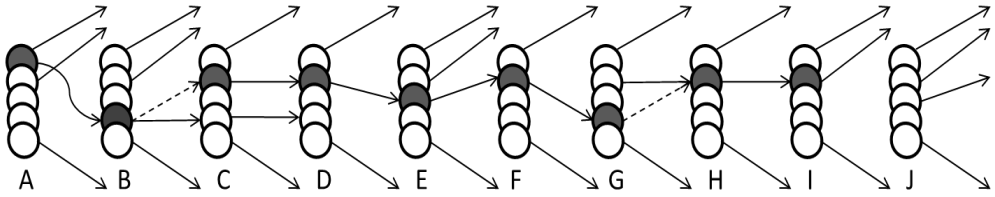

Figure 5-3. Merging of three overlapping PGMs.

A new synaptic connections are added from the end of the predecessor PGM (neurons B and G) to the first neuron of the successor PGM (neuron C), and from the end of the predecessor PGM (neurons C, D, E, F, and G) to the first neuron of the successor PGM (neuron H).

5.3.2 Establishing a Sequence of Linked PA Neurons

After merging of the overlapping PGMs, we may end up with more than one sequence of linked PA neurons (based on the linked episodes). Figure 5-4 shows such a case in which the previously discussed sequence A-J is just a subsequence that follows another sequence K-R. Here, for simplicity, we represent each sequence of the linked episodes by a single predecessor-successor link with a double solid line

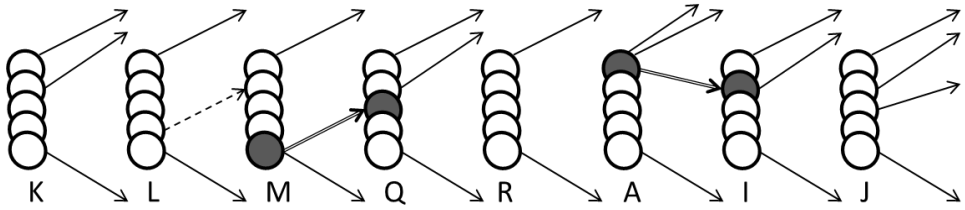

Figure 5-4. A longer sequence of activated minicolumns.

To establish a sequence of linked PA neurons in all the activated minicolumns that are needed in II.5 of the LUMAKG algorithm, we follow the steps specified in the Locating PA neurons algorithm.

Locating PA neurons (LPAN) algorithm:

1. Find the first minicolumn with a PA neuron. If no such column exists, choose a neuron in the last minicolumn with the minimum number of outgoing connections and treat it as a PA neuron. Name this first minicolumn with PA neuron FPA minicolumn.
2. Starting from the predecessor minicolumn to FPA,

a. Choose a neuron in this minicolumn that has a link to the PA neuron in FPA and treat it as a PA neuron.
b. If no such neuron exists, choose a neuron in the predecessor minicolumn with the minimum number of outgoing connections (named here as a MNOC neuron) and treat it as a PA neuron. This establishes a link between the two PA neurons.

Repeat this step for the new PA neuron, selecting a neuron in its predecessor minicolumn with the minimum number of outgoing connections and treat it as a PA neuron, until no predecessor minicolumn is found.

3. Starting from the PA neuron in the first activated minicolumn, follow the path to the last connected PA neuron in the input sequence and repeat this step until no successor minicolumn is found:
   a. If the successor minicolumn has a PA neuron, link the two PA neurons and follow the path to the last connected PA neuron.
   b. If the successor is a UA minicolumn, choose a MNOC neuron in this minicolumn and treat it as a PA neuron. Link the two PA neurons and move to the successor minicolumn.

5.3.3 Design Example

Let us illustrate this location of PA neurons by continuing our example. In Figure 5-4, the first minicolumn with a PA neuron is M, so according to step 1 of the LPAN algorithm, we name M the FPA minicolumn and move to step 2. In the predecessor minicolumn L, there was no neuron that linked to PA in M. Notice that although there

was a link between 4-th neuron in L and the minicolumn M, it did not link to a PA neuron in this minicolumn, so it could not be used.

Following 2.b of the LPAN algorithm, we chose a MNOC neuron in the minicolumn L and treat it as a PA neuron. The selected MNOC neuron in L is treated as a new PA neuron. This established a new link between the two PA neurons as shown by a dashed line in Figure 5-5.

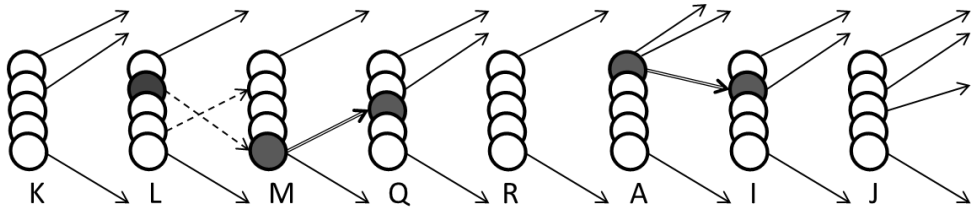

Figure 5-5. A new connection between the UA minicolumn L and a PA neuron in the minicolumn M.

Next, the LPAN algorithm moves back to the minicolumn K. Applying step 2.b of the LPAN algorithm again we chose a neuron in K with the minimum number of the outgoing connections and link it to the PA neuron in the minicolumn L as shown in Figure 5-6.

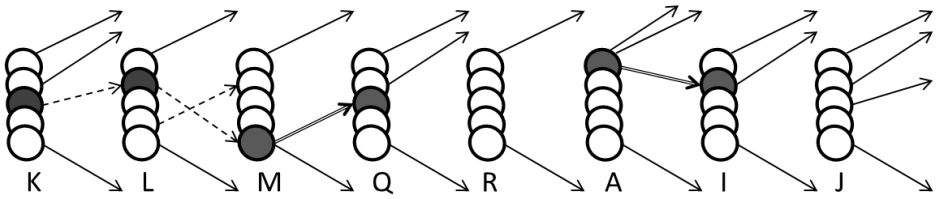

Figure 5-6. A new connection between UA minicolumn K and a PA neuron in minicolumn L.

Since there is no predecessor to K, according to the step 3 of the LPAN algorithm, we follow the path from K to the last connected PA neuron in the input sequence which is the neuron Q. Since the successor minicolumn (R) does not have a PA neuron, we follow step 3.b of the LPAN algorithm and choose a MNOC neuron in this minicolumn, and treat it as a PA neuron. This establishes a new link between these two PA neurons in Q and R as illustrated in Figure 5-7 and we move to minicolumn R.

Subsequently, following step 3.a of the LPAN algorithm, we link PA neurons in minicolumns R and A and move to the PA neuron in minicolumn I. We finish by applying step 3.b of the LPAN algorithm which will choose a PA neuron in the minicolumn J and link the PA neurons in I and J as shown in Figure 5-7.

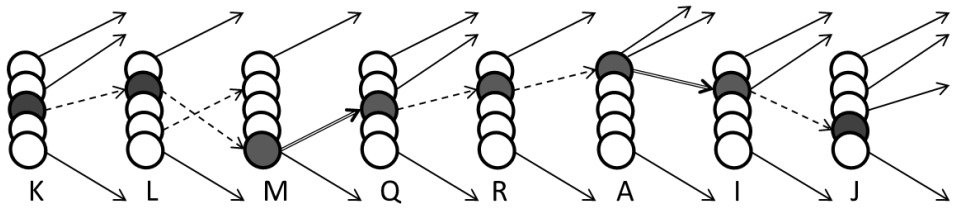

Figure 5-7. New connections between the PA neuron in Q and a selected MNOC neuron in the minicolumn R. Additional connections are established between the PA neurons in R and A and between I and a new PA neuron in J.

This completes the LPAN algorithm and at the same time point II.5 of the LUMAKG algorithm. As a result, we have a sequence of connected PA neurons from the first to the last minicolumn activated by the input sequence. Since there is no successor minicolumn to J, the LUMAKG algorithm moves to III.6 and modifies the synaptic weights between all the established predecessor-successor neurons in the input sequence. Their weights are modified according to the ANAKG algorithm [81].

After application of the ANAKG algorithm to modify weights between the selected PA neurons, we will get all the updated links as shown in Figure 5-8.

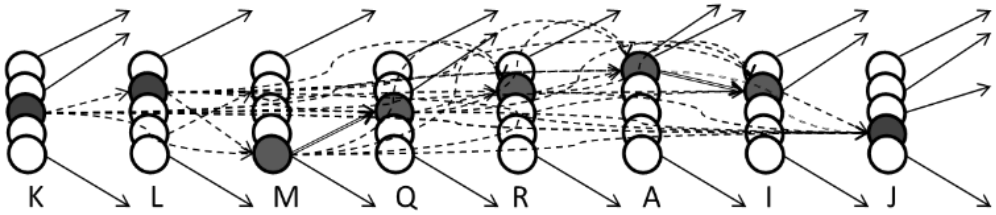

Figure 5-8. Modified synaptic connections for the input sequence.

5.4 Comparative Tests of LUMAKG

Several tests were performed to observe the efficiency of learning, memory capacity, and learning resolution for LUMAKG sequential memory, comparing them with similar features of the ANAKG memory and LSTM. A single layer LSTM network with 256 units was created using the TensorFlow library [121] and was used for testing.

5.4.1 Test Preparation

The first test is used to compare the resolution of recalled sentences using LUMAKG, ANAKG, and LSTM. To test the recall resolution, the memories were self-organized on an input file containing the text from The Children's Book Test or CBT [122]. Note that special characters, *e.g.* commas, periods, *etc*., were discarded and not used in training the memories. We read all sentences that were at least 10 words long from the database, providing us with over 19,000 sentences with over 9,000 unique words. The same sentences were used to obtain LUMAKG, ANAKG, and LSTM memory structures and their respective synaptic connections. After the three memories had been created, their associative memory properties and recall resolution were tested and compared.

To compare how accurately the memories recall stored sequences, we first trained the memories with the first 10 words of the sentences. Subsequently, the first 6 words from each training sequence were used as an input to the LUMAKG, ANAKG, and LSTM memories, and the original test sequences were used as the desired responses. All three memories require only a single presentation of the input data to learn. To observe the effect of increasing the training set size, we started with 100 sentences and gradually

increased this size to 10,000 sentences, in increments of 100 sentences for the first 1,000 sentences and subsequently in increments of 1,000. To illustrate how a number of neurons in minicolumn affects the results, three LUMAKG memory structures with minicolumn sizes 4, 8, and 12 were tested.

5.4.2 Network Response Quality Measures

A variety of heuristics and evaluation measures for information retrieval and related tasks have been proposed, *e.g.* answer scoring and/or ranking [123], passage retrieval [124], and evaluating search engines [125]. These evaluation measures require the use of tools such as parsers and consequently are not well suited for evaluation of the responses generated by the LSTM, ANAKG, and LUMAKG memories. Consequently, here we make use of the Levenshtein distance [126], and a new distance measure called reciprocal word position based on the evaluation metrics from [127].

5.4.2.1 Levenshtein Distance Quality Measure

The quality of results obtained from the LSTM, ANAKG, and LUMAKG memories were first measured by comparing them to the desired output using the Levenshtein distance [126]. Since we are interested in sequences of words rather than individual characters, the Levenshtein distance measured the number of words that must be deleted, inserted, or substituted in order to transform the source sentence to a target sentence. Each word had a unique symbol in the associative memories, and sequences of such symbols represented the output from each memory.

The Levenshtein distance between two strings *a* and *b* (of lengths *u* and *v* respectively) is given by (5-3):

$$d_{a,b}(i,j) = \begin{cases} max(i,j) & if\ min(i,j) = 0 \\ min \begin{cases} d_{a,b}(i-1,j) + 1 & deletion \\ d_{a,b}(i,j-1) + 1 & insertion \\ d_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} & substitution \\ d_{a,b}(i-1,j-1) & do\ nothing \end{cases} & otherwise \end{cases} \quad (5\text{-}3)$$

where $d_{a,b}(i,j)$ is the distance between the first *i* and *j* elements of *a* and *b* respectively. Here, we represent each word in a sentence as a symbol and in computing the Levenshtein distance measure between two sentences, we compare two strings of symbols.

Test I used training and testing sentences as described in the *Section 5.4.1*. We tested the responses of the networks to the same set of inputs, and output sequences obtained by each network were compared with the desired responses using the Levenshtein distance. The larger the Levenshtein distance is, the less similar stored and recalled sequences are, so this distance can be used to compare the quality of the sequential memory.

Figure 5-9 shows the mean Levenshtein distances for LSTM, ANAKG, and LUMAKG memories as a function of the number of symbols used in the training data. The test results show that the average of the mean Levenshtein distance between the desired responses and those generated by LSTM and ANAKG memory across all tests was 3.23 and 3.48 respectively while that for the LUMAKG memory was 2.21, 1.65, and 1.30 for column sizes 4, 8, and 12 respectively.

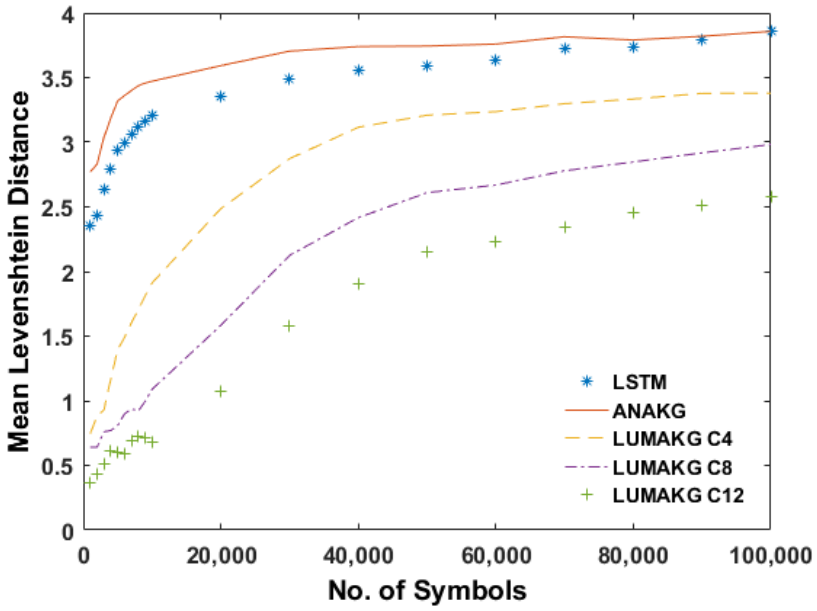

Figure 5-9. Plot of mean Levenshtein distances for the LSTM, the ANAKG network and the LUMAKG networks of different column sizes as a function of the number of symbols.

Since the used symbols may be repeated many times in the sentences, the number of unique symbols (words) grows slower than the number of all symbols used in training. Figure 5-10 shows the mean Levenshtein distances for the LSTM, ANAKG, and LUMAKG memories as a function of the number of unique symbols used. We see the similar dependence of the distances between stored and restored sequences as on the previous figure. This indicates that as the number of words in the memory grow, it is more difficult to restore the original sequence. Thus, if we set some recall standards, this will determine the memory capacity.

The results obtained from LUMAKG were significantly better than those obtained from both LSTM and ANAKG. The performance of LSTM, while initially better than

ANAKG, begins to get worse as network size increases and reaches similar values at 100000 symbols presented. We can observe that as the network grows in size, the quality of recall expressed by the Levenshtein distance is lower. All types of associative memories showed that they could provide a reasonable output given a limited training data set. However, LUMAKG has the promise to significantly increase both the resolution and storage capacity of the associative knowledge graphs and become a foundation for the semantic memory capable of remembering episodes, making associations and accumulating of knowledge.

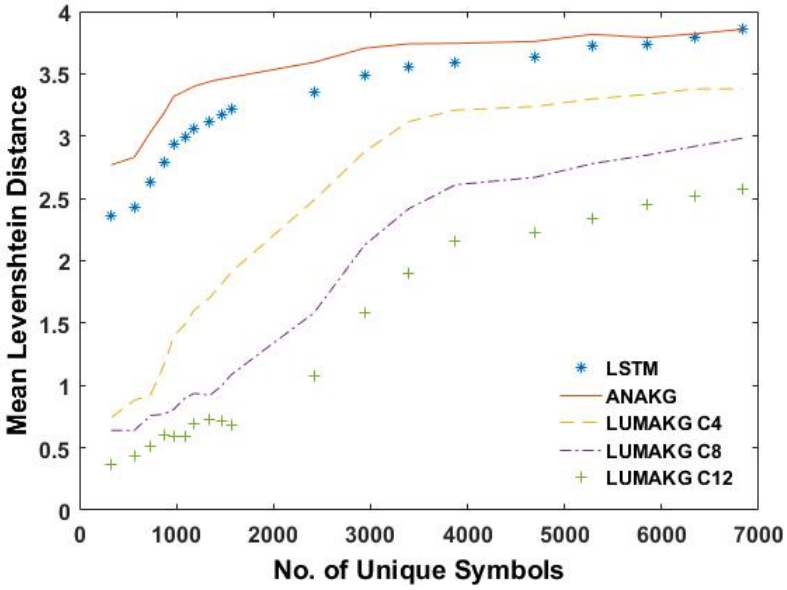

Figure 5-10. Plot of mean Levenshtein distances for the LSTM, the ANAKG network and the LUMAKG networks of different column sizes as a function of the number of unique symbols.

5.4.2.2 Reciprocal Word Position

A challenge in evaluating responses of associative spatiotemporal memories, like those based on LSTM and ANAKG, is linked to the difficulty in determining what the correct response is? Thus, the usefulness of Levenshtein distance, a good measure of text similarity, is limited. To address this problem, we designed a new distance measure called the reciprocal word position (RWP).

The RWP measures the user's effort in extracting the desired response from the output generated by the semantic memory. The RWP distance between two sequences *a* and *b* is calculated as follows:

1. Compare the positions of all the words in the desired output sequence (desired response) *a* to those in the actual memory output sequence (obtained response) *b*,
    a. if the positions of a word in both sequences are the same, the word gets a weight of 1;
    b. if the positions are different by '*n*' locations in the tested sequence the word gets a weight of 1/(*n*+1); and
    c. if a word from the desired response does not exist in the obtained response, it gets a weight of 0;
2. The RWP distance equals to one minus the sum of the weights of all the words in the desired sequence divided by the maximum of the number of words in the desired and actual sequence:

$$d_{RWP}(a,b) = 1 - \frac{\sum_{i=1}^{k} \frac{1}{|p_{io} - p_{id}|+1}}{max(k,l)} \quad (5\text{-}4)$$

where k is the number of words in the desired sequence a, l is the number of words in the obtained sequence b, p_{io} is the position of word i in the obtained sequence b, and p_{id} is the position of word i in the desired sequence a.

3. The RWP distance satisfies the following conditions:

$$d_{RWP}(a,b) = 0 \Leftrightarrow a = b \text{ identity of indiscernibles}$$

$$d_{RWP}(a,b) = d_{RWP}(b,a) \text{ symmetry}$$

$$d_{RWP}(a,c) \leq d_{RWP}(a,b) + d_{RWP}(b,c) \text{ triangle inequality}$$

The measure is normalized since the lowest value is 0 and the highest is 1, and a lower value of RWP indicates a better match between the sequences. For example, assume that the desired output is “likes cold water” and the generated answer is “cold water likes”. Then the second and third words from the desired output are shifted by one position, whereas the first word is shifted by two positions in the generated output, and the resulting RWP distance is: 1- (1/2+1/2+1/3)/3 = 5/9.

The mean RWP measures for the LSTM, ANAKG, and LUMAKG memories are shown in Figure 5-11 as a function of the number of symbols. The test results of applying the RWP distance to the different memory outputs show that the average RWP distance between the desired response and the one generated by LSTM, and ANAKG memory was 0.81 and 0.87 respectively, while the average distance for the LUMAKG memory was 0.58, 0.45, and 0.38 for column sizes 4, 8, and 12, respectively.

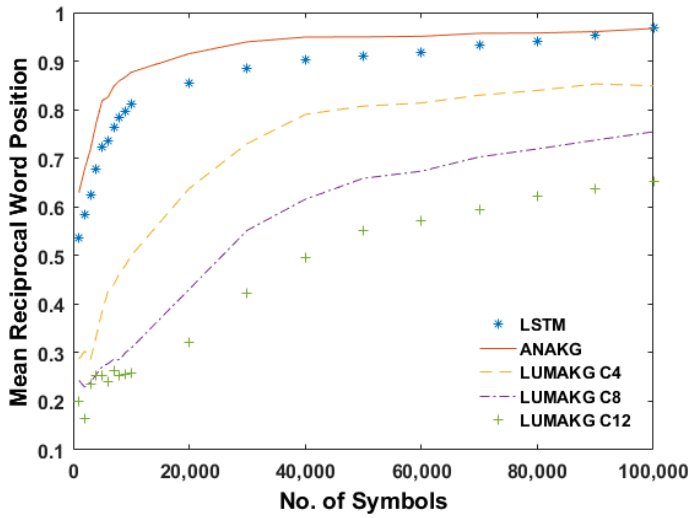

Figure 5-11. Plot of mean RWP for the LSTM, the ANAKG network and the LUMAKG networks of different column sizes as a function of the number of symbols.

These results also show that the performance of LUMAKG based semantic memory is better than LSTM, which is better than ANAKG based semantic memory, and its relative recall quality over both LSTM and ANAKG increases as the column size increases.

5.4.2.3 Recall Quality and Memory Capacity

The third type of tests was to show the dependence of the **memory capacity** on the number of neurons in each minicolumn and the number of objects (individual words) stored. The memory capacity can be established for a specific level of the **recall quality** (R_{QL}), where the recall quality for the results obtained with the Levenshtein distance is defined as:

$$R_{QL} = \underset{S}{mean}\left(1 - \frac{D_{Ls}}{max(L_{s1}, L_{s2})}\right) \tag{5-5}$$

where D_{LS} is the average Levenshtein distance divided by the maximum length (number of words) of the stored and recalled sentences, the average is taken over all the test sentences. The R_{QL} value is between 0 and 1, with 1 representing a perfect recall and 0 a completely wrong recall

We studied R_{QL} as a function of the number of objects used in sentences – objects are individual words and each repetition counts as a new object. In general, the larger the size of the associative memory is, the lower its recall quality is. In our test, we set the recall quality threshold T_{RQ}=70% and tested at which number of objects stored $R_{QL}<T_{RQ}$ to determine the **memory capacity**.

In a similar way, we may establish the memory capacity using the recall quality based on Reciprocal Word Position distance measure. Since distance based on RWP is already normalized, we can define recall quality as

$$R_{QL} = \underset{S}{mean}(1 - d_{RWPs}) \tag{5-6}$$

There is a strict correspondence between the threshold value set to establish the memory capacity and the distance measure used. In addition, although the two distance measures look similar, recall quality based on these measures are not.

For instance, from equation (5-5) it follows that a recall quality threshold of 70% equals a mean Levenshtein distance of 3, but this roughly corresponds to RWP distance equal 0.75 with the recall quality threshold 0.25. Considering that RWP is scaled between 0 and 1 it may be more convenient to specify memory capacity based on this measure.

Figure 5-12 (a) and (b) repeat results from Figure 5-9 and Figure 5-11 respectively. A reference line in Figure 5-12 (a) is used to determine memory capacity for

a Levenshtein distance equivalent to the recall quality threshold of 70%. The test results based on LSTM, ANAKG, and LUMAKG with minicolumn sizes of 4 show that the Levenshtein distance is less than 3 for networks with up to 6075, 2820, and 35230 symbols respectively.

Similarly, the reference line in Figure 5-12 (b) shows that LUMAKG networks with minicolumn sizes of 4, 8, and 12 have RWP less than 0.3 for networks with about 3290, 8995, and 16000 symbols respectively. Memory capacity based on this RWP distance would be zero for LSTM and ANAKG. To make them similar as obtained in the case of the Levenshtein distance, we should change the threshold of average RWP position to about 0.7 which correspond to recall quality value of 30%.

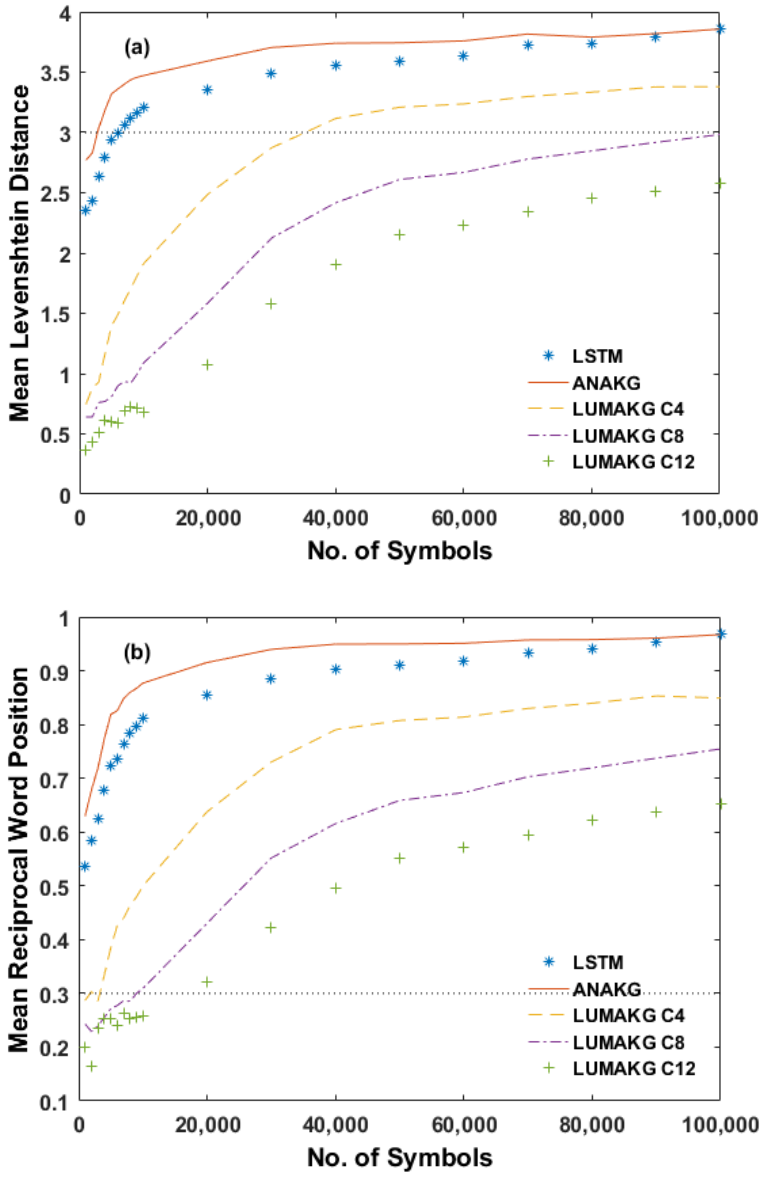

Figure 5-12. Plot of the mean (a) Levenshtein distances and (b) RWP, for the LSTM, the ANAKG networks and the LUMAKG networks of different column sizes as a function of the number of symbols. The reference line at recall quality threshold of 70% was added.

In summary, the results show that the recall quality and memory capacity of LUMAKG networks increases as the size of minicolumn increases and this is better than

ANAKG network. Recall the quality and memory capacity of LSTM is slightly better than ANAKG but not as good as LUMAKG.

5.4.2.4 Computational Complexity

The fourth type of tests was performed to determine the computational complexity of LUMAKG memory in comparison to LSTM and ANAKG memories. We tested the time needed to create the associative memory as a function of the number of objects. The results presented in Figure 5-13 show the learning time for LSTM, ANAKG, and LUMAKG as a function of all objects in the dataset. We see that computational cost for LUMAKG is between 60-200% higher than for ANAKG (depending on the minicolumn size) whereas that for ANAKG is about 6% higher than LSTM.

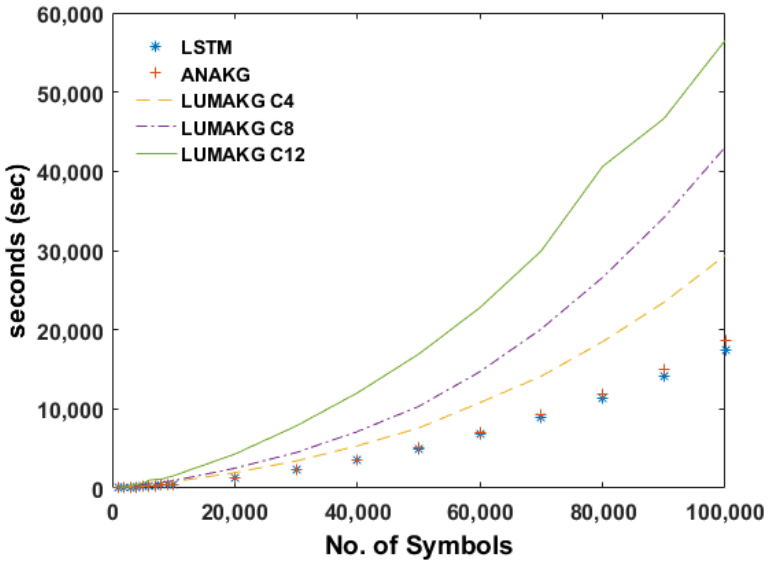

Figure 5-13. Learning times of the LSTM, ANAKG network and the LUMAKG network of different column sizes as a function of the number of symbols.

The overall increase in the simulation time shows less than the quadratic relationship to the size of the training data set, which allows storing a large number of sequences in LUMAKG memory. In addition, an increase in the simulation time due to an increase in the number of neurons in each minicolumn is less than linear. Again, this is a desired property of the developed method, since minicolumns in the human brain contain upwards of 100 neurons each. Tests were performed on a general purpose computer (i7-4790, 3.6GHz, 16 GB RAM).

The number of neurons in LUMAKG memory is k times larger than in ANAKG memory, where k is the number of neurons in each mini-column (in our tests k = 4, 8, or 12). However, the number of synapses does not grow as fast since the number of associative links between all neurons corresponds to the number of transitions between various words. These transitions are just spread over the larger number of neurons. Although the training time is greater for LUMAKG than for ANAKG due to the need of finding predecessor and successor for each element of the training sequence, the quality of results points to better properties of LUMAKG graphs, which make them more suitable to develop short-term associative memories.

One way of addressing the issue of training time would be to consider the requirements of the application (*e.g.* performance vs. computational time requirements) and appropriately determine the minicolumn size. To do this effectively, the above tests need to be repeated, with different datasets and with the randomized order of sequences. There are also some implementation level options for addressing the issue of training time. These include parallelization of the algorithms, and the neuron model to efficiently

use multiple cores or CPUs, offload operations to GPUs, and optimization of libraries used.

5.5 Conclusion

LUMAKG, a generalization of the ANAKG network used in Chapter 4 for the semantic memory, was presented in this chapter. LUMAKG memory supports continuous online learning, self-organization without supervised learning, context-based predictions, and is capable of recognizing time-domain sequences correctly. It improves ANAKG networks with similar properties. ANAKG memories and its derivatives are new types of memories that are under intensive investigation. Their properties are explored with a final goal to use them as basic models for self-organization of the semantic memories.

Test of recall quality, memory capacity, and computational complexity of LUMAKG networks were performed and compared to similar tests of ANAKG and LSTM memories. The effect of varying the number of neurons in minicolumns on the memory performance was also studied. LUMAKG shows better ability to recall sequences stored in the memories than LSTM or ANAKG. Using the Levenshtein distance and another quality measure, we also show that LUMAKG memory has higher capacity and better resolution for short-term memory recall. Future work could extend LUMAKG to a distributed representation of all symbols stored in the memory which will significantly increase its storage capacity. Further studies could also be performed on different types of input data, *e.g.* image and audio, to obtain an assessment of the network properties in different applications.

6 HANDWRITTEN DIGIT RECOGNITION USING ASSOCIATIVE ADAPTATION

In Chapter 2 the sufficiency of the chosen memory capabilities for a cognitive agent was shown. Subsequently in Chapters 4 and 5 the ability of the associative neuron model described in Chapter 3 to form associations, show creativity, recognize sequences and make predictions was shown. In this and the next chapter (Chapter 6 and 7) the ability of networks built with the associative neuron model to recognize objects will be shown through its use in two applications: a) image recognition, and b) multi-label classification, respectively. Note that if "ground truth" or labels drawn from a disjoint set are associated with an image, then recognition is the same as classification. And if individual images are associated with multiple-labels then it becomes a multi-label classification problem. In this chapter the phrase "image recognition" is used to represent a single-label classification problem. The rest of the chapter is organized as follows. Following a short introduction in *Section 6.1* and discussion of some existing neural network based algorithms applied to handwritten digit recognition in *Section 6.2*, the associative adaptation mechanism [86] is discussed in *Section 6.3*. In *Section 6.4* the performance of this network on handwritten digit recognition task are discussed and compared with those from other state-of-the art algorithms. Finally, *Section 6.5* provides the conclusion.

6.1 Introduction

Object recognition is an essential and important cognitive ability for all agents and is of fundamental importance for a variety of tasks. While the means for such

abilities can vary, the task itself can be better understood as one of classification or identification of various sensory stimuli. And irrespective of the sensory means that provides information about the features it is the brain that is responsible for recognizing the stimuli. In fact, the pattern recognition ability of the animal brain is unmatched, and consequently researchers have widely studied the visual systems of social animals such as bees and desert ants [128], [129] to gain insight into the cognitive abilities of higher order animals, such as mammals in general and humans in particular. Such studies, combined with research in other related areas, have inspired various models of learning including deep neural networks [130]. In spite of their biological inspiration the learning and inference mechanisms in these networks is not grounded in biology. While most of the differences are a consequence of the neuron model assumptions, the limitations of which were discussed in Chapter 3, there is one additional difference: the learning mechanism. Backpropagation [131], a training method in which each neuron, following the presentation of a sample from the input space, receives an error signal that is used to update its weight matrix and consequently decrease the output error, is the standard in artificial neural networks. But the implementation of such an error signal in the brain is highly unlikely [132].

A more plausible learning mechanism is the associative adaptation mechanism [86]. Traditional learning mechanisms can be classified into supervised or unsupervised based on the role of the label(s). In supervised learning the neural network has access to the label(s) during training and uses them in learning a function that can predict the class label(s) for the test samples. Meanwhile in unsupervised learning the training data is

unlabeled and the neural network attempts to infer the structural relationships preset in the samples. In the associative adaptation mechanism [86] the neural network has access to the label(s) during training, like in supervised learning, but the label(s) are treated like other features. That is, the neural network construction is not based on knowing which feature(s) play the role of the label(s) and consequently once the network is built any of the feature(s) in the input space can play the role of label space.

An example of the remarkable object recognition capability of the human neocortex is its ability to recognize objects that are either deformed or occur with variations. Examples of such objects include not just simple patterns like handwritten characters but also complex objects like produce in a grocery store. Such recognition tasks have been successfully performed by neural networks that have been first trained with large samples of images. Researchers have created multiple datasets of preprocessed and formatted data that can be used as representative samples to compare performance across algorithms. One such dataset is the MNIST database [70] of handwritten digits that will be used to illustrate the performance of the associative adaptation algorithm on object recognition. It contains 70,000 gray scale images of handwritten digits divided into two sets: a training set of 60,000 samples and a testing set of 10,000 samples. Each digit image has 784 pixels (28X28) with intensities ranging from 0-255.

6.2 Background

Pattern recognition systems have been researched for a long time, and over the last few decades various approaches such as linear classifiers, support vector machines, neural and convolutional nets have been proposed and implemented. These pattern

recognition systems have been applied to a variety of applications such as speech, character, handwriting, and sign recognition. Traditional pattern recognition systems consisted of two modules, a feature extraction module and a trainable classifier module. But over time research into the modules has led to them becoming areas of interest and research in their own right. For the purpose of this work it will be assumed that the input to the memory, and consequently the neural network, is the set of features (or preprocessed and formatted data) and not the raw input from the cognitive agent's sensors.

The recognition algorithm or classification techniques applied to handwritten digit recognition span the gamut of learning techniques, from simple linear classifiers [133] to deep convolutional neural networks [134]. Any feed-forward artificial neural network with two or more hidden layers between the input and output layer is a deep neural network (DNN) [135]. Convolutional neural network (CNN) is a type of deep neural network where the hidden layers consist of convolutional layers, activation function (RELU layer), pooling layers, fully connected layers and normalization layers. CNNs are inspired by the connectivity pattern of neurons in the visual cortexes of animals, where individual neurons respond only to stimuli from small regions of the visual field. While CNNs have long been studied [133] and are easier to train than similar sized feedforward neural networks, as they have fewer connections and parameters, and their performance is comparable, applying them to large scale images was expensive till recently [130]. Technological advancements, leading to wide availability of high-performance computing power at low-cost, has resulted in greater interest and research into techniques

based on principles DNN and CNN. The best performance on the MNIST dataset, with an accuracy of 99.77%, has been by CNN approach [134], with a standard feed forward neural network having an accuracy of 99.65% being tied for the 3rd spot with another CNN [70].

While most approaches to handwritten object recognition have been of the supervised learning variety, there have been a few unsupervised approaches [136], [137]. The best performance, of 95% accuracy, has been found obtain with a two layer spiking neural network using spike-timing-dependent plasticity (STDP) [136]. Note that in a previous work [138] we showed that a simple crossbar structure using memristors [139], an element relating charge and flux, with 784 (28X28) inputs and 300 outputs trained on only 2,600 training data points could achieve 80% accuracy on this task.

6.3 Associative Adaptation

The description of the associative adaptation process in this section is based on the details provided in [86]. The associative adaptation mechanism is grounded in an understanding of the working of biological neural networks. And a neural network constructed using this process can perform multiple computational tasks, such as similarity computation, object recognition, and multi-class and multi-label classification, that are generally performed by separate networks [86]. The fast adaptation mechanism modeled and used here results in depiction of classification results in the form of pulsing frequency, similar to biological networks but different from the artificial neural networks based on perceptron like neuron models that provide outputs in the form of real numbers (*e.g.* [0, 1] or [-1, 1]).

Neural networks in this process, called as associative pulsing neural networks (APNN), are dynamically developed and consist of elements such as associative neurons, receptors, and effectors [86]. While the receptors transform the input signal to the required internal representation, the effectors play a similar role on the output. Receptors are connected to neurons that internally represent the values sensed by the receptors. These internal neurons, especially when representing numerical values, can connect to other neurons representing similar or neighboring values. Each stimulated receptor $R_{v_i}^{a_k}$ by the value v^{a_k} and representing the value $v_i^{a_k}$ constantly charges its connected value neuron $S_{v_i}^{a_k}$ with the strength $x_{v_i}^{a_k}$ computed using (6-1) [86]:

$$x_{v_i}^{a_k} = \begin{cases} \left(1 - \frac{\left|v_i^{a_k} - v^{a_k}\right|}{r^{a_k}}\right)^q & if\ r^{a_k} > 0 \\ \left(\frac{\left|v_i^{a_k}\right|}{\left|v_i^{a_k}\right| + \left|v_i^{a_k} - v^{a_k}\right|}\right)^q & if\ r^{a_k} = 0 \end{cases} \tag{6-1}$$

where r^{a_k} is the current range of values of attribute a_k, and q decreases the sensitivity of a receptor to receptors representing close values. When an input stimulus is presented on the network's input, only the receptor representing the specific input value (or its two neighbors, if a receptor representing the exact value is not present) is stimulated. Weights or the connections between the value neurons representing numerical data is calculated as follows [86]:

$$w_{S_{v_i}^{a_k}, S_{v_j}^{a_k}} = \left(1 - \frac{\left|v_i^{a_k} - v_j^{a_k}\right|}{r^{a_k}}\right)^p \tag{6-2}$$

where $v_i^{a_k}$ and $v_j^{a_k}$ are the values represented by the connected receptors $R_{v_i}^{a_k}$ and $R_{v_j}^{a_k}$ stimulating connected neurons $S_{v_i}^{a_k}$ and $S_{v_j}^{a_k}$, and p decreases the influence of connected attribute neurons representing neighboring values

Object neurons, a special king of neuron that represents an individual training sample, can be used in the constructed network. It, the object neuron, is connected with all value neurons representing values in the pattern it represents. And if there exist any duplicate patterns, they are represented by the same object neuron. Weights of connections from value neurons $S_{v_i}^{a_k}$ to object neurons O_j are computed according to the number of incoming connections to the object neurons using (6-3) [86]:

$$w_{S_{v_i}^{a_k},O_j} = \frac{1}{K} \tag{6-3}$$

The reciprocal connections between the object neurons O_j to the sensory neurons $S_{v_i}^{a_k}$ are equal to their activation threshold value:

$$w_{O_j,S_{v_i}^{a_k}} = \theta \tag{6-4}$$

The computation of weights and thresholds is simple and powerful due to the combination of neuron characteristics, the resulting sparse graph structure, and time approach. In addition, the training data both influences receptors that charge the neurons and also automatically develop the structure that differs for various training data.

The implementation of time and various periods for the internal processes is important as it enables computation of the strength of associations during stimuli integration and during testing the differentiation between the network's answers [140]. The most frequently pulsing neuron represents the most associated values and objects, *i.e.* the response of the network is determined on the basis of the number of pulses of the

most frequently pulsing neurons. The multiple associations, both directly and indirectly, between the neurons representing the values and the objects means that it is not necessary to determine the class attribute during initial network creation. During external stimulation, as happens during the testing phase, the missing attribute values are computed automatically and pointed out as the neuron with the most pulses. This means that the development and training process of this network is different from classical neural networks. For classification applications the training data used contains the desired class label as one of the input features but is treated in a fashion similar to the other features. Since the learning process is neither supervised nor unsupervised, it can be called associative adaption.

Associative pulsing neural networks (APNN) for classification tasks are constructed using the following algorithm [86]:

1. Create an empty network.
2. Add Sensory Input Fields (SIF) for all data attributes, including the desired class attribute. All data attributes are treated the same. Each SIF is the collection of receptors representing aggregated values of a single data attribute. Use AVB-tree [89] to efficiently add, remove, or find represented values in each SIF.
3. For all training samples, use ASSORT-2 [141] to simulate existing receptors and find those representing the attribute values of a new object or create new receptors representing new values for each receptor separately. If a new

receptor is added, create a new value neuron for it and connect the receptor to it. Thus, receptors of each SIF aggregate duplicates of all trained objects.

4. Simulate all receptors representing attribute values of a new trained object simultaneously and let them simulate the connected value neurons. Allow value neurons charged to their pulsing threshold to stimulate neighbor value neurons and object neurons connected to them. Allow new value neurons to mutually connect to other value neurons representing the neighbor values using ASSORT-2 [141]. Use (6-2) to calculate the neighbor weights.
5. Add a new object neuron only if no existing object neuron pulses during the period of time necessary for charging a single neuron and connect it with all value neurons representing the values defining this object. Use (6-3) and (6-4) to compute synaptic weights of connections between value and object neurons.

Only one cycle of training is sufficient for the development of this associative neuronal structure. A network so constructed is ready to classify input data, recognize training samples or to determine the most similar objects (or training patterns) to the input stimulations provided via its receptors.

6.4 Results

A neural network was constructed using the algorithm described in the previous section by using the 60,000 samples in the MNIST training set. This was network was subsequently tested on the 10,000 samples from the testing set. While neural networks built with perceptron's or other similar neuron models provide their outputs as numbers

in the range [0, 1] or [-1, 1], the outputs of this network have to be determined from the number of times individual neurons spike after the input stimuli. Following principles of real neurons, a neuron that spikes the most frequently in comparison to other neurons during a given period of time is the one most associated with the input in the stimulation context. The neuron spiking the most often was taken as the networks output.

A comparison of the performances of the various classification models on the MNIST dataset are shown in Table 6-1.

Table 6-1. Comparison of handwritten digit recognition.

Network Type	**Preprocessing**	**Performance (Accuracy, %)**	**No. of Neurons**
Unsupervised Spiking Neural Net [136]	None	95.00 %	28X28 – N1 Excitatory – N1 Inhibitory N1 = 6,400
Deep Neural Network [142]	Elastic Distortions	99.65 %	784 – 2,500 – 2,000 – 1,500 – 1,000 – 500 – 10
Deep Convolutional Neural Network [134]	Elastic distortions (width normalization)	99.77 %	Committee of 35 conv. net, [elastic distortions]
Associative Adaptation (this work)	None	99.10 %	200,620

While both the unsupervised spiking neural network [136] and the associative adaptation networks were trained with the actual training set of 60,000 images from the MNIST dataset, the network in [136] was presented with the entire MNIST training set 15 times, *i.e.* the training was repeated for 15 cycles before testing. The training set size

to the deep neural network [142] was considerably increased through a process of deforming the images in the training set through a combination of affine and elastic deformations before each training epoch. And the deep convolutional neural network [134] also had its training set size increased through a process of width normalizations and use of multiple deep convolutional network columns per normalization. The results show that the performance of the network constructed through the associative adaptation process while less than that obtained with deep neural network [142] or deep convolutional neural network [134] is reasonable considering both its simplicity of construction and the smaller training set.

6.5 Conclusion

In this chapter an associative adaptation mechanism to construct neural network was described and the resulting network was applied to the problem of handwritten digit recognition. The results showed that the neural network so constructed had a less than 1% error. In future research the associative adaptation mechanism could be used to test its performance of other standard image recognition datasets. Another area of future research could be focused on modifications that result in creation of an object neuron only when no existing object neuron has a contextual similarity above a threshold for the present input. Other areas of possible future research include testing the performance of the network when there exists missing data in the testing set (or maybe even in the training set itself).

7 MULTI-CLASS CLASSIFICATION USING ASSOCIATIVE ADAPTATION

In Chapter 6 a new learning mechanism, called associative adaptation [86], based on the associative neuron model was described and was successfully applied to the object recognition problem from the MNIST dataset. Thus showing the ability of a memory designed with the associative neuron model from Chapter 3 to recognize objects. But real-world objects are not limited to such simple classes. That is, real-world classification problems can be sub-divided into single-label and multi-label classification, and the application in Chapter 6 provides proof for the single-label case. In this chapter the associative adaption algorithm is applied to multi-label classification problem.

7.1 Introduction

In single-label classification the aim is to determine a function or a model that can, based on observed sample label pairs, predict a label for unseen sample(s). The problem of single layer classification can further be subdivided into binary and multi-class problem. While the handwritten digit recognition problem from Chapter 6 is an excellent example of the multi-class classification problem, examples of binary single-label classification problems are spam and malware detection. Similarly, the aim of multi-label classification is to determine a function(s) or model(s) that can, following learning, predict the set of labels for unseen sample(s). The ubiquity of datasets with multiple labels, *e.g.* textual data such as webpages, has motivated research into multi-label classification [143]. Protein function prediction [144], image [145] and document [146] classification are some examples of multi-label classification applications.

The rest of the chapter is organized as follows. In *Section 7.2* the various approaches to multi-label classification traditionally used are described. This is followed by a discussion of the experimental results, including datasets and performance metrics in *Section 7.3*, and conclusion in *Section 7.4*.

7.2 Related Work

A variety of approaches have been tried for multi-label classification. The traditional approaches can be classified into three classes according to the order of label correlations: first-order, second-order, and high-order approaches.

First-order approaches are conceptually the simplest and they address the multi-label classification problem by decomposing it into a number of independent single-label classification problems, *i.e.* a family of q functions, one for each label are learned [147], [148]. First-order approaches while conceptually simple, efficient, and easiest to implement, do not account for label correlations and may be less efficient as a result.

Second-order approaches address the label correlations to some extent due to their consideration of *pairwise* relations between labels. That is, a family of q functions are learned by considering the interactions, such as co-occurrence patterns [149] or ranking constraints [150], between a pair of functions. While second-order approaches are relatively effective as they address label correlation to some extent, they are limited by the fact that label correlations in real-world data can extend beyond second-order.

Finally, higher-order approaches are the most complex due to their consideration of high-order relations between labels. More specifically, these approaches learn a family

of q functions by exploring the relations between a subset of functions [151] or among all functions [152].

In this work the associative adaptation approach is compared against three well-known approaches:

1. ML-KNN [148]: a first order approach that is an adaptation of the well-known k-NN algorithm.
2. BP-MLL [150]: a second-order approach that is derived from the backpropagation algorithm and uses an error function to capture the characteristics of multi-label learning.
3. LIFT [153]: a non-traditional approach that uses label-specific features for multi-label learning. That is, it addresses the multi-label learning problem from the perspective of the input space rather than the output space.

7.3 Experimental Results

In this section the effectiveness of the associative adaptation approach is evaluated on four widely used multi-label datasets and the results are compared with three state-of-the art techniques.

7.3.1 Datasets

Four benchmark multi-label datasets: emotions, image, scene, and yeast were used to evaluate the associative adaptation approach. An issue in multi-label datasets is: how multi-label is a dataset? This is important because not all multi-label datasets are created equally. For example, while in some datasets the number of labels associated with each sample is large in comparison to q, whereas it is small in others. And this could

influence the performance of multi-label classification techniques. Hence, two concepts, *label cardinality* and *label density*, that measure how multi-label a dataset is are generally used.

Label Cardinality: is the average number of labels per example, and is expressed as

$$\text{Label Cardinality} = \frac{1}{s}\sum_{i=1}^{s}|Y_i| \tag{7-1}$$

where s is the number of samples (or examples) in the multi-label data set S, and $Y_i \subseteq L$ is the set of labels of the i-th sample, where $L = \{\lambda_j : j = 1 \ldots q\}$ denotes the finite set of labels in the dataset.

Label Density: normalizes label cardinality by the number of labels, and is expressed as

$$\text{Label Density} = \frac{\text{Label Cardinality}}{q} \tag{7-2}$$

where q is the number of labels.

The detailed information of the selected datasets is provided in Table 7-1.

Table 7-1. Database statistics.

Data set	No. of Samples (s)	Features	No. of Labels (q)	Feature Type	Label Cardinality	Label Density
Emotions[1]	593	72	6	numeric	1.869	0.311
Image[2]	2000	294	5	numeric	1.236	0.247
Scene[1]	2407	294	6	numeric	1.074	0.179
Yeast[1]	2417	103	14	numeric	4.237	0.303

[1] http://mulan.sourceforge.net/datasets.html

[2] http://palm.seu.edu.cn/zhangml/Resources.htm#data

7.3.2 Evaluation Metrics

Performance evaluation in single-label classification is simple and metrics such as accuracy are generally used. But the association of each sample with multiple labels complicates evaluation in multi-label classification tasks. Hence a variety of evaluation metrics for multi-label classification have been proposed in literature and the following: hamming loss, Macro-F1 and Micro-F1 are used here. While hamming loss is an sample/instance based measure, sample/instance based measures evaluate the performance on each sample separately and then return the mean value across the test set, whereas label based metrics evaluate the performance on each class label separately and then return the macro/micro averaged value across all class labels.

Hamming Loss:

$$\text{Hamming Loss} = \frac{1}{p}\sum_{i=1}^{p}\left|Y_i \Delta \hat{Y}_i\right| \tag{7-3}$$

where p is the number of instances in the test set and Δ is the symmetric difference between the actual and predicted label sets Y_i and $\hat{Y}_i$ respectively, normalized over total number of instances. The hamming loss takes into account both the prediction error (incorrect prediction) and the missing error (label not predicted). Lower values of hamming loss are better.

The Macro-F1 and Micro-F1 are extensions of the F1 metric for binary classification. While Macro-F1 scores are obtained by first computing F1 scores on individual class labels and then averaging over all class labels, Micro-F1 scores are computed globally over all instances and all class labels.

Numerically Macro-F1 and Micro-F1 are defined as follows:

$$\text{Macro} - \text{F1} = \frac{1}{\text{q}}\sum_{\text{j}=1}^{\text{q}} \frac{2\sum_{\text{i}=1}^{\text{p}} \text{y}_\text{i}^\text{j}\hat{\text{y}}_\text{i}^\text{j}}{\sum_{\text{i}=1}^{\text{p}} \text{y}_\text{i}^\text{j}+\sum_{\text{i}=1}^{\text{p}} \hat{y}_\text{i}^\text{j}} \tag{7-4}$$

$$Micro - F1 = \frac{2\sum_{j=1}^{q}\sum_{i=1}^{p} y_i^j \hat{y}_i^j}{\sum_{j=1}^{q}\sum_{i=1}^{p} y_i^j+\sum_{j=1}^{q}\sum_{i=1}^{p} \hat{y}_i^j} \tag{7-5}$$

where p is the number of instances in the test set, q is the number of labels, and y_j^i and $\hat{y}_j^i$ are the actual and predicted labels, respectively, for instance i and label j. Higher values of Macro-F1 and Micro-F1 are better.

7.3.3 Performance Comparison

Ten-fold cross-validation was performed on the datasets. Tests were performed on a general purpose laptop (i7-6820HQ CPU, 2.7 GHz, 32 GB RAM). MATLAB implementations of BP-MLL, ML-KNN, and LIFT algorithms were obtained from the author's webpage[154]. The performance metrics (mean ± std. dev.) for the four chosen datasets for the various algorithms are shown in Table 7-2. The best values for a given combination of classification algorithm and dataset are bolded. The up and down arrows in Table 7-2 indicate the desired trend. Up arrow means that the bigger the result the better quality of the technique used, and down arrow means the opposite trend. The results show that the associative adaptation algorithm performs exceptionally well. It is either the best or second-best in all cases.

Table 7-2. Results of 10-fold cross-validation (mean ± std. dev).

Evaluation criterion	Algorithm	Emotions	Image	Scene	Yeast
Hamming loss ↓	**Associative Adaptation**	**0.186±0.017**	0.169±0.008	0.081±0.013	**0.192±0.017**
	LIFT	0.190±0.023	**0.163±0.019**	**0.075±0.011**	0.201±0.013
	ML-KNN	0.196±0.017	0.181±0.010	0.086±0.009	0.199±0.015
	BP-MLL	0.223±0.019	0.249±0.022	0.279±0.013	0.211±0.011
Macro-F1 ↑	**Associative Adaptation**	**0.472±0.019**	**0.408±0.023**	0.671±0.017	**0.548±0.013**
	LIFT	0.436±0.013	0.365±0.021	**0.694±0.010**	0.527±0.015
	ML-KNN	0.258±0.013	0.221±0.012	0.583±0.012	0.478±0.018
	BP-MLL	0.332±0.032	0.289±0.023	0.556±0.019	0.450±0.013
Micro-F1 ↑	**Associative Adaptation**	**0.481±0.015**	**0.401±0.012**	0.681±0.008	0.572±0.015
	LIFT	0.432±0.021	0.358±0.018	**0.683±0.013**	**0.575±0.021**
	ML-KNN	0.285±0.019	0.245±0.017	0.650±0.010	0.527±0.017
	BP-MLL	0.337±0.029	0.298±0.035	0.581±0.015	0.487±0.017

Simulation times for the ten-fold cross validation for the various algorithms and datasets are shown in Table 7-3. The results show that 10-fold cross-validation with ML-KNN takes the lowest time whereas the associative adaptation algorithm takes the highest. The associative adaptation algorithm, as previously described, creates a neuron for each unique input value. Consequently, the neural network structure resulting from the associative adaptation algorithm is very large and has about 37K, 530K, 614K, and 226K neurons for *emotions*, *image*, *scene*, and *yeast* datasets respectively. Simulation time for associative adaptation algorithm can be decreased through various means such as parallelization to efficiently use multiple cores, offload operations to GPUs, and use optimized libraries. Note that the BP-MLL and LIFT codes make use of optimized libraries like “Neural Network Toolbox” and LIBSVM package [154] respectively.

All algorithms take the lowest time for *emotions* dataset, the smallest of the four. While the simulation time of ML-KNN and LIFT algorithms for the other three datasets (*image*, *scene*, and *yeast*) increases with increase in number of labels (q), the simulation times for associative adaptation and BP-MLL algorithms increase with increasing number of unique values in the datasets.

Table 7-3. Simulation times for 10-fold cross-validation.

Evaluation criterion	**Algorithm**	**Emotions**	**Image**	**Scene**	**Yeast**
Simulation Time	**Associative Adaptation (hr)**	7.28	97.62	132.43	32.65
	LIFT (s)	10.7	116.9	118.3	476.9
	ML-KNN (s)	4.81	63.6	67.5	109.2
	BP-MLL (hr)	2.66	15.75	19.01	12.21

7.4 Conclusion

The associative adaptation approach from Chapter 6 was successfully used to construct neural networks for various multi-label datasets. The experimental results show that this approach compares well with widely used state-of-the art techniques in this area. Future research areas could include testing on multi-class multi-label datasets and extremely large datasets. Another area of future research could be comparing the performances with varying size of the training dataset.

8 CONCLUSIONS AND FUTURE WORK

Individual researchers have studied cognitive architectures, neural nets, and various applications. But as far as we can tell there have been very few attempts at neural network implementation of either complete cognitive architectures or large subsystems of them. Hence, the objective of this dissertation was to demonstrate the feasibility of implementing memory structures capable of performing various memory related tasks that are necessary for a cognitive agent using a biologically plausible associative pulsing neuron model. The complexity of the memory structures and learning mechanism required for a cognitive agent is dependent on the problem(s) it is designed to learn to solve, more complex structures and learning mechanisms are required to learn more complex goals. In *Chapter 1* the scope of the work was defined and it was assumed that the ability to create semantic relationships, demonstrate creativity, recognize sequences, make predictions, and recognize objects were necessary for the cognitive agent considered here to learn to solve a problem and demonstrate its intelligence. In *Chapter 2* a cognitive agent with these capabilities and grounded in the principles of motivated learning was shown to learn to achieve its goals, demonstrating that these capabilities were necessary for a cognitive agent designed to solve a specific problem. Subsequently in *Chapter 3* a biologically plausible associative pulsing neuron model that addresses various limitations of existing neuron models, such as computational cost and accounting for timing influences, was described. Modifications to this model to accommodate growth and sensitivity of neruons and their strength of connections, that result in

automatic threshold changes and specialization of neurons, were also proposed. In subsequent chapters various structures were implemented using this model.

First, semantic and episodic memories were developed using the principles of the ANAKG algorithm and tested on their associative properties and ability to recognize sequences. The results demonstrated that episodic memories were better at sequence recognition. Declarative memories, consisting of semantic and episodic memories, were also developed and their testing showed that these structures were capable of generalizing and consequently demonstrate creativity through its responses to questions whose context was non-unique or not present in the training sequences. Subsequently the ANAKG algorithm was generalized to its mini-column form, i.e., LUMAKG. Results of testing memory structure developed using the LUMAKG algorithm demonstrated that its recall quality, i.e., ability to recall training sequences, and memory capacity was better than LSTM and ANAKG memory structures.

Finally, the ability of a memory structure based on the associative pulsing neuron model to recognize objects was demonstrated using neural network structures created using the associative adaptation process. The performance of such networks was first tested on the handwritten digit recognition task. The results showed that network had an recognition error less than 1%. Subsequently the performance of the networks resulting from the associative adaptation process were successfully appled to the multi-label recognition task. The results showed that the performance of these networks were comparable to widely used state-of-the art techniques in this area.

Research is a never ending process of continuous improvement, always building upon existing knowledge. While the completion of this dissertation work means that one chapter of this process has ended, a new chapter of furthering knowledge begins. There are various possible directions for future research. One such area, and possibly the most immediate, is to improve the computational efficiency and speed. While computational efficiency could involve studying means of simplifying the structures, computational speed could be improved through hardware acceleration via use of dedicated hardware such as graphical processing units (*e.g.* NVIDIA's Tesla cards) or FPGAs. A second area of future research could involve the integration of all these abilities into a cognitive agent and test its performance in various environments, subsequently implementing memory for cognitive agents based on more complex cognitive architectures, including high-level memory functions such as planning and decision making. Other areas of future research could focus on neuromorphic hardware and development of the sensory/motor sub-systems to implement the information pipeline present in a cognitive agent. These are just a few possible research directions for future work.

REFERENCES

[1] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, “A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE.” [Online]. Available: http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html. [Accessed: 03-Mar-2015].

[2] A. M. TURING, “Computing Machinery and Intelligence,” *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950.

[3] J. A. Starzyk, “Motivation in embodied intelligence,” in *Frontiers in Robotics, Automation and Control*, A. Zemliak, Ed. InTech, 2008.

[4] J. A. Starzyk and J. Graham, “MLECOG: Motivated Learning Embodied Cognitive Architecture,” *IEEE Syst. J.*, vol. 11, no. 3, pp. 1272–1283, Sep. 2017.

[5] D. Marr, “Artificial intelligence—a personal view,” *Artif. Intell.*, vol. 9, no. 1, pp. 37–48, 1977.

[6] D. Ferrucci, A. Levas, S. Bagchi, D. Gondek, and E. T. Mueller, “Watson: Beyond Jeopardy!,” *Artif. Intell.*, vol. 199–200, pp. 93–105, Jun. 2013.

[7] J. E. Laird, *The Soar cognitive architecture*. MIT press, 2012.

[8] I. Kotseruba and J. K. Tsotsos, “40 years of cognitive architectures: core cognitive abilities and practical applications,” *Artif. Intell. Rev.*, pp. 1–78, Jul. 2018.

[9] J. R. Anderson, *How can the human mind occur in the physical universe?* Oxford University Press, 2009.

[10] P. Wang, “Natural Language Processing by Reasoning and Learning,” Springer,

Berlin, Heidelberg, 2013, pp. 160–169.

[11] U. Faghihi and S. Franklin, “The LIDA Model as a Foundational Architecture for AGI,” Atlantis Press, Paris, 2012, pp. 103–121.

[12] S. Schaat, A. Wendt, S. Kollmann, F. Gelbard, and M. Jakubec, “Interdisciplinary Development and Evaluation of Cognitive Architectures Exemplified with the SiMA Approach,” in *EuroAsianPacific Joint Conference on Cognitive Science*, 2015, pp. 25–27.

[13] D. V. Pynadath, P. S. Rosenbloom, and S. C. Marsella, “Reinforcement Learning for Adaptive Theory of Mind in the Sigma Cognitive Architecture,” Springer, Cham, 2014, pp. 143–154.

[14] B. Goertzel and G. Yu, “A Cognitive API and Its Application to AGI Intelligence Assessment,” Springer, Cham, 2014, pp. 242–245.

[15] W. Bridewell and P. Bello, “Incremental Object Perception in an Attention-Driven Cognitive Architecture.,” in *CogSci*, 2015.

[16] J. K. Tsotsos, “Attention and Cognition: Principles to Guide Modeling,” Springer, Singapore, 2017, pp. 277–295.

[17] U. Faghihi, P. Poirier, and O. Larue, “Emotional Cognitive Architectures,” Springer, Berlin, Heidelberg, 2011, pp. 487–496.

[18] T. C. Henderson, A. Joshi, and W. Wang, “The cognitive symmetry engine,” *Sch. Comput. Univ. Utah, Salt Lake City, UT, USA, Rep. UUCS-13-004*, 2013.

[19] S. L. Epstein, “Metaknowledge for autonomous systems,” in *Proceedings of AAAI Spring Symposium on Knowledge Representation and Ontology for Autonomous*

Systems. AAAI, 2004, pp. 61–68.

[20] D. Martín, M. Rincón, M. C. García-Alegre, and D. Guinea, "ARDIS: Knowledge-Based Dynamic Architecture for Real-Time Surface Visual Inspection," Springer, Berlin, Heidelberg, 2009, pp. 395–404.

[21] J. B. Maxwell, "Generative music, cognitive modelling, and computer-assisted composition in musicog and manuscore," ? by Home Dept & Faculty of Senior Supervisor: Special Arrangements, 2014.

[22] J. McCarthy and P. J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in *Readings in Artificial Intelligence*, Elsevier, 1981, pp. 431–450.

[23] D. F. Sherry and D. L. Schacter, "The Evolution of Multiple Memory Systems," *Psychol. Rev. Nadel*, vol. 94, no. 4, pp. 439–454, 1987.

[24] E. Tulving, "How many memory systems are there?," *Am. Psychol.*, vol. 40, no. 4, pp. 385–398, 1985.

[25] B. Gordon, "Preserved Learning of Novel Information in Amnesia: Evidence for Multiple Memory Systems," *BRAIN Cogn.*, vol. 7, pp. 257–282, 1988.

[26] M. G. Packard, R. Hirsh, and N. M. White, "Differential Effects of Fornix and Caudate Nucleus Lesions on Two Radial Maze Tasks: Evidence for Multiple Memory Systems," *J. Neurosci.*, vol. 9, no. 5, pp. 1465–1472, 1989.

[27] T. S. Collett and M. Collett, "Memory use in insect visual navigation," *Nat. Rev. Neurosci.*, vol. 3, no. 7, pp. 542–552, Jul. 2002.

[28] C. T. Fernando *et al.*, "Molecular circuits for associative learning in single-celled

organisms.," *J. R. Soc. Interface*, vol. 6, no. 34, pp. 463–9, May 2009.

[29] P. Hagmann *et al.*, "Mapping the Structural Core of Human Cerebral Cortex," *PLoS Biol.*, vol. 6, no. 7, p. e159, Jul. 2008.

[30] J. Duncan and A. M. Owen, "Common regions of the human frontal lobe recruited by diverse cognitive demands," *Trends Neurosci.*, vol. 23, no. 10, pp. 475–483, Oct. 2000.

[31] Y. Liu, J. A. Starzyk, and Z. Zhu, "Optimized approximation algorithm in neural networks without overfitting," *IEEE Trans. neural networks*, vol. 19, no. 6, pp. 983–995, 2008.

[32] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning," in *The elements of statistical learning*, Springer, 2009, pp. 485–585.

[33] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT press, 1998.

[34] J. Hawkins and S. Blakeslee, *On Intelligence*. Henry Holt and Company, 2007.

[35] J. A. Starzyk, "Motivated Learning for Computational Intelligence," *Mach. Learn. Concepts, Methodol. Tools Appl. Concepts, Methodol. Tools Appl.*, p. 120, 2011.

[36] M. Taddeo and L. Floridi, "Solving the symbol grounding problem: a critical review of fifteen years of research," *J. Exp. Theor. Artif. Intell.*, vol. 17, no. 4, pp. 419–445, Dec. 2005.

[37] X. Xu, J. Graham, Basawaraj, J. Zhu, J. A. Starzyk, and P. Zhang, "A Biopsychically Inspired Cognitive System for Intelligent Agents in Aerospace Applications," in *Infotech@Aerospace 2012*, 2012.

[38] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[39] R. Sun, *Anatomy of the mind: exploring psychological mechanisms and processes with the Clarion cognitive architecture*. New York, New York, USA: Oxford University Press, 2016.

[40] R. Sun, “The importance of cognitive architectures: an analysis based on CLARION,” *J. Exp. Theor. Artif. Intell.*, vol. 19, no. 2, pp. 159–193, Jun. 2007.

[41] D. Vernon, G. Metta, and G. Sandini, “A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents,” *IEEE Trans. Evol. Comput.*, vol. 11, no. 2, pp. 151–180, Apr. 2007.

[42] P. Ye, T. Wang, and F.-Y. Wang, “A Survey of Cognitive Architectures in the Past 20 Years,” *IEEE Trans. Cybern.*, vol. 48, no. 12, pp. 3280–3290, Dec. 2018.

[43] R. Vigo, “Musings on the utility and challenges of cognitive unification: Review of Anatomy of the Mind, Ron Sun. Rensselaer Polytechnic Institute (2016). 480 pp.,” *Cogn. Syst. Res.*, vol. 51, pp. 14–23, 2018.

[44] M. L. Anderson, “Embodied Cognition: A field guide,” *Artif. Intell.*, vol. 149, no. 1, pp. 91–130, Sep. 2003.

[45] M. Wilson, “Six views of embodied cognition,” *Psychon. Bull. Rev.*, vol. 9, no. 4, pp. 625–636, Dec. 2002.

[46] R. A. Brooks, “Intelligence without representation,” *Artif. Intell.*, vol. 47, no. 1–3, pp. 139–159, Jan. 1991.

[47] R. A. Brooks, *Cambrian intelligence : the early history of the new AI.* MIT Press, 1999.

[48] A. Clark, *Being there: Putting brain, body, and world together again*. MIT press, 1998.

[49] A. Clark, "Embodied, Situated, and Distributed Cognition," in *A Companion to Cognitive Science*, W. Bechtel and G. Graham, Eds. Malden, MA, USA: Blackwell Publishing Ltd, 1999, pp. 506–517.

[50] R. Pfeifer, C. Scheier, and I. Illustrator-Follath, *Understanding intelligence*. MIT Press, 2001.

[51] E. Thelen and L. B. Smith, *A dynamic systems approach to the development of cognition and action*. MIT press, 1996.

[52] R. A. Brooks, "Intelligence without reason, computers and thought lecture," *Proc. IJCAI-91, Sydney, Aust.*, pp. 569–595, 1991.

[53] R. A. Brooks, *Flesh and machines: How robots will change us*. Pantheon Books New York, 2002.

[54] J. R. Anderson, "ACT: A simple theory of complex cognition.," *Am. Psychol.*, vol. 51, no. 4, pp. 355–365, 1996.

[55] D. Dörner and C. D. Güss, "PSI: A computational architecture of cognition, motivation, and emotion.," *Rev. Gen. Psychol.*, vol. 17, no. 3, pp. 297–317, 2013.

[56] E. M. Izhikevich, "Which Model to Use for Cortical Spiking Neurons?," *IEEE Trans. Neural Networks*, vol. 15, no. 5, pp. 1063–1070, Sep. 2004.

[57] S. Haykin, "Neural Networks: A Comprehensive Foundation," 1998.

[58] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, Jan. 2015.

[59] J. Hawkins, S. Ahmad, and D. Dubinsky, “Hierarchical temporal memory including HTM cortical learning algorithms,” *Techical report, Numenta, Inc, Palto Alto http//www. numenta. com/htmoverview/education/HTM_CorticalLearningAlgorithms. pdf*, 2010.

[60] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. New York, New York, USA: John Wiley & Sons, 1949.

[61] S. Haykin, *Neural Networks and Learning Machines*. 2009.

[62] N. Caporale and Y. Dan, “Spike Timing–Dependent Plasticity: A Hebbian Learning Rule,” *Annu. Rev. Neurosci.*, vol. 31, no. 1, pp. 25–46, Jul. 2008.

[63] T. Kohonen, “The self-organizing map,” *Proc. IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[64] P. RODRIGUEZ, J. WILES, and J. L. ELMAN, “A Recurrent Neural Network that Learns to Count,” *Conn. Sci.*, vol. 11, no. 1, pp. 5–40, Mar. 1999.

[65] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[66] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.

[67] P. Raif and J. A. Starzyk, “Motivated learning in autonomous systems,” in *Neural Networks (IJCNN), The 2011 International Joint Conference on*, 2011, pp. 603–610.

[68] J. A. Starzyk and D. K. Prasad, “A computational model of machine consciousness,” *Int. J. Mach. Conscious.*, vol. 3, no. 02, pp. 255–281, 2011.

[69] E. Tulving and others, “Episodic and semantic memory,” *Organ. Mem.*, vol. 1, pp. 381–403, 1972.

[70] Y. LeCun, C. Cortes, and C. J. C. Burges, “MNIST handwritten digit database.” [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed: 04-Mar-2015].

[71] D. Purves *et al.*, Eds., “Eye Movements and Sensory Motor Integration,” in *Neuroscience*, 3rd ed., Sunderland (MA), USA: Sinauer Associates, 2004.

[72] “CREATIVITY | definition in the Cambridge English Dictionary.” [Online]. Available: https://dictionary.cambridge.org/us/dictionary/english/creativity. [Accessed: 20-Apr-2019].

[73] A. Horzyk, J. A. Starzyk, and Basawaraj, “Emergent creativity in declarative memories,” in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8.

[74] R. Yuste, “From the neuron doctrine to neural networks,” *Nat. Rev. Neurosci.*, vol. 16, no. 8, pp. 487–497, Aug. 2015.

[75] N. Brunel and M. C. W. van Rossum, “Lapicque’s 1907 paper: from frogs to integrate-and-fire,” *Biol. Cybern.*, vol. 97, no. 5–6, pp. 337–339, Dec. 2007.

[76] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, Dec. 1943.

[77] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve,” *J. Physiol.*, vol. 117, no.

4, pp. 500–544, Aug. 1952.

[78] C. Meunier and I. Segev, “Playing the Devil’s advocate: is the Hodgkin–Huxley model useful?,” *Trends Neurosci.*, vol. 25, no. 11, pp. 558–563, Nov. 2002.

[79] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[80] W. Gerstner and W. M. Kistler, *Spiking neuron models : single neurons, populations, plasticity*. Cambridge University Press, 2002.

[81] A. Horzyk, “How does generalization and creativity come into being in neural associative systems and how does it form human-like knowledge?,” *Neurocomputing*, vol. 144, pp. 238–257, Nov. 2014.

[82] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Networks*, vol. 10, no. 9, pp. 1659–1671, Dec. 1997.

[83] D. Ferster and N. Spruston, “Cracking the neuronal code.,” *Science*, vol. 270, no. 5237, pp. 756–7, Nov. 1995.

[84] J. J. Hopfield, “Pattern recognition computation using action potential timing for stimulus representation,” *Nature*, vol. 376, no. 6535, pp. 33–36, Jul. 1995.

[85] P. S. Churchland and T. J. (Terrence J. Sejnowski, *The computational brain*. MIT Press, 1992.

[86] A. Horzyk and J. A. Starzyk, “Multi-Class and Multi-Label Classification Using Associative Pulsing Neural Networks,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1–8.

[87] A. Longstaff, *BIOS Instant Notes in Neuroscience*. New York, New York, USA:

Garland Science, 2011.

[88] J. W. Kalat, *Biological psychology*. Wadsworth, CA, USA: Nelson Education, 2015.

[89] A. Horzyk, “Deep Associative Semantic Neural Graphs for Knowledge Representation and Fast Data Exploration.,” in *KEOD*, 2017, pp. 67–79.

[90] A. Horzyk, J. A. Starzyk, and J. Graham, “Integration of Semantic and Episodic Memories,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 28, no. 12, pp. 3084–3095, Dec. 2017.

[91] M. Prinz, B. Prinz, and E. Schulz, “The growth of non-pyramidal neurons in the primary motor cortex of man: a Golgi study.,” *Histol. Histopathol.*, vol. 12, no. 4, pp. 895–900, Oct. 1997.

[92] J. A. Starzyk and H. He, “Spatio–Temporal Memories for Machine Learning: A Long-Term Memory Organization,” *IEEE Trans. Neural Networks*, vol. 20, no. 5, pp. 768–780, May 2009.

[93] B. P. Bean, “The action potential in mammalian central neurons,” *Nat. Rev. Neurosci.*, vol. 8, no. 6, pp. 451–465, Jun. 2007.

[94] N. S. Simons-Weidenmaier, M. Weber, C. F. Plappert, P. K. Pilz, and S. Schmid, “Synaptic depression and short-term habituation are located in the sensory part of the mammalian startle pathway,” *BMC Neurosci.*, vol. 7, no. 1, p. 38, May 2006.

[95] H. Zhang *et al.*, “Synaptic Fatigue is More Pronounced in the APP/PS1 Transgenic Mouse Model of Alzheimers Disease,” *Curr. Alzheimer Res.*, vol. 2, no. 2, pp. 137–140, Apr. 2005.

[96] H. Eichenbaum, “The hippocampus and declarative memory: cognitive mechanisms and neural codes,” *Behav. Brain Res.*, vol. 127, no. 1–2, pp. 199–207, Dec. 2001.

[97] E. T. Rolls, “A computational theory of episodic memory formation in the hippocampus,” *Behav. Brain Res.*, vol. 215, no. 2, pp. 180–196, Dec. 2010.

[98] E. J. Gibson, “Perceptual Learning in Development: Some Basic Concepts,” *Ecol. Psychol.*, vol. 12, no. 4, pp. 295–302, Oct. 2000.

[99] L. Shastri, “Episodic memory and cortico–hippocampal interactions,” *Trends Cogn. Sci.*, vol. 6, no. 4, pp. 162–168, Apr. 2002.

[100] A. Nuxoll and J. E. Laird, “A Cognitive Model of Episodic Memory Integrated with a General Cognitive Architecture,” *Iccm*, 2004.

[101] J. A. Starzyk and H. He, “Anticipation-Based Temporal Sequences Learning in Hierarchical Structure,” *IEEE Trans. Neural Networks*, vol. 18, no. 2, pp. 344–358, Mar. 2007.

[102] D. L. Wang and B. Yuwono, “Anticipation-based temporal pattern generation,” *IEEE Trans. Syst. Man. Cybern.*, vol. 25, no. 4, pp. 615–628, Apr. 1995.

[103] Basawaraj, J. A. Starzyk, and A. Horzyk, “Lumped mini-column associative knowledge graphs,” in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–8.

[104] J. R. Binder and R. H. Desai, “The neurobiology of semantic memory,” *Trends Cogn. Sci.*, vol. 15, no. 11, pp. 527–536, Nov. 2011.

[105] E. Plaku, L. E. Kavraki, and M. Y. Vardi, “Discrete Search Leading Continuous

Exploration for Kinodynamic Motion Planning.," in *Robotics: Science and Systems*, 2007, pp. 326–333.

[106] M. A. Kramer, "Autoassociative neural networks," *Comput. Chem. Eng.*, vol. 16, no. 4, pp. 313–328, Apr. 1992.

[107] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 5528–5531.

[108] J. Weston, S. Chopra, and A. Bordes, "Memory Networks," Oct. 2014.

[109] M. Iyyer, J. Boyd-Graber, L. Claudino, R. Socher, and H. Daumé III, "A Neural Network for Factoid Question Answering over Paragraphs," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 633–644.

[110] A. Graves, G. Wayne, and I. Danihelka, "Neural Turing Machines," Oct. 2014.

[111] J. Hawkins, S. Ahmad, and D. Dubinsky, "Cortical learning algorithm and hierarchical temporal memory," *Numenta Whitepaper*, pp. 1–68, 2011.

[112] Y. Cui, S. Ahmad, and J. Hawkins, "Continuous Online Sequence Learning with an Unsupervised Neural Network Model," *Neural Comput.*, vol. 28, no. 11, pp. 2474–2504, Nov. 2016.

[113] H. Jaeger, "Adaptive nonlinear system identification with echo state networks," in *Advances in neural information processing systems*, 2003, pp. 609–616.

[114] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553,

pp. 436–444, May 2015.

[115] J. M. McFarland, Y. Cui, and D. A. Butts, “Inferring Nonlinear Neuronal Computation Based on Physiologically Plausible Inputs,” *PLoS Comput. Biol.*, vol. 9, no. 7, p. e1003143, Jul. 2013.

[116] E. M. Izhikevich, “Simple model of spiking neurons,” *IEEE Trans. Neural Networks*, vol. 14, no. 6, pp. 1569–1572, Nov. 2003.

[117] E. M. Izhikevich, “Polychronization: Computation with Spikes,” *Neural Comput.*, vol. 18, no. 2, pp. 245–282, Feb. 2006.

[118] J. Hawkins and S. Ahmad, “Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex,” *Front. Neural Circuits*, vol. 10, p. 23, Mar. 2016.

[119] “GitHub - numenta/nupic: Numenta Platform for Intelligent Computing is an implementation of Hierarchical Temporal Memory (HTM), a theory of intelligence based strictly on the neuroscience of the neocortex.” [Online]. Available: https://github.com/numenta/nupic. [Accessed: 12-Jul-2018].

[120] O. Sporns, G. Tononi, and R. Kötter, “The Human Connectome: A Structural Description of the Human Brain,” *PLoS Comput. Biol.*, vol. 1, no. 4, p. e42, 2005.

[121] “TensorFlow.” [Online]. Available: https://www.tensorflow.org/. [Accessed: 27-May-2018].

[122] “bAbI - Facebook Research.” [Online]. Available: https://research.fb.com/downloads/babi/. [Accessed: 02-Jun-2018].

[123] S. K. Ray, S. Singh, and B. P. Joshi, “A semantic approach for question

classification using WordNet and Wikipedia," *Pattern Recognit. Lett.*, vol. 31, no. 13, pp. 1935–1943, Oct. 2010.

[124] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton, "Quantitative evaluation of passage retrieval algorithms for question answering," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval - SIGIR '03*, 2003, p. 41.

[125] S. Büttcher, C. L. A. Clarke, and G. V Cormack, *Information retrieval: Implementing and evaluating search engines*. Mit Press, 2016.

[126] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet physics doklady*, 1966, vol. 10, no. 8, pp. 707–710.

[127] D. R. Radev, H. Qi, H. Wu, and W. Fan, "Evaluating Web-based Question Answering Systems.," in *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC'02)*, 2002.

[128] R. Wehner, B. Michel, and P. Antonsen, "Visual navigation in insects: coupling of egocentric and geocentric information," *J. Exp. Biol.*, vol. 199, no. 1, pp. 129–140, 1996.

[129] S. Sommer, C. von Beeren, and R. Wehner, "Multiroute memories in desert ants," *Proc. Natl. Acad. Sci.*, vol. 105, no. 1, pp. 317–322, 2008.

[130] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks." pp. 1097–1105, 2012.

[131] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation (No. ICS-8506).," *Calif. Univ San Diego La*

Jolla Inst Cogn. Sci., 1986.

[132] R. C. O'Reilly and Y. Munakata, *Computational explorations in cognitive neuroscience: Understanding the mind by simulating the brain*. Cambridge, MA: MIT Press, 2000.

[133] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[134] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column Deep Neural Networks for Image Classification," Feb. 2012.

[135] G. Hinton *et al.*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Process. Mag.*, vol. 29, Nov. 2012.

[136] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. Comput. Neurosci.*, vol. 9, p. 99, Aug. 2015.

[137] K. Greff, A. Rasmus, M. Berglund, T. Hao, H. Valpola, and J. Schmidhuber, "Tagger: Deep Unsupervised Perceptual Grouping." pp. 4484–4492, 2016.

[138] J. A. Starzyk and Basawaraj, "Memristor Crossbar Architecture for Synchronous Neural Networks," *Circuits Syst. I Regul. Pap. IEEE Trans.*, vol. 61, no. 8, pp. 2390–2401, 2014.

[139] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found.," *Nature*, vol. 453, no. 7191, pp. 80–3, May 2008.

[140] A. Horzyk and J. A. Starzyk, "Fast neural network adaptation with associative pulsing neurons," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–8.

[141] A. Horzyk, “Neurons Can Sort Data Efficiently,” Springer, Cham, 2017, pp. 64–74.

[142] D. C. Ciresan, U. Meier, L. M. Gambardella, and J. Schmidhuber, “Deep, big, simple neural nets for handwritten digit recognition,” *Neural Comput.*, vol. 22, no. 12, pp. 3207–3220, 2010.

[143] G. Tsoumakas, I. Katakis, and I. Vlahavas, “Mining Multi-label Data,” in *Data Mining and Knowledge Discovery Handbook*, Boston, MA: Springer US, 2009, pp. 667–685.

[144] G. Yu, C. Domeniconi, H. Rangwala, G. Zhang, and Z. Yu, “Transductive multi-label ensemble classification for protein function prediction,” in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1077–1085.

[145] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, “Cnn-rnn: A unified framework for multi-label image classification,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2285–2294.

[146] T. N. Rubin, A. Chambers, P. Smyth, and M. Steyvers, “Statistical topic models for multi-label document classification,” *Mach. Learn.*, vol. 88, no. 1–2, pp. 157–208, Jul. 2012.

[147] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown, “Learning multi-label scene classification,” *Pattern Recognit.*, vol. 37, no. 9, pp. 1757–1771, Sep. 2004.

[148] M.-L. Zhang and Z.-H. Zhou, “ML-KNN: A lazy learning approach to multi-label learning,” *Pattern Recognit.*, vol. 40, no. 7, pp. 2038–2048, Jul. 2007.

[149] N. Ghamrawi and A. McCallum, “Collective multi-label classification,” in *Proceedings of the 14th ACM international conference on Information and knowledge management - CIKM ’05*, 2005, p. 195.

[150] Min-Ling Zhang and Zhi-Hua Zhou, “Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization,” *IEEE Trans. Knowl. Data Eng.*, vol. 18, no. 10, pp. 1338–1351, Oct. 2006.

[151] J. Read, B. Pfahringer, G. Holmes, and E. Frank, “Classifier chains for multi-label classification,” *Mach. Learn.*, vol. 85, no. 3, pp. 333–359, Dec. 2011.

[152] W. Cheng and E. Hüllermeier, “Combining instance-based learning and logistic regression for multilabel classification,” *Mach. Learn.*, vol. 76, no. 2–3, pp. 211–225, Sep. 2009.

[153] M.-L. Zhang and L. Wu, “Lift: Multi-Label Learning with Label-Specific Features,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 1, pp. 107–120, Jan. 2015.

[154] M.-L. Zhang, “Min-Ling Zhang’s Homepage.” [Online]. Available: http://palm.seu.edu.cn/zhangml/. [Accessed: 15-Mar-2019].

Thesis and Dissertation Services