An investigation of the cortical learning algorithm

Anthony C. Samaritano
Rowan University

AN INVESTIGATION OF THE CORTICAL LEARNING ALGORITHM

by

Anthony C. Samaritano

A Thesis

Submitted to the
Department of Electrical and Computer Engineering
College of Engineering
In partial fulfillment of the requirement
For the degree of
Master of Science in Electrical and Computer Engineering
at
Rowan University
November 2, 2016

Thesis Advisor: Robi Polikar, Ph.D.

Acknowledgments

I would like to express my sincerest gratitude and appreciation to Dr. Robi Polikar for his help, instruction, and patience throughout the development of this thesis and my research into cortical learning algorithms. Dr. Polikar took a risk by allowing me to follow my passion for neurobiologically inspired algorithms and explore this emerging category of cortical learning algorithms in the machine learning field. The skills and knowledge I acquired throughout my research have definitively molded me into a more diligent and thorough engineer. I have, and will continue to, take these characteristics and skills I have gained into my current and future professional endeavors.

I would like to thank my mother, Dawn Samaritano, and fiancé, Lauren Anderson, for the continued and unwavering love and support through this journey. Without your support, I would not have been about to finish this research. Thank you for propelling me forward at every turn and the countless hours of reading and feedback throughout the writing of the thesis.

Abstract

Anthony C. Samaritano
AN INVESTIGATION OF THE CORTICAL LEARNING ALGORITHM
2017-2018
Robi Polikar, Ph.D.
Master of Science in Electrical and Computer Engineering

Pattern recognition and machine learning fields have revolutionized countless industries and applications from biometric security to modern industrial assembly lines. The fields continue to accelerate as faster, more efficient processing hardware becomes commercially available. Despite the accelerated growth of the pattern recognition and machine learning fields, computers still are unable to learn, reason, and perform rudimentary tasks that humans and animals find routine. Animals are able to move fluidly, understand their environment, and maximize their chances of survival through adaptation – animals demonstrate intelligence. A primary argument in this thesis that we have not yet achieved a level of intelligence similar to humans and animals in the pattern recognition and machine learning fields, not due to a lack of computational power but, rather, due to lack of understanding of how the cortical structures of mammalian brain interact and operate.

This thesis describes a cortical learning algorithm (CLA) that models how the cortical structures in the mammalian neocortex operate. Furthermore, a high level understanding of how the cortical structures in the mammalian brain interact, store semantic patterns, and auto-recall these patterns for future predictions are discussed. Finally, we demonstrate that the algorithm can build and maintain a model of its environment and provide feedback for actions and/or classification in a similar fashion to our understanding of cortical operation.

Table of Contents

Table of Contents (continued)

Table of Contents (continued)

Table of Contents (continued)

List of Figures

List of Tables

Chapter 1

Introduction

The human brain is one of the most widely studied areas in biology. Many researchers have devoted their entire careers to studying, demystifying and understanding how the brain operates. Only recently have researchers been able to noninvasively peer into the brain to understand the flow of information throughout the cortical structures of the brain and discover how the different regions of the brain interact. Furthermore, the brain imaging technology continues to develop at a rapid pace, which in turn has provided neurobiology researchers with unprecedented opportunities to peer into the brain *in vivo*.

The accelerating rate at which neurobiology researchers are gathering and publishing their findings on the cortical structures of the brain opens a very unique multidisciplinary opportunity for pattern recognition and machine learning researchers. This opportunity may mark the first time in history that we can start applying our high level understanding of cortical function to real world algorithms. Given our current understanding of neurobiology, it is possible to hypothesize that intelligence, in the context of neurobiology, is a system encoded in the cortical structure of highly evolved animal species. If this hypothesis is true, it is possible, using high fidelity brain imaging technology and other *in vivo* methods, to characterize intelligence by understanding the flow of information throughout the brain and the cortical structures involved in processing sensory information. Traditionally, neurobiology and machine learning have been treated as two separate fields of study, yet the boundary between the two is increasingly becoming fuzzier. One of the main goals of this thesis is, therefore, to illustrate the unique properties that the brain demonstrates, and relate those properties to machine learning and pattern recognition.

Furthermore, using the knowledge gathered from neurobiology research and correlated to machine learning and pattern recognition fields, this thesis illustrates how 'intelligent' machines of the future can use a new class of machine learning algorithms – that we call cortical learning algorithms (CLA) – to exhibit true intelligent behavior. In the following sections, we argue why it is important to study the brain for machine learning research. This chapter then briefly introduces Jeff Hawkins' work as the primary motivating factor for this work. This chapter concludes with the primary contributions and broader impact of this thesis.

Why Study the Brain?

The machine learning and pattern recognition fields have produced many algorithms capable of clustering and classifying data, predicting future patterns, etc. Typically, these machine learning algorithms have tunable parameters that help them minimize an error criterion (i.e. fitness function). Although these tuned algorithms perform well at predicting and classifying datasets, they are still subject to a number of fundamental problems including the overfitting and sample complexity – the number of training samples required to train the classifier to a desirable level of performance.

The brain, conversely, approaches learning in a different manner than typical machine learning and pattern recognition algorithms today – the brain learns and adapts to the environment by using very few training examples and without completely rebuilding its models. Moreover, because of the neocortex's (the focus of this thesis) uniform structure [1], it is possible to theorize the neocortex utilizes the same fundamental cortical operations (or algorithm) across all learning modalities – e.g. somatosensory, olfaction, vision, hearing, etc. This uniform structure has led to the following hypothesis upon which this thesis is based: *the neocortex has a common cortical algorithm utilized for both memory* *and prediction*. If this hypothesis is true, this means that the neocortical functions used for vision, somatosensory perception, audition, language, etc. all follow a single, fundamental learning algorithm.

At first glance, the biological brain appears to challenge the No Free Lunch theorem [2], which states that for every problem an algorithm succeeds, there is another problem on which it will fail. To be clear, the purpose of this thesis is not to challenge the No Free Lunch theorem by designing a brain, but to explore how the brain is able to efficiently learn such a large set of problems or modalities by utilizing a common learning algorithm. In other words, what is the set of assumptions that the neocortical common learning algorithm makes in order to quickly learn so many modalities while being general enough to support such a large class of problems? Furthermore, if there is a common learning algorithm driven by the uniform structure of the neocortex, can we (machine learning and pattern recognition researchers) emulate its function for use in real world applications? Therefore, it is necessary to focus on brain function and brain structure for further understanding of biological intelligence. By studying the brain, machine learning and pattern recognition researchers can leverage neurobiology researcher's efforts to attempt to apply neurobiology research to a new class of biologically inspired algorithms - cortical learning algorithms (CLA).

The focus of this thesis is to describe cortical interactions within the neocortex. The neocortex of the mammalian brain appears to be essential to biological intelligence and will be explored in detail throughout this thesis. Furthermore, brain structures involved with processing emotion, releasing hormones, etc. are not be discussed in this thesis as they are, likely, not the source of logical intelligence and are not responsible motor control. To be clear, this thesis is not arguing that structures like the pituitary glands, striatum and other chemical and hormone producing regions are not essential for the healthy function and growth of the brain, but may not be essential to intelligence itself. This assumption may prove to be incorrect but, for the preliminary research conducted for this thesis, it greatly simplifies and bounds the problem by ignoring the hormone producing brain regions – instead, this thesis focusing on neocortical structure, and how action potentials and neurotransmitters are processed between the neocortex and thalamus.

To develop algorithms and methods that model the principles fundamental to intelligence in the brain, we must first explore neurobiology research to determine the field's current model of cortical architectures, and understand the interactions between various brain regions. This thesis's purpose is to present the research necessary to understand basic cortical function as defined by neurobiology research, identify the properties that cortical circuits inherently demonstrate and relate them to the machine learning and pattern recognition fields. Furthermore, using the aforementioned cortical structures, neurobiology research, and properties of cortical circuits, a cortical learning algorithm (CLA) is presented that demonstrates all the properties of an intelligent machine as defined in this thesis. Additionally, the CLA is applied to various synthetic and real datasets to measure its performance, identify the CLAs properties in practice, and draw conclusions for future work.

Related Work

This thesis relates heavily to Jeff Hawkins work on developing Hierarchical Temporal Memory, Sparsely Distributed Representations and Cortical Learning Algorithm framework proposed in *On Intelligence* [3], implemented at *Numenta, Inc.* [4]. Further work and Ph.D thesis research by Dileep George [5] improves upon the memory-prediction framework proposed in *On Intelligence* and develops mathematical models for the microcircuits of the brain. Furthermore, recent developments in deep learning and deep architectures [6] [7] [8] strongly resemble the hierarchical structure of multiple Cortical Learning Algorithms developed for this research. The related work is discussed in detail throughout Chapter Chapter 3 of the thesis.

Motivation

Machine learning and pattern recognition have proven extremely important and influential throughout the fields' more than half century history and have contributed immensely to successful careers, organizations, research institutions and industries – i.e. the fields have continued to accelerate and generate extensive value in our society. The goal of this thesis is to build upon the success and explore new territories of machine learning and pattern recognition by analyzing emerging cortical learning algorithms and techniques influenced by neurobiology research (the study of cells, organization, and functionality of the nervous system that process information and mediate behavior).

The motivation for exploring the brain for fundamental structures and functionality of intelligence is galvanized by the desire to further unite machine learning and neurobiology fields. Further motivation for this thesis is to develop alternate, biologically inspired cortical learning algorithms that do not utilized traditional mathematical models of the neural structures - the perceptron and multi-layer perceptron (MLP) models. As machine learning and pattern recognition researchers, it is prudent to revisit our understanding of neural networks and update our models using the latest in neurobiology and cortical structure research. Furthermore, deep belief network researchers have proven that, by adapting neural networks with known principles of cortical structures, it is possible to greatly improve the state of the art of perception problems (e.g. object recognition) through unsupervised, hierarchical methods [6] [9].

Contributions

Several researchers have built cortical learning algorithms and started to apply their algorithms to real world, commercial applications (e.g. Numenta, Inc. and Vicarious) – as such cortical learning algorithms themselves are not a novel idea and notable cortical learning algorithms have been developed [5] [10]. Although CLAs are not a novel idea, the methods, structure, and functionality used to implement the CLA for this thesis are novel. For example, an alternate CLA model is proposed based on the thalamus research of Sherman et al. [11] [12] as well as utilizing an arbitrarily large *n* by *m* hierarchical structure of CLAs for classification. Many of the cortical learning algorithms developed in the past have focused on perception problems (e.g. image recognition) and anomaly detection. Although image recognition and anomaly detection are both befitting applications for cortical learning algorithms and hierarchical temporal memory structures, the focus of this thesis is to develop methods and approaches to utilizing cortical learning algorithms for motor control applications. Therefore, the contribution of this thesis to the pattern recognition and machine learning fields are:

1. Implementation of an open .NET C# CLA library for use future applications.
2. Development of a fast, simple cortical learning algorithm based on the initial research of Jeff Hawkins et al. (described in [3] [4] [5]) without the use of spatial and temporal pooling, and without the use of name cells.
3. Quantify the effects of hierarchical structuring of CLAs on classification tasks.
4. Quantify the effects of CLA sparsity on classification tasks.

5. Novel approach to passing information up the CLA hierarchy derived from neurobiology research [11].
6. Theory of how cortical learning algorithms may be applied to motor control applications.

Broader Impacts

In writing and developing this thesis, I hope to impact the scientific community in the following ways:

- Create a well-structured thesis to understand the fundamentals of cortical learning algorithms, brain function, and machine learning necessary to build a cortical learning algorithm.
- Develop a fully functional, open source hierarchical temporal memory algorithm library for use in research applications and continued development of the cortical learning algorithm.
- Evaluate the cortical learning algorithm through classification performance utilizing hierarchical structuring and hierarchical classification
- Understand and document the variables (e.g. sparsity) of CLAs
- Identify future work necessary to continue cortical learning algorithms and hierarchical temporal memory research.
- Theorize how cortical learning algorithms and hierarchical temporal memory systems can be applied to motor control applications.

Thesis Structure

- Chapter 2 describes the cortical structures of the brain by starting at the lowest level, the neuron, and build upon by describing the brain's architecture to define the cortical structures of interest. Next, we identify which cortical structures are most likely responsible for biological intelligence and present their structures.

- Chapter 3 reviews applicable machine learning principles and high level descriptions of machine learning algorithms, including deep machine learning architectures in order to relate cortical learning algorithms to the current state-of-the-art.
- Chapter 4 develops a relationship between the properties of the brain and machine learning applications in hopes of illustrating the potential properties and applications of an intelligent machine.
- Chapters 5 describes the proposed cortical learning algorithm developed in this study, and its various derivatives including supervised, unsupervised, and alternate CLA models. Furthermore, pseudocode and graphical examples of the various algorithms are described. Finally, Chapter 5 outlines the experiments conducted in order to validate the CLA design and assess its performance.
- Chapter 6 describes the experimental setup, validation and verification exercises used to test the developed CLA, and illustrates the results of the various experiments used to quantify the CLA and hierarchical classification performance.
- Chapters 7 discusses the results of the cortical learning algorithm and hierarchical structuring experiments, draw conclusions, and discusses future work.

Chapter 2

Understanding the Brain

This chapter briefly describes the cortical structure of the neocortex and analyzes the cortical structures of the neocortex. This chapter begins with the basic unit of the nervous system (the neuron) and expands to cortical column and neocortical regions. Furthermore, the thalamus and hippocampus are briefly described due to their highly coupled interaction with the neocortex.

Cortical Structures

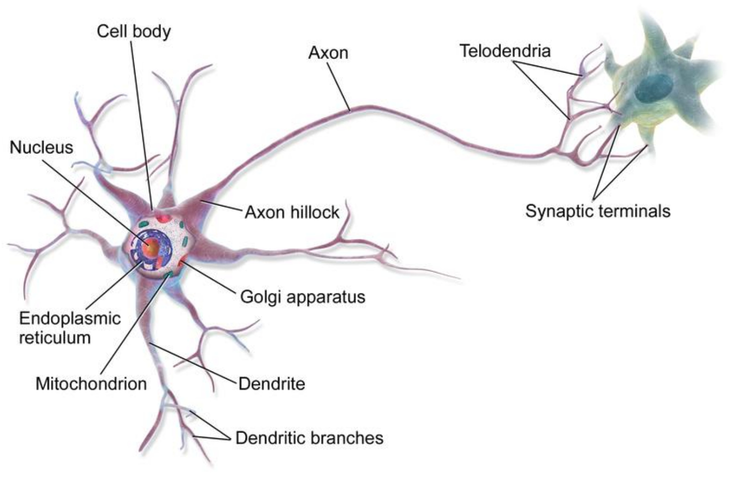


Figure 1. Unipolar Neuron *[13]*

Neuron. The fundamental unit of the brain is the neuron. The average human brain contains an estimated 100 billion neurons and 100 trillion synapses [14] that create an incredibly complex neural network more capable than any super computer on earth today at perception and complex multidimensional control problems. The brain contains many types of neurons but they all function in a similar manner with a nearly uniform function:

neurons are signal processors. The neuron is made up of three main parts: soma (or cell body), dendrites and axon as illustrated in Figure 1.

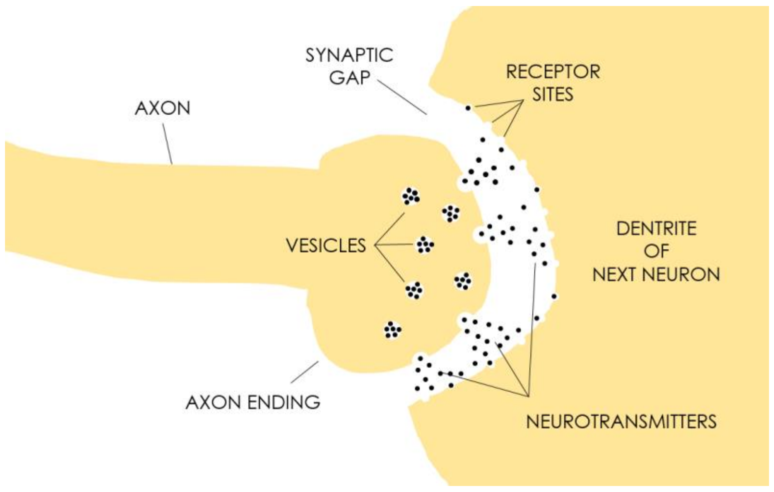


Figure 2. Neurotransmitters and Receptor Site

The soma, or cell body, is the prominent part of the neuron containing the nucleus and surrounding organelle. Aside from building proteins and basic cellular operations, the soma is the originator of action potentials that ultimately transmit neurotransmitters. The cell body of a neuron can also contain synapses for incoming electrochemical neurotransmitters of neighboring cells.

Dendrites are branch-like protrusions that originate at the cell body of a neuron. They act to conduct the electrochemical signals from other neurons and play a crucial role in regulating production of action potentials. Electrochemical neurotransmitters interface between axons and dendrites via synapses. Synapses contain neurotransmitter receptor sites that accept excitatory or inhibitory neurotransmitters. This explanation of excitatory or inhibitory neurotransmitters is over-simplified but, for the purposes of this thesis, is sufficient for first-order modeling of the neuron.

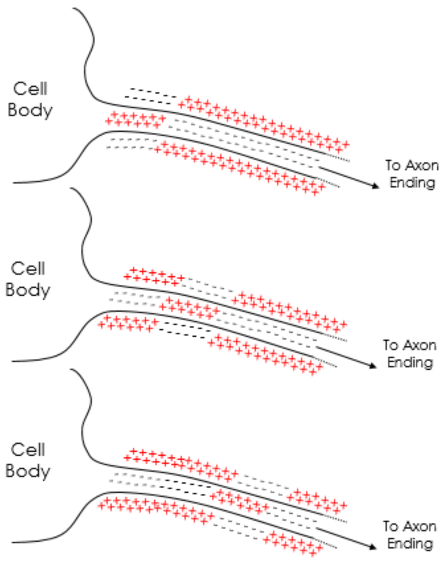


Figure 3. Ion channel carrying action potential over time

Axons are long tail-like structures that are used to carry action potentials to the axon ending signaling neurotransmitters from the soma to neighboring cells. Axon endings carry excitatory or inhibitory neurotransmitters to other neurons that can create or inhibit neuron excitation of the neighboring cell. It is worth noting that axons are the primary component of what is commonly referred to as “white matter” in the mammalian brain. The white in “white matter” is observed from the myelin sheath that coats an axon. The myelin sheath acts as an insulator allowing ions (i.e. action potentials) to travel significantly faster from the originating soma to neighboring cell’s synapses to release their intended neurotransmitters.

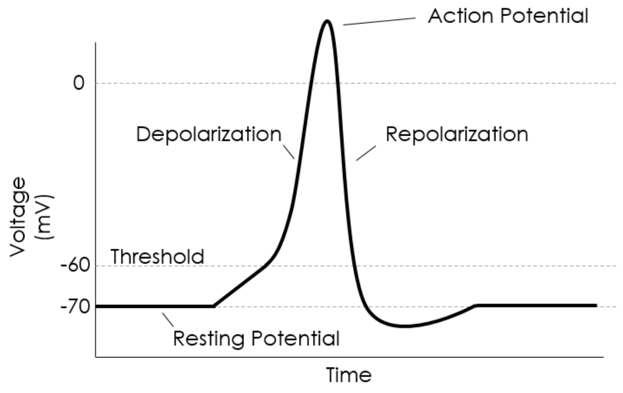


Figure 4. Example of a single action potential

As illustrated in Figure 3, the ion channels of an axon allow a positive electrical membrane potential to flow from the cell body, towards the axon ending. The positive electrical membrane potential produced is commonly referred to as an action potential. Action potentials, or spikes, are a short lasting electrical events in which the electrical membrane potential of a cell rapidly rises in voltage and then quickly falls back to a baseline voltage, or resting potential. Moreover, action potentials frequently occur as a series of temporally spaced pulses (commonly referred to as pulse trains). The pulse train spikes are the neurons main communication mechanism to neighboring cells and other biological tissue via the axon.

Figure 4 illustrates a generic, single action potential as voltage vs time. The voltage is measured across the neuron’s membrane and an action potential usually occurs on millisecond time scales. Figure 5 displays a common pulse train of action potentials. Pulse trains are commonly recorded during cell “bursting” or burst firing. A burst is a phenomenon normally observed as neuron activations in the brain and spinal cord [15].

Again, for the purposes of this thesis, a simplified model of bursting is assumed for intercellular communication.

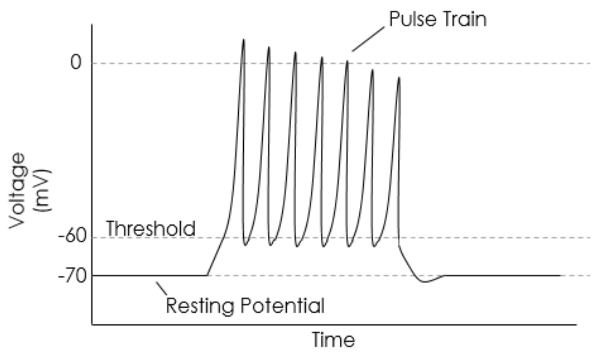


Figure 5. Example of a neuron pulse train

Neocortex. The neocortex (which literally means “new bark” in Latin) is the outermost region of the brain and is well-developed in most mammals. As previously stated, the main focus of this thesis is the neocortex. It is essential to focus on the neocortex because it is common among all intelligent species and is essential to understanding intelligence and intelligent behavior. The neocortex is the most obvious location to search for high level intelligence as the human neocortex is highly evolved and is responsible for our superior planning abilities, complex motor control, complex reasoning abilities, creativity, etc. Furthermore, using modern brain imaging technologies such as functional magnetic resonance imaging (fMRI), researchers have been able to map high-level cognitive functions such as speech, facial recognition and memory recollection to the neocortex. This thesis primarily focuses on mammalian cortex (primarily the neocortex) because a large majority of neurobiology research is grounded in the mammalian neocortex – i.e. there is a large volume of research to leverage.

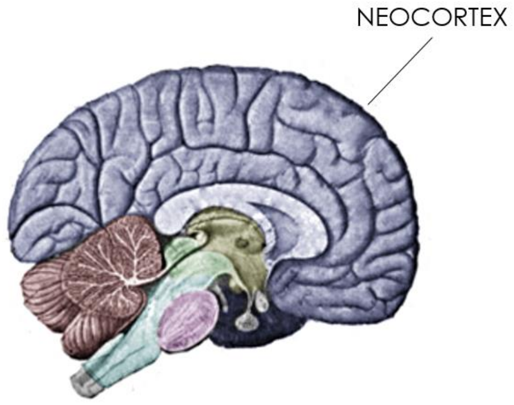


Figure 6. Neocortex Location *[16]*

The neocortex is a sheet of neurons arranged in a columnar structure known as cortical columns. The layers of cortical columns comprise a large sheet of neural tissue that is folded in on itself to fit within the human skull and envelopes the old brain. This giant sheet of cells alone is estimated to contain approximately 30 billion neurons [3]. The neocortex is able to store all memories, skills, knowledge, and life experiences. It can process modal signals such as feel, see, smell, hear and taste – furthermore, it can navigate and control biological machines in three dimensional space. In order to truly understand intelligence, we must study how information is processed and stored in the neocortex. The following sections define a high level understanding of the neocortical structure.

Neocortical structure. In 1978, Vernon Mountcastle published a widely cited paper [1] outlining the remarkably uniform appearance and structure of the neocortex.

Mountcastle observed that every region of the brain - whether it be the visual cortex, auditory cortex, prefrontal cortex, etc. - share the same basic structure [3] [1]. Except for minor discrepancies, such as neocortical thickness and cell density, Mountcastle argues that all regions of the neocortex must be performing the same basic operation (or algorithm) to learn. Mountcastle describes the neocortex as a “large number of modular elements linked together in echeloned parallel and serial arrangements” [1]. This arrangement is well documented in neuroscience and consists of a basic unit of cells – the “minicolumn” (or cortical column). Mountcastle argues that cognitive functions such as hearing and vision are not the same, yet their signals are processed in identical fashion. He argues that functions such as motor control operate in the same way as well. This hypothesis is highly supported by experimental evidence where small mammals’ optic nerves were rewired from the visual cortex to the auditory cortex at birth and the animals learn to see via the auditory cortex [17].

This uniformity across the entire neocortical structure cannot be understated and is, in fact, the basis of this thesis and cortical algorithm developed in this effort. Mountcastle’s insight suggests that the brain (neocortex in particular) is a complex system comprised of a network of cortical columns that act as the basic unit. It is likely that the cortical columns relationships and interconnectivity between the cortical layers and neighboring columns produce intelligence, or are a fundamental component of intelligence as we observe in humans and other species. As machine learning and pattern recognition researchers, we can utilize this knowledge of the neocortical structure to start developing and implementing algorithms that model this complex neural structure.

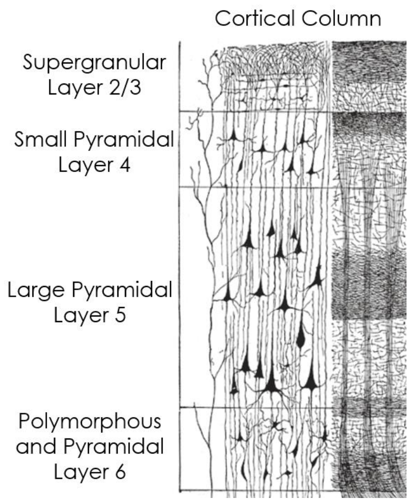


Figure 7. Cortical Column *[18]*

The cortical column consists of six distinct layers of neurons and is widely considered to be the basic unit of neocortical function [3] [1] [19]. Each cortical column is approximately 2 mm in height by 30 micrometers [19] in diameter. The six distinct layers of neurons are highly interconnected. It is important to note that layer I is widely cited as containing very few neurons but many axons – although many neurons from layer II/III have dendrites in layer I [3]. Because layer I contains mostly axons, it is normally ignored when describing the neocortical structure.

The six layers of the cortical column have distinct roles in processing signals. Layer IV is the input layer of the neocortical sheet (thalmocortical connections) [12] [19]. A thalmocortical connection is a neural connection that originates from the thalamus and terminates in the neocortex. Layers II and III, which are normally lumped together, receive signals from Layer IV. Furthermore, Layers V and VI are outputs of the neocortex. Layer VI sends signals from the neocortex to the thalamus (corticothalmic connections) whereas Layer V sends signals to subcortical structures, other regions (corticortical connections) and to the spinal cord [20] [12] [11]. Corticothalmic connections are a neural connection that originates from the neocortex and terminates in the thalamus whereas a corticortical connection originates from one column of the neocortex and terminates in a neighboring column of the neocortex.

Table 1

Firing Modes and Neural Pathways [12]

Firing Modes	***Tonic*****:** Neural mode induced by a modulator pathway to suppress neuron from reaching excited state in the case of driver afferent signal. ***Burst*****:** Excited state of neuron which is induced by driver afferent signal.
Neural Pathways	***Driver*****:** Message carrying pathway from neuron to neuron. ***Modulator*****:** Signaling pathway that sends excitatory or inhibitory behavior from neuron to neuron.

Neural pathways. There are two primary types of inter-neural pathways, drivers and modulators [12]. Drivers are the primary message carrying mechanism for thalamocortical and corticortical afferents, whereas modulators strictly alter the effectiveness of the driver signals/connections [12] [21]. For example, a driver afferent signal originates from the periphery, or sensory organs, arrive at the thalamus and is routed via thalamocortical connection to the neocortex [22]. Modulators, which can be found in connections originating from Layer VI in the neocortex to the thalamus, control the firing mode of the neurons to which they are connected.

Although there are many different neurotransmitters, subcategories of drivers and modulators and firing modes, it is sufficient to over generalize the various modes of signaling throughout the brain in order to reduce the developed cortical algorithm's complexity while investigating the brain's basic functionality. Table 2 summarizes firing modes and neural pathways identified by neurobiology researchers [12]:

Neocortical regions.

Layer IV. Layer IV of the cortical column is the primary, feed-forward input layer to the cortical column – i.e. the sensory/motor inference layer. It receives strong driver signals from thalamocortical connections. These driver signals may originate from the periphery organs (relayed via first order thalamus), from lower order regions of the thalamus or from neighboring cortical columns [11].

Layers II and III. Layers II and III receive signals from Layer IV of their cortical columns and closely interact with Layer II and III of neighboring cortical columns. Connections through Layer II and III are thought to be, primarily, modulating signals that place their neighboring columns' cells in tonic or burst modes [3] [11]. Furthermore, the layer II and layer III likely responsible for high-order inference memory – i.e. layer II and layer III is where memory and prediction intersect.

Layer V and Layer VI. Layer V and Layer VI act as the output of the neocortex. Layer V contains large pyramidal neurons which produce driver signals. These driver signals produced by layer V neurons terminate in the higher order regions (HO) of the thalamus, higher regions of the neocortex and/or routed to associated muscle groups via neural pathways though the brainstem.

Table 2

Neocortical Layer Description

Layer I	***Structure***: Mostly axons and sparsely arranged granular pyramidal neurons. ***Functionality***: Corticortical signal transport.
Layer II and III	***Structure***: Mostly axons and sparsely arranged granular pyramidal neurons. ***Functionality***: Modulatory connections to neighboring columns of Layer II and III cells.
Layer IV	***Structure***: Large star-shaped neurons (astrocytes). ***Functionality***: Cortical column input, feedforward output to cortical column.
Layer V	***Structure***: Mix of normal pyramidal neurons and very large pyramidal neurons. ***Functionality***: Motor output, feedforward output to thalamus and neocortical regions.
Layer VI	***Structure***: Normal pyramidal cells. ***Functionality***: Feedback modulator to thalamus.

Thalamus. The thalamus is an extremely important cortical structure that sits above the brainstem and is about the size and shape of a walnut.

The thalamus acts as the brain’s signal router; it is the interface between the body and cortex. It is worth noting that it is not accurate to describe the thalamus as simply a signal router - recent research suggests the thalamus plays key roles in other brain functions such as attention and synchronous communication between different brain regions, among others [21].

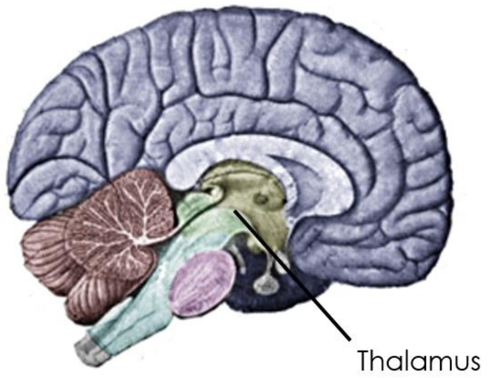


Figure 8. Thalamus Location *[16]*

The importance of the thalamus cannot be overstated. For example, a patient with a damaged thalamus is considered brain dead - no sensory information can reach the patient's cortex. In mammals, the thalamus possesses billions of corticothalamocortical (connections from the neocortex, to the high order thalamus, then back to the neocortex) nerve fibers that are directly connected to all neocortical regions [11]. This complex circuitry allows regions of the neocortex to send processed signals from, for example, the visual cortex to the parietal lobe.

The thalamus, along with corticocortical afferents, enables the mammalian brain to be wired in a hierarchical network [11]; this hierarchy of low-level to high-level brain regions is likely a key component to intelligence. Thalamic research has shown that the thalamus is broken into two major types of thalamic regions, first order (FO) and higher order (HO) regions [12] [22]. The FO regions of the thalamus receive signals from, for example, sensory organs and relay the sensory signals to the respective primary sensor regions of the neocortex via thalamocortical connections. The primary sensory region then sends its output signals back to the HO regions of the thalamus via corticothalamic connections. The HO regions of the thalamus then routes the signals to higher order regions of the neocortex via thalamocortical connections [11].

The corticothalamocortical connections of the brain are modeled via driver signals from one cell region to another in the cortical learning algorithm developed for this thesis. The driver signals are used for both classification and motor control signals. The simplified neural model is illustrated in Figure 9 where circles represent the granular neurons found in Layer II/III of the cortex, the stars represent the astrocytes neurons found in Layer IV and the triangles represent the pyramidal neurons found in Layers V/VI. Furthermore, in Figure 9, illustrates the simplified neural model utilized for the algorithm in this thesis where signals flow up a cortical hierarchy from periphery modalities to higher regions in the brain via corticothalamocortical connections [11].

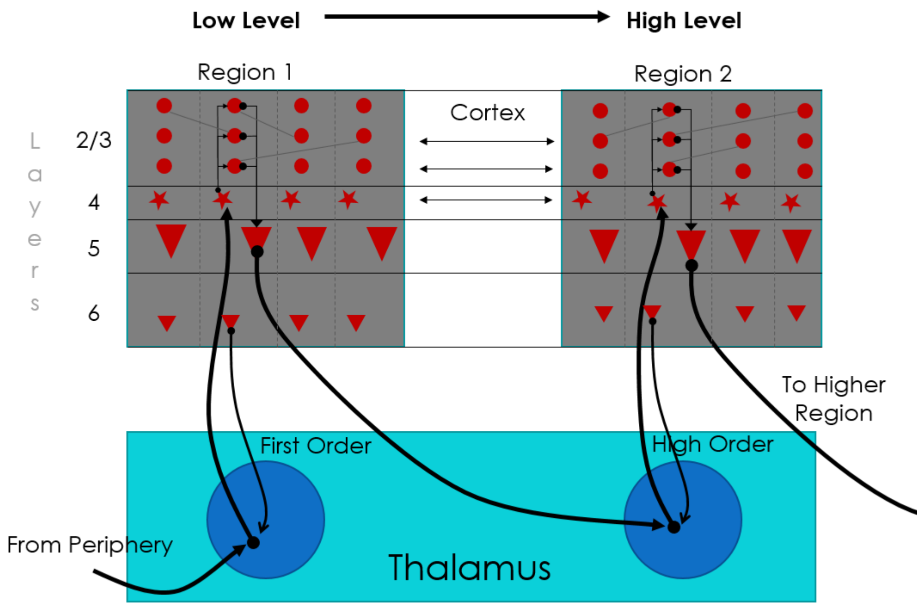


Figure 9. First order and high order thalamic connections

Hippocampus. The hippocampus is one of three large brain structures that are positioned between the midbrain and under the neocortical sheet (illustrated in Figure 10). The hippocampus sits atop of the logical hierarchy of brain structures [3]. All new and novel signals from the lower regions are identified, classified and stored for later recollection in the hippocampus. The hippocampus is important in the formation of new memories, spatial memory, and navigation. The processing and storage mechanism of the hippocampus is necessary for efficient use of brain resources [3]. Moreover, without the hippocampus (bilateral hippocampal damage) it is impossible to form new memories – this phenomenon is referred to as anterograde amnesia.

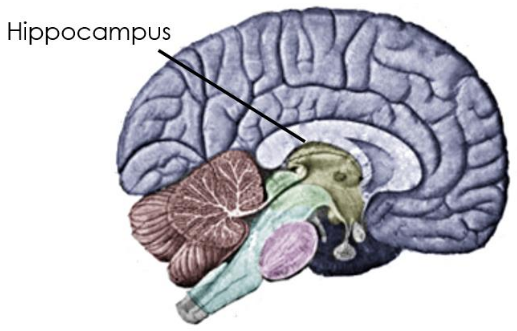


Figure 10. Hippocampus Location *[16]*

The hippocampus's unique neurological position in the cortex makes it responsible for the ultimate realization of signals processed in lower brain regions. As signals propagate up the neural hierarchy, anything new and/or novel projects onto the hippocampus where the new memories can be quickly processed and stored for later classification in the neocortex.

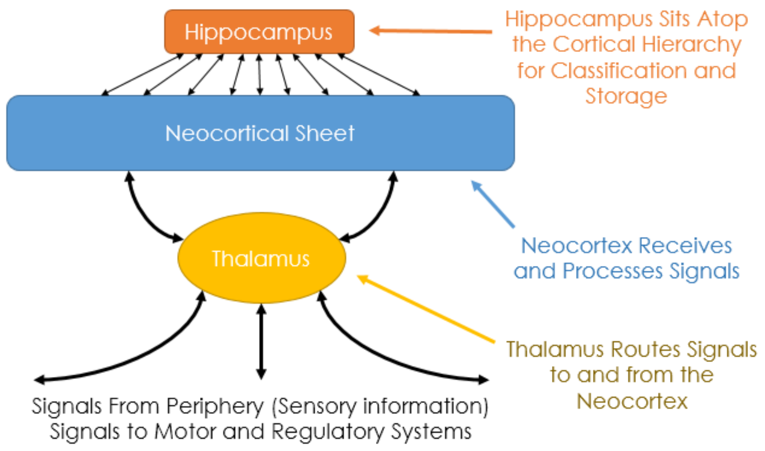

Hippocampus
Hippocampus Sits Atop the Cortical Hierarchy for Classification and Storage
Neocortical Sheet
Neocortex Receives and Processes Signals
Thalamus
Thalamus Routes Signals to and from the Neocortex
Signals From Periphery (Sensory information)
Signals to Motor and Regulatory Systems

Figure 11. Cortical Region Interaction

Ideally, every time a novel signal arrives at the hippocampus, the hippocampus retains the novel signal in the appropriate lower neocortex structure. The next time this previously novel signal starts to propagate up the brain's hierarchy of regions, the pattern (which was previously stored by the hippocampus in the neocortex) will be identified and classified by the neocortex. This classification by the neocortex can then be processed as a single, invariant representation rather than being processed further by the hippocampus. Therefore, this frees the hippocampus to focus on other, more novel signals.

The hippocampus is where memory formation occurs [8]. Our ancestors were able to use the hippocampus as a mechanism for spatial recognition and decision making when confronted with new/novel experiences and locations. In modern day humans, we use the hippocampus to process new/novel events or ideas - such as meeting a new colleague, reading a news story or learning a new skill. Each of these novel experiences are auto-correlated to current or past experiences at the highest level. When a novel event occurs, such as meeting a new work colleague, that first meeting is auto-correlated in your hippocampus to the current location, what you were doing prior to meeting your new colleague, the subject of conversation you and your new colleague talked about, etc. It is then the hippocampus's job to take that newly formed memory and store it, including all auto-correlated events associated with the newly formed memory, into the neocortex (our long term memory).

It is important to note that the neural function that allows the hippocampus to "write" new memories to the neocortex is not fully understood by researchers, although, the neural function responsible for storing new and novel memories likely involves back propagation of signals from the top of the hierarchy to the lower levels. This thesis focuses on the high level functionality of the hippocampus – the ability to auto-correlate novel ideas, places or events and efficiently store them for classification automatically in the future.

Chapter 3

Machine Learning

The purpose of this chapter is to familiarize the reader with applicable machine learning theories and techniques as they relate to the proposed cortical learning algorithm (CLA). Three high level classes of machine learning algorithms (supervised, unsupervised and reinforcement learning) are discussed. Moreover, both shallow and deep machine learning architectures are explored for context. A basic background in machine learning is assumed.

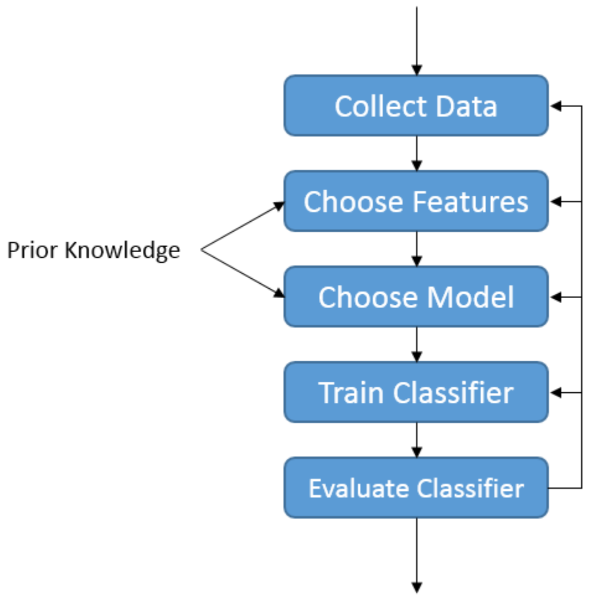


Figure 12. Common classier design *[23]*

Supervised Learning

A supervised learning algorithm uses labeled training data to model a system for classification of new, unknown examples of unlabeled data in the same feature space.

Supervised learning is often associated with "classification," and typically involves adaptively changing the parameters of a model, or classifier, until the model of the output fits the training data [24]. Figure 12, illustrates a common approach when designing a classier for real-world applications.

Training data contains labels (classes) and features. Labels are normally a scalar value that uniquely identifies the entity within a group.

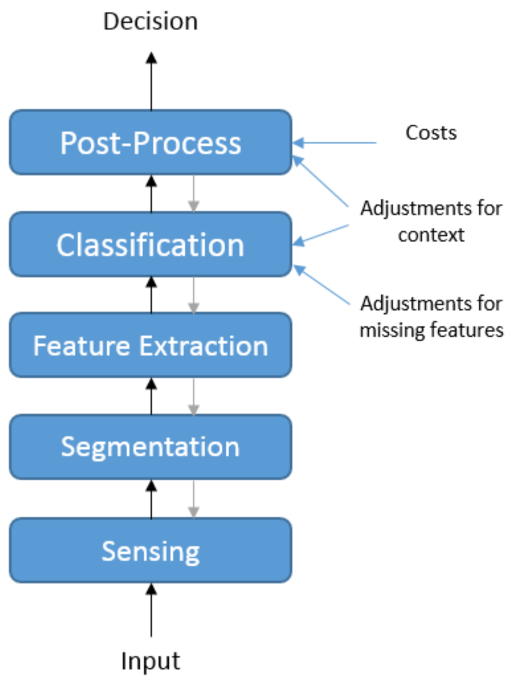


Figure 13. Common classifier system architecture *[23]*

An applicable example of a supervised learning algorithm is the multilayer perceptron (MLP). The MLP uses a cost function to minimize the error between labeled classes and labeled data. MLPs use back propagation to relate a known class (label) to various numerical examples, or features, of that class. By using a cost function (such as mean-squared error between actual and desired response), the back propagation algorithm minimizes the average error between a known class and the features that represent that class. Once a supervised MLP has been trained, it is then possible for the algorithm to accept unlabeled data and infer to the corresponding class. A common architecture used in trained classifier operation can be seen in Figure 13.

Reinforcement Learning

Reinforcement learning algorithms are used to solve the problem of sequential decision making by a system receiving delayed rewards as a function of performance towards a known goal or cost function [24] – i.e. reinforcement learning algorithms give their systems explicit goals, can sense their surrounding environments, and can choose actions to influence their environments to yield higher rewards. The reinforcement system is not told which actions to take but instead must discover which actions yield the most reward by heuristically searching for an optimal solution [25]. As with most systems, there is a challenging tradeoff with reinforcement learning algorithms - between exploitation and exploration. For a system to receive a reward, it must perform actions that have proven to be effective in the past. If the system is only using techniques it has identified as lucrative in the past, then it is simply exploiting the reinforcement learning algorithm to receive a reward, but may miss a better solution by not exploring alternate solutions. Whereas if the system is only searching for new ways to solve a problem, it is only exploring its solution space, and may suffer too much cost at little or no benefit. A healthy balance between the previously described exploitation and exploration must be tuned per application of the reinforcement learning algorithm.

It is important to note that reinforcement learning is different than supervised learning because supervised learning takes advantage of examples provided by a knowledgeable external supervisor (i.e. a labeled dataset). However, supervised learning is not capable of learning from interactions within a new environment as it is usually impractical to obtain datasets of desired system behavior in all situations where a system must perform. Thus, reinforcement learning algorithms perform best when a system is navigating uncharted territories where a supervised learning algorithm would likely fail.

Unsupervised Learning

Unsupervised learning is a subset of machine learning and pattern recognition commonly used in applications such as data mining. Unlike supervised learning, all data passed into an unsupervised learning algorithm is unlabeled and unlike reinforcement learning, there is no error or reward signal to assess the solution. Unsupervised learning, in most applications, is simply trying to find structure in unlabeled data. Unsupervised learning has huge implications in fields such as "Big Data" where huge amounts of unlabeled data are available, yet manually classifying that data is logistically or economically unfeasible. Many techniques exist to achieve unsupervised learning such as clustering algorithms, sparse coding or hidden Markov models.

There are many unsupervised learning algorithms in the context of neural networks, one of which is of importance in recent years - deep learning. Deep learning is a subset of machine learning algorithms that model high level abstractions of input data using a hierarchy of lower level abstraction layers until a class can be identified at a higher layer. Deep learning will be covered in detail in subsequent sections due to its applicability to the CLA and hierarchical temporal memory structure.

Applicable Machine Learning Algorithms

Much of the past quarter century in machine learning has provided us with non-parametric learning algorithms that can approximate any input-output mapping (e.g. MLP, decision trees, support vector machines, etc.). Given sufficient training data and computational resources, the non-parametric learning algorithms have proven invaluable in machine learning research and industry applications. However, new architectures (e.g. deep belief networks), which take advantage of hierarchical structures, have become increasingly popular in solving perception and highly nonlinear controls problems (e.g. driverless cars, natural language, etc.) [7] [26] [9]. This pivot to perception and nonlinear control has proven that our current “off-the-shelf” algorithms do not perform well on perception and highly nonlinear problems – problems that humans find trivial. Additionally, deep belief networks share the same foundation and properties as the CLA proposed in this thesis. For the purposes of this thesis, deep learning (or deep belief) networks are highlighted due to their applicability to the CLA. Before defining deep architectures, it is necessary to touch upon poplar shallow architectures as a means for comparison.

Shallow architectures, typically, use one layer of abstraction in order to execute their objective function. Many of the popular non-parametric, shallow architectures (i.e. – Support Vector Machines, Multilayer Perceptron) struggle when confronted with complex perceptual tasks such as computer vision, especially when there is complicated intra-class variability [7]. This single layer’s job is to match input patterns to output patterns (training data), normally accomplished through back propagation techniques. In contrast, deep belief networks are implemented using stacked, hierarchical architectures – analogous to the neocortical structure described in the previous chapter. Moreover, akin to the proposed cortical learning algorithm, deep belief networks treat machine learning tasks as different aspects of a generalized problem – deep belief networks seek to build a model of the network's observable world in an unsupervised fashion. In the following sections we examine both shallow and deep architectures.

Shallow architectures. Shallow architectures in machine learning have been the basis for most research throughout the research field's history. In this section, various shallow architectures are explored that are commonly used in machine learning applications.

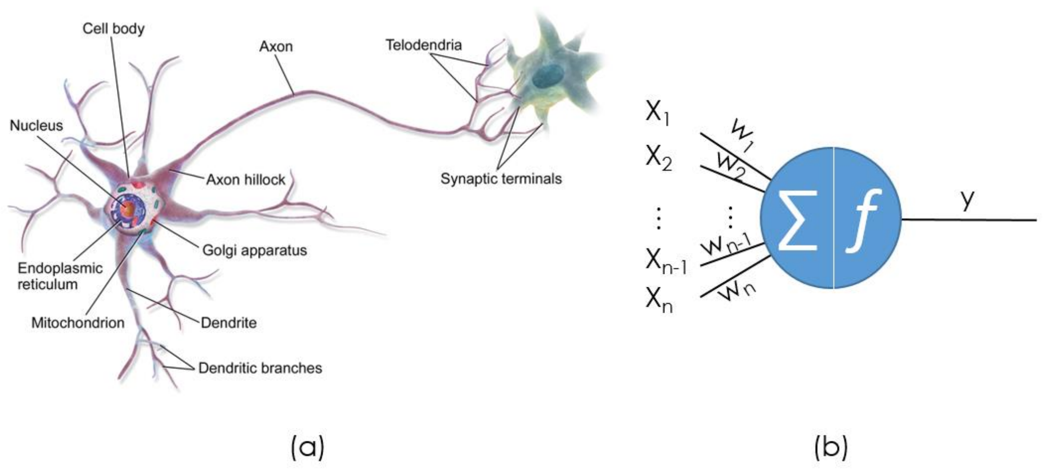


Figure 14. (a) Unipolar neuron (b) Perceptron model

Neural networks. Neural networks constitute a large field of machine learning and pattern recognition. Inspired by the central nervous system of animals, the traditional perceptron network uses weighted connection between nodes to classify a given input. The most notable neural networks are the multilayer perceptron network (MLP) and radial basis function (RBF). Both networks have been staples in the machine learning field for decades and have been exhaustively studied and tested on a wide verity of applications. The focus of this section is artificial neural networks (ANNs) and MLP because of its biologically inspired association to the cortical learning algorithm.

$$net = \sum_{i=0}^{n} \boldsymbol{x} \cdot w^T + w_0$$

Equation 1

$$y = f(net)$$

Equation 2

Where:
$\boldsymbol{x}$ is the feature vector
w is the perceptron weights
w_0 is the bias
net is the net sum
f is the activation function
y is the output

Artificial neural networks (ANNs) can be trained using supervised learning algorithms, unsupervised learning algorithms or reinforcement learning algorithms. ANNs were devised to mimic the decision making ability of the brain and brain function as understood in the late 1950's when the perceptron model was introduced. The perceptron, seen in Figure 14b, utilizes a weighted sum of an n-dimensional feature vector passed through an activation function to produce an output. The perceptron creates a linear decision boundary and is characterized by the following equation:

The activation function, *f*, of a perceptron model is typically one of three types of functions: thresholding function, identity function or sigmoid function. Each function is characterized by the following equations:

Thresholding function

$$f(net) = \begin{cases} 1, if\ net\ \geq 0 \\ 0, otherwise \end{cases}$$

or

Equation 3

$$f(net) = \begin{cases} 1, if\ net\ \geq 0 \\ -1, otherwise \end{cases}$$

Identity function

$$f(net) = net$$

Equation 4

Sigmoid function

$$f(net) = \frac{1}{1 + e^{-\beta \cdot net}}$$

Equation 5

Each activation function can be selected by the user for desired outputs for a given application. The choice of activation function determines whether or not the perceptron creates a polarizing output (-1 or 1), binary output (0 or 1), linear output (identity function) or nonlinear output (sigmoid).

The perceptron, by itself, is only useful in the case of linearly separable feature vectors. For example, a simple perceptron model is unable to solve the logical XOR problem [27]. Considering most practical applications of machine learning are non-linear, the perceptron is limited in its usefulness. Therefore, the multilayer perceptron (MLP) has proven valuable for nonlinear problems. The MLP is an artificial neural network that became very popular in the 1980's and has been exhaustively studied and tested throughout the subsequent decades. As proven in Rumelhart et. al [28], the MLP was able to overcome the XOR problem by utilizing a layer of hidden units as illustrated in Figure 12.

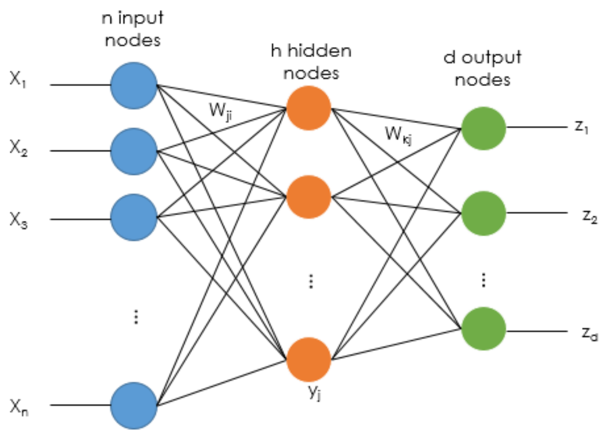


Figure 15. MLP architecture

As seen in Figure 15, the MLP architecture is a feedforward perceptron network comprised of an input layer, hidden layer(s) and an output layer. Thus, if the right connections from input to a large enough set of hidden units, it is possible to always find a representation that will perform mapping from input to output through the hidden units [28]. This architecture allows for an abundance of variations including activation functions, learning rate, number of hidden nodes, number of hidden layers, etc. This structure is trained by adjusting its weights as the squared error on labeled training data is minimized in relation to a cost function. This learning rule is called the back propagation. Furthermore, it has been shown that any decision boundary function of arbitrary dimensionality can be achieved with a single hidden layer [6].

Statistical approaches. Statistical algorithms in machine learning use a probabilistic approach for classification decisions. Generally, probabilistic approaches are characterized by having an explicit underlying probabilistic model which provides a probability of being in each class rather than simply a classification. There are dozens of examples of statistical learning algorithms and their practicality in real world applications cannot be understated. Although statistical algorithms are pervasive throughout the machine learning field, for the purposes of this thesis, this section provides only a very brief explanation of statistical algorithms due to our focus on biologically inspired algorithms.

The most obvious example of a statistical machine learning algorithm is the Bayes classifier. This approach can be best exemplified by the following statistical equations that calculates the posterior probability of a class given the likelihood, evidence and prior probability. The Bayes and Naïve Bayes classifier chooses the class with the highest posterior probability. Therefore, statistically speaking, the Bayes classifier is the best classifier one can build (described by Equation 6).

$$P(\omega_j|x) = \frac{p(x|\omega_j) \cdot P(\omega_j)}{p(x)} \qquad \textit{Equation 6}$$

Where:
$P(\omega_j|x)$ is the Posterior Probability
$p(x|\omega_j)$ is the Likelihood
$P(\omega_j)$ is the Prior Probability
$p(x)$ is the Evidence

There are many examples of statistical approaches in machine learning including (but not limited to) linear, quadratic, and logistic discriminant, K-nearest neighbor, causal networks, etc. Each of the aforementioned statistical approaches to classification are well documented [29] [24] [23] and continue to be applied in a wide range of research and applications.

Decision trees. Decision trees are members of a more general class known as graphical models. Decision trees can be described using graph theory and graph terminology [24]. Decision trees use a series of rules in order to build models of their training data – these rules allow the decision trees to essentially memorize the data. This makes them incredibly prone to over fitting and are generally considered weak classifiers. Once trained, decision trees are not computationally complex - they are a series of “if” statements. Their instability and low computational complexity make the classifier perfect as a base classifier for ensemble systems.

They are surprisingly similar to the CLA in that decision trees and the CLA are deterministic and memorize incoming data. The main difference between decision trees and the CLA is that the CLA “forgets” instances of training data by degrading Hebbian connections over time. Furthermore, the CLA can operate completely unsupervised whereas decision trees need labeled data to set up rules for the decision tree to follow and classify.

Deep architectures. Deep architectures have a distinct advantage over shallow architectures in that they can trade space for time [7]. In other words, the deep structure of stacked, hierarchical hidden layers in the deep architecture allows for the lower levels to represent low-level abstractions and upper level high-level features. This tradeoff between space and time has been coined the depth-breadth tradeoff [7].

Deep architectures are fundamental to the operation of deep belief networks. Most deep belief networks (commonly known as deep learning networks) are composed of a multi-layer neural network architecture which contains several layers of parameterized, non-linear hidden layers. Each layer, or module is subject to learning layer at a time in a hierarchical fashion. The following sections will capture the fundamental operating principles of deep belief networks and relate deep belief networks to the CLA proposed in this thesis.

Deep belief networks. Deep networks, neural networks consisting of many hidden layers, had been found to empirically perform no better (often worst) than neural networks with one or two hidden layers [30]. These finding hampered the progress of deep networks until recent methods introduced by Hinton et al. [26] for training deep layered networks based on greedy, layer-wise unsupervised learning showed promising results [7]. Hinton et al. demonstrated that it is possible to teach a deep, densely connected belief network one layer at a time by assuming the higher layers do not exist when the lower layers are training [26]. Deep belief networks (DBNs) pair feedforward layers with a feedback layer that aims to recreate the input of the layer from its output. This method guarantees that most of the information contained in the input is preserved at the output of the layer [7] and results in the deep belief network's architecture. After the initial unsupervised feedforward/feedback, layer by layer training has occurred, the network is then refined using a supervised back propagation method. This unsupervised feedforward/feedback pairing followed by supervised refinement results in the upper layers of the DBN representing abstract concepts that explain the input space while the lower layers extract low-level features from the feature space. In other words, the DBN's lower layers learn simple concepts first, then the upper layers build on the simple concepts to create more abstract concepts (as seen in Figure 16).

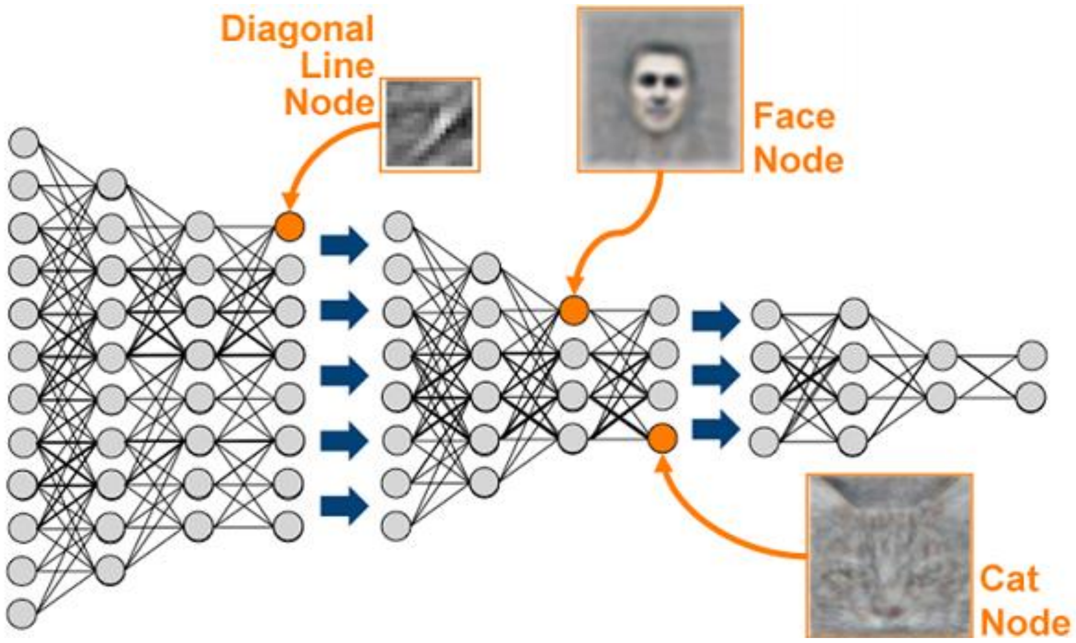


Figure 16. Deep architecture example [5]

A greedy layer-wise training algorithm proposed by Hinton et al. [26] is the basis of training deep belief networks and operates by training the network one layer at a time [6]. Initially, deep belief network takes empirical data as an input and models it resulting in an empirical distribution of the first layer of the deep belief network. The following equation denotes the empirical distribution of training the deep belief network:

$$p^1(g^1) = \sum_{g^0} p^0(g^0)Q(g^1|g^0) \qquad \textit{Equation 7}$$

Where:

p	is the empirical distribution
p^1	is the "empirical" distribution over the first layer
g^0	is the empirical data
g^1	is the first layer
$Q(g^1\|g^0)$	is the posterior probability associated with training the first layer

The idea of training a deep belief network using a greedy layer-wise training algorithm is that after the top-level layer, layer L, of a L-level deep belief network, one can change the interpretation of the network parameters to insert them in a (L+1)-level deep belief network – i.e. the distribution $p(g^{L-1}|g^L)$ from the neural network associated with layers L-1 and L is kept as part of the deep belief network generative model [6]. In other words, it is possible to train a deep belief network layer by layer resulting in increasingly abstracted representation of the input layer, g^0. Following the initial greedy layer-wise training algorithm to build the deep belief network's layers, it is necessary to fine-tune the structure by tuning all of the layers parameters together. This is accomplished using a supervised, back propagation method to fine tune the network and minimize the error of the trained deep belief network.

Deep belief networks allow for top down, multilayer neural network connections to generate sensory data rather than classifying the data [31]. Furthermore, deep belief networks form many hidden layers a hierarchy of abstraction layers where the lowest layers learn low-level features and higher layers learn generalized representations of the low-level abstractions – i.e. as low-level, sensory signals move up the hierarchy of hidden layers, the low-level features build high-level, generalized classifications (illustrated in Figure 16).

The resulting hierarchy of hidden layers in the deep belief network shares a similar structure and functionality to the hierarchy of cortical learning algorithms structures that are the focus of this thesis.

The architecture and training process using in DBNs are similar to the methods developed and implemented in this thesis using the CLA and hierarchical structuring methods. Bengio et al. concluded that unsupervised learning layer by layer greatly increases the supervised optimization over traditional supervised deep or shallow architectures [6]. It is likely that these same conclusions will hold true for hierarchical structured CLA cell regions as deep belief networks. It is worth noting that, although the architecture and training process of DBNs are similar to the CLA by utilizing unsupervised learning layer by layer, the mathematical methods used in DBNs are not similar to the algorithmic methods used in the CLA. Therefore, a strong understanding of DBNs is not necessary to understand the implementation of the CLA. For more information on the methods and processes used to implement DBNs, please reference Bengio et al. [6] and/or Hinton et al. [26].

Chapter 4

Relationship Between the Brain and Machine Learning

This chapter provides a link between properties of brain structure and function (e.g. hierarchy, sparse activation, etc.) and common machine learning techniques (e.g. anomaly detection, data fusion, etc.). The correlation between machine learning techniques and brain structure and function is important for illustrating the potential use of biologically inspired brain algorithms in machine learning and pattern recognition fields.

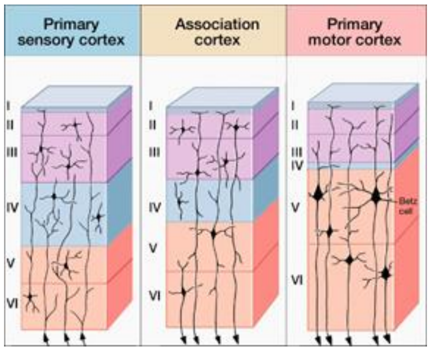


Figure 17. Illustration of the uniformity in neocortical column structure across the neocortex [32]

Properties of the Brain

The brain is a biological pattern recognition machine. The brain allows for complex interactions with the world and demonstrates a diverse set of cognitive functions such as sight, language and planning. Considering the broad spectrum of tasks and functions the brain can accomplish, one may intuitively believe that there must be a wide variety of specialized neurological structures to accomplish these various tasks. However, neurobiology research reveals this is, likely, not the case [17] [3]. In fact, a very uniform structure can be observed across the neocortex and other brain regions associated with cognition and learning [1].

As illustrated in Figure 17, various brain regions (that demonstrate completely different functions in the brain) all share the same fundamental 6 layer structure. The discovery of the uniform structure of the neocortex is widely attributed to V. B. Mountcastle et al. [3] [1]. Because of the brain's incredibly uniform structure, it is reasonable to hypothesize that there must be an underlying algorithm that composes the formation of intelligence which the brain utilizes for all of its complex cognitive functionality [3]. This section describes the assumptions and pattern recognition properties of the brain to characterize the methodology used to develop the cortical learning algorithm.

Online learning. The brain is an online learning machine - it tries to make sense of the world in real time in order to maximize its chances of survival. Online learning, in the context of machine learning, takes place when data becomes available in a sequential, temporal manner. Online learning algorithms update their models as each new piece of data becomes available. Like online learning algorithms, brain structures have evolved to characterize large volumes of high frequency sensory data streams in order to store and learn its environment in an online fashion. Simultaneously, the brain is constantly searching for previously learned patterns to make decisions in real time as well as store any new patterns of information. This process of storage and recollection allows the brain to recall the learned patterns for future reference and has been coined the *memory-prediction framework* by Jeff Hawkins [3].

Many traditional machine learning algorithms learn in an offline setting – offline learning is a useful technique in many engineering applications but limits the algorithms to a static model of the world until new data is available for retraining. Offline learning normally takes place using batches of data – i.e. stored datasets that are used to train the offline learning algorithm to model the dataset. Offline learning, generally, does not allow the algorithms' learned model of the world to be modified in real time – i.e., as new data becomes available. Conversely, online learning allows the algorithm to modify its model of the world which – when implemented by the brain - provides the brain with an evolutionary advantage by continuously learning from each new input. Online learning lends itself well to common machine learning problems such as learning in nonstationary environments or learning previously unseen new classes.

To create an intelligent system, it is necessary to build a system that learns in (near) real time. Post processing information (offline learning) can still be a useful tool to refine and reinforce a model built by the system, but this thesis hypothesizes that online learning is an essential property to any real time intelligent system.

Hierarchy. Hierarchy in the brain, as described in Chapter 2, is essential to the cognitive operation of the mammalian brain. Hierarchy in the brain is observed as low level sensory signals that move from the primary sensory regions of the brain towards the high level regions of the brain [5] [3] – e.g., from the primary sensory regions of the brain to the frontal lobe and hippocampus. As these low level sensory signals move to higher level regions of the brain, it is theorized that the signals become more and more abstract. An increasingly abstract signal in the brain can be described as an existing signal that is becoming increasingly representative of an idea rather than a signal that has physical meaning from peripheral sensory organs. In other words, an abstracted signal represents ideas in the high level regions of the brain whereas a low-level signal directly represents a modalities raw sensory stream. The abstraction of low-level signals creates a logical hierarchy of brain regions used for intelligent tasks such as complex motor control and perception. The hierarchical structure of the brain starts with sensory organs and ultimately ends at the hippocampus. The following paragraphs illustrate the flow of signals up the hierarchy of the nervous system.

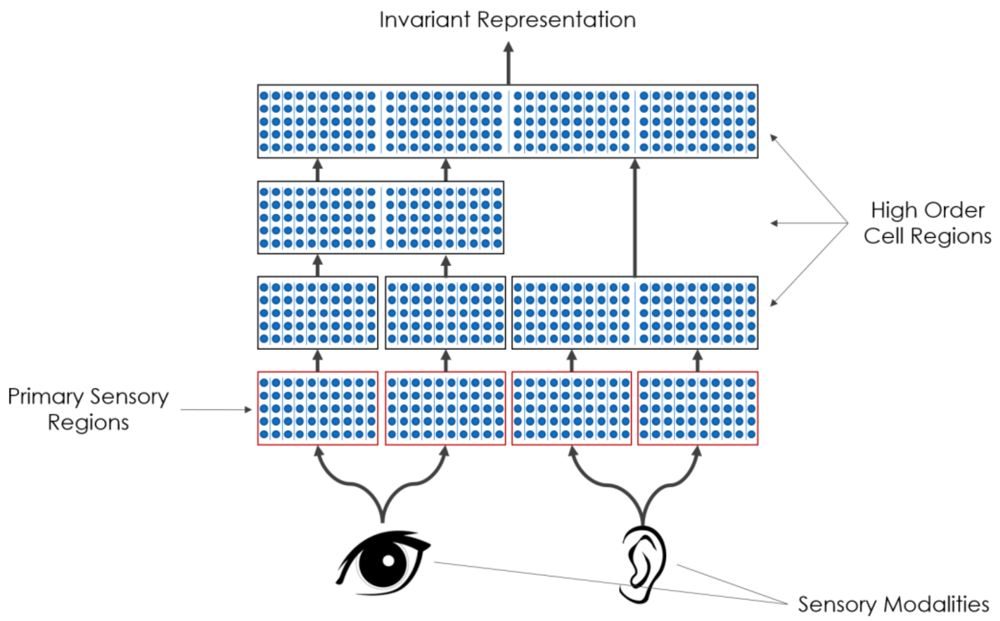


Figure 18. Example hierarchy of cell regions

The hierarchy of the brain, more specifically the nervous system, starts at the low-level, quickly varying sensory signals produced by the sensory organs (e.g. eyes, ears, skin, etc.). These sensory signals are carried via nervous system to the brainstem, then arriving at the first order (FO) thalamus. The FO thalamus routes the sensory signals to their corresponding primary sensory areas in the neocortex via white matter. The primary sensor areas of the brain are the cerebral areas that receive sensory information from thalamic nerve projects and are the foundation of the logical neocortical hierarchy. As illustrated in Figure 18, signals that arrive at the primary sensory regions of the brain and report their signals “up” the cortical hierarchy. It is important to note that “up” or “down” the neocortical hierarchy is not a physical location above or below the corresponding neocortical cell regions – i.e., if a signal is moving “up” the hierarchy it is passing processed, slower varying signals to the next region via corticocortical and corticothalamocortical connections.

The neural hierarchy operates in a fashion analogous to the hierarchical structures of most large corporations. Starting at the employee level where many employees may work for a single manager - that manager (and many more managers) then report to a mid-level manager. The layers of managers repeat until the top of the hierarchy, the CEO and board of directors.

The analogy between corporate organizational structure and the neural hierarchy is also true for communication between the hierarchies. Information can flow up the hierarchy, being reported one level at a time up to the top of the hierarchy (if necessary). Furthermore, signals can also feedback down the hierarchy from higher regions to lower regions or laterally within the same level of the hierarchy. Feedback signals from higher regions are analogous to a manager informing or giving directions to his or her employee, whereas lateral signals are analogous to two mid-level managers reporting status of their organization to each other.

The visual cortex is an excellent example of how the cortical hierarchy operates. The visual cortex consists of many cell regions including the primary visual cortex, V1, and the extrastriate areas V2, V3, V4, and V5 (to name a few). Raw sensory information from the optical nerve arrives at the thalamus and is routed to V1, the visual primary sensory area. In the primary sensory area of V1, quickly varying raw light signals from the retina are processed and reported up the hierarchy to V2 then V3 and so forth. As signals move from V1 to V2, V3 and so forth, the signals become slower, more generalized and more abstract (much like deep belief networks). At the higher level regions of the visual cortex, like V4 and V5, the visual cortex is processing colors and complex shapes like circles, stars, motion, human faces, cats, etc. This high level image information is used in high level functions of the brain such as reasoning, perception, and complex motor commands.

The neurological hierarchy of regions, in short, takes rapidly varying, low-level signals and converts them to slowly varying (or increasingly invariant), high-level ideas. Creating invariant signals at the higher level regions of the brain is necessary for the auto association of signals and formation of ideas by the regions of the brain. Furthermore, the auto association of these high level signals is used for perception of the outside world.

Time sequences. The brain operates in the time domain as sequences of patterns that are stored and recalled as a function of time. Time sequences are sequences of information stored temporally. An example of a time sequence is musical notes played sequentially over time to form a song. There are numerous examples how the brain temporally stores patterns – the following is a simple experiment that shows how difficult it is to divert from a learned pattern stored as a time sequence: recalling the alphabet from “z” to “a”. This task becomes very non-trivial because most people have not learned the alphabet in reverse order [3]. It takes a large amount of conscious effort to reverse the learned pattern of letters from childhood and recite them from “z” to “a”.

Without using time, tactile and auditory senses become impotent. For example, simple tactile contact with a surface will not allow the brain to infer what type of surface the skin is touching (i.e. type of material). Once the fingertip starts to slide across the said surface, the brain can infer what the surfaces material properties are and classify the surface as wood or tile, for example. This is a nonobvious example of time sequence memory: not until the fingertip begins to travel across the surface, as a function of time, can the brain infer that the very grainy, non-uniform structure is a wooden tabletop, for example.

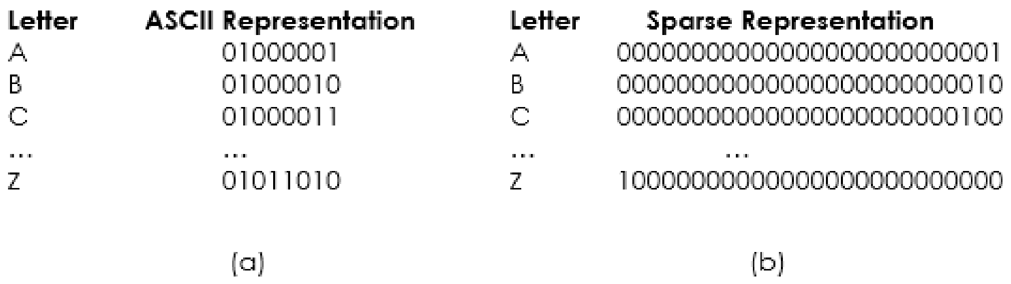
Furthermore, a more obvious example of time sequences is auditory signals over time. The brain perceives auditory signals as a time series of frequencies. These time series of frequencies may represent the chorus of a song, sound of a passing car or a sentence spoken by a close friend. All are examples of stored sequences of auditory patterns: songs are a time series of frequencies prepared by an artist, the sound of a passing car is a series of frequencies characterized by the Doppler effect, and spoken language is a series of syllables, learned during childhood, that represent words and ideas.

In machine learning, recognition of time sequences can be grouped into temporal pattern recognition algorithms that encode or decode patterns over time. Time is embodied in temporal pattern recognition algorithms using temporal order and time duration [33]. There have been many temporal pattern recognition algorithms such as short-term memory (STM) models, template matching using Hebbian Learning, Multilayer Perceptron (MLP) approach, and spiking neural networks, among others. Wang et al. [33] and Kasabov et al.

[34] describes a variety of STM models and neural network models for temporal pattern recognition that are inspired by the temporal pattern recognition properties of the brain.

There are a limitless number of example from all senses and high level thought that illustrate the importance of time sequences in the brain. It is reasonable to assume that time sequences in the brain play a crucial role in learning, inference and prediction, all of which are included in the understanding of a truly intelligent system. Therefore, any system that models intelligence must take into account the role time plays in storing patterns.

Sparsity. Sparsity in the brain is a storage and semantic meaning mechanism that is the basis for neocortical function. Sparsity can be defined as thinly distributed or scattered. In the context of the brain, sparsity is a measure of how many neurons are active at any given time. Sparsity is best understood when contrasted against traditional storage and meaning representations. For example, in a computer, information is represented as a stream of logical 1's and 0's called bits. Traditionally, computer architects and computer scientists intend on minimizing the amount of bits necessary to store information in order to conserve memory. This approach of information storage can be referred to as dense representation. Dense representation encodes a piece of information using an arbitrary assignment. A good example of this is ASCII code. ASCII code was developed to standardize the letter and number representation for the transmission of digital information. In ASCII the letter "t" has a binary code of "01110100." Each bit in the binary code has no specific value, "t" was merely assigned the 8-bit word as part of the ASCII convention. This dense representation contrasts sparse representation – as illustrated by example in the following figure, Figure 19:



Letter	ASCII Representation
A	01000001
B	01000010
C	01000011
...	...
Z	01011010

(a)

Letter	Sparse Representation
A	00000000000000000000000001
B	00000000000000000000000010
C	00000000000000000000000100
...	...
Z	10000000000000000000000000

(b)

Figure 19. (a) Dense representation (ASCII encoding) (b) Sparse representation

As seen in Figure 19b, sparse representation stores letters by semantically assigning each letter a specific bit to represent meaning rather that the arbitrary assignment of an encoded binary value as in the dense representation, Figure 19a.

Sparse representation trades memory efficiency for semantic meaning. In other words, sparse representation of information does not care to minimize the amount of bits needed to store information. Instead, sparse representation attempts to store information in a manner that guarantees semantic significance of the individual bits.

In the brain, neurons are activated sparsely – meaning that only a very few number of neurons are active simultaneously [35]. This sparse neural activity semantically correlates to the constant flow of time sequences projecting onto the neocortex. In other words, the brain sparsely actives neurons throughout to represent the world. The following is a grossly oversimplified example of sparsity in the brain: when the brain identifies a cat, the “cat” neurons [9] in the brain become active while the “dog”, “frog”, “rhino”, etc. neurons stay inactive. The sparsity allows the brain, at the highest level, to semantically represent the cat and identify it in the world.

In pattern recognition and machine learn, sparsity is a common tool used by researcher for computer vision as well as other pattern recognition applications. In computer vision, sparse signal representation has proven to be a powerful tool for acquiring, representing and compressing high-dimensional signals [36]. Using sparse representations, researchers have been able to achieve state of the art results by sparsely representing data as a subset of a learned dictionary of information. In other words, sparse representations are able to sparsely represent a high-dimensionality signal with low-dimensionality structures. The research conducted by Wright et al. [36] demonstrates the power of using sparsity for high-dimensionally problems in pattern recognition much like sparsity is utilized in the high dimensionality sensory input stream to the brain.

Attention. The brain, at any given time, is handling millions of sensory inputs from the periphery. Everything an individual sees, hears, feels, smells and tastes is nothing more than a stream of action potentials relayed via axons by the thalamus from the body's periphery. Furthermore, the brain is constantly sending output driver signals to skeletal muscles, organs, etc. to keep the body operating and maximize the organism's chance of survival. In order to make sense of the surrounding environment, the brain must prioritizes some sensory signals over other sensory signals. This prioritization of sensory signals can be understood as attention. Attention, in the context of this thesis, is the brain's ability to dynamically focus on specific sensory signals over other sensory signals.

The two feedforward pathways responsible for focusing attention are direct corticocortical and corticothalamocortical pathways, often following parallel paths (Figure 20). Specifically, a driver signal from lower cortical regions travels from cortical region to cortical region via direct driver connections and from cortical region to high order (HO) thalamus then to cortical region [11].

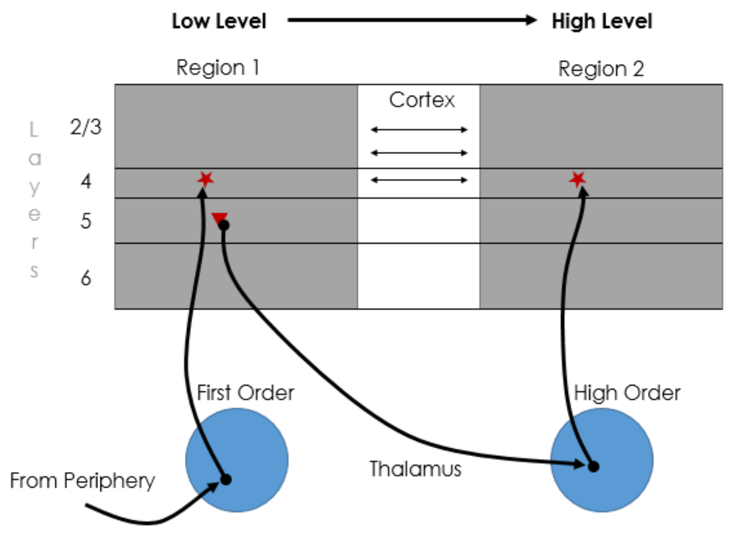


Figure 20. Corticocortical and corticothalamocortical connections from low level regions to high level regions

An example of attention and memory storage can be illustrated by two strangers meeting for the first time. At first meeting, when two strangers shake hands and exchange name they have no prior knowledge of each other. The brain, during the initial meeting, has now focused its attention specifically on this unknown person. It is no longer focusing its attention on the other conversations happening concurrently or what it had for breakfast, it is now focusing its attention on the conversation with the person it has just met. After the brief conversation, the brain has learned the other person’s name, and any other details picked up during meeting. The information about this newly met acquaintance, at the highest level in the brain (the hippocampus), has now been auto-associated to their face, subject of conversation, etc. and the memories are stored in the cortex. The next time the two people meet, the stored memories can be auto-recalled by the brain at lower levels freeing the higher regions and allowing them to focus attention elsewhere.

Another example of attention is learning a new motor skill. Motor commands, such as kicking a soccer ball or throwing a baseball, consist of complex set of instructions sent from various motor processing regions of the brain, to the midbrain, then to the spinal column, finally terminating in precise progression at the skeletal muscles. The signals delivered by the brain direct the skeletal muscles contract and relax to allow the athlete to kick a soccer ball or throw a baseball. Additionally, using the eyes and various somatosensory organs, the brain tracks the movement of the limbs completing a feedback loop between motor command and motor response. This complex iteration coordinates muscles and motor commands, while optimizing the desired output, and takes an enormous amount of attention to learn. This learning process (e.g. the best way to shoot a soccer ball at a goal) comes in the form of intense practice. For example, the first time an individual kicks a soccer ball he or she needs to focus intently on where to plant the stationary foot, how to swing the kicking foot forward, and where precisely to strike the ball. Each time that the feedback loop closed and the soccer ball is struck, the brain stores the complex sequence of commands to use later. After sufficient practice, the act of shooting a soccer ball becomes "second nature." – i.e. the individual has learned the proper sequence of motor commands in order to kick the ball and for the intended goal or action

It is reasonable to argue that attention is simply a neural pathway from lower regions of the cortical hierarchy to the top of the hierarchy (the hippocampus). By focusing attention, bringing lower level ideas to the hippocampus, the brain is able to make sense of complex details. Once the brain makes sense of these details, the hippocampus (responsible for memory storage) can store the newly formed ideas in the cortex. By storing the ideas in the cortex, the brain can automate the attention process in the future.

Modelling of attention in pattern recognition and machine learning is largely based off selective attention and prescriptive attention models that are inspired by biological processes. These attention models have been applied in applications such as handwritten digit recognition and face recognition by exploiting the fact that real images often contain vast areas of data that are insignificant from the perspective of pattern recognition [37]. This selective attention to detail is analogous to the corticothalamocortical pathways, described earlier in this section, the brain uses to select signals that contain useful information. For more information, the attention research demonstrated in Salah et al. [37] describes selective attention and prescriptive attention models and results used in pattern recognition of handwritten digit recognition and face recognition.

Invariant representation. The neocortex is a large, six layer sheet of nerve cells of about the same area as common dinner napkin when unfolded [19] [3]. The neocortical sheet can be divided into several regions. Each region is arranged in a hierarchical fashion where the lowest region is an input region and the subsequent regions are higher order regions. The regions pass feedforward information up the hierarchy via corticothalamocortical connection (as illustrated in Figure 9). These cortical brain structures allow the neocortex to create invariant representations. An invariant representation, in the context of the brain, is a signal (or group of neurons) that remains active to represent an experience (e.g. physical object, state, feeling, etc.) being observed by the brain.

A primary sensory area is a region of the cortical structure that directly receives sensory information from thalamic nerve projection (e.g. primary auditory cortex that is responsible for interpreting the signals produced by the cochlea). When sensor inputs are generated, they ultimately arrive at the primary sensory area as quickly changing, fluctuating signals. Quickly changing, fluctuating signals are difficult to classify in real world, real time applications. In the machine learning and pattern recognition fields, common techniques such as preprocessing, feature selection algorithms and feature extraction are used to reduce complex, highly parallel signals into their core components for classification. The mammalian brain is quite different - sensory organs have evolved to filter out unnecessary information but have also evolved to incorporate any environmental signal that maximizes the chances of survival. Thus, the brain is continuously flooded with millions of sensory signals from the body's periphery. How does the brain make sense of all of this information? In a few words, invariant representations.

We will use three contrived brain regions (R1, R2, and R3) located in the neocortex as an example. The signals in the lowest region, R1 (which represents the primary sensory region), are quickly varying. The primary sensory region, R1, begins to characterize common patterns from the high frequency periphery data as invariant signals. The lower brain regions then project the signals "up" to the next region in the hierarchy of brain regions. As we move up the hierarchy of brain regions, signals become increasingly invariant. For example, the primary sensory region, R1, may receive quickly changing frequency information from the inner ear (cochlea). R1's job is to characterize the frequencies sensed by the cochlea into an invariant representation that represent low level sounds that are passed that to the next highest region, R2. Region R2's job is to take the invariant representation from R1 and characterize the sounds (e.g. notes, voices, etc.). Furthermore, the invariant representations from region R2 is then passed up to the highest level region in this brain system, R3. Region R3 accepts the invariant representation of region R2 and characterizes the high level sounds for recognition. In other words, sensory signals that travel into the brain are characterized into increasingly invariant representations of the world. This is precisely why when an audience hears the first few notes of Beethoven's 5th Symphony, they are immediately able to characterize the remainder of the symphony as Symphony No. 5. Another real world example of invariant representation is when talking to a close friend. The brain automatically scans the close friend's face via saccades performed by the eye. Primary sensory regions of the visual cortex, V1, characterize low level details of the face. The low level signals from V1 propagate up the cortical hierarchy becoming increasingly invariant. In the higher regions of the visual cortex, invariant signals terminate in the fusiform gyrus (the face recognition region of the brain) where an invariant representation of the close friend remains.

Signal agnostics and plasticity. All sensory data streaming into the brain are relayed as a stream of action potentials, or neural spikes – i.e. the brain does not care if the action potentials represent auditory information, visual information, temperature information, etc. – the brain is simply making predictions and storing information about the patterns it has observed in the past. Because the brain does not discriminate against arriving action potentials, the brain demonstrates signal agnostics.

Signal agnostics has been demonstrated in neurobiology research [17] [38] [39] using cross-modal experiments where the inputs of one sensory modality are re-directed to a different modality, in effect "rewiring" the brain in mice. For example, when the visual signals originating from the retinas were "re-wired" to the auditory cortex of infant mice, the mice still learned to see through their auditory cortex. The research shows evidence that cortical areas have inherent propensity for the processing of subtypes of information (i.e. auditor cortex performs well at processing temporal signals and visual cortex performs well at processing spatial signals) due to millions of years of evolution. However, even though the primary sensory regions have an inherent propensity, the experiments show that the cortex is highly plastic and can adapt to its inputs (e.g. teaching the auditory cortex to see) because of the signal agnostic and plasticity properties of the cortex [38].

Another demonstration of learning through signal agnostics is leveraging existing senses to learn external signals – this is called sensory substitution [40]. D.M. Eagleman et al. have demonstrated many different experiments using sensory substitution to give participants super-human senses using simple vibrating motors on their skin. For example, the team was able to teach a deaf person to hear letters by simply activating a vibrating motor array (same motors found in cell phones) on their back. The audio patterns picked up from a microphone were encoded into motor vibrations and relayed over the vibrating motor array. Other experiments by Paul Bach-y-Rita et al. have proven similar sensory substitution over a tongue electrode array. Sensory substitution demonstrates the brain's plasticity and agnostics to incoming signals.

Signal agnostics in most pattern recognition and machine learning algorithms is common. For example, given normalized input signals, a MLP model does not discriminate between the normalized input signals, the MLP is simply trying to classify its inputs. In other words, an MLP will attempt to classify any dataset it has been trained on – it does not matter whether that dataset represents pictures of cats or audio of natural language. Alternatively, plasticity is less prevalent in machine learning and pattern recognition. The aforementioned MLP that was trained to identify pictures of cats will have a difficult time identifying pictures of sharks, for example, without disrupting the prior knowledge of cats.

Many traditional machine learning and pattern recognition algorithms are designed to learn a dataset but do not adapt to changing data distributions over time. This inability to adapt makes many traditional machine learning and pattern recognition algorithms develop static models of their world, unlike the brain. An example of adaptive, or plastic, algorithms in pattern recognition are adaptive resonance theory (ART) introduced by Carpenter et al. [41] describes an algorithm that identifies objects as a result of the interaction of top-down observed expectations with bottom-up sensory information and offers a solution to the plasticity/stability problem where new information can be learned without disturbing existing knowledge, much like the brain.

In order to create an intelligent machine, it is reasonable to hypothesize that the machine must encode sensory information as a generalized input (such as action potentials). Guaranteeing that signals entering the intelligent machine are signal agnostic allows for a generalized learning algorithm to understand the fundamental patterns in the signals and build a model of its environment (like the CLA algorithm).

Pattern Recognition Properties of an Intelligent System

In the context of machine learning and pattern recognition, all of the aforementioned neural properties of the brain lead to a suite of very useful machine and pattern recognition properties that the brain utilizes to demonstrate intelligence. This section highlights and describes many of the common machine learning and pattern recognition properties that are exhibited by the brain and attempts to demonstrate how and/or why the brain is capable of these properties.

Resistance to catastrophic forgetting. Catastrophic forgetting is a commonly observed problem of traditional machine learning algorithms. Catastrophic forgetting is the susceptibility of a classifier to abruptly (and often completely) forget information or patterns previously learned when learning new patterns or information. The brain, however, does not suffer from catastrophic forgetting (aside from traumatic events or diseases such as Alzheimer's). This makes evolutionary sense: if the brain were to catastrophically forget information about past locations or vital information just so it can learn new information, it would significantly reduce the organism’s chances of survival in the given environment. Therefore, the cortical learning algorithm must demonstrate resistance to catastrophic forgetting.

Data fusion. The brain is continuously streaming in large volumes of high frequency data points from the periphery via sensory organs. This high frequency information is continuously being utilized to build a model of its environment. Data fusion, in the context of machine learning, is the process of integrating information from multiple features from multiple sources of data that represent the same real world object or environment. A biological example of data fusion may be the integration of various sensory organs to describe a single object – such as an apple. The representation of an apple incorporates the way its waxy skin feels on the fingertips, the color of the skin observed with the eyes, the sweetness on taste buds and the very distinct sound heard when the apple is bitten. The brain is capable of incorporating the numerous sensory information streams about an object (e.g. an apple) into one singular representation. In other words, the brain is able to combine many independent sensory signals into a single representation.

A similar example of data fusion, in the context for control theory, is the sensor fusion between a gyroscope and accelerometer when tracking inertial measurements. These two independent sensors, when used together, provide more complete information about the way an object is moving in three dimensional space. Unfortunately, just using a single accelerometer cannot always extrapolate angular rate of an object moving in space and a single rate gyroscope cannot sense the influence of gravity or body accelerations on an object. When the sensory information from both sensors are combined, however, it is possible to precisely track the movement of an object throughout 3 dimensional space. This technique of data fusion can be used to control a car, plane, multi-rotor helicopter, etc.

Data fusion in pattern recognition and machine learning can be classified into the following categories: complementary, redundant, and cooperative data fusion [42]. Complementary data fusion is used when input sources represent different or overlapping parts of a scene. Redundant data fusion is used when data sources represent the same scene which can be fused to increment confidence. Cooperative data fusion is used when information provided is combine to form new information that is typically more complex than the original information. A review of data fusion techniques in machine learning can be found in Castanedo et al. [42] research published in The World Scientific Journal.

The brain, just like a data fusion algorithm, is able to combine different sensory streams into an abstracted understand of the world around it. All cortical learning algorithms must be able to combine any sensory input (i.e. signal agnostics) and create a high level understanding of the various signals to properly model the operating environment.

Noise resistance. Sensory organs and man-made sensors are not perfect. When operating in an uncontrolled environment, there are many sources of noise that may corrupt a system. The brain is capable of automatically isolating important sources of information and correcting for imperfections within its own biological system. The brain's ability to reduce noise allows two people to carry out a conversation in a very loud environment or pick specific flavors out of complex dishes. The ability of the brain to "filter out" what it understands to be noise in the signal demonstrates noise resistance within the cortical system.

Noise resistance, in machine learning and artificial intelligence, is a common problem that is usually handled via filtering (preprocessing) raw data before using them with a machine learning algorithm. Further examples of noise reduction in machine learning include: wavelet transforms, nonlinear filters, anisotropic diffusion, fuzzy logic, etc. For examples of how noise reduction algorithms are used, Hu et al. [43] illustrate a comparative intelligibility study of noise reduction on corrupted microphone data. The brain does not have the ability to preprocess data (like many machine learning algorithms) because it runs in an online fashion – i.e. the brain cannot store raw sensory information for processing later. Instead the brain has the ability to focus attention on specific pathways during learning in order to filter the noisy signal. Although we do not conclusively know the specific method for the noise resistance in the brain, a plausible hypothesis is that the hierarchical structure of the brain naturally denies noise in the raw sensory stream to propagate up the cortical hierarchy. In other words, low level brain regions actively dampen the sensory stream resulting in a progressively cleaner signal as the signals move higher up the hierarchy of brain regions. The inherent ability for the brain to reduce noise because of its hierarchical structure allows for the high level, low noise signals perceived by the brain.

Cortical learning algorithms must filter noise without preprocessing data streams in an online fashion. The cortical learning algorithm developed for this thesis assumes that the above hypothesis that the hierarchical structure of the brain inherently filters noisy signals is true. This noise resistance of the CLA is demonstrated later.

Anomaly detection. The brain is constantly making predictions about what it will experience next within its environment. If the brain does not make a correct prediction about its environment, it becomes active (or a bursting state) - i.e. when a signal violates the predicted model of the environment, the brain becomes surprised. The brain has specifically evolved to detect anomalies within its environment. The active state of the brain is analogous to anomaly detection. In machine learning and pattern recognition, anomaly detection is the observation of events or patterns that do not conform to the developed models – i.e. detection of outliers. Some examples of popular techniques of anomaly detection in pattern recognition include density based techniques (like k-nearest neighbor), use of support vector machines, cluster based outlier detection, fuzzy logic based outlier detection, etc. An overview of anomaly detection techniques in pattern recognition can be found in Patcha et al. [44] overview of anomaly detection algorithms.

In the context of evolutionary survival, the brain's natural tendency to seek out anomalies and focus its attention on the anomalies makes sense. Whenever the brain's model of the world is violated, it works to analyze the cause of the violation and assess if the violation is a threat. The next time that this specific violation or anomaly occurs, the brain will have learned how threatening the anomaly is to its survival and how to react. This very useful tool that has naturally evolved in the brain is commonly used in a variety of applications in machine learning and pattern recognition fields. For example, Numenta (Jeff Hawkins company) is using cortical learning algorithms and hierarchical temporal memory to detect anomalous behavior in web server operations. When a violation of the server's model is sensed, Numenta's software (Grok) alerts the server's operators so they can analyze and take action on potential server issues before any serious or catastrophic events take place.

Non-stationary environments. The environment in which a biological system resides is ever changing. There are always new social interactions, differences in weather patterns, new locations and even new predators that a biological system must sense, interpret and model to maximize its chances of survival. Therefore, the brain is well adapted to handle learning and modeling of an ever changing environment. In pattern recognition and machine learning fields, such environments are named non-stationary environments. In non-stationary environments (in the context off machine learning and pattern recognition) the underlying data distribution for a given class changes over time – i.e. the features that represent a class at t_0 do not represent the same class at a later time, t_n.

Traditional machine learning and pattern recognition algorithms do not handle non-stationary environments well because they have been developed assuming the class boundary remains constant over time. However, this assumption need not be true. Recently, more and more pattern recognition and machine learning problems have gone online (i.e., data arrives incrementally over time) and it is not guaranteed that the class boundary remains static. When a data stream experiences changes (or drifts) to the class's boundary over time, the non-stationary environment is said to experience concept drift. In order for a machine learning algorithm to operate in real world environments, it must be resistance to concept drift by reclassifying, or reassessing the class boundary over time [45].

Algorithms such as Learn^{++}.NSE use an ensemble of classifiers that learn over time in a non-stationary environment where the class boundary does not remain static. For more information on pattern recognition and machine learning algorithms relating to non-stationary environment, Ditzler et al. [46] have conducted a survey of algorithms that perform in non-stationary that include Hierarchical ICI, Learm^{++}.NSE, massive online analysis, among others.

Non-stationary environments can lead to catastrophic forgetting and other nontrivial engineering problems when designing machine learning and artificial intelligence algorithms. The brain handles operation in non-stationary environments by constantly making and testing predictions based on the current state and auto associating those predictions with previously learned models. Therefore, it is necessary for machine learning algorithms, which model the brain, to operate in non-stationary environments and assume that data used to model its environment will experience concept drift.

New class. Supervised machine learning algorithms use labeled training data from a known number of classes in order to later classify incoming, unlabeled data. If a new, unknown class is introduced to the system post training, the traditional machine learning algorithm will not adapt and wrongly classify the data. For example, traditional machine learning algorithms, such as a multilayer perceptron (MLP), can be used to identify dogs and cats. Once the algorithm is trained using training data that contains only pictures of cats and dogs, the hypothetical learning algorithm can classify photos of dogs and cats correct for 100% of all instances. However, if a photo of a bird was introduced to the trained learning algorithm – the algorithm would output “cat” or “dog” (depending on the features it has extracted from the photo), not a bird. Because the algorithm was trained using only two labels (or classes), it would never be able to classify anything but cats or dogs correctly. Although this example of adding a new class to a trained learning algorithm is very simple, it illustrates that an online learning algorithm that models the brain must be able to dynamically add new classes to the system without implicitly retraining the entire system.

Chapter 5

Cortical Learning Algorithm

The Cortical Learning Algorithm (CLA), outlined by Jeff Hawkins [3] and Numenta, Inc. [4], is a biologically inspired algorithm that takes advantage of our understanding of the neocortical structure and the neocortical operation. The focus of this thesis, like the CLA outlined by Numenta, is the neocortex – i.e., the memory and prediction structure of the brain. Although each brain system discussed in previous chapters have a distinct role in processing and interpreting continuously streaming signals from the sensory organs, the neocortex appears to be the structure of the brain with the largest influence on intelligence and intelligent behavior - therefore, the neocortex was chosen as a starting point for this research.

The CLA mimics the basic functionality of the cortical columns of the neocortex and emulates many of the properties of the brain that were outlined in Chapter Chapter 4. In order to sufficiently model the functionality of the neocortex, the following properties were engineered into the CLA developed for this thesis: online learning, time sequences, sparsity, cellular regions, hierarchy, prediction and invariant representation. To reiterate, the purpose of this thesis is not only to model the functionality of the neocortex and build upon the work of other CLA researchers, but to lay the framework for motor control via cortical learning algorithms – I believe that by modeling the aforementioned principles, new insights to cortically inspired motor control algorithms may emerge.

The CLA was designed with user experience and portability in mind for later use in relevant applications. A couple key features were set as requirements for the CLA: scalability, internet protocol (IP) communication, and easy visualization. Furthermore, the

CLA was designed to easily switch between unsupervised learning, supervised classification and control tasks. The purpose of these requirements were to: 1) validate that cortical learning algorithms can learn in a networked and highly coupled fashion in order to build hierarchical CLA structures 2) understand if supervised and unsupervised learning are possible via the same CLA algorithm (with minor modifications and 3) to understand if the CLA is capable of motor control.

This chapter discusses the design goals, design process, and the overall system architecture of the CLA algorithm. Furthermore, CLA pseudo code is provided as well as descriptions of the experiments designed to test the CLA algorithm.

Neocortical Principles

This section describes the various neocortical principles (e.g. online learning, time sequences, sparsity, cellular regions, hierarchy, prediction and invariant representation) and how the principles are modeled and implemented.

Online learning and time sequences. In order to model online learning in time sequences, the CLA operates in discrete time steps where patterns are observed by the CLA one step at a time. This functionality is implemented by passing in a series of numbers, one discrete time step at a time.

Cell regions and hierarchy. The neocortex is comprised of distinct regions where signal processing occurs. These regions of cells are responsible for everything from speech, vision, and motor commands to high level reasoning. The regions form logical hierarchies where low level feature extraction from the periphery feeds sensory signals up the hierarchy of brain regions to form high level abstractions. As signals move up the hierarchy they become increasingly invariant and are used for high level pattern recognition. The regions of cells are made up of a variable number of groups of cortical columns. These groups of cortical columns (referred to as cell regions in this thesis) act as a modular unit of parallel processing capability. It is likely, in the context of neurobiology, that larger cell regions correspond to more complex input signals (e.g. the visual cortex is much larger than the auditory cortex) - therefore, a cortical learning algorithm must be able to dynamically add cortical columns to a cell region as the complexity (i.e. - size of the input vector) increases.

Modeling these cell regions is a requirement for cortical learning algorithms, much like the one developed for this thesis. Furthermore, it is possible to create independent processing units by breaking the CLA into discrete regions. These independent processing units can work in parallel – e.g., on different threads, processing cores or even different networked computers. The independence of the discrete cell regions modeled in the CLA allow for distribution of the workload amongst many machines. Moreover, it is then possible to logically stack the discrete cell regions into a hierarchical structure. CLA hierarchical structuring is very analogous to the thousands of different cell regions in the biological brain all processing millions of sensory information simultaneously.

Sparsity. Sparsity is fundamental property of the neocortical structure and is a key requirement of the developed CLA. Sparsity, in the context of this thesis, is a measure of how many cortical columns are active at a given time within a region – i.e. each cell region must maintain a specified level of sparsity. For the CLA to perform nominally, it is necessary to maintain a specific level of sparsity during CLA operation. For example, if a level of sparsity of 1% is necessary to maintain proper CLA functionality for a 1 dimensional input vector, we must guarantee that, during every time step, a maximum of 1 column of cells is active and for every 99 other columns of cells that are not active. For a two dimensional input vector, we must guarantee that a maximum of two column of cells are active and a minimum of 198 other columns of cells are not active and so forth. Therefore, sparsity is maintained during CLA operation by limiting the number of columns of cells active within a cell region during each discrete time input.

Prediction and invariant representations. In the brain, as signals move up the hierarchy of regions, information changes less often. This is due the abstraction of low level features and results in the invariant representation of a pattern at the higher levels of the hierarchy. In other words, invariant representations are slowly changing signals (or representations) that stay active while a pattern is being identified.

The invariant representation of objects, sounds, tastes, etc. allows the neocortex to make predictions as information is flowing into the brain via sensory modalities. For example, when someone begins singing "twinkle, twinkle little ____," it is possible to fill in the blank. This prediction comes from the abstraction and invariant representation of a common childhood song learned over time. The brain abstracts the audio frequencies picked up from the ear (more specifically the cochlea) into high level, invariant representations of words. Those high level words, in that specific order yield a prediction of "star" following "twinkle, twinkle little." In other words, the brain is constantly making predictions (in real time) to identify patterns. Every time a pattern is correctly predicted, that pattern is reinforced in the brain. The reinforcement of patterns over time creates a belief, or prediction. This functionality and fundamental property of the brain is encoded into the CLA and hierarchy of cell regions by constantly assessing what the next pattern that the CLA is likely to observe – i.e. during each time step (or time sequence), the CLA makes a prediction about what it will observe during the next time step, prior to observation. If the CLA correctly predicts the next time step, it knows that its model is valid. Although, if the prediction is wrong, it knows that is has identified a new pattern.

Cortical Learning Algorithm Design

This section describes the design of the CLA using the neocortical principles of the brain as well as additional engineering requirements used to design the CLA. The additional requirements used to design the CLA are derived with usability and portability in mind.

Modularity. In order to build an algorithm for potential use in machine learning and pattern recognition applications, the CLA is structured for modularity. Modularity, in the context of the CLA, is defined as the ability for the CLA to dynamically add processing units as needed. This section discusses how the CLA achieves a modular solution and the CLA's system architecture.

The described modularity was attained through a high-level object oriented language, visual C#. Visual C# allows the algorithm to grow in an online learning environment by dynamically calling processing units in software. Furthermore, the hierarchical nature of the neocortical structure was taken advantage of during the implementation of the system. The neocortex operates in a hierarchical fashion, therefore it is possible to take advantage of the hierarchical structure when implementing the system in code. As seen in Figure 20, information processed in a lower level region is passed in parallel via cortico-cortical and cortico-thalamo-cortical connections to higher regions of the neocortex [22] [21] [20] [47]. Information processed in lower regions is relayed to high regions in an increasingly invariant form [3], therefore it is possible to think of the information flow between cortical regions as packets of information being passed from one processing unit to another - a hierarchical web of decentralized processors.

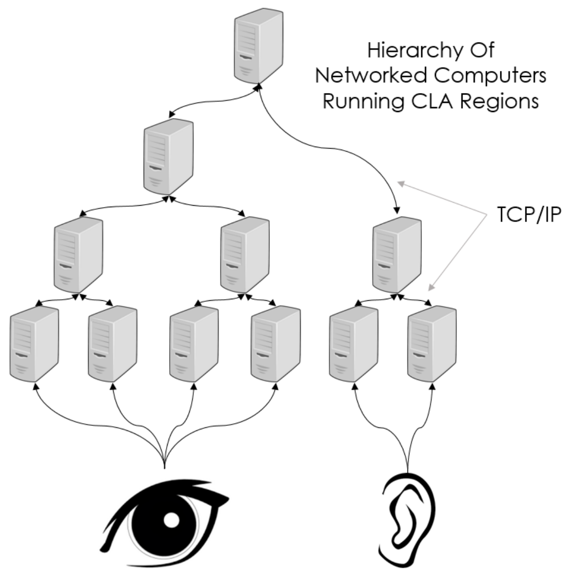


Figure 21. Example of hierarchy of networked computers

The overall system was designed to incorporate the cell regions and hierarchical principles outlined in Chapter 2 as proposed by highly tested and validated assumptions of the hierarchical structure of the brain [3] [11]. This hierarchical design can take advantage of individual system threads and networked computers allowing for a highly scalable overall system architecture. The system architecture is covered in subsequent sections.

Communication between cell regions is accomplished by serializing a CLA communication packet and sending the packet over transport layer protocols. For this thesis, UDP (User Datagram Protocol) was selected for inter cell region communication.

Networked communication between cell regions allows for a scalable network of CLAs by sending CLA communication packet over IP if necessary – i.e. if more processing units are required for a given problem, the user can dynamically add a processing units to the network via local host or connected machines.

Each CLA communication packet contains all necessary input and output information for the networked CLAs including source CLA's identification number and CLA's output information.

Visualization. Visual inspection of the cellular regions developed by the CLA is important to verifying the operation of the algorithm as well as inspection of the Hebbian connections created by the algorithm. In order to visualize the CLA, after training, the algorithm logged the cellular structure of the cell region as well as the weighted value of the Hebbian connections between individual cells- i.e., all cells and axons of the cell region were saved as a visualization file for post-processing. Visualization of the saved cell region can be loaded in Gephi, an open graph visualization platform [48]. The data visualization of cell regions proved invaluable during testing, debugging and presenting the CLA. Gephi allows the user to select individual or multiple cells for analysis. The strength of the Hebbian connections are illustrated in connection thickness - i.e. the thicker the cellular connection, the stronger the connection weight. This visualization tool allows for quick, visual debugging of the CLA by checking cell connections and showing how the CLA was predicting future signals.

CLA structure. This section describes the structure of the CLA, implementation of the CLA, and how the biological elements of the brain are modeled into artificial entities. There are only four elements to the CLA: supergranular cells, name cells, cell columns and cell region. A basic understanding of computer science is assumed – i.e. knowledge of data structures and functions (e.g. classes, lists, methods, etc.).

Table 3

Supergranular Cell Lists

Method Name	Method Description
Input Connections	Input connections list holds pointers to neighboring supergranular cells that are connected to a supergranular cell.
Output Connections	Output connection list holds pointers to neighboring supergranular cells that a supergranular cell is connected to.
Connection Strengths	Connection strength list holds the output connection strength of each connection to other supergranular cells.

Supergranular cell. The supergranular cell is an artificial class in the CLA modeled after the theoretical processes of a biological supergranular cell found in layer 2/3 of the neocortex and is the fundamental pattern recognition component in the CLA. The supergranular cell connects to neighboring cells in the cell region via Hebbian connections which create a network of cells used to predict future time steps. The supergranular cell model contains several lists (Table 3) for handling outgoing and incoming signals from connected cells.

Table 4

Supergranular Cell Methods

Method Name	Method Description
Activate	Activates Cell functionality when evoked putting the cell into an active or predictive state.
Send and Receive	Sends “action potential” to all output connections. Receive accepts “action potential” and calls the activate function.
Modify Connections	Modify connections methods either create, strengthen, decay or destroy Hebbian connections between supergranular cells.

Table 5

Column Class List and Variables

Variable Name	Variable Description
Cells	Cells list is a list of either supergranular or name cells contained in the column
Prediction Value	The prediction value of a column holds a scalar value relative to how likely the column is to fire in the next time step. This value can be normalized by the region in order to predict the next input.

Table 6

Column Class Methods

Method Name	Method Description
Activate	The Activate method is called when the region receives that column’s index.
Burst	The Burst method is called when the column receives a signal, yet none of the cells contained in the column were in a predictive state. The burst method activates all cells contained in the column.
Modify Cells	The column class is able to modify, add or destroy cells contained in the column. This functionality allows for the column to scale as needed for the application

Name cell. The name cell is a special class of supergranular cell that models layer 5 pyramidal cells and layer 2/3 supergranular cells. The purpose of a name cell is to identify strong Hebbian connections between supergranular cells and create an invariant representation of the connected cells. The name cell model assumes that a strong Hebbian connection between supergranular cells indicates that a pattern has been identified. Because the name cell is a subclass of the supergranular cell superclass, the name cell inherits the same functionality as the supergranular cell with the addition of a list of output regions. The list of output regions is used to pass the invariant representation of the connections it has characterized to high regions of the system. Therefore, as signals move from lower level regions to higher level regions, the signals become slower and increasingly invariant.

Table 7

Region Lists and Variables

Variable Name	**Variable Description**
Cells Activity	Monitors Cell activity in region.
Columns	List of columns in cell region.
Column Prediction	List of column prediction values used to predict which column will fire next.
Name Cells	List of name cells contained in the region.
Output list	List of active name cells used pass invariant representations to the next higher region.
Region Bookkeeping	Variables used for region bookkeeping include current time step, input and output regions, input to region and outputs from region, etc.
Region Parameters	The region parameters are used to tune the region for specific applications.

Column. The column class is a container that holds either supergranular or name cells. When a signal arrives at the cell region, the signal holds a list of indices. The indices indicate which column of cells will become active during that time step. The column class models the cortical column (or minicolumn) as described throughout neurobiology literature [19] [47] [22] [3] [20]. The column class models signals sent from the thalamus to layer 4 neurons in the neocortical sheet and contains a variety of lists and methods in order to manage and distribute the signals to the supergranular cells and name cells.

Table 8

Region Methods

Method Name	Method Description
Activate	Activate function starts a single time step for the CLA. Furthermore the activate function activates cell columns as specified by the input string.
Input and Output	The input and output methods parse incoming column indices for column activation. The output method uses name cell activation to create output string of indices to pass to higher level regions.
Feedback	The feedback method takes inputs from higher level regions and plays back patterns stored in the Hebbian connection of the cell structure.
Modify Columns	The column class is able to modify, add or destroy columns contained in the region. This functionality allows for the region to scale as needed for the application
Modify Cells	The region can modify cells by commanding the columns class to call appropriate cell modification methods.
Send Predictions	The send prediction method of the region class allows for the region to normalize column prediction values in order to predict which column/name cells will be active in subsequent time steps.
Sleep	The region sleep method, which is called periodically, cleans and re-baselines the region. The sleep method is a housekeeping method designed to destroy unused Hebbian connections. This method greatly increases the efficiency of the CLA.
Export/Import	The export and import methods are used to store and recall cell regions' structures from the machine's hard drive.
Calculate Metrics	The calculate metrics method keeps track of the rate of bursting throughout the cell region. Higher bursting rates indicates that the cell region is not predicting

Region. The CLA can be isolated to a single cell region – in other words, the cell region was designed with the ability to independently run the CLA. Each cell region can interface with low level regions, high level regions and/or periphery (sensory) inputs. The region holds a list of cell columns, and input and output parsers. The structure of the cell region is illustrated in Figure 22. The region class contains a number of lists, variables and methods for dynamic generation of the region, prediction, commands and general bookkeeping.

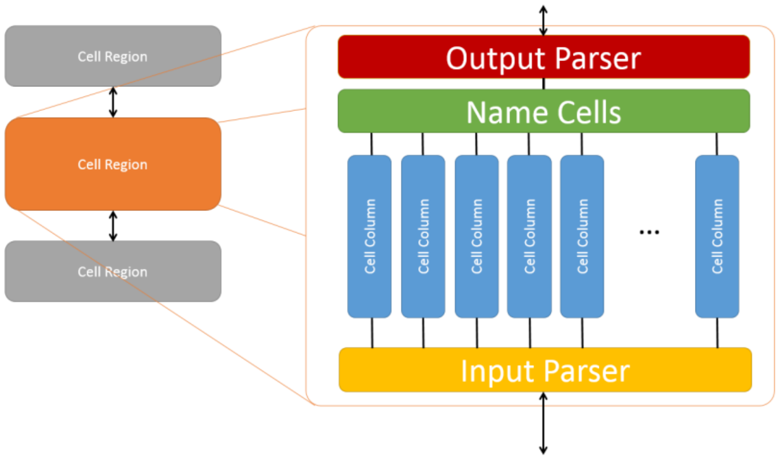


Figure 22. Cell Region structure and interface

As illustrated in Figure 22, each cell region contains an input and output parser. For purposes of this thesis, ASCII (American Standard Code for Information Interchange) encoding was chosen to transmit data between regions. Therefore, the input and output parser recognized comma separated ASCII characters that represent the cell column to be activated during that time step.

The input parser is an abstraction layer between the chosen communication method and the cell columns in the region. The input parser interprets incoming inputs from modalities or lower level regions. When new inputs are available to the region, the input parser converts the ASCII string into column indices to become activated during that time step. The output parser is designed to act in a similar matter to the input parser. The output parser accepts name cell activations and turns the name cell activations into output ASCII text to feed forward to the next cell region.

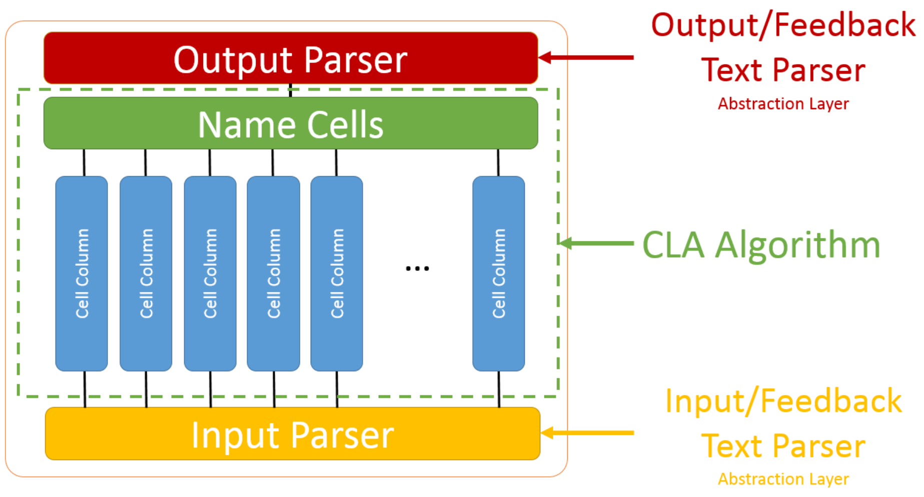


Figure 23. Illustration of the function of both the input and output parser relative to the CLA.

Name cell method CLA graphical example. The following section outlines the graphical process of a new cell region during initial training. The purpose of this section is to illustrate the CLA structures (cells, columns, region) interaction during training and operation. Subsequent sections further illustrate how signals propagate up the cortical hierarchy.

Initialized regions. An initialized cell region contains columns of cells. The cell region's data structure is analogous to the neocortical regions of cells that are aligned in a columnar structure. Prior to any signals being applied to the cell region, the region contains no Hebbian connections between cells in the region – i.e. the region is a "new born" with no bias or prior knowledge. The Figure 24 illustrates a new cell region:

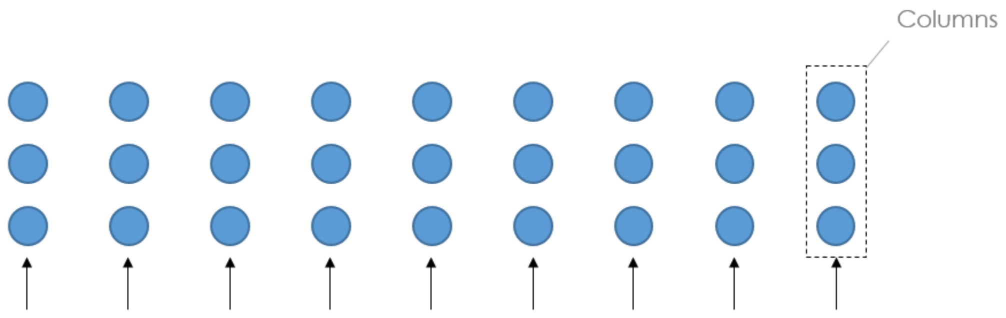


Figure 24. Newly initialized cell region

Introduction of temporal signals. After a cell region has been initialized, the region is ready to accept new inputs from its connected modality. The following figures illustrate the first three time steps of signals introduced to the CLA.

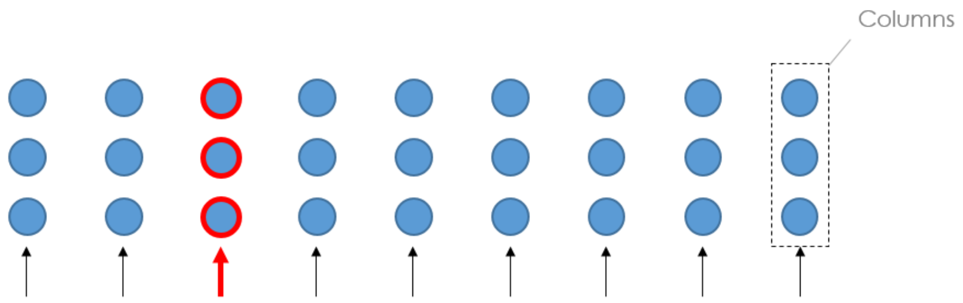


Figure 25. First time step of temporal signal

Figure 25 shows that the third column of cells accepting the first temporal signal to the CLA. Because the cell region was not predicting any cells in column three to become active, the CLA activates all cells in the column (shown in red).

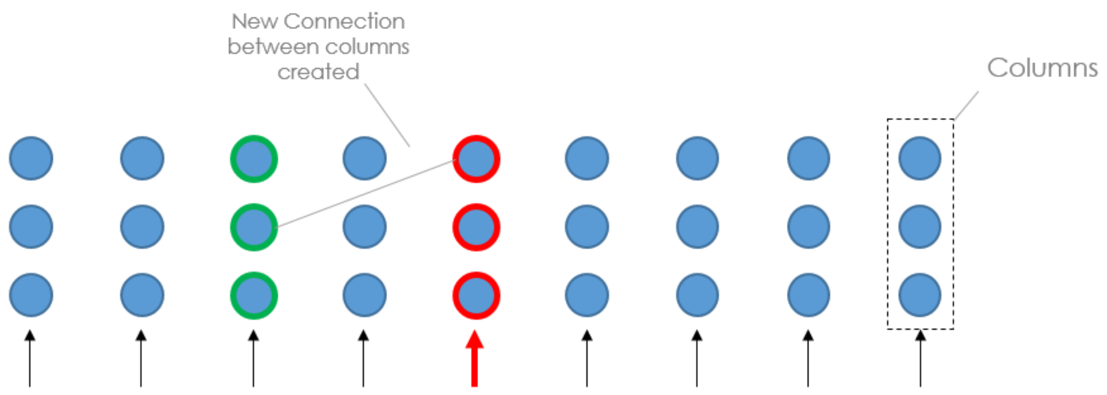


Figure 26. Second time step of temporal signals

In Figure 26, the CLA receives the second temporal signal to the cell region. Again, during this time step, the CLA did not predict any cells to become active, therefore the entire column becomes active. Furthermore, because there were cells active in the previous time step, the CLA picks two cells, one currently active and one previously active, and creates a new Hebbian connection between the cells. By introducing this new Hebbian connection, the CLA has now stored prior knowledge within the region for later prediction. This process is repeated for the next time step, time step three (shown in Figure 27).

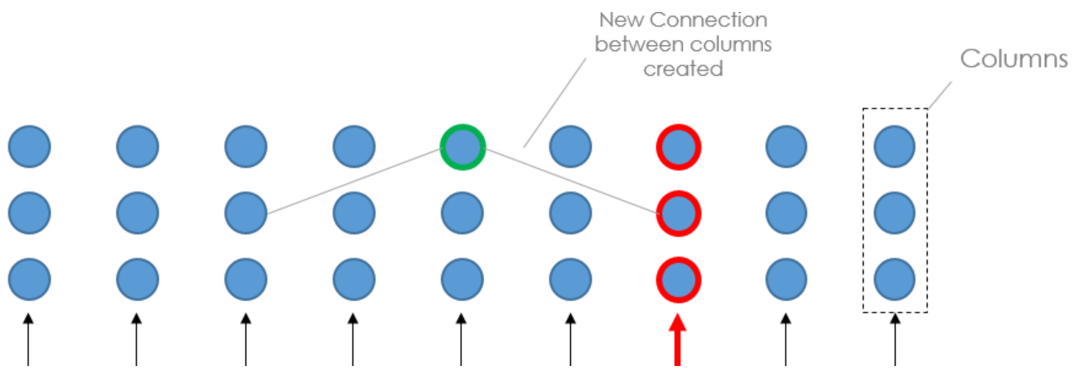


Figure 27. Third time step of temporal signals

Strengthening of Hebbian connections. In this section, the cell region has accepted multiple time steps creating many Hebbian connections. The following figures illustrate a series of Hebbian connections being predicted and strengthened over three time steps.

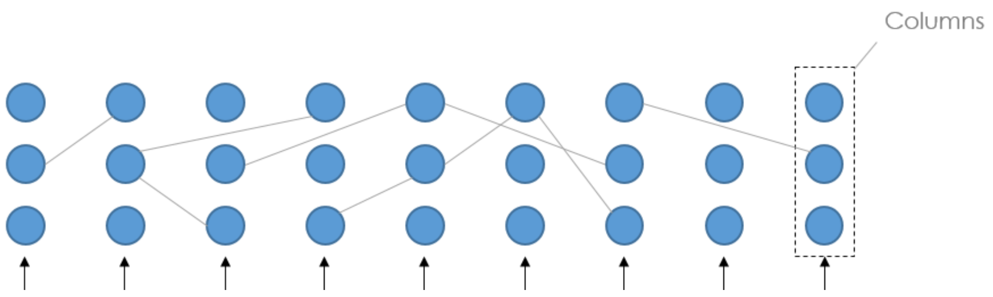


Figure 28. Cell region with Hebbian connections

Figure 28 displays the cell region with many learned Hebbian connections between cells. Each connection was created via temporal input signals. Because the region contains prior knowledge of the modality it represents in the form of Hebbian connections, it is possible for the cell region to actively predict what cells will become active in future time steps. Figure 29 displays a learned signal received by the cell region demonstrating prior knowledge and reinforcing the Hebbian connections.

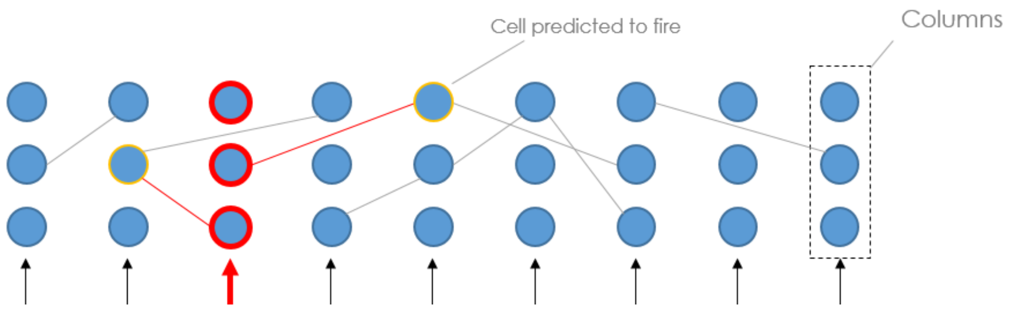


Figure 29. First signal of pattern

As illustrated in Figure 29, a signal arrives at the cell region causing the column of cells to become active because none were predicted to become active. When each cell becomes active, it sends an action potential to all connected cells (illustrated by the red Hebbian connections). Cells that receive an action potential are placed in a predictive state – i.e. neuron polarization. Cells that are placed into a predictive state allow the region to identify the likelihood of what will become active in subsequent time steps.

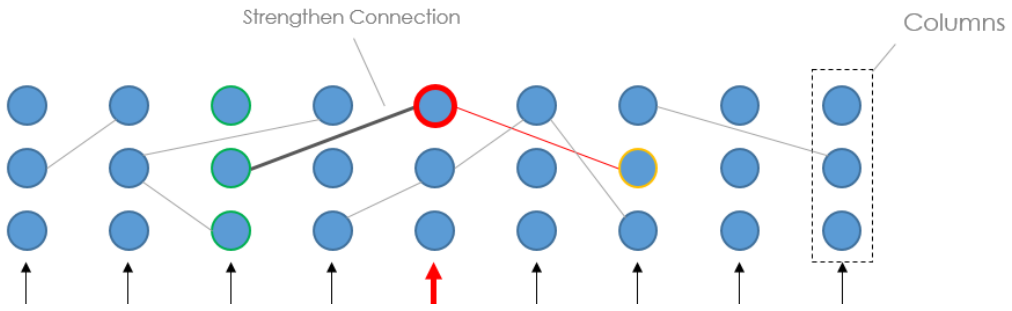


Figure 30. Second signal of pattern

Figure 30, illustrates the next time step of a previously observed pattern. Because the column had a cell in a predictive state, only that predictive cell becomes active in this time step. Again, the active cell sends an action potential to its connected cells. Furthermore, because the Hebbian connection used to predict the activation of this column was correct, the Hebbian connection is strengthened. The strengthening of the Hebbian connection is analogous to the strengthening of connections between neurons in the neocortex. The following figure illustrates the end of the learned pattern and strengthening of correct Hebbian connections.

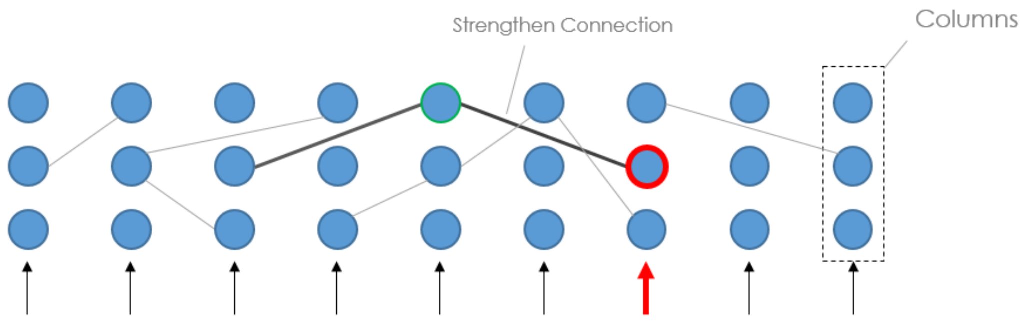


Figure 31. Last signal of pattern

Creating permanent Hebbian Connections. In the previous sections, the cell region created new Hebbian connections to store a pattern and strengthened the learned pattern, respectively. In this section, the cell region stores the pattern as permanent Hebbian connections. This action represents the operation of the CLA developed for this thesis. Figure 32 illustrates the cell region with strengthened Hebbian connections before the temporal signal is applied to the cell region.

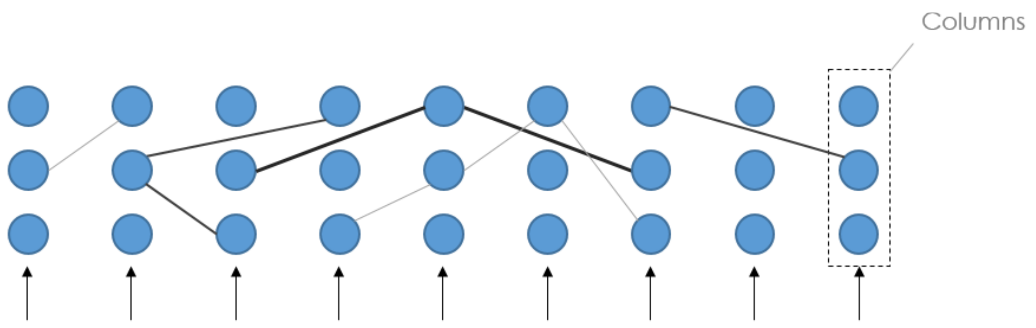


Figure 32. Cell region with strengthened Hebbian connections

Figure 32, illustrates the cell region before the temporal signal is applied (observed by darker, thicker inter-cell connections). In the following series of figures, the same signal is applied to the cell region. However, in this example, the Hebbian connections have reached a variable strength threshold that trigger the permanence of the Hebbian connection. Furthermore, the cell region creates a name cell in order to represent the permanent connection.

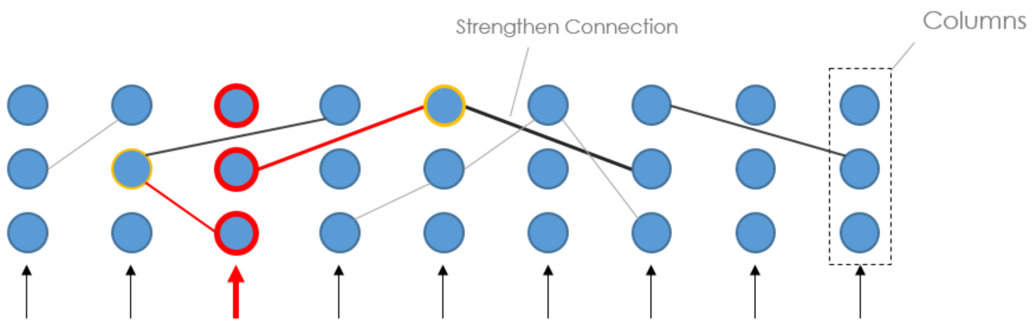


Figure 33. First signal of pattern

As illustrated in Figure 33, the first signal from the connected modality is applied to the cell region. Just like previous examples, the column becomes active and sends action potentials to all connections.

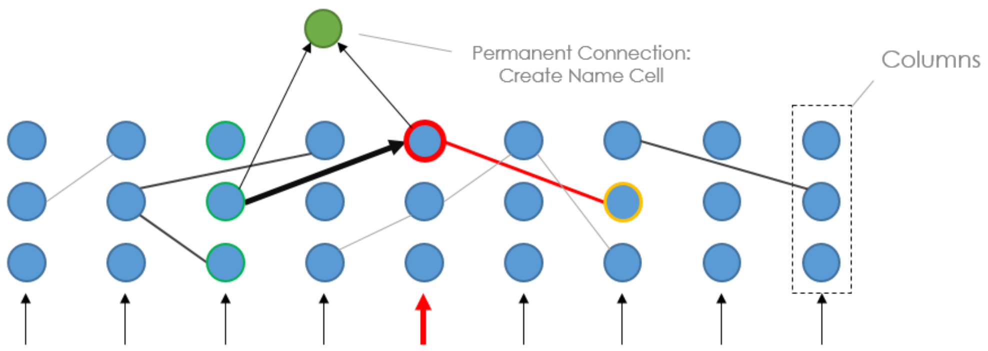


Figure 34. Creation of first name cell

Unlike previous examples, the Hebbian connection that represents the signal transition reaches a permanence threshold. Once the cell region reaches (???) the threshold, it creates a name cell in order to represent the strong Hebbian connection. The name cell will be used to represent the learned pattern and relay the signal to higher cell regions in the hierarchical structure. Finally, Figure 35, the signal creates a second name cell in order to represent the entire signal.

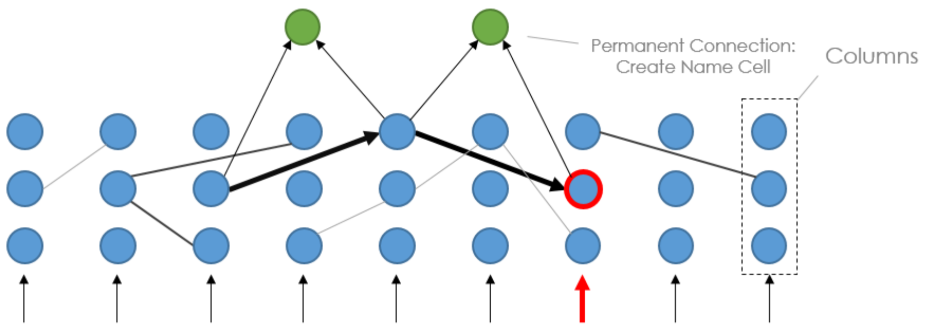


Figure 35. Creation of second name cell

In this series of figures, the cortical learning algorithm has been illustrated – from initialization to name cell characterization. In section 0, graphical examples are illustrated to further characterize the CLA operation.

CLA graphical example. The CLA does not utilize name cells for passing information up the cortical hierarchy. This section describes the method for passing feedforward information up the cortical hierarchy using the same fundamental CLA algorithm.

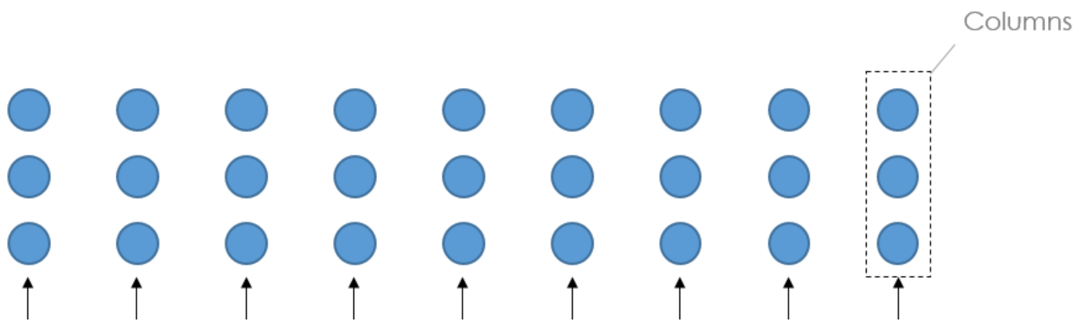


Figure 36. Initialization of CLA

Initialized region. Region initialization of the CLA is identical to the original CLA described in the previous example.

Introduction of temporal signals. As with the original CLA, after a cell region has been initialized, the region is ready to accept new inputs from its connected modality. The following figures illustrate the first three time steps of signals introduced to the CLA.

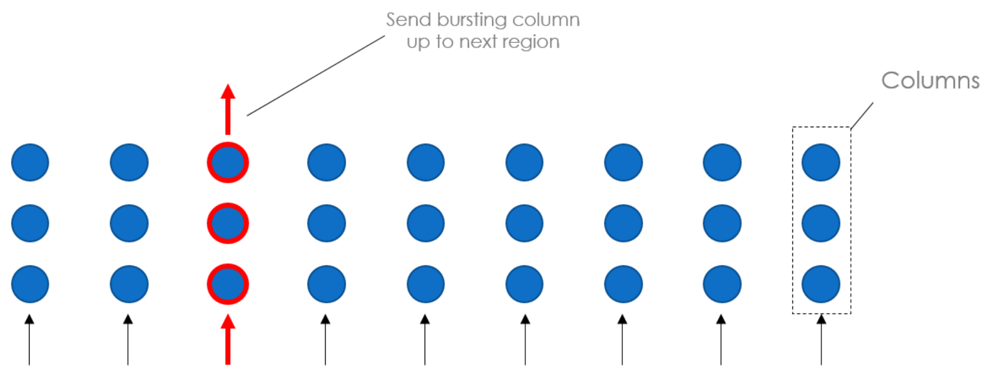


Figure 37. First time step of temporal signal

Figure 37, shows the third column of cells accepting the first temporal signal to the CLA. Because the cell region was not predicting any cells in column three to become active, the CLA activates all cells in the column (shown in red) and creates a feedforward output representing the bursting column (all cells in the column become active). The column "bursts" because it did not contain any cells in a predictive state – i.e. when a column bursts, it is surprised it received an input.

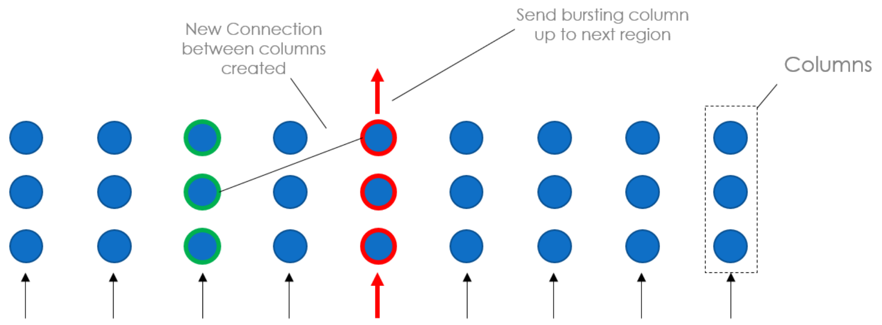

New Connection between columns created
Send bursting column up to next region
Columns

Figure 38. Second time step of temporal signals

In Figure 38, the CLA receives the second temporal signal to the cell region. Again, during this time step, the CLA did not predict any cells to become active, therefore the entire column becomes active. Furthermore, because there were cells active in the previous time step, the CLA picks two cells, one currently active and one previously active, and creates a new Hebbian connection between the cells and because the column bursts, the CLA creates a feedforward output representing the bursting column. By introducing this new Hebbian connection, the CLA has now stored prior knowledge within the region for later prediction. This process is repeated for the next time step, time step three (shown in Figure 39) as well as a new feedforward output due to the column bursting activation.

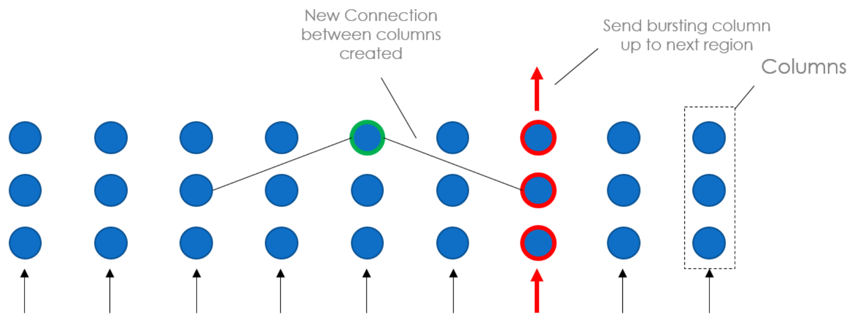

New Connection between columns created
Send bursting column up to next region
Columns

Figure 39. Third time step of temporal signals

Strengthening of Hebbian connections. The following figures illustrate a series of Hebbian connections being predicted and strengthened over, and outputs produced during three time steps using the CLA.

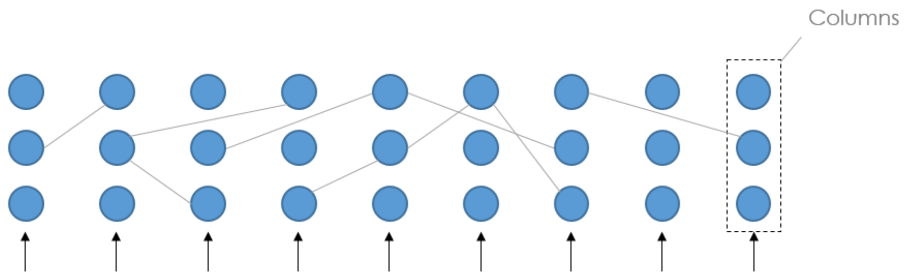


Figure 40. Cell region with Hebbian connections

Figure 40 displays the cell region with many learned Hebbian connections between cells. Each connection was created via temporal input signals. Because the region contains prior knowledge of the modality it represents in the form of Hebbian connections, it is possible for the cell region to actively predict what cells will become active in future time steps. As illustrated in Figure 42, a signal arrives at the cell region causing the column of cells to become active (or burst) because none were predicted to become active. When each cell becomes active, it sends an action potential to all connected cells (illustrated by the red Hebbian connections). Cells that receive an action potential are placed in a predictive state – i.e., neuron polarization. Cells that are placed into a predictive state allow the region to identify the likelihood of what will become active in subsequent time steps. Furthermore, because the cell region "burst" we output that column for feedforward learning. As illustrated in Figure 42, a signal arrives at the cell region causing the column of cells to become active (or burst) because none were predicted to become active. When each cell becomes active, it sends an action potential to all connected cells (illustrated by the red Hebbian connections). Cells that receive an action potential are placed in a predictive state – i.e., neuron polarization. Cells that are placed into a predictive state allow the region to identify the likelihood of what will become active in subsequent time steps. Furthermore, because the cell region "burst" we output that column for feedforward learning.

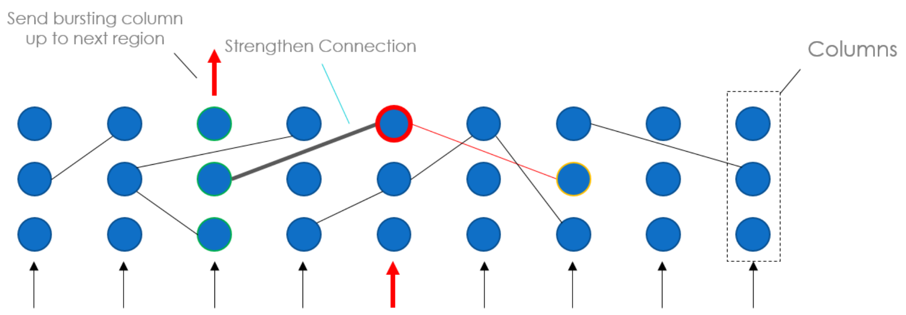


Figure 41. Connection strengthened

Figure 41, displays a learned signal received by the cell region demonstrating prior knowledge and reinforcing the Hebbian connections.

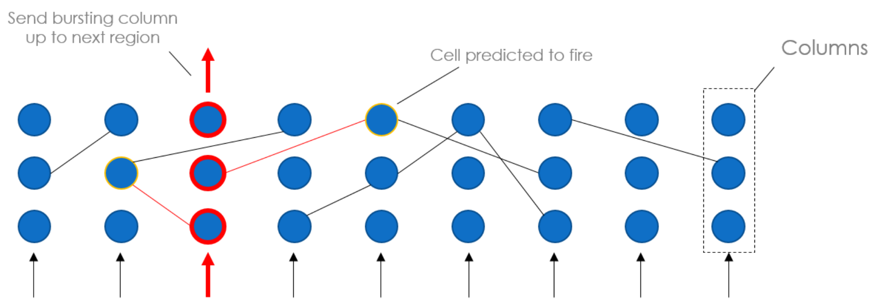


Figure 42. First signal of pattern

As illustrated in Figure 42, a signal arrives at the cell region causing the column of cells to become active (or burst) because none were predicted to become active. When each cell becomes active, it sends an action potential to all connected cells (illustrated by the red Hebbian connections). Cells that receive an action potential are placed in a predictive state – i.e., neuron polarization. Cells that are placed into a predictive state allow the region to identify the likelihood of what will become active in subsequent time steps. Furthermore, because the cell region becomes active, we output that column for feedforward learning.

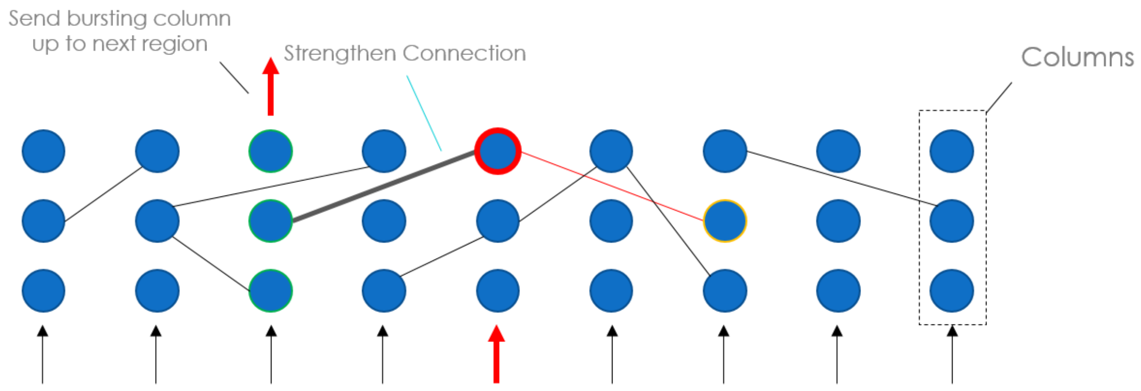


Figure 43. Second signal of pattern

Figure 43 illustrates the next time step of a previously observed pattern. Because the column had a cell in a predictive state, only that predictive cell becomes active in this time step. Again, the active cell sends an action potential to its connected cells. Furthermore, because the Hebbian connection used to predict the activation of this column was correct, the Hebbian connection is strengthened. The strengthening of the Hebbian connection is analogous to the strengthening of connections between neurons in the neocortex. Finally, because the region correctly predicted the column activation, it continues to apply the previous output, as displayed in Figure 43. Figure 44 illustrates the end of the learned pattern and strengthening of correct Hebbain connections. Again, because the region correctly predicted the column activation, it continues to apply the previous output.

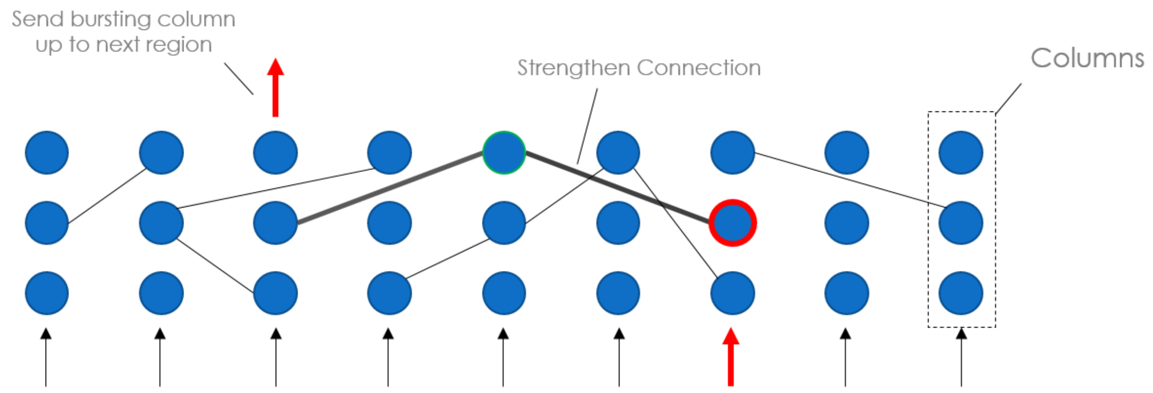


Figure 44. Last signal of pattern

Applications

Unsupervised pattern recognition. Our connected world contains an abundance of information but it is often economically infeasible to manually classify the data for use in supervised learning algorithms. Furthermore, data are often time sensitive – i.e. the usefulness of collected data decays exponentially as time progresses - and manual classification of data is very time consuming. Therefore, it is necessary to use unsupervised learning algorithms to find underlying patterns in the data, in real time. Today, this is accomplished via data mining techniques. Data mining is the computational process of discovering patterns in large datasets and is used throughout the Big Data industry to transform unlabeled data into comprehensible structures for further use [49]. Data mining algorithms have allowed researchers to understand and use Big Data to utilize unlabeled data and take actions on the data. Using the CLA, it is possible to the build models of large quantities of data in real time and taking actions immediately. The following paragraphs provide two examples of applications of hierarchical structures that build models of Big Data. One example comes from deep learning research between Stanford and Google [9], and the other example is of Numenta's Grok [10] (a commercially available hierarchical CLA). These examples provide useful applications of hierarchical structures on real world data – i.e., new unsupervised hierarchical algorithms similar to the CLA described in this thesis.

The Stanford and Google collaboration, *Building High-level Features Using Large Scale Unsupervised Learning*, was a large scale demonstration of deep learning and was conducted using 15 million unlabeled frames from YouTube on a 16,000 core supercomputer. The purpose of this experiment was to test DBNs (Deep Belief Networks) on a large scale and simulate high-level class-specific neuron using unlabeled data. The experiment was conducted to demonstrate that it is possible to train neurons to be selective for high-level concepts using entirely unlabeled data and achieved a 70% relative improvement over the state-of-the-art in 2012 [9] –The Stanford and Google team proved that it is possible to use hierarchical structures to model completely unlabeled data. Furthermore, this experiment proves that a structure, very similar to the hierarchical CLA structure can build high-level models without any human interaction.

At Numenta, Inc., Jeff Hawkins and his team are using their open source CLA algorithm, NuPIC [4], to build a commercial product named Grok [10]. Grok uses NuPIC in order to detect anomalies in online servers. The early detection of issues in servers aid in building highly reliable networked systems at scale. Furthermore, the Grok system performs complex pattern detection, automated modeling, and adaptive learning using the open source NuPIC CLA. This system requires no more than an Amazon web server to run the CLA and has proven very useful in real world applications [10] such as IT analytics, rogue behavior detection, and geospatial tracking.

There are many opportunities and possible applications for cortical learning algorithms and hierarchical structures (classified under deep architectures) in the pattern recognition and machine learning fields. There are many recent examples of using this new family of deep architectures in research and industry applications [50] [9] [10] [6] [7]. As discussed in the Pattern Recognition Properties of an Intelligent System section 0, these biologically inspired algorithms have the potential to build upon the state-of-the-art in many classification applications including data fusion, non-stationary environments, anomaly detection, etc. Furthermore, it is important to note that cortical learning algorithms and other neurobiology inspired learning algorithms/hardware are currently in their infancy (e.g. NuPIC [4], Human Brain Project [51], DARPA Synapse [52], etc.).

Motor control. The nervous system evolved around motor control and the mammalian neocortex has a well-defined and well documented primary motor cortex. Although the primary motor cortex is responsible for much of the motor control neural activity in the brain recent research suggests that motor control signals propagate from primary sensory regions (such as the visual cortex)of the brain as well as the primary motor cortex [53] [11]. The cortical structure involved in motor control is also the same cortical structure used in perception and prediction [3] [12]. The plentiful research surrounding biological motor control via the cortex will have very important applications in the future of robotic control.

Classically, in order to control a robot, researchers and engineers must explicitly map feed forward commands and feedback sensors to individual motor controllers, utilize complex filtering techniques and develop data fusion algorithms. The design and tuning process involved in a single dimensional control problem, depending on the application, could take dozens of engineering hours to accomplish, which is very economically expensive as well as taxing on the controls engineers. Because the CLA and deep learning architectures (discussed in this thesis) are analogous to the structure of the low level regions and primary motor cortex of the mammalian brain, it is plausible that the CLA could enhance motor control in robotic and industrial applications and open the commercial market to complex, high-dimensional robotic systems analogous to biological skeletal systems.

Cortical Learning Algorithms

We now provide the pseudocode for unsupervised learning, supervised learning, and classification. Unsupervised learning CLA is the baseline algorithm upon which the supervised and classification algorithms are based. The basic components of the CLA are illustrated as a UML diagram in Figure 45. Moreover, a graphical representation of the algorithm is illustrated to further familiarize the reader with the CLA algorithms – the graphical examples, although simple, should sufficiently demonstrate the functionality of the algorithms.

UML diagram. In the UML diagram shown in Figure 45 illustrate the components of the CLA and hierarchical structuring. The CLA is comprised of regions that hold columns of cells. The Cell class outputs action potentials to a list of output cells and holds corresponding connection strengths. An action potential is generated via the Cell Activate() function which calls a connected cell's ReceiveInput() method.

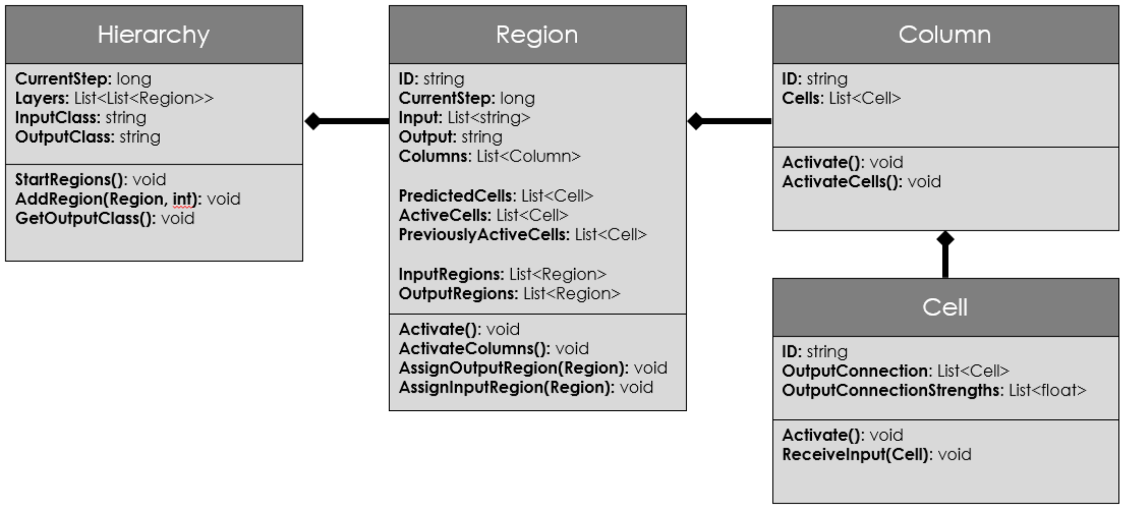


Figure 45. UML diagram

Columns contain a list of cells. By calling the Column Activate() method, an input from the region is processed; the Column Class then decides to "burst" if it contains no predicted cells, otherwise it activates predicted cells.

The Region class maintains a list of predicted, active, and previously active cells that the Cell and Column class can access during activation. The Region class also contains a list of input regions and output regions. Input and output regions are used to pass information processed by the Region class up or down a hierarchy. The Region Activate() method is called when the region receives a new input. The Activate() function activates the corresponding column in the region's column list via the ActivateColumns() method.

When Activate() returns, the region produces a new output that can be passed up the hierarchy.

The Hierarchy class manages multiple regions and the structure of the regions. In other words, the Hierarchy class is the manager of multiple CLAs – it manages starting and stopping CLA threads, data input streams, and classification tasks. Furthermore, the Hierarchy class monitors all CLA outputs which are used for supervised learning.

Unsupervised learning name cell CLA.

Algorithm pseudocode.

```
ActivateColumn(*column* C):
$P_{prev}$ is previously predicted cells
$c_i$ is a cell contained in $C$ (i.e. $c_i \in C$)

**If** $\{c_i \in C\} \nexists P_{prev}$ **do**
$\forall\{c_i \in C\}$ **ActivateCell**($c_i$)
Else
  **For** $\forall\{c_i \in P_{prev}\}$ **do**
    **ActivateCell**($c_i$)
  End do
End do


ActivateCell(*cell* C):
$Cell$ is the current cell being evaluated
$Cells_{active}$ is the list of currently active cells in the parent region
$X$ is the output connections contained by the cell
$x_i$ is an output connection contained in $X$ (i.e. $x_i \in X$)
$cell_i$ is the cell(s) connected to $x_i$
$P_{curr}$ is the list of currently predicted cells
$N_{active}$ is the list of currently active name cells

**For** $\forall\{x_i \in X\}$ **do**
**If** $Cell$ is a name cell **do**
  add: $Cell$ to $N_{active}$
Else
  add: $cell_i$ to $P_{curr}$
  add: $Cell$ to $Cells_{active}$
  End do
End do
```

```
Unsupervised Learning CLA:
$\boldsymbol{A}$ is the input vector
$\alpha_i$ is the input $\{\alpha_i \in \boldsymbol{A}\}$ that is linked to column $l_i$
$L$ is the list of columns contained in the region
$l_i$ is column $i$ contained in $L$ (i.e. $l_i \in L$)
$P_{prev}$ is previously predicted cells
$P_{curr}$ is currently predicted cells
$Cells_{prev}$ is the list of previously active cells
$Cells_{active}$ is the list of currently active cells
$Cells_{region}$ is the list of all cells in the region
$T_{current}$ is the current time step of the region
$N$ is the list of all name cells in a region
$N_{active}$ is the list of name cells active during the current time step
$\beta$ is the connection strength increment value
$\gamma$ is the connection permanence threshold

$\alpha_i = \emptyset$
Parse $\boldsymbol{A}$ into $\alpha_i$where $\{\alpha_i \in \boldsymbol{A}\}$
$\forall\{\alpha_i\}$ **ActivateColumn**(column $l_i$) corresponding to $\alpha_i$
**For** $\forall\left\{cells\left\{cell_i \in Cells_{region}\right\}\right\}$ **do**
  **If** $cell_i \in Cells_{prev}$ **do**
    **For** $\forall\left\{hebbian\ connections\left\{x_{ij} \in cells_i.connections\right\}\right\}$ **do**
      **For** $\forall\left\{active\ cells\left\{cells_k \in Cells_{active}\right\}\right\}$
  **If** $x_{ij}$ is connected to $c_k$ **do**
  **If** $x_{ij}$ less than $\gamma$ **do**
Increment Hebbian connection $x_{ij}$ by $\beta$ (i.e. $x_{ij}$+= $\beta$)
**If** $x_{ij}$ greater than $\gamma$ **do**
  Add: new name cell, $n$, to $N$
  Add: new Connection from $cell_i$ to $n$
  Add: new Connection from $cell_k$ to $n$
End do
End do
        Else
          Create Connection $x_{i(j+1)}$ connecting $cell_i\ to\ cell_k$ with initial strength $\beta$
        End do
      End do
    End do
  End do
End do
**For** $\forall\left\{cells\left\{cell_i \in Cells_{region}\right\}\right\}$ **do**
  **For** $\forall\left\{hebbian\ connections\left\{x_{ij} \in cell_i.connections\right\}\right\}$ **do**
    **If** $x_{ij}$ less than 0 **do**
```

```
        Destroy $x_{ij}$
      else
        Decrement $x_{ij}$: (i.e. $x_{ij}$−=1)
      End do
    End do
End do
**For** $\forall\{columns\ \{l_i \in L\}\}$ **do**
    $l_i$ predictive state = $\emptyset$
End do
**For** $\forall\{predicted\ cells\ \{p_i \in P_{curr}\}\}$ **do**
Increment $p_i$.parentColumn predictive state (i.e. $p_i$.parentColumn.predictiveState +=1)
End do
$P_{prev} = P_{curr}$
$P_{curr} = \emptyset$
$T_{current} = T_{current} + 1$
**For** $\forall\{name\ cells\ \{n_i \in N_{active}\}\}$ **do**
convert $n_i$ ID to output string
End do
**For** $\forall\{names\ cells\ n_i \in N\}$ **do**
    **If** $n_i$ activity is <.2 **do**
      $n_i\ activity = 0$
    Else
      $n_i\ activity = \frac{n_i\ activity}{2}$
    End do
End do
```

Graphical representation. Below is a graphical representation of the unsupervised learning algorithm. The graphical representation of the algorithm displays four time steps of a pre-trained cell region containing 9 cortical columns and 8 name cells.

The first time step, t_0, displays a CLA cell region after it has been trained. As illustrated in Figure 46, there are thin connections and thick connections between the cells in each column. These connections represent Hebbian connections where the thin connections are weak and the thick connections are strong. The stronger the Hebbian connection between cells, the more often the connection has been observed by the CLA –

i.e., patterns that activate subsequent columns. Furthermore, connections between cells (blue) and name cells (green) represent a strong semantic meaning.

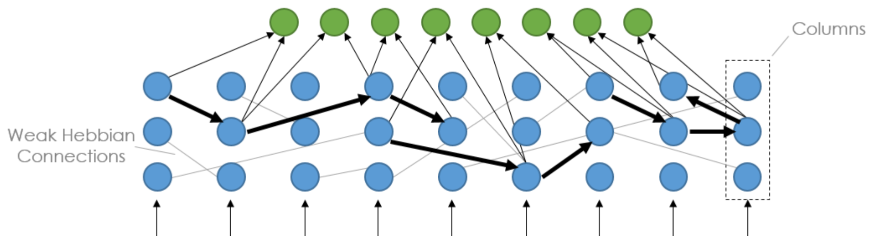


Figure 46. Graphical representation of an unsupervised CLA at t_0

In the first time step, t_1, the cell region receives a signal that activates column number 4. Because there were no cells predicted to become active in column number 4, the CLA orders the entire column of cells to send out signals to their connections – i.e. the column “bursts.” In the graphical representation, the red cells and connection represent cell activations and outgoing signals, respectively. This bursting action places the cell region in a very active state where many of the active cell’s connections are predicting that they will become active soon.

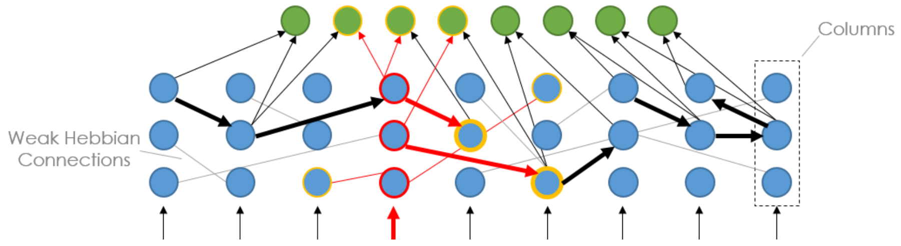


Figure 47. Graphical representation of an unsupervised CLA at t_1

In the next time step, t_2, the cell region receives an input to the 6th column. Because the 6th column contains a cell that was predicting it would become active, only that cell becomes active. During the cell's activation, it sends signals to the cells that it has Hebbian connections to. As illustrated in Figure 48, there are fewer active and predicted cells because the CLA made a correct prediction from prior observations in the form of a strong Hebbian connection. Also, as seen in Figure 48, a cell in the 7th column is predicting it will become active during the next time step. Finally, the 4th name cell has become active because it has identified a prior pattern. This name cell activation can now be passed to the next higher region in the cortical hierarchy.

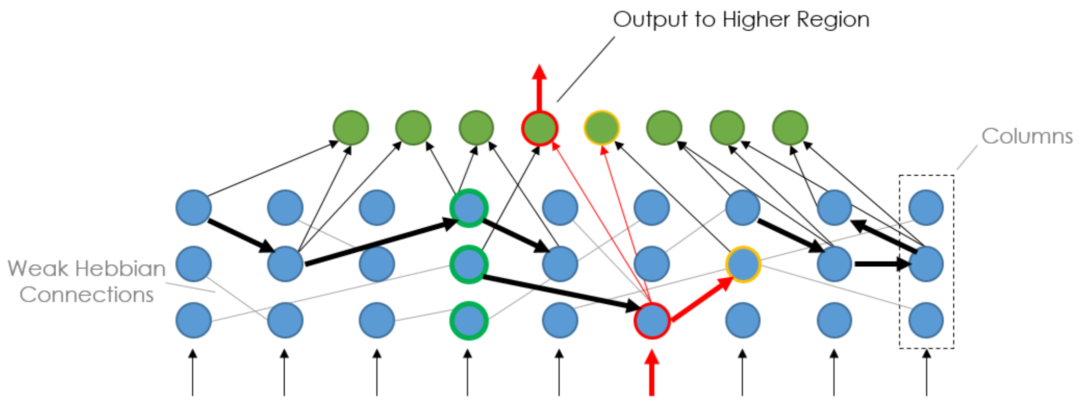


Figure 48. Graphical representation of an unsupervised CLA at t_2

The final time step of this graphical example validates the prediction made by the CLA in t_2. As illustrated in Figure 49, column 7 becomes active via lower input. Column 7's activation triggers the predicted cell to become active and send outputs to connected cells. Again, because this cell is connected to a name cell (name cell 5), and that name cell was predicted to receive an input from t_2, the 5th name cell becomes active creating an output from the cell region to be sent to above regions in the cortical hierarchy for further processing.

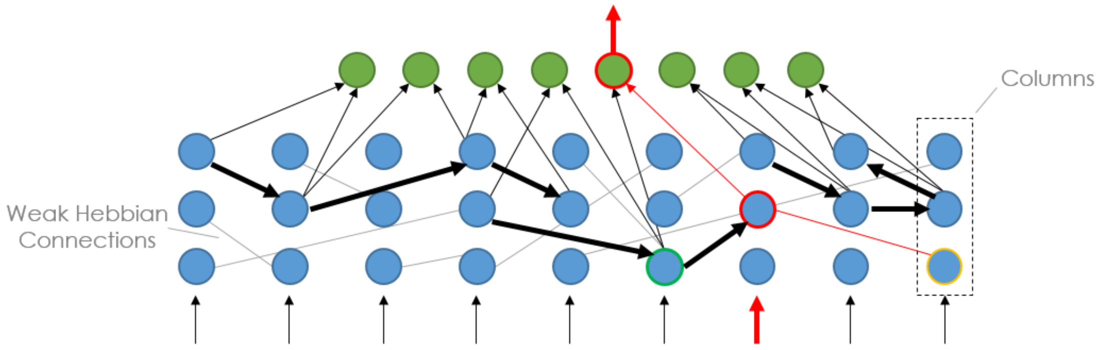


Figure 49. Graphical representation of an unsupervised CLA at t_3

Supervised learning name cell CLA.

Algorithm pseudocode.

```
Supervised Learning CLA
$\boldsymbol{A}$ is the input vector
$\alpha_i$ is the input $\{\alpha_i \in \boldsymbol{A}\}$ that is linked to column $l_i$
$S$ is the input class
$T$ is the list of class columns
$t_i$ is the class column $i$ contained in $T$ (i.e. $t_i \in T$)
$O$ is the class output
$L$ is the list of columns contained in the region
$l_i$ is column $i$ contained in $L$ (i.e. $l_i \in L$)
$P_{prev}$ is previously predicted cells
$P_{curr}$ is currently predicted cells
$Cells_{prev}$ is the list of previously active cells
$Cells_{active}$ is the list of currently active cells
$Cells_{region}$ is the list of all cells in the region
$T_{current}$ is the current time step of the region
$N$ is the list of all name cells in a region
```

```
$N_{active}$ is the list of name cells active during the current time step
$\beta$ is the connection strength increment value
$\gamma$ is the connection permanence threshold

$\alpha_i = \emptyset$
Parse $\boldsymbol{A}$ into $\alpha_i$where $\{\alpha_i \in \boldsymbol{A}\}$
**If** $S$ contains corresponding class column $t_s \in T$ **do**
  **For** $\forall\{name\ cells\ \{n_i \in N_{active}\}\}$ **do**
    Add: $n_i$ to $t_s$
  End do
Else
  Add: new class column $t_s$ to $T$
End do
$\forall\{\alpha_i\}$ **ActivateColumn**() column $l_i$ corresponding to $\alpha_i$
**For** $\forall\left\{cells\{cell_i \in Cells_{region}\}\right\}$ **do**
  **If** $cell_i \in\ C_{ellsprev}$ **do**
    **For** $\forall\left\{hebbian\ connections\{x_{ij} \in cell_i.connections\}\right\}$ **do**
      **For** $\forall\{active\ cells\{cell_k \in Cells_{active}\}\}$
  **If** $x_{ij}$ is connected to $cell_k$ **do**
  **If** $x_{ij}$ less than $\gamma$ **do**
Increment Hebbian connection $x_{ij}$ by $\beta$ (i.e. $x_{ij}{+}{=}\ \beta$)
**If** $x_{ij}$ greater than $\gamma$ **do**
  Add: new name cell, $n$, to $N$
  Add: new Connection from $cell_i$ to $n$
  Add: new Connection from $cell_k$ to $n$
End do
End do
        Else
          Create Connection $x_{i(j+1)}$ connecting $cell_i\ to\ cell_k$ with initial strength $\beta$
        End do
      End do
    End do
  End do
End do
**For** $\forall\left\{cells\{cell_i \in Cell_{region}\}\right\}$ **do**
  **For** $\forall\left\{hebbian\ connections\{x_{ij} \in cell_i.connections\}\right\}$ **do**
    **If** $x_{ij}$ less than 0 **do**
      Destroy $x_{ij}$
    else
      Decrement $x_{ij}$: (i.e. $x_{ij}{-}{=}1$)
    End do
  End do
End do
```

```
For $\forall\{columns\ \{l_i \in L\}\}$ do
  $l_i$ predictive state = $\emptyset$
End do
For $\forall\{predicted\ cells\ \{p_i \in P_{curr}\}\}$ do
Increment $p_i$.parentColumn predictive state
End do
$P_{prev} = P_{curr}$
$P_{curr} = \emptyset$
$T_{current} = T_{current} + 1$
For $\forall\{name\ cells\ \{n_i \in N_{active}\}\}$ do
convert $n_i$ ID to output string
End do
For $\forall\{class\ columns\ \{t_i \in T\}\}$ do
  $class\ total_i = \sum number\ of\ active\ name\ cells \in t_i$
End do
$O$ = index of $Max\{class\ total\}$
For $\forall\{names\ cells\ n_i \in N\}$ do
  If $n_i$ activity is <.2 do
    $n_i\ activity = 0$
  Else
    $n_i\ activity = \frac{n_i\ activity}{2}$
  End do
End do
```

Graphical representation. Below is a graphical representation of the supervised learning algorithm. The graphical representation of the algorithm displays four time steps of a pre-trained cell region containing 9 cortical columns and 8 name cells. The intent of this graphical example is to demonstrate to the reader that the CLA creates a probabilistic analysis of the incoming, online data stream and classifies the pattern using prior knowledge during initial training.

Figure 50 represents a cell region that was trained using a supervised CLA with three classes. As seen in the figure, the structure is very similar to the unsupervised CLA graphical example. The primary difference between the supervised and unsupervised CLA is the grouping of name cells into classes.

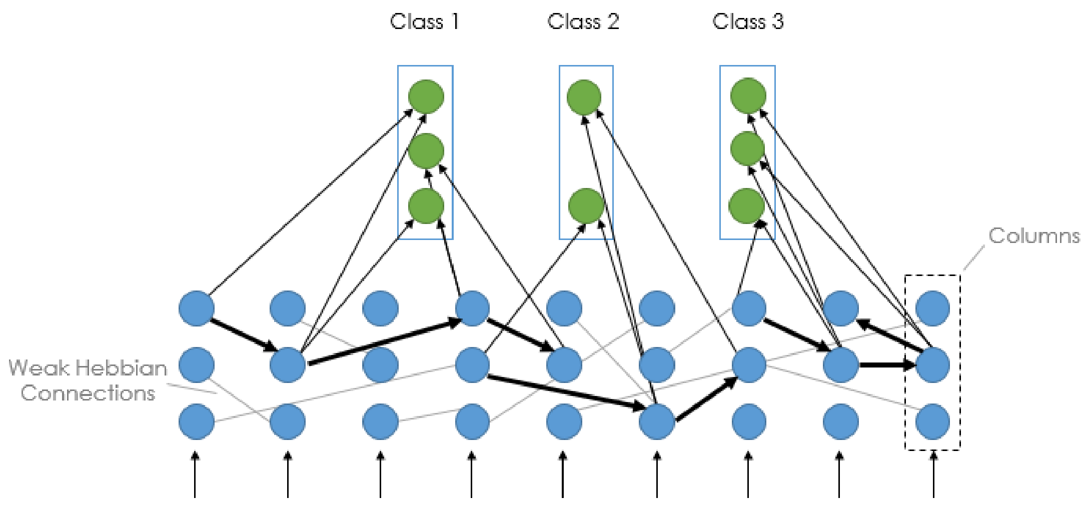


Figure 50. Graphical representation of a supervised CLA at t_0

The same pattern is introduced to the supervised learning algorithm as was introduced in the unsupervised graphical representation of the unsupervised CLA. The same pattern was picked to show the differences between unsupervised and supervised learning after training.

At time step t_1, the supervised CLA uses the same bursting method as the unsupervised CLA because the column was not predicted to fire – casing all cells in the column to become active and send outputs to their respective Hebbian connections. The only difference between the unsupervised and supervised CLA is that the name cells are grouped into known classes. As seen in Figure 51, the CLA identifies the pattern as part of either class 1 or class 2. Because this is the first time step, the CLA must wait to observe the next temporal step in the pattern before it can definitively identify the pattern.

At time step t_2, the CLA has received a second temporal piece of the unknown pattern. As predicted, the cell in column 6 becomes active sending its Hebbian outputs to connected cells.

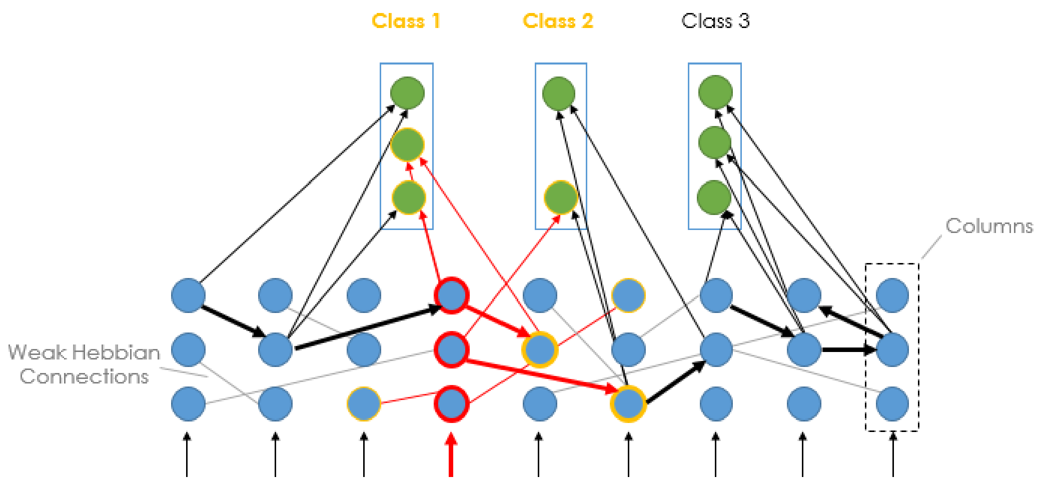


Figure 51. Graphical representation of a supervised CLA at t_1

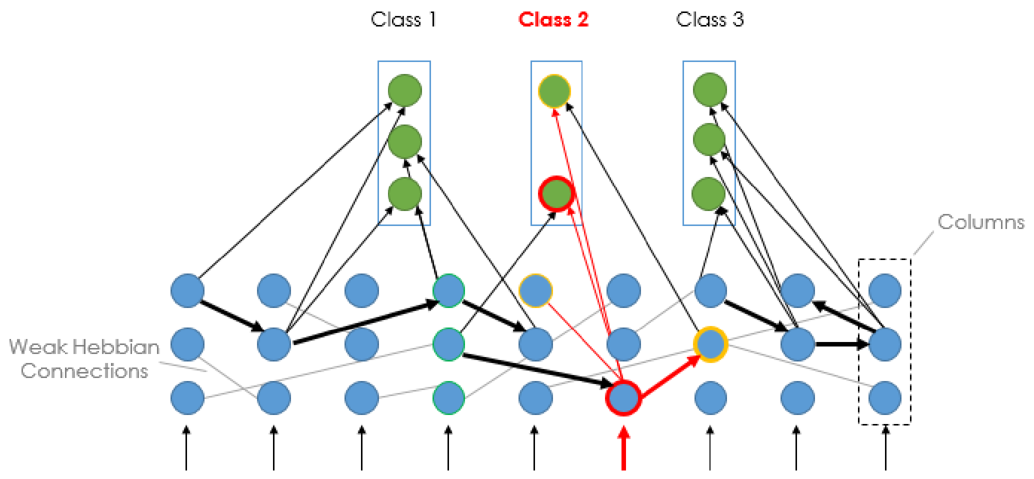


Figure 52. Graphical representation of a supervised CLA at t_2

As shown in Figure 52, the predicted name cell in the Class 2 group becomes active due to column 6 input. Because that name cell is the only name cell active, it is highly likely that the pattern belongs to Class 2, therefore, the supervised CLA outputs the class label, Class 2.

Although during time step t_2 the supervised CLA outputted Class 2, the pattern has not ended. In time step t_3, the final temporal piece of the pattern is introduced to the supervised CLA. In this step, the cell region correctly holds the output of the CLA at Class 2 (seen in Figure 53).

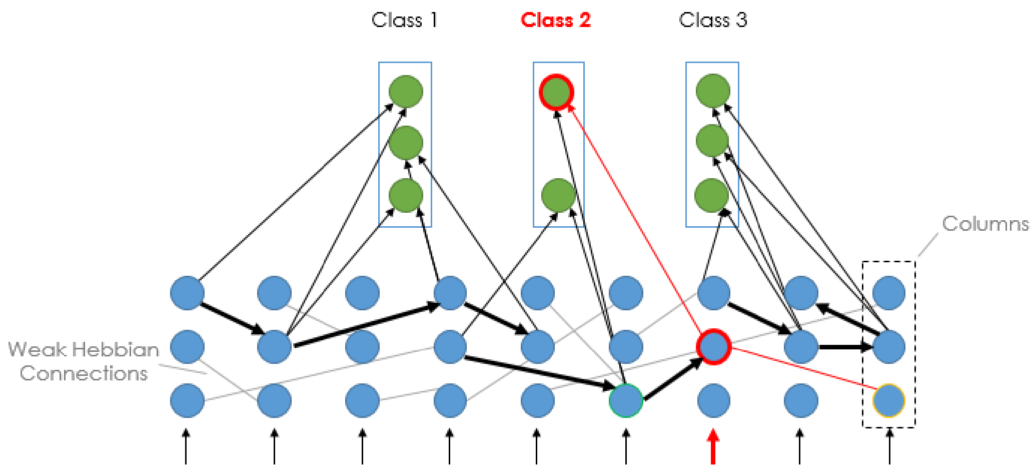


Figure 53. Graphical representation of a supervised CLA at t_3

As illustrated above, Class 2 is identified as the current temporal pattern observed by the supervised CLA. It's worth noting that Class 2 was identified two time steps in a row. This example correctly illustrates the concept of invariant representations – where lower signals change faster than the output of the region. The invariant representation of Class 2 can be reported to a higher level region in a cortical hierarchy or simply outputted as the class label for classification.

Model CLA. The CLA uses the same cortical learning algorithm as the previously described algorithms without the use of name cells for feedforward outputs. Instead, this model of the CLA utilizes all column bursting activation as feedforward output of the cell region.

Algorithm pseudocode.

```
Model Unsupervised Learning CLA (single time step):
$\boldsymbol{A}$ is the input vector
$\alpha_i$ is the input $\{\alpha_i \in \boldsymbol{A}\}$ that is linked to column $l_i$
$L$ is the list of columns contained in the region
$l_i$ is column $i$ contained in $L$ (i.e. $l_i \in L$)
$P_{prev}$ is previously predicted cells
$P_{curr}$ is currently predicted cells
$Cells_{prev}$ is the list of previously active cells
$Cells_{active}$ is the list of currently active cells
$Cells_{region}$ is the list of all cells in the region
$T_{current}$ is the current time step of the region
$\beta$ is the connection strength increment value

$\alpha_i = \emptyset$
Parse $\boldsymbol{A}$ into $\alpha_i$where $\{\alpha_i \in \boldsymbol{A}\}$
$\forall\{\alpha_i\}$ ActivateColumn() column $l_i$ corresponding to $\alpha_i$
For $\forall\left\{cells\left\{cell_i \in Cells_{region}\right\}\right\}$ do
  If $cell_i \in Cell_{prev}$ do
    For $\forall\left\{hebbian\ connections\left\{x_{ij} \in cell_i.connections\right\}\right\}$ do
      For $\forall\left\{active\ cells\left\{cell_k \in Cells_{active}\right\}\right\}$
  If $x_{ij}$ is connected to $cell_k$ do
  If $x_{ij}$ less than $\gamma$ do
Increment Hebbian connection $x_{ij}$ by $\beta$ (i.e. $x_{ij} += \beta$)
End do
        Else
          Create Connection $x_{i(j+1)}$ connecting $cell_i\ to\ cell_k$ with initial strength $\beta$
        End do
      End do
    End do
  End do
End do
For $\forall\left\{cells\left\{cell_i \in Cells_{region}\right\}\right\}$ do
  For $\forall\left\{hebbian\ connections\left\{x_{ij} \in cell_i.connections\right\}\right\}$ do
    If $x_{ij}$ less than 0 do
      Destroy $x_{ij}$
    else
```

```
        Decrement $x_{ij}$: (i.e. $x_{ij}$−=1)
      End do
    End do
End do
For $\forall\{columns\ \{l_i \in L\}\}$ do
    $l_i$ predictive state = $\emptyset$
End do
For $\forall\{predicted\ cells\ \{p_i \in P_{curr}\}\}$ do
Increment $p_i$.parentColumn predictive state
End do
$P_{prev} = P_{curr}$
$P_{curr} = \emptyset$
$T_{current} = T_{current} + 1$
For $\forall\{columns\ \{l_i \in L\}\}$ do
    If $l_i$ is bursting do
        Convert $l_i$ ID to output string
    End do
End do
```

Graphical representation. Below is a graphical representation of the alternate cortical learning algorithm. The graphical representation of the algorithm displays four time steps of a pre-trained cell region containing 9 cortical columns. The intent of this graphical example is to demonstrate to the reader that the CLA creates a probabilistic analysis of the incoming, online data stream and classifies the pattern using prior knowledge from initial training.

The figure below, Figure 54, illustrates a pre-trained cell region at t_0.

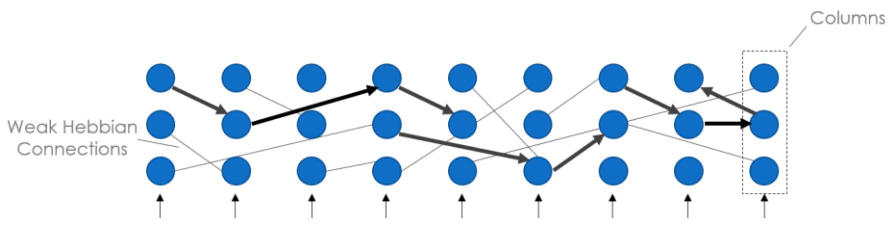


Figure 54. Graphical representation of alternate CLA at t_0

In Figure 55, time step t_1, the fourth column in the region becomes active. Because no cells in the column were predicted to become active during t_1, the entire column of cells burst which, subsequently, causes the column to create a feedforward output signal for use in higher regions in the hierarchy.

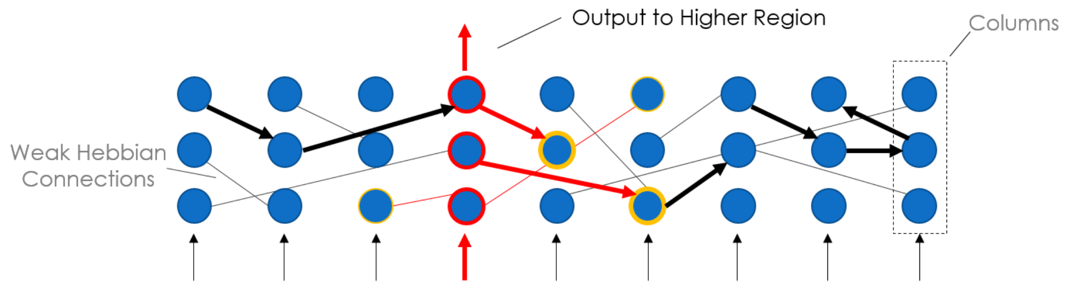


Figure 55. Graphical representation of alternate CLA at t_1

In Figure 56, time step t_2, the sixth column in the cell region becomes active. Because the sixth column contained a predicted cell, the column does not burst – thus, the fourth column output remains high.

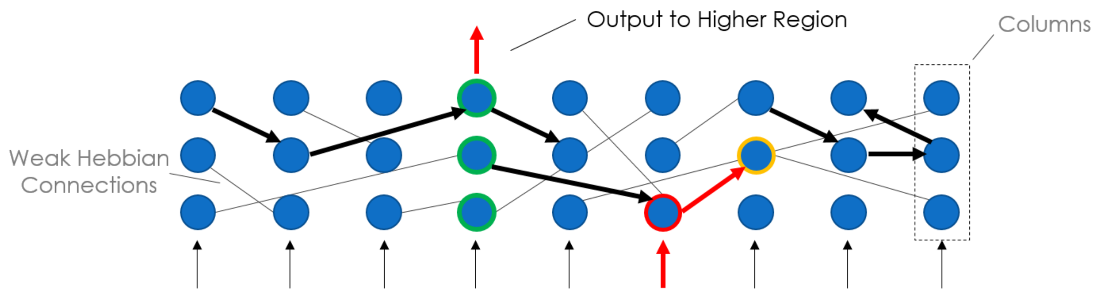


Figure 56. Graphical representation of alternate CLA at t_2

In Figure 57, time step t3, the seventh column in the cell region becomes active. Similarly to t2, because the seventh column contained a predicted cell, the column does not burst – thus, the fourth column output remains high.

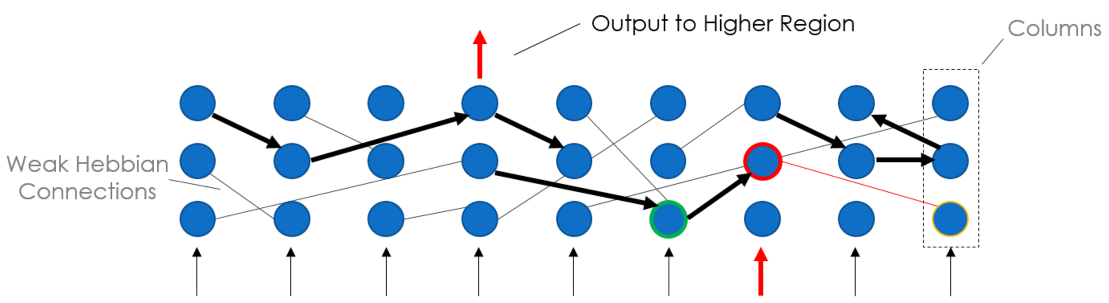


Figure 57. Graphical representation of alternate CLA at t3

In Figure 58, time step t4, the first column fires but does not contain any predicted cells. Therefore, the column bursts and activates a feedforward output representing column number one. Again, this output is send to higher regions in the hierarchy.

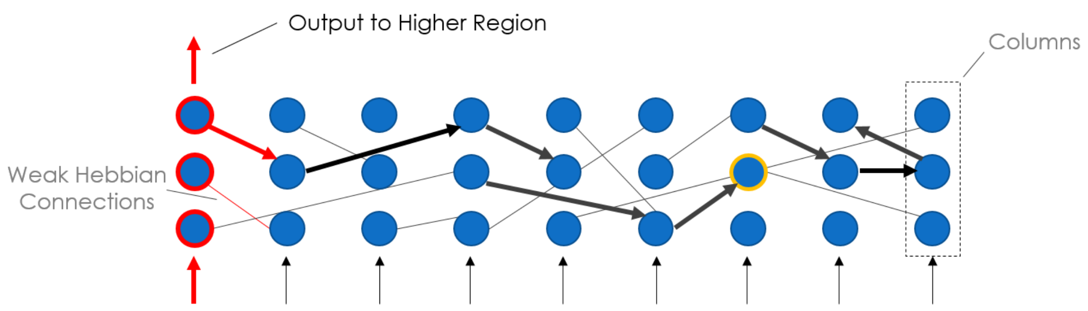


Figure 58. Graphical representation of alternate CLA at t4

Hierarchical classification.

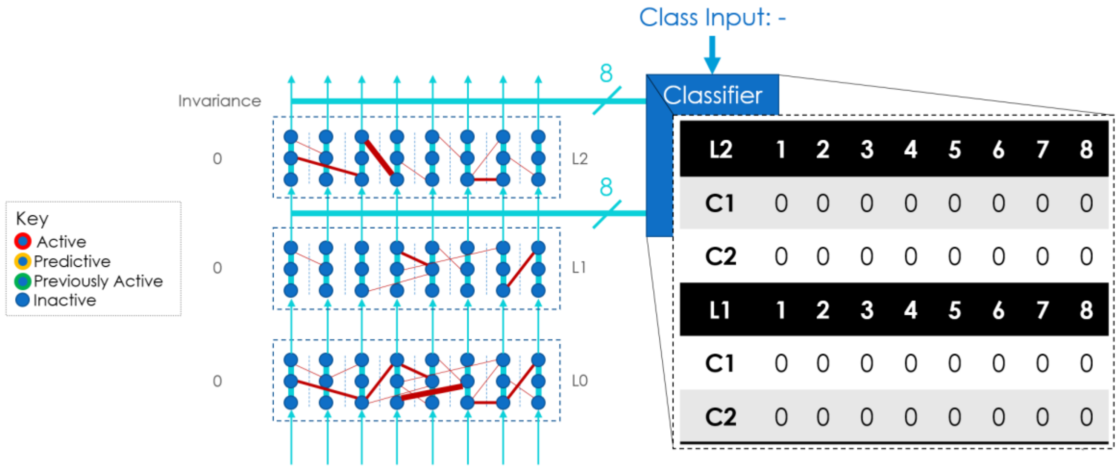
Training and classification pseudocode.

```
**Training**(class $S$):
$O$ is the list of all CLA outputs contained in the hierarchy
$o_i$ is an active CLA output contained in $O$ (i.e. $o_i \in O$)
$count_i$ is the list linked to $o_i$
$c_{ij}$ is a list of class counts corresponding to each output in $count_i$ (i.e. $count_{ij} \in count_i$)
$S$ is the current class introduced the hierarchy

For $\forall\{active\ output\{o_i \in O\}\}$ do
For $\forall\left\{class\ count\{count_{ij} \in count_i\}\right\}$ do
If S corresponds to $count_{ij}$ do
  Increment $count_{ij}$ (i.e. $count_{ij} += 1$)
End do
End do
End do

Classification:
$O$ is the list of all CLA outputs contained in the hierarchy
$o_i$ is an active CLA output contained in $O$ (i.e. $o_i \in O$)
$count_i$ is the list linked to $o_i$
$count_{ij}$ is a list of class counts corresponding to each output in $c_i$ (i.e. $c_{ij} \in c_i$)
$m_j$ is the total value corresponding to each class
$Class$ is the class output as integer

For $\forall\{active\ output\{o_i \in O\}\}$ do
For $\forall\left\{class\ count\{count_{ij} \in count_i\}\right\}$ do
    $m_j += count_{ij}$
  End do
End do
Return $Class = index\ of\ Max\{m_j\}$ (i.e. return class index)
```

Graphical representation. Classification via a hierarchy of CLAs is achieved utilizing output signals from the various CLAs contained in the hierarchy. In other words, when a feedforward output is activated, the output is correlated to the current class being introduced. In the following section, a graphical example describes how both training and classification are achieved via the hierarchy.

In this example, classifier training of a linear hierarchy of initialized alternate CLA models is executed over the course of ten discrete time steps. The classification example contains two classes. The purpose of this graphical example is to 1) illustrate how signals propagate up the linear hierarchy 2) demonstrate how output signals are correlated to class input during training and 3) demonstrate invariance propagates throughout a hierarchy of CLAs.

In, Figure 59, time step t_0, the initialized alternate CLAs begin their training process. The classifier table contains eight columns, which represent the eight columns of cells in the example. Both layer 1 and layer 2 (labeled L1 and L2, respectively) contain rows for class 1 and class 2. When a class is introduced to the classifier during training, and a column output from L1 and/or L2 are active, the training algorithm increments the column corresponding to the class.



L2	1	2	3	4	5	6	7	8
C1	0	0	0	0	0	0	0	0
C2	0	0	0	0	0	0	0	0

L1	1	2	3	4	5	6	7	8
C1	0	0	0	0	0	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 59. Initialized linear hierarchy of alternate CLAs

In Figure 60, time step t_1, the first input is applied to the linear hierarchy while class 1 input applied to the classifier for training. The signal arrives at the lowest layer, L0, which is analogous to a primary sensory region in the brain. As illustrated in previous examples, because the input was not predicted by column 4, the column enters a bursting state. Therefore, all cells become active in the column and send signals to all cells connected placing the connected cells in a predictive state. Furthermore, as displayed in Figure 60, because the column burst, it creates an output from column 4 to layer 1. The signal then propagates to layer 2 via bursting column in L1. Finally, because class 1 is introduced to the classifier for training and L1 and L2 contain an output from column 4, the classifier increments class 1 (rows C1 in the classifier table), column 4 for each layer.

During time step t_2, time step t_2, illustrated in Figure 61, the second input to the primary sensory region (layer 0) is applied to column 5. Column 5 contained multiple predicted cells in t_1, therefore the column does not enter a bursting state – instead, only the predicted cells in the column become active, and send signals to their connected cells. Furthermore, because the column did not burst, no feedforward signal is generated by layer 0 – therefore, the previous output remains active and the regions above (layer 1 and 2) do not need to process signals during this time step. Finally, similarly to t_1, because class 1 is introduced to the classifier for training and L1 and L2 contain an output from column 4, the classifier increments class 1 (rows C1 in the classifier table), column 4 for each layer.

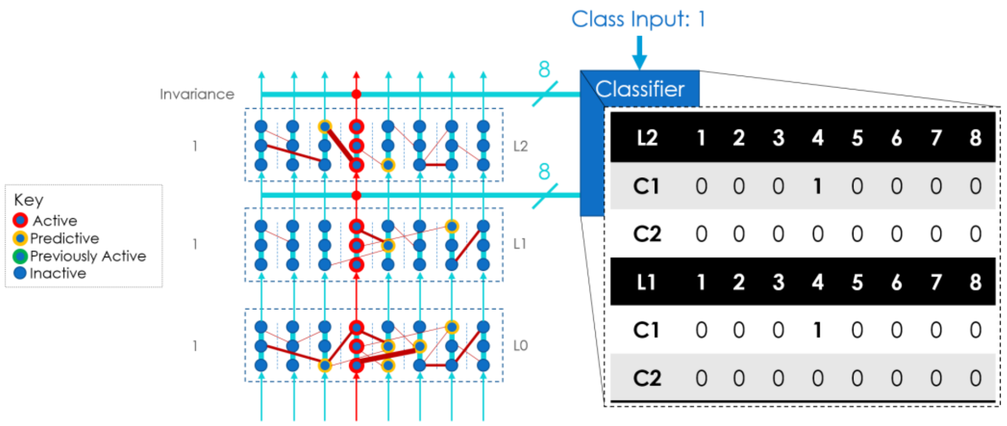


Figure 60. Initialized linear hierarchy of alternate CLAs at t_1

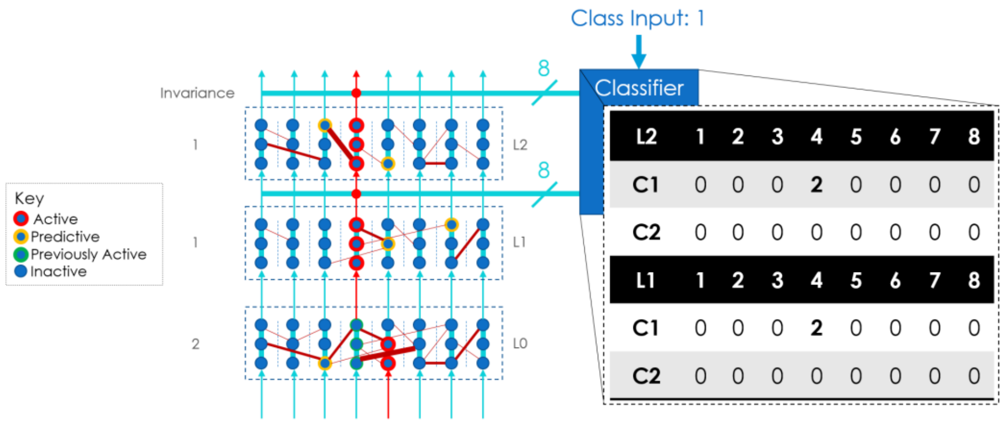
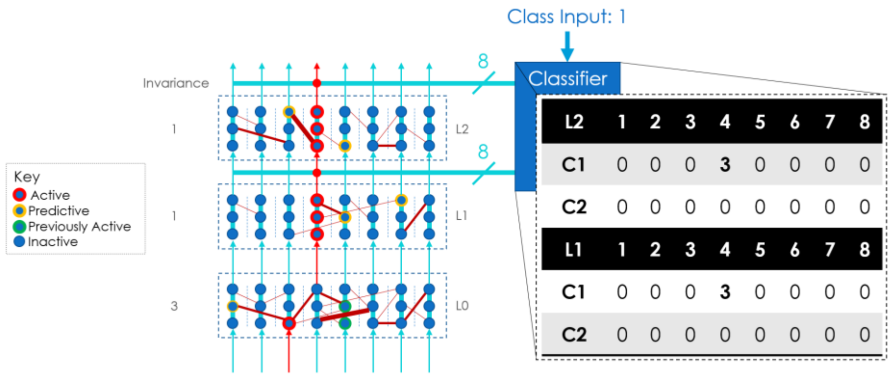


Figure 61. Initialized linear hierarchy of alternate CLAs at t_2

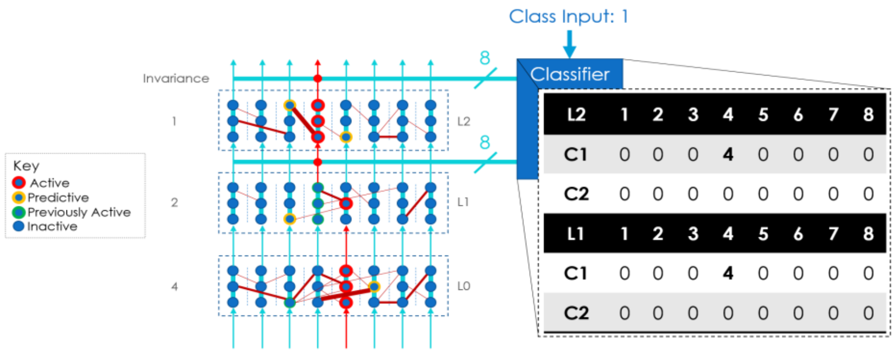
During time step t_3, illustrated in Figure 62, the third input to the primary sensory region (layer 0) is applied to column 3. Column 3 contained a predicted cell in t_2, therefore the column does not enter a bursting state - only the predicted cells in the column become active and send signals to their connected cells. Again, because the column did not burst, no feedforward signal is generated by layer 0 – therefore, the regions above do not need to process signals during this time step. Finally, the classifier increments class 1 (rows C1 in the classifier table), column 4 for each layer.



L2	1	2	3	4	5	6	7	8
C1	0	0	0	**3**	0	0	0	0
C2	0	0	0	0	0	0	0	0
L1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C1	0	0	0	**3**	0	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 62. Initialized linear hierarchy of alternate CLAs at t$_3$

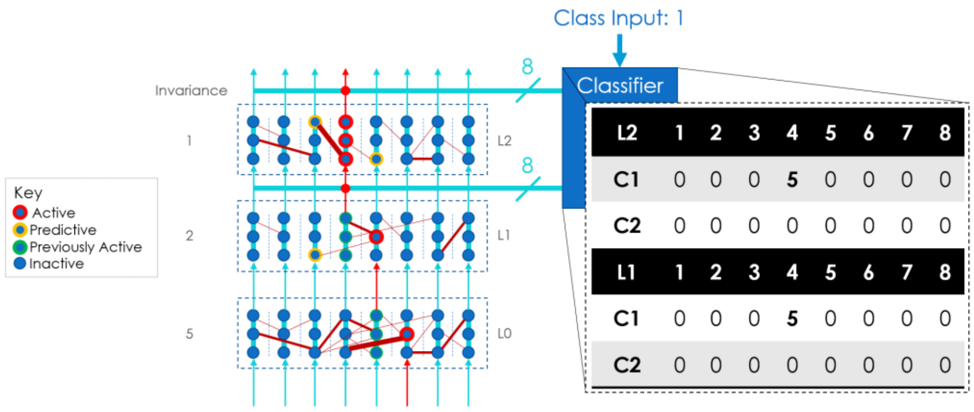
During time step t$_4$, illustrated in Figure 63, the primary sensory region receives an input at column 5 – which was not predicted to become active. Therefore, the column bursts creating an output signal at column 5. The column 5 feedforward output to layer 1 was predicted and, therefore, the layer 1 region does not enter a burst mode allowing the output to layer 1 to remain invariant. As illustrated by the "Invariance" column to the left of the linear hierarchy, the lower layers are changing quickly while the upper layer is not changing (or remaining invariant). Again, the classifier increments class 1 (rows C1 in the classifier table), column 4 for each layer.



L2	1	2	3	4	5	6	7	8
C1	0	0	0	**4**	0	0	0	0
C2	0	0	0	0	0	0	0	0
L1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C1	0	0	0	**4**	0	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 63. Initialized linear hierarchy of alternate CLAs at t$_4$

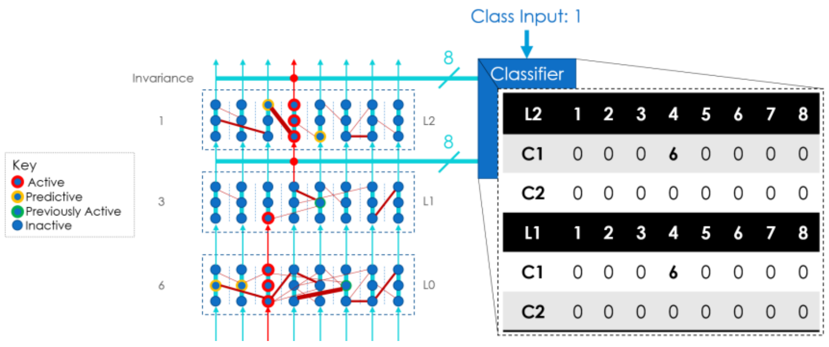
Time step t_5, illustrated in Figure 64, is similar to time steps t_2 and t_3. Again, the classifier increments class 1 (rows C1 in the classifier table), column 4 for each layer while the layer 0 region handles the incoming signal.



L2	1	2	3	4	5	6	7	8
C1	0	0	0	5	0	0	0	0
C2	0	0	0	0	0	0	0	0
L1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C1	0	0	0	5	0	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 64. Initialized linear hierarchy of alternate CLAs at t_5

During time step t_6, illustrated in Figure 65, is similar to time step t_4 where the layer 0 region burst and is handled by layer 1. Again, the classifier increments class 1 (rows C1 in the classifier table), column 4 for each layer.

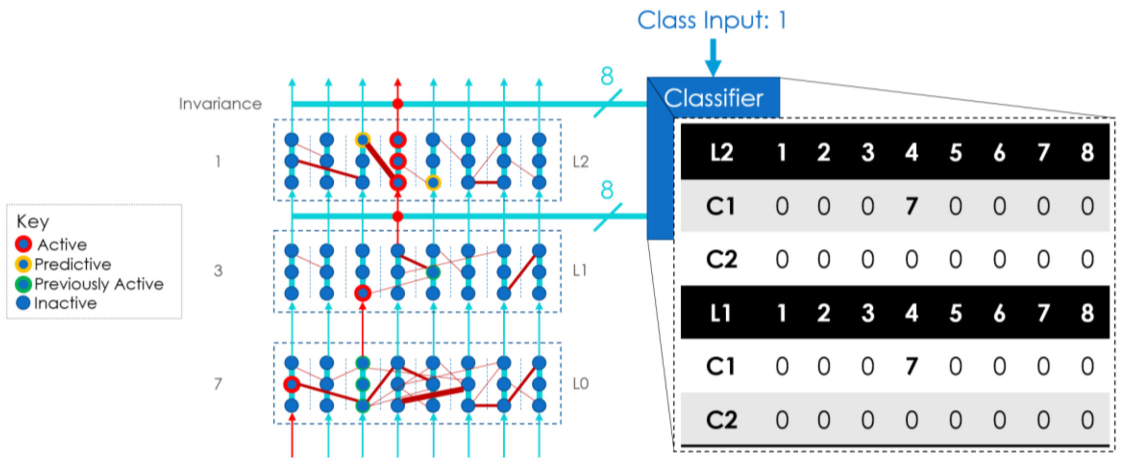


L2	1	2	3	4	5	6	7	8
C1	0	0	0	6	0	0	0	0
C2	0	0	0	0	0	0	0	0
L1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C1	0	0	0	6	0	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 65. Initialized linear hierarchy of alternate CLAs at t_6

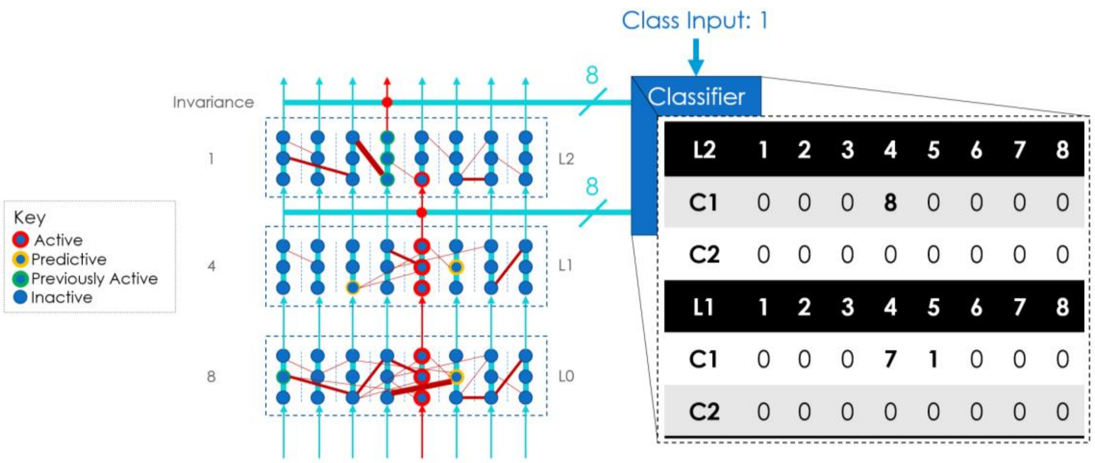
Time step t7, illustrated in Figure 66, is similar to time step t5. Again, the classifier increments class 1 (rows C1 in the classifier table), column 4 for each layer while the layer 0 region handles the incoming signal.

Time step t8, illustrated in Figure 67, a new signal is applied to column 5 of the primary sensory region, L0. Because no cells were predicted in L0, the column bursts and a signal is passed to L1 via feedforward activation. Similarly to the signal arriving at L0, L1 contained no predicted cells causing column 5 to burst and a signal is passed to L2 via feedforward activation. Finally, L2 receives the feedforward signal at column 5 but contains a predicted cell – therefore, L2 does not burst and maintains its feedforward output at column 4. The new output from L1 is captured and the classifier increments class 1, column 5 for L1 and class 1, column 4 for L2.



L2	1	2	3	4	5	6	7	8
C1	0	0	0	7	0	0	0	0
C2	0	0	0	0	0	0	0	0
L1	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
C1	0	0	0	7	0	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 66. Initialized linear hierarchy of alternate CLAs at t7

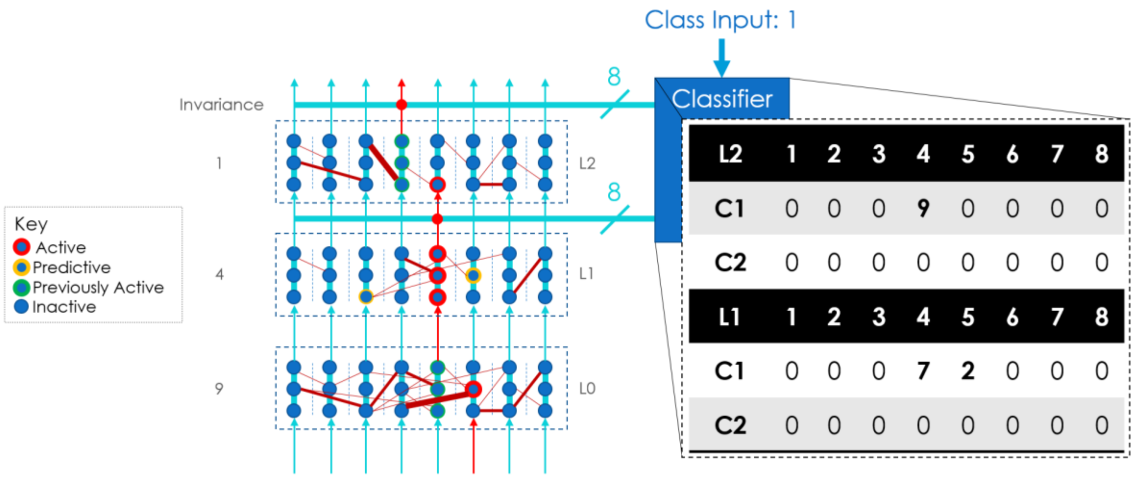


L2	1	2	3	4	5	6	7	8
C1	0	0	0	**8**	0	0	0	0
C2	0	0	0	0	0	0	0	0

L1	1	2	3	4	5	6	7	8
C1	0	0	0	**7**	**1**	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 67. Initialized linear hierarchy of alternate CLAs at t_8

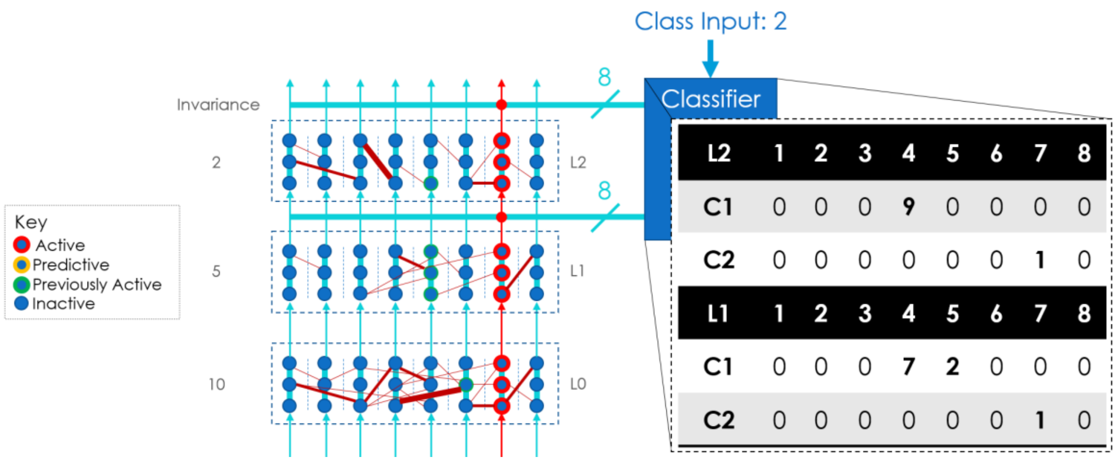
In time step t_9, illustrated in Figure 68, L0 receives a predicted input a column 5.



L2	1	2	3	4	5	6	7	8
C1	0	0	0	**9**	0	0	0	0
C2	0	0	0	0	0	0	0	0

L1	1	2	3	4	5	6	7	8
C1	0	0	0	**7**	**2**	0	0	0
C2	0	0	0	0	0	0	0	0

Figure 68. Initialized linear hierarchy of alternate CLAs at t_9



L2	1	2	3	4	5	6	7	8
C1	0	0	0	9	0	0	0	0
C2	0	0	0	0	0	0	1	0

L1	1	2	3	4	5	6	7	8
C1	0	0	0	7	2	0	0	0
C2	0	0	0	0	0	0	1	0

Figure 69. Initialized linear hierarchy of alternate CLAs at t_{10}

The final time step in this example, t_{10} (illustrated in Figure 69), a new class is applied to the linear hierarchy, class 2. Furthermore, a new signal is applied to the linear hierarchy at column 7. L0, L1, and L2 have not predicted column 7 to become active, so each layer enters a burst state propagating the signal up the linear hierarchy via feedforward outputs. Finally, the new output from L1 and L2 is captured by the classifier and it increments class 2, column 7 for L1 and L2.

Experiments

To test the developed CLA, a set of experiments were devised to verify the CLA algorithm's design and evaluate the CLA's performance. Preliminary testing of the CLA was conducted on a set of synthetic test signals built dynamically by a MATLAB script. The synthetic datasets are designed to test all of the CLA's principles and behaviors as well as evaluate the CLA.

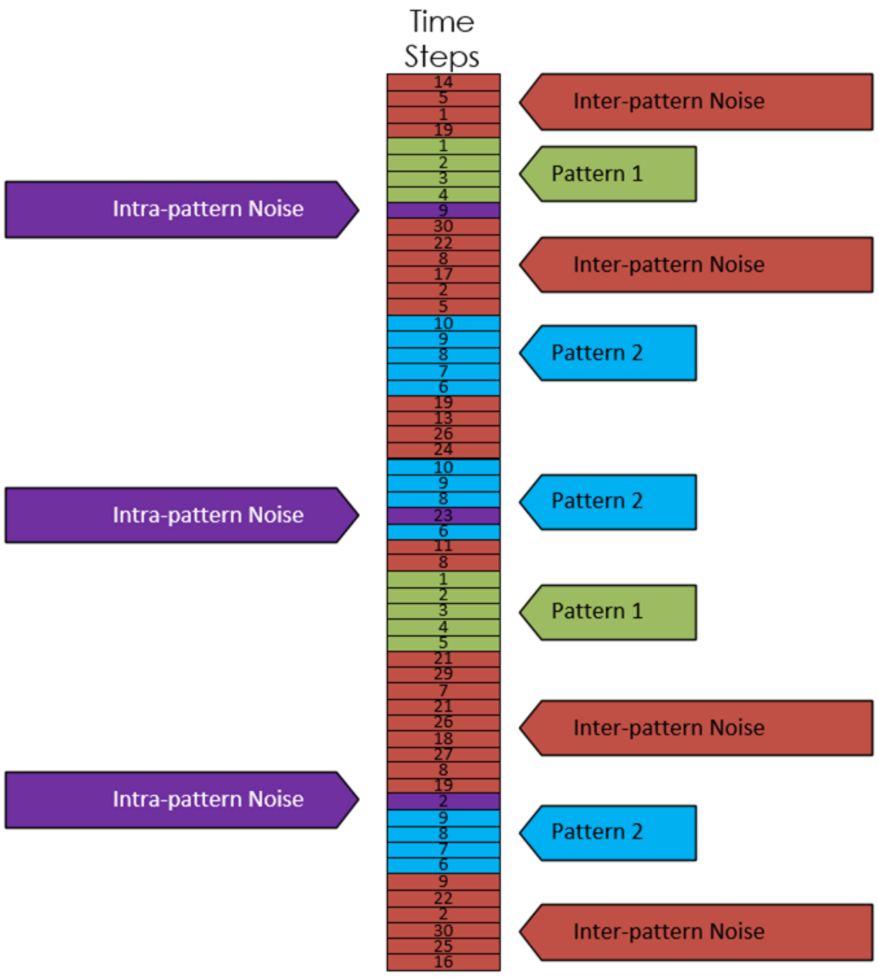


Figure 70. CLA validation synthetic dataset example

Synthetic test signals. To properly test and debug the described system, a dynamic, synthetic dataset that can produce a wide variety of signals is designed. The developed CLA accepts abstracted signals extracted from data distributions. These data distributions can represent any number of signals ranging from auditory signals represented in the frequency domain to temporal stock price data. The CLA is specifically designed to operate in the time domain. Therefore, it is essential that we emulate a sensor changing over time. The CLA semantically identifies patterns that occur over time.

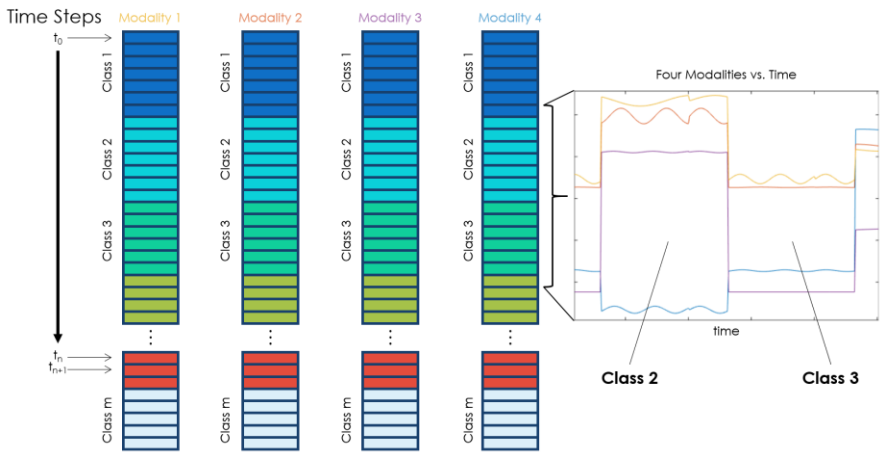


Figure 71. Example of CLA hierarchy classification synthetic dataset

CLA validation synthetic dataset. The CLA validation synthetic datasets represent patterns that occur over time with uniform random noise. The purpose of this synthetic dataset generator is to test and validate the functionality of a single Cell region. For each dataset, a known number of patterns, with a known temporal length, are devised. For example, a single dimensional dataset with 60 temporal time steps may contain 2 individual patterns that randomly occur over time. Each pattern may have 5 sequential time steps. Figure 70 displays how such a pattern is introduced to the CLA. *Note: Pattern 1 is 1, 2, 3, 4, 5 and Pattern 2 is 10, 9, 8, 7, 6 with time progressing from top to bottom.*

It is possible to corrupt the signal with varying amounts of noise. The synthetic dataset has been devised to introduce noise in two ways: inter-pattern noise, and intra-pattern noise. In Inter-pattern noise the dataset includes randomly inserted signals (with equal probability) between predefined signals.

The inter-pattern noise provides two distinct functions. It allows for the predefined signals to be temporally spaced, and simulates complete randomness that contains no important information. Therefore, we maintain a high level of inter-pattern noise in order to test if the CLA can make sense of and identify the predefined patterns, and that the CLA is not memorizing noise that contains no useful information.

The intra-pattern noise tests the robustness of the CLA. For example, if the predefined patterns are of length 5, we can randomly change one of the 5 time steps in the pattern to corrupt the signal. The significance of intra-pattern noise is demonstrated in the following sections.

CLA hierarchy classification synthetic dataset. The purpose of the CLA hierarchy classification synthetic dataset is to evaluate classification performance on an arbitrary multimodal signal containing an arbitrary number of classes. Multimodal, in the context of this thesis and neurobiology, refers to number of sensory input streams (i.e. a modality in neurobiology can refer to sensory stream such as visual, auditory, somatic, etc.). Again, the synthetic dataset is a time series - each modality containing a single vector of discrete time steps. For example, if we generate a four modality signal, the CLA hierarchy classification synthetic dataset generator creates four independent, single dimensional outputs correlated with respect to time. Figure 71, shows an example of a multimodal CLA hierarchy classification synthetic dataset containing four modalities.

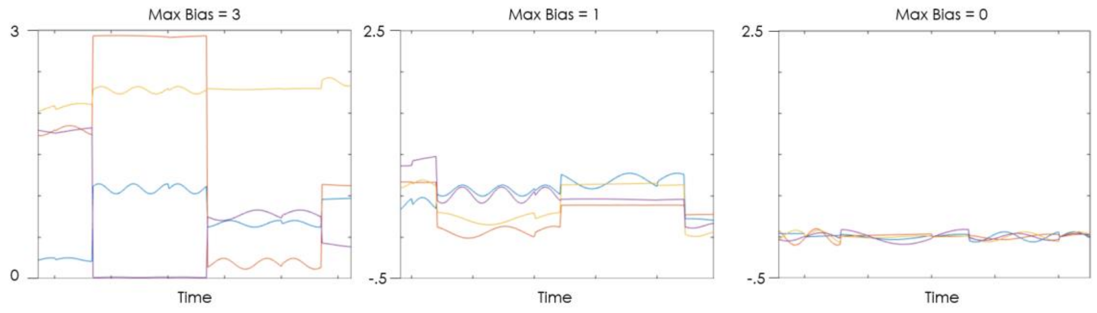


Figure 72. Illustration of varying signal bias

Each class contains at least 1 random sinusoidal signal per modality. Furthermore, the signal bias of the signal can be changed arbitrarily. When the signal bias is high, the patterns are linearly separable – i.e., trivial to classify. By varying the bias, the classification problem becomes much harder. Figure 72 illustrates the arbitrary bias variable of the CLA hierarchy classification synthetic dataset on the same scale. Therefore, when there is no bias, the only differentiating information in the signal is the sinusoidal signal.

CLA testing. This section outlines the methods and motivation behind the various tests performed to demonstrate the developed CLA core functionality. Core functionality of the CLA include temporal pattern recognition, spatial pattern recognition, and Hebbian connection construction and manipulation.

Single dimensional functionality. The CLA was designed to include both single dimensional discrete inputs (i.e., single column activation per time step) and multi-dimensional discrete inputs (i.e., multiple column activation per time step). During the single dimensional functionality testing, the CLA was subjected to the basic synthetic test signals described in previously in this chapter. The purpose of these tests is to validate the performance of the CLA when synthetic data containing single and multiple patterns are introduced – i.e., the single dimensional functionality tested how Hebbian connections are constructed and manipulated when one column is active per discrete time step.

Multi-dimensional functionality. The multi-dimensional functionality test is conducted to validate the spatial performance of the CLA when multiple columns are activated per discrete time step. Expanding on the results of the single dimensional functionality testing, the structure and performance of the CLA are assessed based on how

Hebbian connections are constructed and manipulated when multiple columns are active per discrete time step.

Name cell generation. Name cells are generated when a Cell region identifies a learned pattern. The purpose of the name cell generation experiment is to demonstrate the name cells' significance to invariant representations and name cell generation based on learned Hebbian connections.

CLA functionality. The original CLA algorithm developed for this thesis utilizes name cells for classification and feedforward outputs. The final CLA developed for this thesis does not utilize name cells in order to pass feedforward information up the cortical hierarchy – it simply passes bursting columns up a cortical hierarchy to form invariant representations. Therefore, this model is tested in a similar fashion as both the single and multi-dimensional functionality. The output of the CLA model are recorded and validated by recording when bursting occurred and recording the output of the alternate CLA. If the bursting and output are correlated with time, the test passed.

Hierarchical structuring functionality. The hierarchical structuring experiments are conducted using the alternate CLA model. The Cell regions are arranged in an ascending, hierarchical structure. The purpose of this experiment is to prove that highly variant, quickly changing signals can be characterized and passed to a higher region. As the signals are passed from lower to high level regions, the signals become increasingly invariant signals that can be used for classification and state estimation. Figure 73 represents two possible embodiments of Cell region hierarchical structures studied in this thesis.

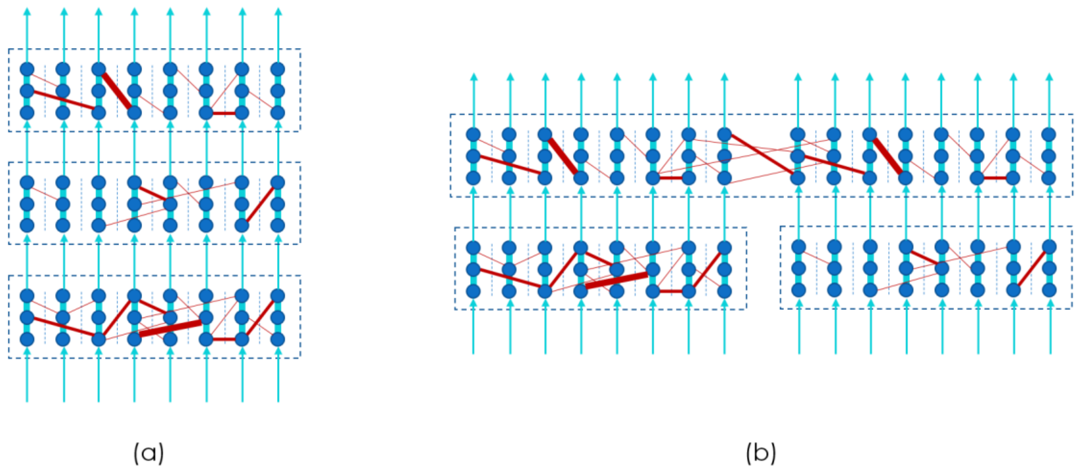


Figure 73. (a) Linear hierarchical structure (b) Pyramidal structure.

As illustrated in Figure 73the number of columns in each region are matched 1:1 as signals propegate up the hierarchy – i.e. all regions in the heirarchy contain the same number of columns as the region(s) below. Furthermore, as illustrated in Figure 73b, the pyrimidal structure allows lower region's output signals to connect via Hebbian connections – thus, drawing semantic connections across modailties. In other words, in Figure 73b, each lower level cell region represents a single modality – the region above accepts both cell regions outputs. By accepting both regions' outputs, the higher region can semantically create Hebbian connections between different modalities. The fundimental heirarchical functionality is validated by recording the inputs and outputs of each region contained in the heirarchy. Furthermore, invariance is recorded by counting how many time steps each region's output remained active.

Sparsity. The effect of CLA sparsity were tested by measuring the classification performance of a linear hierarchy of alternate CLA models on a linearly separable synthetic dataset. All variables are held constant besides the number of available columns – e.g., the same signal is tested on a hierarchy of regions containing 10 columns, 20 columns, 40 columns, 80 columns, etc. with only one column active during each time step. The performance is recorded and evaluated based on number of columns vs classification performance.

Convergence. Convergence, as it relates to the CLA, occurs when the CLA reaches a nominal burst rate. The burst rate of the CLA is the number of columns that "burst" and send their output to cell regions up the hierarchy. When the CLA burst rate is high, the CLA has poor prior knowledge of the signal being introduced to the CLA. As the CLA learns, the burst rate decreases and reaches an equilibrium – this is how we identify whether or not the CLA has converged. As expected, when novel signals are introduced to a CLA that has already converged, we see spikes in the burst rate of the CLA indicating either anomalies in the data or new classes being introduced.

Real data. One of the motivations of this thesis is to identify how the CLA can be utilized for motor control. In all controls applications, the state of the system must be identified for control of the system. For example, control of a helicopter is different during hover in ground effect as compared to forward flight at max speeds – for each state (hover and forward flight) control system gains and control signals differ. Traditionally, states of a system are identified and programmed by engineers and designers and is a very time consuming process. However, if the CLA can differentiate between multi-modal signals and identify system states in an unsupervised manner, it may be possible to utilize those same signals used for identifying system state to control the system.

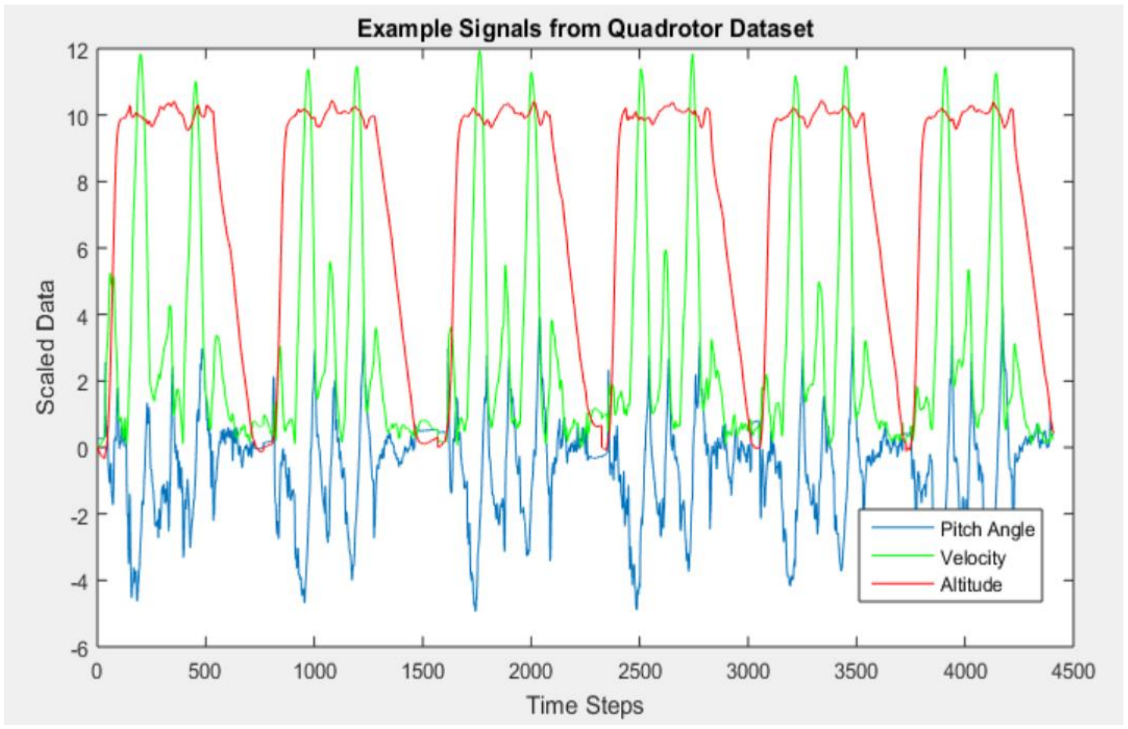


Figure 74. Example flight test data

In this experiment, we gathered flight test data from a fully automated flight performed by a PIXHAWK autopilot (example of dataset is illustrated in Figure 74). The data is then downloaded, and classified into the following states: Ground, Takeoff, Hover, Cruise, and Landing. Because the flight test data is a time series, the states are active during each maneuver up until transition to a subsequent state. Once the data is classified, the data is post processed by the CLA and the classification performance is recorded.

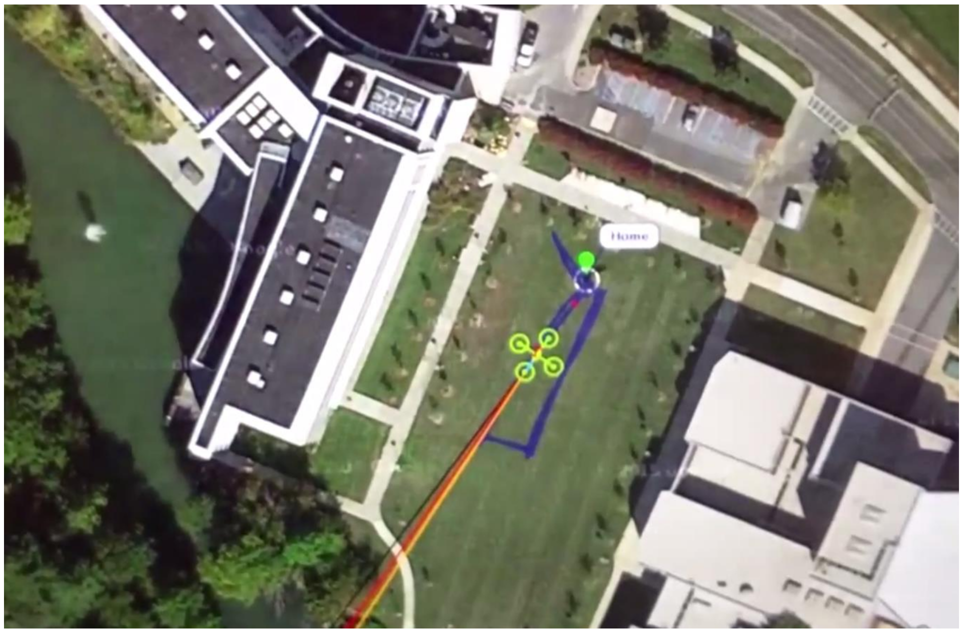


Figure 75. Feedback to ground station computer

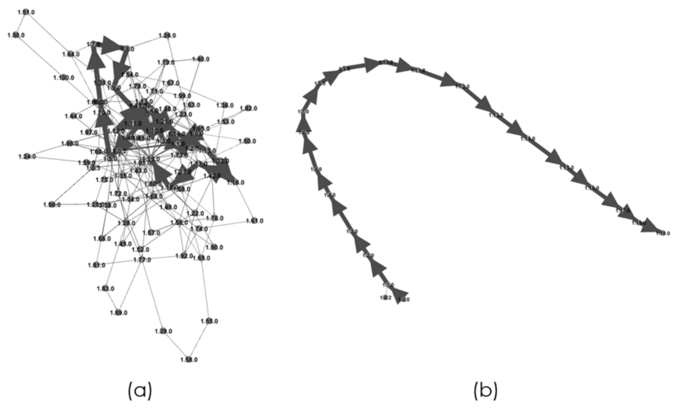
Figure 76. Quadrotor during automated flight

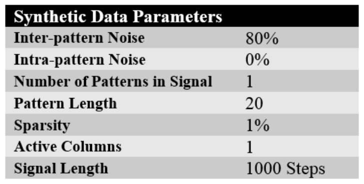
Chapter 6

Results

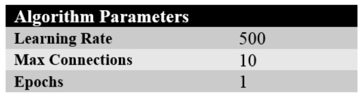
Functionality

Single pattern.





Synthetic Data Parameters	
Inter-pattern Noise	80%
Intra-pattern Noise	0%
Number of Patterns in Signal	1
Pattern Length	20
Sparsity	1%
Active Columns	1
Signal Length	1000 Steps

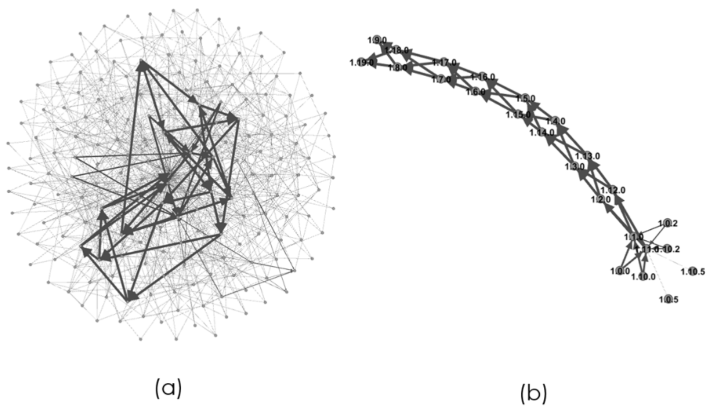


Algorithm Parameters	
Learning Rate	500
Max Connections	10
Epochs	1

Figure 77. Single pattern results

Figure 77 illustrates the CLA learning a single pattern. In this experiment, the CLA was introduced to a signal containing a single pattern. The pattern was introduced 1 out of every 5 times steps. When introduced, the entire pattern was applied to the CLA for 20 time steps. The region contained 100 columns and the pattern activated columns 1 through 20 sequentially for 20 time steps. Figure 77 a displays the CLA structure after 1 epoch of 1000 time steps. As seen in Figure 77a, the structure contains many Hebbian connections between the nodes of the structure. Furthermore, a string of very strong connections are apparent in Figure 77a. These very strong connections represent the single pattern introduced to the CLA. In Figure 77b, using Gephi, weak connections were filtered out revealing the single pattern structure within the CLA. Figure 77b shows nodes 1 through 20 with strong Hebbian connections – these Hebbian connections perfectly correlate with the pattern introduced to the CLA during this experiment. This experiment has proven to be predictable and repeatable and, thus, validates the CLA's ability to create Hebbian connections using time series data.

Spatial pattern.



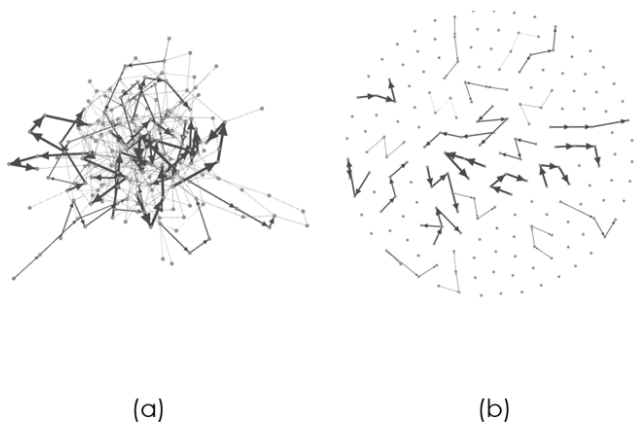
Synthetic Data Parameters	
Inter-pattern Noise	80%
Intra-pattern Noise	0%
Number of Patterns in Signal	1
Pattern Length	20
Sparsity	1%
Active Columns	2
Signal Length	1000 Steps

Algorithm Parameters	
Learning Rate	200
Max Connections	10
Epochs	1

Figure 78. Partial pattern results

In Figure 78, a pattern containing spatial data is introduced to the CLA. A spatial pattern contains more than one active column per time step – in this experiment, we chose to activate two columns. Only one pattern was introduced for test and validation. Furthermore, the likelihood of the pattern being introduced to the CLA was 1 out of 5 and the signal was applied for 1 epoch of 1000 time steps. As illustrated in Figure 78a, the CLA built very strong Hebbian connection correlating to the spatial pattern introduced to the region. Again, using Gephi, the weak Hebbian connections were filtered out revealing only the strong Hebbian connections perfectly correlating to the signal applied. In Figure 78b, there is an obvious "interweaving" occurring between the strong Hebbian connections creating a tower-like structure. The nodes at each "level" of the structure perfectly correlate to columns activated in the same time step – e.g. column 6 and 16 (represented by 1.6.0 and 1.16.0, respectively) appear next to each other because column 6 and 16 were activated in the same time step. This experiment is predictable and repeatable, thus, validating the performance of the CLA when more than one column is activated per time step.

Multiple patterns.



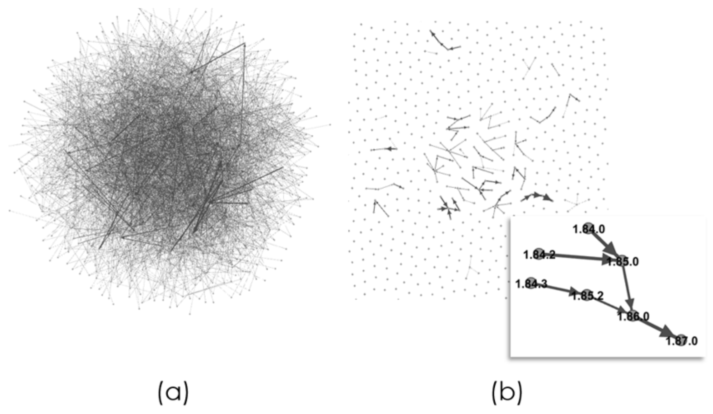
Synthetic Data Parameters	
Inter-pattern Noise	80%
Intra-pattern Noise	0%
Number of Patterns in Signal	25
Pattern Length	4
Sparsity	.5%
Active Columns	1
Signal Length	10000 Steps

Algorithm Parameters	
Learning Rate	150
Max Connections	10
Epochs	1

Figure 79. Multiple patterns results

In the multiple patterns experiment, a signal containing more than one pattern is applied to the CLA. The synthetic signal contained 25 unique patterns of length 4. In other words, there are 25 unique patterns that could be applied for four subsequent time steps. Again, there was a 1 out of 5 chance that a pattern was randomly selected and applied to the CLA, otherwise noise was introduced. Figure 79 displays the CLA structure after 1 epoch of training, 10,000 time steps. Furthermore, the CLA in this experiment contained 200 columns where only one column was active per time step. As seen in Figure 79, the CLA contains many strong Hebbian connections. After using Gephi to filter out the weak Hebbian connections and reorganizing the graph, 25 unique patterns emerge, and are illustrated in Figure 79b that perfectly correlate to the 25 signals applied to the CLA. This experiment has proven to be predictable and repeatable and, thus, validates the CLA's ability to create Hebbian connections when multiple patterns are contained in the signal.

Noisy signal.



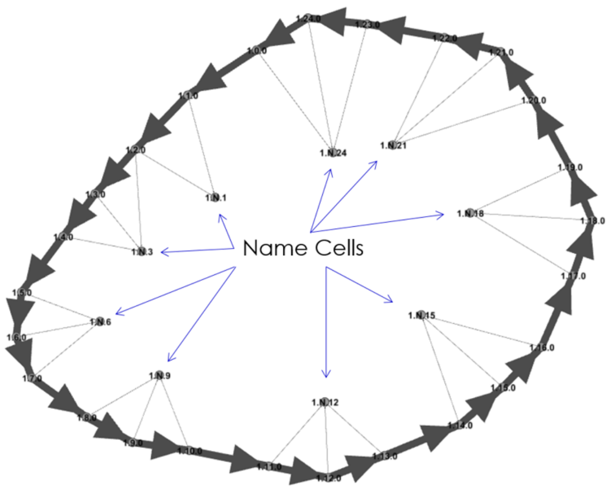
Synthetic Data Parameters	
Inter-pattern Noise	90%
Intra-pattern Noise	25%
Number of Patterns in Signal	25
Pattern Length	4
Sparsity	.5%
Active Columns	1
Signal Length	10000 Steps

Algorithm Parameters	
Learning Rate	200
Max Connections	10
Epochs	5

Figure 80. Noisy signal results

The noisy signal experiment, only with additional inter-pattern noise and intra-pattern noise applied to the synthetic signal. In this experiment, additional inter-pattern noise decreased the chances of 1 of the 25 available patterns being applied to the CLA. Furthermore, when a signal was being applied to the CLA, intra-pattern noise was introduced. In other words, there was 25% chance that when the signal was being applied, random noise would be introduced to the CLA. The resulting structure of the CLA, illustrated in Figure 80, contained many more connections than previous experiments as the CLA attempted to extract useful data out of the very noisy signal. Again, using Gephi to filter out weak Hebbian connections and reorganizing the graph, we were able to properly identify the 25 unique patterns contained in the Hebbian connections (illustrated in Figure 80b). In Figure 80b one of the patterns is highlighted to illustrate that the CLA was able to correctly identify 1 of the 25 patterns introduced to the CLA. These results predictable and repeatable and, thus, validates the CLA’s ability to identify multiple patterns contained in the signal even within a very noisy environment.

Name cells. Name cells, as described in Chapter 5 and [3], are used to group, identify, and output patterns within a CLA. In this experiment, we show a number of name cells generated when a single pattern is introduced. As shown in Figure 81, name cells are generated across strong Hebbian connections. Name cells are generated automatically, and only one name cell can represent an individual Hebbian connection between cells. Furthermore, name cells are limited to fewer than 3 total connections, thus generalizing the connections they represent. Although the pattern illustrated in Figure 81 only represents a very simple signal these result were predictable and repeatable across all synthetic signals tested and validate the CLA’s ability to build and maintain name cells for classification and hierarchical communications.



Synthetic Data Parameters	
Inter-pattern Noise	0%
Intra-pattern Noise	0%
Number of Patterns in Signal	1
Pattern Length	25
Sparsity	1%
Active Columns	1
Signal Length	1000 Steps

Algorithm Parameters	
Learning Rate	200
Max Connections	10
Epochs	1

Figure 81. Name cell creation results

Classification

Hierarchy. Classification was achieved as described in Chapter 5. This experiment tests the effects on performance by increasing the size of the hierarchy by 1) adding more columns of cell regions; and 2) increasing the number of hierarchical layers.

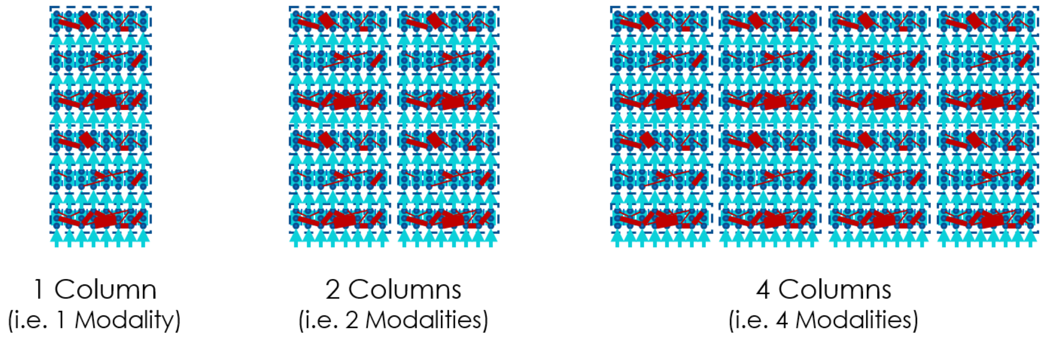


Figure 82. Increasing the number of columns

Table 9

Learning rate per hierarchical layer

Hierarchical Layer Number	Learning Rate
1	10
2	20
3	40
4	80
5	160
6	320

Figure 82 illustrates the difference between classification hierarchies of varying number of columns. Each column in the hierarchy represents a modality input to the hierarchical structure of CLAs. In this experiment, hierarchical structure of CLAs ranging from 1 column to 16 columns were tested for classification performance.

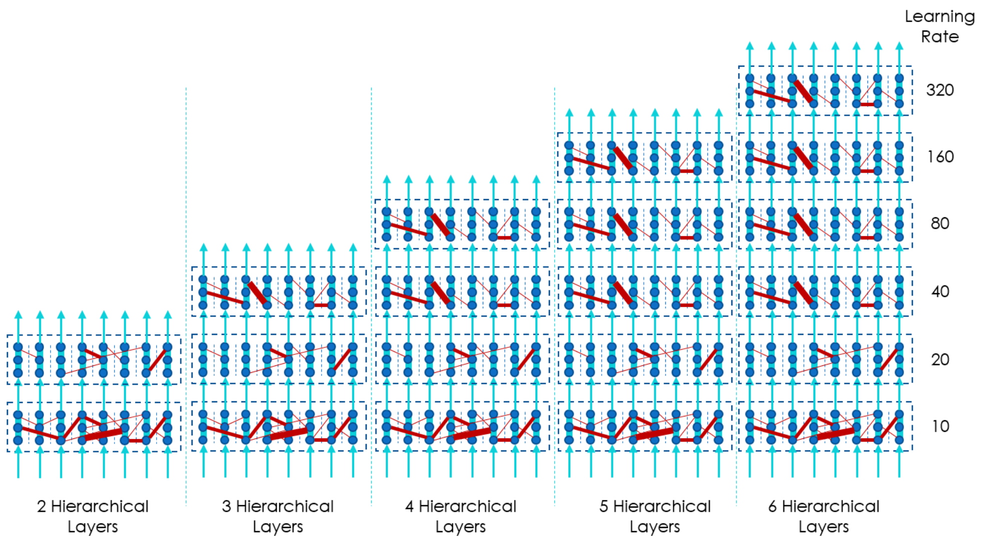


Figure 83. Increasing the number of hierarchical layers

Figure 83 illustrates the difference between hierarchical layers of varying size. In this experiment, along with varying the number of modalities, we also varied the number of layers in the hierarchy to test classification performance. As seen in Figure 83 and Table 9, the learning rate increases as we move up the hierarchy.

Figure 84 displays the synthetic test signal used in this experiment. The synthetic signal contains 30 unique classes randomly activated one after another. The dataset contains 10,000 time steps and each pattern is a length of 100 time steps. Finally, a random bias between 0 and 2 was added to each signal.

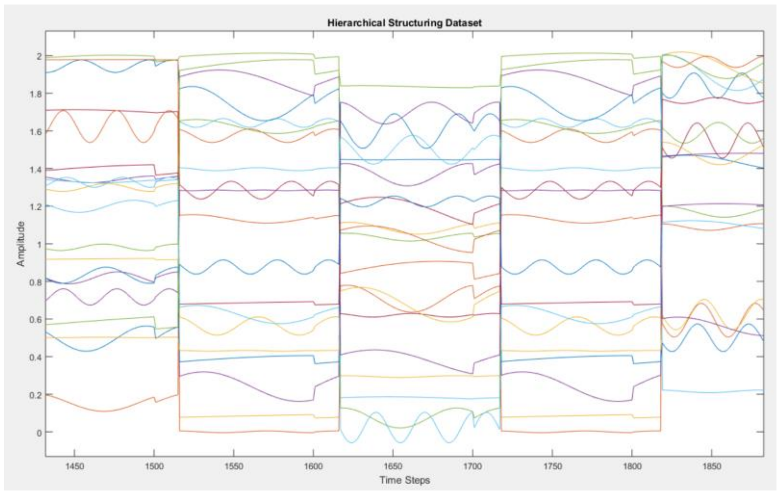
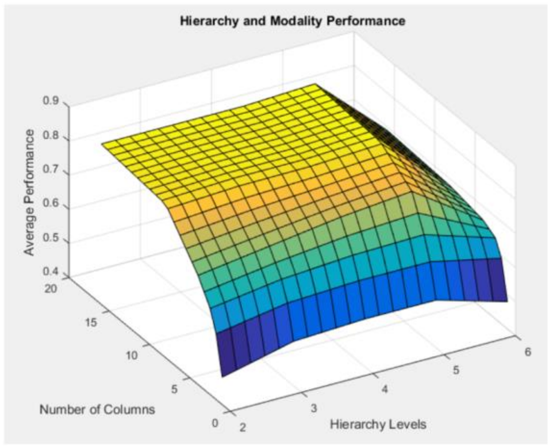
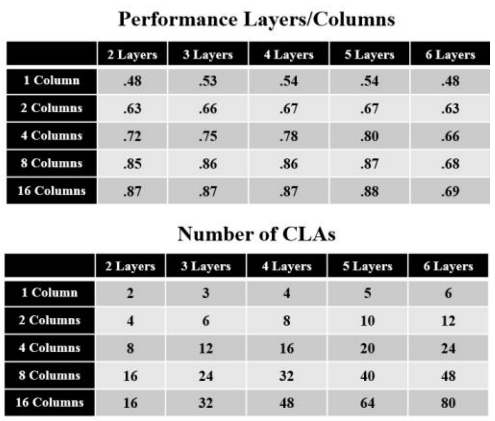


Figure 84. Hierarchical structuring dataset (subset)

For each modality, the data was normalized from 0 to 1 then binned. Each bin represents a column of cells. For example, if we initialize a cell region with 25 columns, there will be 25 bins for incoming data. When incoming data falls within one of the bins, that bin sends a signal to its respective column of cells.

The final results of the hierarchical classification can be seen in Figure 85 as a 3D mesh. Classification was accomplished using cross validation [24] – i.e., 90% of signal used for training, 10% used for assessing classification performance. As displayed in Figure 85, increasing the number of modalities increases the classification performance with diminishing returns after approx. eight modalities. Furthermore, although not as apparent, increasing the levels of hierarchy increases the performance of the hierarchy until a 6th layer was added.





Performance Layers/Columns

	2 Layers	3 Layers	4 Layers	5 Layers	6 Layers
1 Column	.48	.53	.54	.54	.48
2 Columns	.63	.66	.67	.67	.63
4 Columns	.72	.75	.78	.80	.66
8 Columns	.85	.86	.86	.87	.68
16 Columns	.87	.87	.87	.88	.69

Number of CLAs

	2 Layers	3 Layers	4 Layers	5 Layers	6 Layers
1 Column	2	3	4	5	6
2 Columns	4	6	8	10	12
4 Columns	8	12	16	20	24
8 Columns	16	24	32	40	48
16 Columns	16	32	48	64	80

Figure 85. Average performance of hierarchical structures

Sparsity. Sparsity, as described in Chapter 5, is the number of active columns vs. inactive columns. This experiment illustrates the effects of sparsity on classification of a hierarchy containing 10 columns and 3 layers totaling in 30 CLAs running synchronously. Furthermore, the synthetic signal contains only 5 classes and no bias.

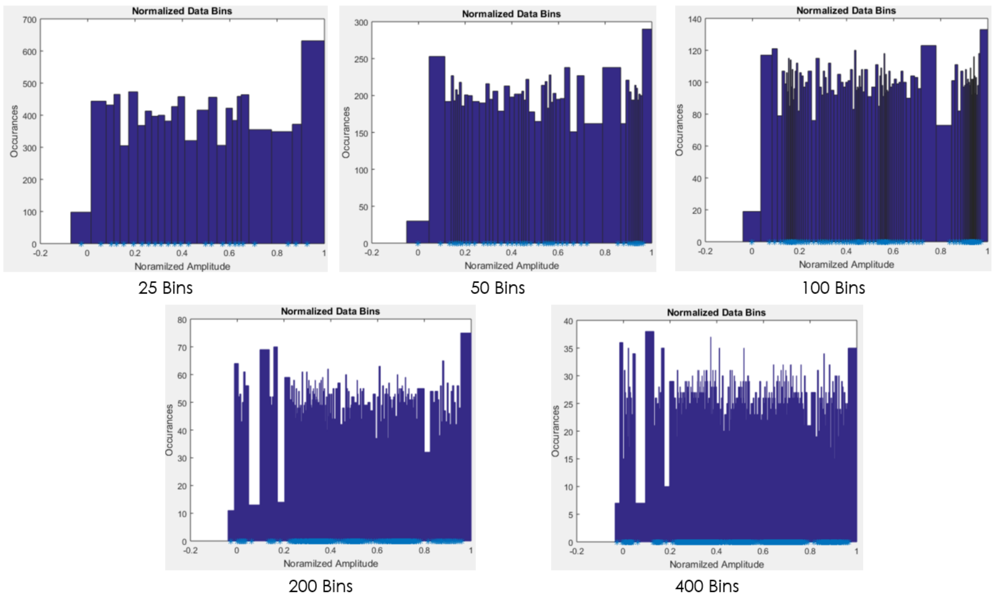


Figure 86. Varying levels of sparsity

Table 10

Level of sparsity

Number of Bins	Sparsity
25	4%
50	2%
100	1%
200	.5%

Different levels of sparsity are accomplished by changing the number of bins over a single dataset. The number of bins corresponds to the number of columns in a CLA. Because we are only activating one column per time step, we can vary the level of sparsity for a given dataset by separating the data into different numbers of bins. Figure 86 shows the same dataset, but broken in 25, 50, 100, etc. bins. The level of sparsity (because we are only activating 1 bin per time step) of each region can be found in Table 10.

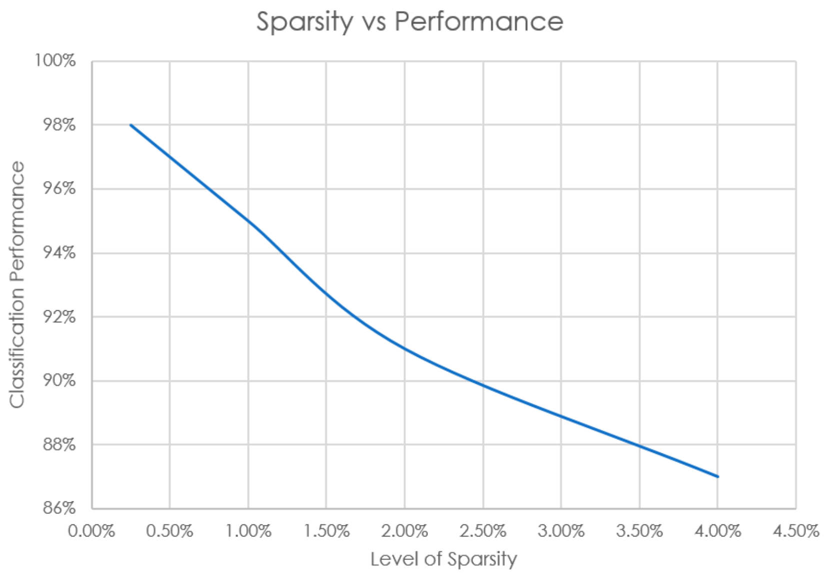


Figure 87. Sparsity vs performance

As shown in Figure 87, as level of sparsity increases, the classification performance decreases – i.e., there is an inverse relationship between level of sparsity and classification performance.

Noise. This experiment identifies the effects of corrupting a signal with random noise from a normal data distribution. In this experiment, during synthetic data generation random noise was introduced using MATLAB's built in *randn()* function. The magnitude of the random noise from a normal data distribution was adjusted to a signal to noise ratios (SNR) specified in. The synthetic dataset contained 5 classes with no bias.

For classification experiment, the hierarchy of CLAs was held constant with 1% sparsity levels, 10 modalities, and 3 hierarchical layers totaling in 30 CLAs running synchronously. The following table displays the results of classification while testing this hierarchical CLA structure against variable SNRs introduced to the hierarchical structure. No filtering was applied to the noisy signals.

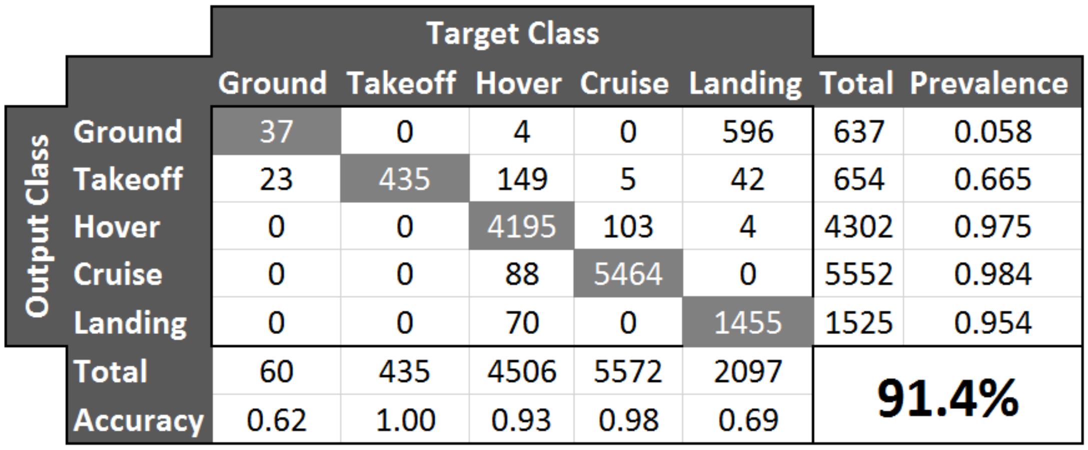
Table 11

Hierarchical classification performance with noise

SNR (db)	**Classification Performance**
INF	95%
12.0	94%
4.4	92%
1.9	79%
0.0	73%
-3.5	66%
-6.0	63%
-8.0	57%
-9.5	55%
-10.9	47%
-12.0	41%
-14.0	29%

CLA flight state identification. This section outlines the results from real flight data of a PIXHAWK controlled unmanned aerial system (UAS). The flight data classification dataset contains approximately 12,670 time steps in which 12 automated flights were executed. State information from the flight computer was recorded at 10Hz resulting in 21 minutes of classified flight data. The hierarchical classifier training was conducted in 3 epochs – i.e. the dataset was introduced 3 times to the classifier. Furthermore, training and classification were conducted concurrently during each epoch.

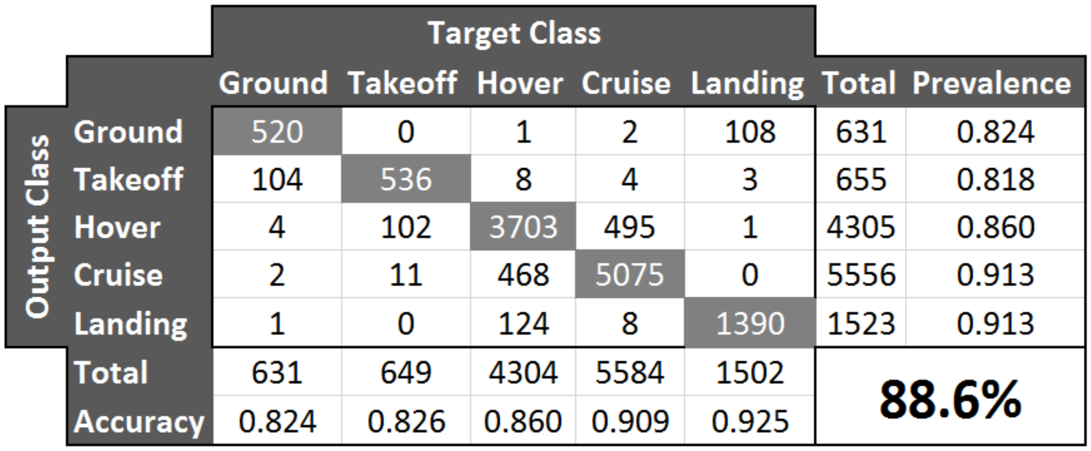
The following figure, Figure 88, displays the results of the 10x cross validation of the classification exercise using a confusion matrix. The hierarchy of CLAs was held constant with 1% sparsity levels, 8 modalities, and 3 hierarchical layers totaling in 24 CLAs running synchronously.



		Target Class						
		Ground	Takeoff	Hover	Cruise	Landing	Total	Prevalence
Output Class	Ground	37	0	4	0	596	637	0.058
Output Class	Takeoff	23	435	149	5	42	654	0.665
Output Class	Hover	0	0	4195	103	4	4302	0.975
Output Class	Cruise	0	0	88	5464	0	5552	0.984
Output Class	Landing	0	0	70	0	1455	1525	0.954
	Total	60	435	4506	5572	2097	91.4%	
	Accuracy	0.62	1.00	0.93	0.98	0.69		

Figure 88. Confusion matrix of flight data classification using CLA

NuPIC flight state identification. The purpose of this section is to quantify the classification performance of the flight state identification dataset using the open source hierarchical learning algorithm NuPIC, supported by Numenta. The performance of NuPIC in this experiment is used for a direct comparison between the performances of the CLA versus the performance of a similar, commercial-off-the-shelf hierarchical learning algorithm. Default classification parameters for NuPIC used in this experiment based are off of the Online Prediction Framework (OPF) examples in NuPIC version 0.5.7. The NuPIC OPF provides the closest comparison to CLA developed for this thesis.



		Target Class						
		Ground	Takeoff	Hover	Cruise	Landing	Total	Prevalence
Output Class	Ground	520	0	1	2	108	631	0.824
	Takeoff	104	536	8	4	3	655	0.818
	Hover	4	102	3703	495	1	4305	0.860
	Cruise	2	11	468	5075	0	5556	0.913
	Landing	1	0	124	8	1390	1523	0.913
	Total	631	649	4304	5584	1502	88.6%	
	Accuracy	0.824	0.826	0.860	0.909	0.925		

Figure 89. Confusion matrix of flight data classification using NuPIC

Chapter 7

Discussion

This thesis has presented the fundamental principles behind cortical learning algorithms, which are based on the basic cortical interactions of the mammalian brain. The CLA developed for this thesis primarily models the supergranular layers of the neocortex (Layers II and III) where patterns are believed to be stored and recalled. By logically stacking the cell regions into a hierarchy, low-level signals can be abstracted into higher-level patterns that demonstrate both invariance (in respect to time) and noise rejection from noisy signals. Furthermore, by using feedforward signals, a hierarchy of CLAs is able to classify patterns in a supervised learning scheme.

As confirmed in Chapter Chapter 6, the CLA developed for this thesis performs as described in Chapter Chapter 5 and demonstrates the pattern recognition properties explained in Section 0. In Section 0, the described CLA demonstrates temporal and spatial detection, storage, and prediction of patterns in an unsupervised learning scheme. The CLA resists catastrophic forgetting by creating permanent Hebbian connections after γ, the permanence threshold, has been reached. Data fusion of various modalities is combined using two different yet complimentary methods; 1) combination of generalized column activations mimicking action potentials via driver signals originating from the thalamus and 2) combination of feedforward "bursting" signals with feedback class inputs during supervised learning. The combination of these two methods semantically associate different modalities with respect to time. The CLA is able to identify anomalies as bursting columns – i.e. when the CLA does not predict an input, its model is violated which indicates an anomaly may have been identified. Finally, during hierarchical classification, new classes can be added dynamically to the CLA models.

In Section 0, we used classification performance as a metric to assess the properties of hierarchical structures of CLAs. As seen in Figure 85, for the 30 class dataset introduced to the hierarchy of CLAs, the hierarchical structure's performance is greatly influenced by the number of columns of CLA's included in the structure. Furthermore, increasing the levels of hierarchy (i.e. stacked CLAs per column) increased the performance of the overall hierarchical structure as well, although the effect of adding more levels in the hierarchy did not have as strong of an effect as increasing the number of columns or modalities. In fact, after 6 levels were introduced to the hierarchy, the structure started to perform poorly. We believe that the degraded performance was caused by overlearning – i.e. because the learning rate increases as we move up the hierarchy, the chances of overlearning increases. Therefore, we conclude that the classification performance is highly dependent of the learning rate of the CLAs.

The effects of CLA sparsity was measured via hierarchical classification performance and recorded in Chapter 6. As shown in Figure 87, there is an inverse relationship between classification performance and level of sparsity – i.e. as the ratio of active columns to non-active columns increases, the performance of the classification task decreases. In the testing conducted for this thesis, we observed up to 10% decrease in classification performance when a ratio of 1:25 active vs inactive columns was tested. Furthermore, we observed an obvious increase in the rate of speed at which algorithm runs when the sparsity level increased – i.e. 1:25 active vs inactive columns runs faster than 1:100 active vs inactive columns. This finding, although not quantified in this thesis, makes sense - the computational complexity increases as we add more columns to the CLA, thus increasing the time required for computing the CLA per time step. We theorize that, depending on the application, there is an optimal computation speed vs classification performance (illustrated in Figure 90) for real-time applications – i.e. an optimal sparsity level at which the CLA should be initialized for real-time applications.

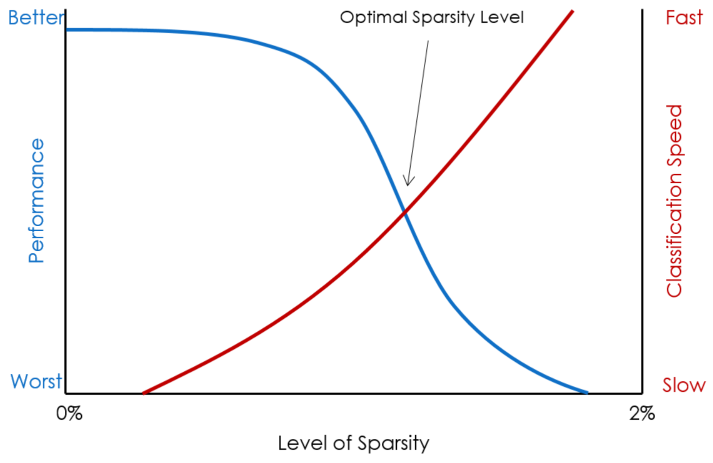


Figure 90. Example of speed vs performance

In Chapter 6, we assessed the hierarchy of CLAs performance when normally distributed noise was introduced at varying signal to noise ratios (SNRs). As displayed in Table 11, the hierarchy of CLAs, predictably, degraded noticeably as lower SNRs were applied to the signals. The control dataset’s classification performance was 95%. The probability of correct classification due to random guessing was .2 for all datasets testing in the noisy signal experiment. As the SNR decreased, the hierarchy of CLAs approached the probability of random guessing. At 0dB SNR (noise level = signal level) the performance dropped 20% off of the control dataset's classification performance. At -6dB SNR (noise level was twice that of the signal) the hierarchy of CLA's classification performance was observed at 63% - approximately 43% higher than random guessing. Although this experiment suggests that hierarchies of CLAs demonstrate resistance to noise, more testing should be conducted to confirm a similar results on both real and synthetic datasets with and without filtering. Furthermore, it is yet to be seen if this resistance to noise demonstrated by the hierarchy of CLAs will be beneficial in real-world applications.

In Chapter 6, we attempted to classify real flight state from an unmanned aerial system (UAS). As displayed in the confusion matrix (Figure 88), the system correctly identified the state of the UAS with 91.4% accuracy overall of the 5 class time series dataset. The dataset contained unbalanced examples of each of the 5 classes. The bias of the unbalanced datasets were obvious from the results – i.e. the class with the most data "Cruise" performed best at 98% correct classification whereas "Ground" and "Landing" performed at 62% and 69% correct classification, respectively.

When the results of the real flight data using the CLA are compared to NuPIC for flight state classification, we observe a small overall classification performance increase using the CLA versus NuPIC – CLA performance is 91.4% whereas NuPIC performance is 88.6%. Although the CLA performed better overall, NuPIC performed significantly better than the CLA during the ground state (the ground state has the least number of samples in the dataset). The CLA excelled in identifying the state of the system during transitions while NuPIC was still identifying the transition. An example of where the CLA identified a class transition from hover (class 3) to forward flight (class 4) faster than

NuPIC is shown in Figure 91. Clean transitions from one class to another class increased the overall 10x cross validation of the CLA over the NuPIC despite NuPIC's superior performance with unbalanced data.

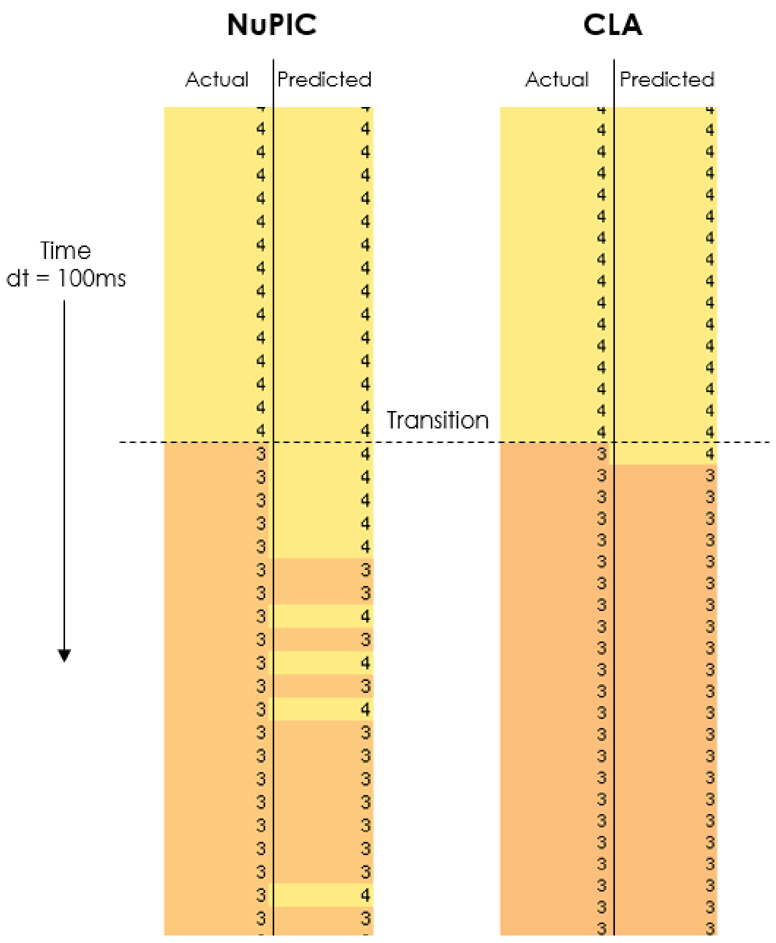


Figure 91. Class transition of NuPIC and CLA

The functionality demonstrated by the CLA is accomplished using a single fundamental algorithm, described in Section 0 and demonstrated in Chapter 6. The CLA – which incorporates online learning, sparsity, hierarchy, and signal agnostics – provides anomaly detection, noise reduction, invariant representations, data fusion and resistance to catastrophic forgetting in a single algorithm. There is still much more research to be completed before CLA algorithms will be able to perform at levels comparable to the CLA's closest machine learning and pattern recognition relative, deep belief networks (DBNs). With that said, cortical learning algorithms of the future provide promising results as they share the benefits of DBNs, yet operate in an online learning fashion and are designed to incorporate the feedback channels necessary for motor control in future robotic applications.

The CLA belongs on highly parallelized computer architectures. The execution of a discrete time step can be optimized to run in parallel which would greatly increase the speed that the algorithm can run. Moreover, when running multiple CLAs in a hierarchical structure, it is possible to distribute the CLAs onto multiple threads, multiple cores, or even multiple machines. This thesis's implementation of the CLA can run multiple CLAs in a hierarchy on multiple threads allowing discrete time steps of the hierarchy to run more efficiently on multithreaded processors. In retrospect, the CLA should be implemented for highly parallel hardware architectures (such as GPUs). Furthermore, because most of the nodes of the CLA are in an inactive state at any given time during the CLA's operation, a lot of energy is wasted maintaining the state of the CLA in memory and performing calculations in a serial fashion on inactive members of the cell regions. Although each CLA runs in a single thread, the hierarchy of CLAs prove to perform quickly when running as a hierarchy of synchronized, multithreaded CLAs – suggesting that many (e.g. tens of thousands) CLAs arranged in hyper-modal, deep hierarchies is feasible on computing clusters and/or supercomputers.

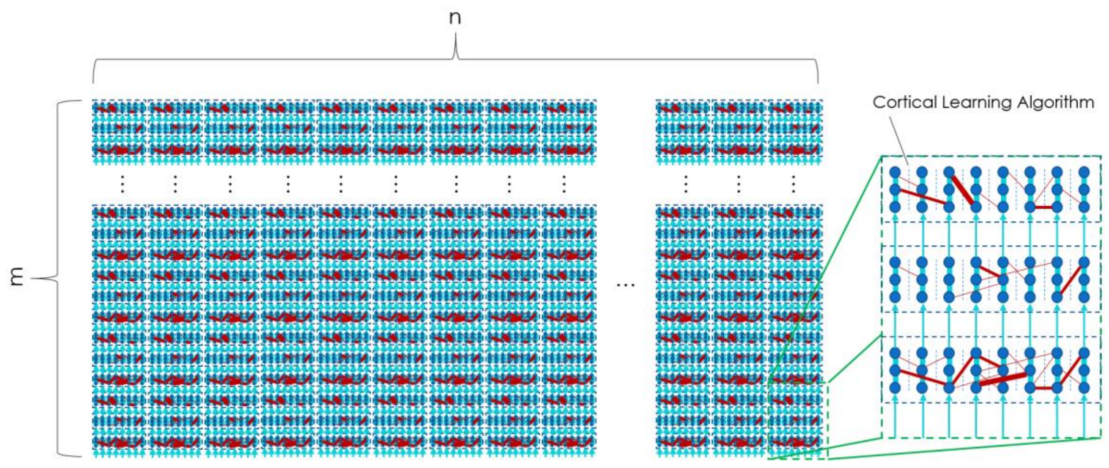


Figure 92. Example of arbitrarily large hierarchy of CLAs

The figure above, Figure 92, an *n* column by *m* row hierarchy of CLAs is illustrated as an example of an arbitrarily large hierarchical structure. The columns represent *n* modalities and rows represent *m* layers or “deepness” of the hierarchy (i.e. the deepness of the hierarchy increases with quantity of layers). In this configuration, all CLAs are synchronize to the current discrete time step. As previously mentioned, this thesis strongly suggests that the example synchronized hierarchical structure is feasible on computing clusters and/or supercomputers. It is worth noting that the architecture illustrated in Figure 92 is only one of many possible architectures of CLA hierarchies – i.e. the architecture for motor control or feature extraction may be much different than the architecture used for classification in this thesis.

The CLA developed for this thesis explores various properties of neurobiology inspired cortical learning algorithms. Future work on cortical learning algorithms must consider new neurobiology discoveries in the feedback paths of the brain, interactions between the thalamus and neocortex, and neocortex and the hippocampus. Also, future work on cortical learning algorithms must focus on hippocampal and thalamic functionality as new neurobiology discoveries on the brain regions are published. It is important to note that neurobiological discoveries are accelerating to create a better understanding of the brain every year, therefore researchers in this field must keep a close eye on emerging findings in neurobiology to include into their cortical learning algorithms. Furthermore, as neurobiology research evolves and the need for increasingly intelligent machines continues, new hardware architectures will emerge to address the cortical learning algorithms properties. An example of this push to build hardware that is applicable to highly neurobiologically inspired algorithms is DARPA’s (Defense Advanced Research Projects Agency) SyNAPSE (Systems of Neuromorphic Adaptive Plastic Scalable Electronics) project [52]. The goal of the project is to create hardware that achieves power savings for neural systems by utilizing sparse hardware resources while processing large volumes of information, much like the mammalian neocortex. These new hardware architectures can be utilized to produce extremely powerful and efficient cortical learning algorithms in future research.

Finally, while researching neurobiological algorithms and DBNs, it became apparent that much of the research was devoted to tackling problems such as image recognition, high-level natural language processing, and other complex perceptual applications. This focus on complex perceptual problems has yielded groundbreaking results that continue to improve upon the state of the art of pattern recognition and machine learning, yet one can argue that these techniques are not addressing the real goal of machine learning and pattern recognition fields. One can argue that the goal of machine learning and pattern recognition is to produce methods, algorithms, etc. to model and enable artificially intelligent agents capable of learning behaviors without prior knowledge and little human intervention. Furthermore, it has been hypothesized that the nervous system evolved specifically to address movement of an organism and three-dimensional navigation [54]. If that hypothesis holds true, cortical learning algorithms of the future should be designed to address feedforward sensory information in order for the agent to provide feedback commands to motor centers for movement in three dimensional space – i.e. pattern recognition and machine learning researchers should focus research on enabling artificially intelligent agents capable of learning complex movements without prior knowledge and little to no human intervention using cortical learning algorithms. I hypothesize that by focusing on movement and three dimensional navigation, intelligent behavior and complex perceptual understanding of the agent's surrounding world will emerge, just like with the evolution of the nervous system in biological systems. The emergence of intelligent behavior and complex perceptual understanding from movement and three dimensional navigation may lead to image recognition, high-level natural language processing, and other complex perceptual applications.

The brain is one the most complex structures we have ever studied. It has continued to perplex neurobiology researchers for centuries and is the only source of intelligence in the known universe. Furthermore, the brain is our only obvious proof of concept for intelligence. Biological evolution, over billions of years, has built a machine capable of extraordinarily hard perception and control tasks. The nervous system of most species in the animal kingdom is responsible for regulation and control of trillions of individual cells. Beyond regulation and control, the nervous system (more specifically the brain) is capable of forming complex memories, socializing, performing incredibly fine motor commands, etc. As machine learning and pattern recognition researchers, it is necessary to better understand the neurobiology field as the field continues to grow due to rapidly improving brain imaging technologies in pursuit of artificially intelligent agents.

Contributions and Future Work

This thesis has yielded an experimental cortical learning algorithm (CLA) based – primarily - on Jeff Hawkins et al. ([5] [3]) theory of neocortical operation as well as neurobiologist SM Sherman et al. ([11] [12]) thalamus research. Through synthetic dataset validation and verification exercises on the developed CLA, this thesis has demonstrated:

1. A fast, simple cortical learning algorithm without the use of spatial and temporal pooling, and without the use of name cells.
2. Effects of hierarchical structuring of CLAs on classification tasks.
3. Effects of CLA sparsity on classification tasks.
4. CLA classification via arbitrarily large *n* by *m* hierarchical structure of CLAs.

Furthermore, this thesis contributes the following to the field of pattern recognition and artificial intelligence:

1. An open .NET C# CLA library for use future applications.
2. Novel approach to passing information up the CLA hierarchy derived from neurobiology research [11].
3. Theory of how cortical learning algorithms may be applied to motor control applications.

Finally, this thesis's purpose is to supply the reader with all necessary information needed to learn, adapt, and/or design their own algorithm for experimentation in this emerging neurobiologically inspired field of cortical learning algorithms. We would like to expand on this work in the future by focusing on the following items:

1. Apply the current hierarchy of CLAs architecture to real-world time series classification problems.
2. Implement basic motor control applications using the CLA via a reinforcement learning.
3. Quantify the performance of the CLA on time series feature extraction datasets.
4. Compare the performance of the developed CLA classification algorithm against a variety of classifiers.
5. Increased focus on thalamic and hippocampal interactions with the neocortex via emerging neurobiology research.
6. Create theories and methods to automate CLA parameter tuning (e.g. generalized convergence of CLA learning rate).
7. Automate and quantify hierarchical structuring architectures for a variety of applications (e.g. classification, feature extraction, data mining, etc.).
8. Rebuild the CLA for parallel processors – i.e. computer clusters, GPUs, FPGAs, neuromorphic chips, etc.

Conclusion

The current state of neurobiology research, due to advancements in *in vivo* brain imaging and other *in vivo* techniques, has enabled a detailed view into the operation of the brain. This has allowed researchers to utilize the many decades of brain structure research and understand how the various structures interact. Understanding these complex interactions throughout the various structures of the brain are, likely, key to understanding intelligence. Moreover, a balance between research, theory and implementation of these biologically inspired algorithms must be maintained to continue progression the field. Although studying the brain and brain function may open doors to understanding the basis of biological intelligence, the brain is not an end-all general learning algorithm. Although the CLA is founded on neurobiology research, the no-free-lunch theorems [2] holds true for the CLA and, ultimately, any neurobiology inspired algorithms. Further research into cortically inspired algorithms should focus on the useful tasks that most animals excel at - perception and control. It is important to note that all intelligent behavior (such as planning, reasoning, language and understanding) is, plausibly, emergent from perception and control - therefore, perception and control should be the focus of cortically inspired future research.

This research has concluded that, at a basic level, interactions between the supragranular layers of the neocortex (Layers II and III) can store semantic meaning, feedback, auto-correlate patterns, and produced signals to be used for classification. Furthermore, the CLA algorithm can add new classes in real time, detect anomalies, and natively achieve data fusion while resisting sensitivity to noise and reducing vulnerability to catastrophic forgetting. Surprisingly, all the aforementioned properties and applications of the CLA are achieved without modification to the basic cortical learning algorithm. Finally, this research has quantified hierarchical interactions between CLAs when arranged in logical hierarchies of CLAs while understanding best practices when constructing a CLA hierarchy.

References

[1] V. B. Mountcastle, An Organizing Principle for Cerebral Function, Cambridge, MA: MIT Press, 1978.

[2] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization.," *Evolutionary Computation, IEEE Transactions,* pp. 67-82, 1997.

[3] J. Hawkins and S. Blakeslee, On Intelligence, New York: Times Books, 2004.

[4] Numenta, Inc., "Hierarchical Temporal Memory including HTM Cortical Learning Algorithms," 2011.

[5] D. George, "How The Brain Might Work: A Hierarchical And Temporal Model For Learning And Recognition," Stanford University, Stanford, CA, 2008.

[6] Y. Bengio et al., "Greedy layer-wise training," *Advances in Neural Information Processing Systems,* p. 19, 2007.

[7] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards AI," *Large-scale kernel machines,* pp. 1-41, 2007.

[8] E. A. M. Demis Hassabis, "Deconstructing episodic memory with construction," *Trends in cognitive sciences,* vol. 11, no. 7, pp. 299-306, 2007.

[9] Q. V. Le et al., "Building High-level Features Using Large Scale Unsupervised Learning," in *International Conference on Machine Learning*, Edinburgh, Scotland UK, 2012.

[10] Numenta, Inc., "Breakthrough Science of Anomally Detection: Grok," Numenta, Inc., 8 Nov 2014. [Online]. Available: http://numenta.com/grok/. [Accessed 8 Nov 2014].

[11] S. M. Sherman, "Thalamocortical Interactions," *Current Opinion in Neurobiology,* pp. 22:575-579, 2012.

[12] S. M. Sherman and R. W. Guillery, Exploring the Thalamus and its Role in Cortical Function, Cambridge, MA: MIT Press, 2006.

[13] R. W. Williams and K. Herrup, "The control of neuron number," *Annual Review of Neuroscience,* vol. 11, no. 1, pp. 423-453, 1988.

[14] BruceBlaus, Artist, *Neuron.* [Art]. Web.

[15] D. Cooper, "The significance of action potential bursting in the brain reward circuit," *Neurochemistry international,* vol. 41, no. 5, pp. 333-340, 2002.

[16] Looie496, Artist, *Vertebrate-brain-regions.png [Public domain].* [Art]. Wikimedia Commons, 2014.

[17] M. Sur et al.: The Role of Patterned Activity in Development and Plasticity of Neocortical Circuits, Cambridge, MA: MIT Press, 1999.

[18] W. H. Calvin, Cortical Columns, Modules, and Hebbian Cell Assemblies, Cambridge, MA: MIT Press, 1995.

[19] *Gray754.* [Art]. Wikimedia Commons.

[20] Y. B. Saalmann and S. Kastner, "Cognitive and Perceptual Functions of the Visual Thalamus," *Neuron,* no. 71, pp. 209-223, 2011.

[21] Y. B. Saalmann, "Intralaminar and medial thalamic influence on cortical synchrony, information transmission and cognition," 2014.

[22] R. W. Guillery and S. M. Sherman, Branched thalamic afferents: what are the messages that they relay to cortex?, Chicago: Brain Research, 2011.

[23] R. Polikar, "Machine Learning and Pattern Recognition Lectures," 2013.

[24] H. &. S. Duda, "Pattern Classification, 2/e Wiley," 2000.

[25] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 1998.

[26] G. E. Hinton and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation,* 2006.

[27] M. Minsky and S. Papert, Perceptrons, Cambridge, MA: MIT Press, 1969.

[28] D. E. Rumelhart et al., "Learning Internal Representations by Error Propagation," *Parallel distributed processing: Explorations in the microstructure of cognition,* vol. 1, no. 1986, 1986.

[29] D. Michie et al, Machine Learning, Neural and Statistical Classification, 1994.

[30] G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning,* pp. 257-277, 1992.

[31] G. E. Hinton, "Learning multiple layers of representation," *TRENDS in Cognitive Sciences,* vol. 11, no. 10, pp. 428-434, 2007.

[32] Brainmind, "brainmind.com," brainmind.com, 2010. [Online]. Available: http://brainmind.com/TemporalLobes.html. [Accessed 25 February 2015].

[33] D. Wang, "Temporal Pattern Processing," *The Handbook of Brain Theory and Neural Networks,* vol. 2, pp. 1163-1167, 2003.

[34] N. Kasabov et al., "Dynamic evolving spiking neural networks for on-line spatio- and spectro-temporal pattern recognition," *Neural Networks,* vol. 41, pp. 188-201, 2013.

[35] L. Baldassarre et al., "Structured Sparsity Models for Brain Decoding from fMRI data," in *2012 International Workshop on Pattern Recognition in NeuroImaging (PRNI),* London, 2012.

[36] J. Wright et al., "Sparse Representation For Computer Vision and Pattern Recognition," *Proceedings of the IEEE,* vol. 98, no. 6, pp. 1031-1044, 2010.

[37] A. A. Salah et al., "A selective attention-based method for visual pattern recognition with application to handwritten digit recognition and face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 3, pp. 420-425, 2002.

[38] J. R. Newton and M. Sur, "Rewiring Cortex: Functional Plasticity Of The Auditory Cortex During Development," *Plasticity and Signal Representation in the Auditory System,* pp. 127-137, 2005.

[39] J. Newton et al., "Developmental Studies on Rewiring the Brain: What They Tell Us about Brain Evolution," *Evolution of Nervous Systems,* vol. 3, pp. 103-112, 2007.

[40] B. Parsons et al., "Motor-sensory recalibration modulates perceived simultaneity of cross-modal events," *Frontiers in Psychology,* pp. 4-46, 2013.

[41] G. A. Carpenter and S. Grossberg, "The ART of Adaptive Pattern Recognition," *Computer,* vol. 21, pp. 77-88, 1988.

[42] F. Castanedo, "A Review of Data Fusion Techniques," *The Scientific World Journal,* vol. 2013, p. 19, 2013.

[43] Y. Hu and P. Loizou, "A comparative intelligibility study of single-microphone noise redution Algorithms," *The Journal of the Acoustical Society of America,* pp. 1777-1786, 2007.

[44] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks,* vol. 51, no. 12, pp. 3448-3470, 2007.

[45] T. R. Hoens and N. V. Chawla, "Learning in Non-stationary Environments with Class Imbalance," in *Knowledge Discovery and Data Mining,* Beijing, 2012.

[46] G. Ditzler et al., "Learning in Nonstationary Environments: A Survey," *IEEE Computational Intelligence Magazine,* vol. 10, no. 4, pp. 12-25, 2015.

[47] H. Alitto and W. M. Usery, "Corticothalamic feedback and sensory processing," *Current Opinion in Neurobiology,* no. 13, pp. 440-445, 2003.

[48] Gephi 0.8.2, "Open-source network analysis and visualization software package," 2014.

[49] Association for Computing Machinery, "Data Mining Curriculum," Association for Computing Machinery's Special Interest Group.

[50] H. Lee et al., "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations," in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

[51] H. B. Project, "Brain Simulation Platform - Human Brain Project," Human Brain Project, European Commission, 2013. [Online]. Available: https://www.humanbrainproject.eu/brain-simulation-platform1. [Accessed 2015].

[52] DARPA, "SYSTEMS OF NEUROMORPHIC ADAPTIVE PLASTIC SCALABLE ELECTRONICS (SYNAPSE)," DARPA, 2014. [Online]. Available: http://www.darpa.mil/Our_Work/DSO/Programs/Systems_of_Neuromorphic_Adaptive_Plastic_Scalable_Electronics_%28SYNAPSE%29.aspx.

[53] F. Matyas et al., "Motor Control by Sensory Cortex," *Science,* vol. 330, no. 26 November 2010, pp. 1240-1243, 2010.

[54] D. M. Wolpert et al., "An Internal Model for Sensorimotor Integration," *Science,* vol. 269, no. 29 September 1995, pp. 1880-1882, 1995.

[55] M. Mohri, R. Afshin and T. Ameet, Foundations of Machine Learning, Cambridge, MA: MIT, 2012.

[56] T. M. Cover, "Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition.," Cover, T.M..