Sensory Memory for Grounded Representations in a Cognitive Architecture

Pulin Agrawal PAGRAWAL@MEMPHIS.EDU
Department of Computer Science, University of Memphis, Memphis, TN 38152 USA

Stan Franklin FRANKLIN.STAN@GMAIL.COM
Institute for Intelligent Systems, University of Memphis, Memphis, TN 38152 USA

Javier Snaider JAVIERSNA@GMAIL.COM
Department of Computer Science, University of Memphis, Memphis, TN 38152 USA

Abstract

Vector LIDA was proposed in 2014 as major overhaul to the representation system in the LIDA model. It replaced the nodes and links representation system used earlier with Modular Composite Representation (Modular Composite Representation) vectors for representing knowledge in LIDA. Although, it was mostly trivial to swap out nodes and links with these vectors, as described in the Vector LIDA paper, it was not trivial to obtain these vectors from sensors so that symbol grounding is maintained. In this paper, we propose a sensory memory system and a method for obtaining Modular Composite Representation vectors from it in a way that enables symbol grounding. We propose to build sensory memory out of Hierarchical Temporal Memory's Cortical Learning Algorithm's regions, an online learning algorithm, which produce sparse distributed representations of spatio-temporal patterns recognized from its stimulus from the environment. These sparse distributed representations can then be translated into Modular Composite Representation vectors in a way that preserves the semantic meaning encoded in these sparse representations, and thus enable the LIDA's vector representations to be grounded by the sensors in the environment. One of the main commitments of the LIDA model is that "all learning happens via consciousness". To allow Cortical Learning Algorithms to learn and adapt to the environment and still honor this commitment, we need to translate the contents of the conscious broadcast to sparse distributed representations, so that Cortical Learning Algorithms can learn based on consciousness. To summarize, we propose a Cortical Learning Algorithms based Sensory Memory for the Vector LIDA and produce a method of translating back and forth between the representations used by Cortical Learning Algorithms, i.e. sparse distributed representations, and the representations used by Vector LIDA, i.e. Modular Composite Representation vectors. This will allow agents based on Vector LIDA to have grounded symbols for representation.

1. Introduction

"Once a problem is described using an appropriate representation, the problem is almost solved" (Winston, 1992). Many authors have stressed the importance of well-chosen representations in trying to solve a problem or perform a task. For a cognitive architecture the task of choosing the right representation system becomes crucial, because its agents may be expected to represent a variety of complex forms of information and to perform an equally varied set of tasks. The importance of representation for symbolic and connectionist cognitive architectures has been discussed in more detail in Franklin (1997). Sun (1999) talks about the need to integrate the two most popular forms of representations, symbolic and connectionist, to reap the benefits of both.

Many cognitive architectures choose to use a symbolic representation system because that provides a robust way to manipulate the knowledge represented by an agent and to think and talk about it for people observing the agent or interacting with it. We believe that these symbolic representations need to be grounded in the environment of the agents that are built using these cognitive architectures. If we want an agent to be able to perform well in a highly complex and dynamic environment, it is important to have symbol grounding in order to capture the dynamism of the environment. Symbol grounding is achieved by the sensory memory of a cognitive architecture. If the representations that come out of the sensory memory faithfully deliver the required complexity and dynamism from the environment to the rest of the system, we can say that the representation system of the cognitive architecture is grounded in the environment.

But it is not enough to just produce grounded representations that faithfully represent the environment. A cognitive architecture also needs to make sure that the processes and memory systems it employs are able to exploit all the needed nuances that are captured by these representations. For example, the recognition memory system must be able to recognize learned objects/concepts from representations of what the agent has sensed in the environment. Sensory memory and the recognition memory should be able to communicate seamlessly and with high fidelity so as to provide symbols in the recognition memory the expressive power of the representation to be used by other parts of the system.

Here we provide a solution to the two problems of trying to create a sort of hybrid representational system that has the benefits of both symbolic and connectionist representations and building a sensory memory that produces grounded representations that can be communicated seamlessly and with high fidelity to the rest of the cognitive architecture. We solve these problems in the context of a systems-level cognitive architecture called LIDA (Franklin et al., 2016), short for Learning Intelligent Decision Agent.

Recent advancements to the representation system in LIDA model, by the virtue of using Modular Composite Representations (Modular Composite Representations) as a part of Vector LIDA project (Snaider & Franklin, 2014b), makes LIDA a solid contender among cognitive architectures. The value of using the new representation system for LIDA was argued for in Snaider & Franklin (2014b). This new representational system makes it much more powerful by preserving all the benefits of its earlier symbolic representations in the form of nodes and links and adding a lot of processing capabilities that were not possible with nodes and link. In addition, this system makes the representations very noise robust. We feel that the earlier sensory memory system does not do justice to the augmented capabilities of LIDA model with this new representation system. The earlier sensory memory still uses nodes and links, which are not expressive enough for information-rich dynamic environments that Vector LIDA is capable of representing with the help of Modular Composite Representation vectors.

In this paper, we propose a new Sensory Memory System for LIDA that compliments the new vector-based representational system of Vector LIDA by delivering richer representations than nodes and links. Sensory representations now will be in the form of high dimensional sparse distributed representations, which have greater expressive power and thus enables stronger symbol grounding. This creates a need for translation between the two types of representations, viz. Modular Composite Representation and sparse distributed representation, for the sensory memory and recognition memory to be able to communicate. In this paper, we also describe a method of doing high fidelity translation between sparse distributed representation and Modular Composite Representation vectors that preserves the richness of information during translation.

We describe the two major contributions mentioned above in the Sections 3 (HTM as Sensory Memory) and 4 (Translation Problem) respectively. Section 2 will give the reader some background about the concepts that we talk about in the rest of the paper. Section 5 describes the experiments done to validate the claim of high fidelity seamless translation between Modular Composite Representation and sparse distributed representation vectors. Section 6 will conclude with a discussion of the results and contributions of this paper.

2. Background

Cognition is said to be made up perception-action cycles by neuroscientists and psychologists (Cutsuridis et al., 2011; Dijkstra et al., 1994; Freeman, 2002; Fuster, 2004, 2002; Neisser, 1976). An agent (natural or artificial) samples its environment, internal or external, to initiate the process of perception; with the rest of the system employing other cognitive processes it completes the cycle by performing an action. This is what a cognitive cycle is. One important cognitive process in this cycle is attention. Attention filters for the most salient aspects of the current situation for the agent, and broadcasts them to the whole cognitive system to use as conscious content in accordance with Global Workspace Theory (Baars, 1993).

2.1 LIDA Model and Global Workspace Theory

The LIDA model (Figure 1) is built out of these cognitive atoms, or cognitive cycles, running in parallel in an agent. Various processes of cognition, like planning, imagining and reasoning, happen over multiple cognitive cycles. These processes are enabled by various modules working together. They work by manipulating and restructuring data during a cognitive cycle working towards the fulfillment of the agenda of the agent. For example, attention codelets produce coalitions, including salient content from aspects of the current situation, that compete for consciousness. Similarly, structure building codelets build new structures from the contents of the current situation. Each cognitive cycle is marked by the event of conscious content broadcast, which initiates learning in possibly all the various memories of the LIDA model. This will be relevant later in this paper because this is one of the main conceptual commitments of LIDA model to which we must adhere in the process of building a Sensory Memory. The commitment comes from the Conscious Learning Hypothesis (Franklin et al., 2013), and states that significant learning takes place via the interaction of consciousness with the various memory systems. This hypothesis has been argued for in detail

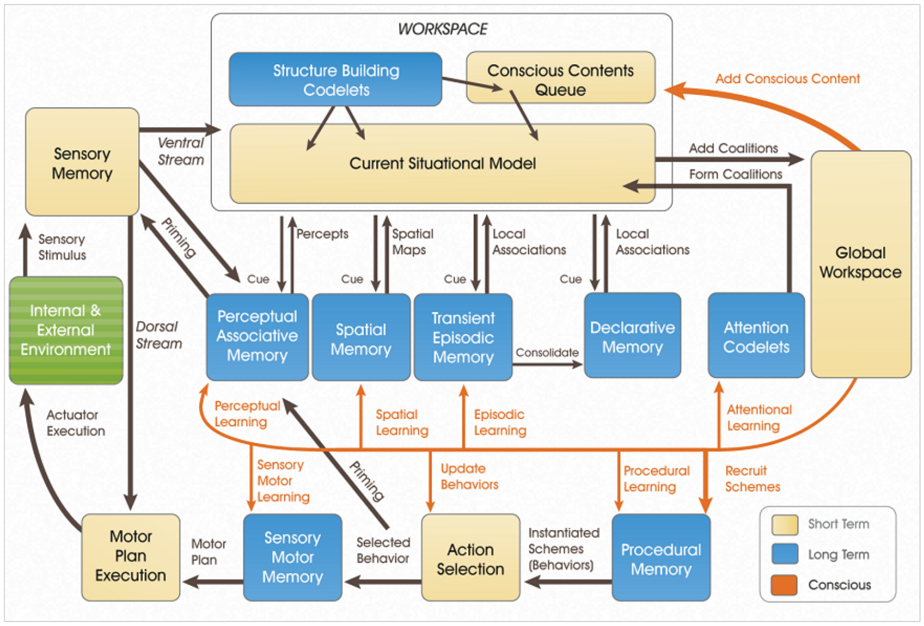

Figure 1. The Learning Intelligent Decision Agent cognitive architecture.

in Franklin et al. (2013). Consciousness refers to functional consciousness here, as opposed to phenomenal consciousness. Consciousness acts like an attention filter in LIDA.

Further details about the LIDA model are beyond the scope of this paper and can be found in the LIDA Model Tutorial (Franklin et al., 2016). In the next two sections, we discuss the two important memories in LIDA Model that are relevant for discussions in this paper.

2.1.1 Sensory Memory

Sensory Memory in LIDA uses feature detectors. Feature detectors are daemons which are constantly looking for incidences of their interest in the environment. One feature detector is tasked with looking at one particular feature in the entire perceptual scene. When it detects that feature it puts a node in Sensory Memory that lasts for a few milliseconds. This node represents the feature and tells the agent that that particular feature was recently present in the environment. Since the nodes in the Sensory Memory last only a few milliseconds, it is a short-term memory.

Sensory Memory plays an important role in fulfilling the Embodied Cognition commitments of the LIDA model (Franklin et al., 2013). Embodied Cognition claims that the body as well as the brain (if it has one) of an agent is situated in an environment, and the relationship between the environment, the body, and the mind affects all cognitive processing (de Vega et al., 2012). The LIDA model adheres to Embodied Cognition by using only perceptual symbols (Barsalou, 1999). Sensory memory is directly involved in creating these perceptual symbols that faithfully represent the environment of the agent.

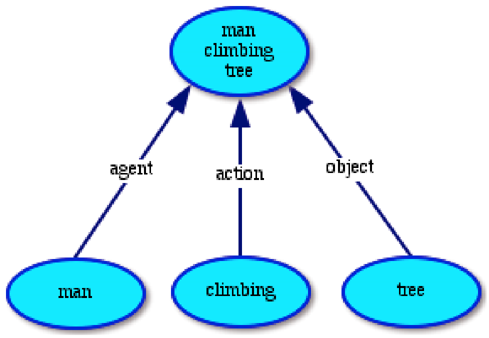

Figure 2. A node and links structure in PAM representing an event.

2.1.2 Perceptual Associative Memory

PAM is the recognition memory of LIDA. It uses nodes and links to represent information. Nodes represent features, objects, feelings, actions, events, categories etc. Links represent relations between these nodes, like feature-of, causation, category membership etc (see Figure 2). These are the things an agent has learned to recognize in its environment, and their relevant relationships to one another. Structures in PAM can be accessed by cues from the Current Situational Model (see Figure 1). They are activated from other PAM nodes or from Sensory Memory. This allows the agent to form percepts from the sensory stimulus that are then sent back to Current Situational Model to enable the agent to have a better understanding of the current situation that it is in with regard to its internal and external environment.

2.2 Vector LIDA

Vector LIDA was presented as a major overhaul of the representation system of the LIDA model. It uses high dimensional Modular Composite Representation (Modular Composite Representation) vectors, instead of nodes and links as previously described. These Modular Composite Representation vectors are stored in Integer Sparse Distributed Memory (Snaider et al., 2013), which is a modified version of Kanerva's (1988) Sparse Distributed Memory, for storing integer vectors instead of binary vectors. There are several advantages to using Modular Composite Representation vectors for representation, and ISDM for memory. It enhances the learning mechanisms, improves scalability and provides better integration with episodic memory and sensory memory. For more details on that and other benefits please refer to the Vector LIDA paper (Snaider & Franklin, 2014b).

Since Vector LIDA is based on representing information using Modular Composite Representation vectors and processing these vectors, we briefly describe Modular Composite Representation vectors in the following section.

2.3 Modular Composite Representation Vectors

Modular Composite Representation (Snaider & Franklin, 2014a) vectors are long integer vectors. They are similar to Spatter Code (Kanerva, 1994) in its properties, where Spatter Code uses binary vectors. Modular Composite Representation is a reduced description model. This means that complex representations can be compressed/reduced into one vector, and can be reconstructed out of this vector when needed. The interesting thing to note here is that each of the components of the complex representation is itself a long vector of the same dimension as the compressed vector. The following paragraphs provide a little more technical description of Modular Composite Representation vectors.

A system using Modular Composite Representation vectors defines the size of the vectors it is going to use, which remains constant for that system's lifetime. Every Modular Composite Representation vector has that many dimensions, where each dimension can take a value from a defined range of integer values, for example, (0, 1, ..., 15) (Aggleton & Pearce, 2001). More formally, a Modular Composite Representation vector is a vector in a multidimensional space, $v \in \mathbb{Z}_r^n$, where n is the number of dimensions of the space and r denotes the range of integer values for each dimension.

Two basic operations defined on these vectors are Binding and Grouping. These are important concepts to understand for getting a grasp on the translation between sparse distributed representation and Modular Composite Representation vectors.

- **Binding** is like a multiplication operation, symbolized by $\otimes$. It is defined as the modular sum in each dimension. It works like the bitwise XOR for binary vectors. For example, multiplying vectors $X, Y \in \mathbb{Z}_{16}^n$ to get integer vector $Z = X \otimes Y$, if the dimension i in X and Y has values 11 and 14 respectively, then in the resulting dimension in Z, Z_i will be 9. Formally,

  $$Z_i = mod_{16}(X_i + Y_i)$$

  The unbinding operation can thus be defined as the modular subtraction in each dimension or binding the first operand to the complement of the second operand $X = Z \otimes Y^{-1}$. The complement, or the inverse vector, in this model is such that each dimension in Y^{-1} is the complement of that value in the same dimension. Continuing our previous example each dimension in Y^{-1} would be,

  $$Y_i^{-1} = mod_{16}(16 - Y_i) = 16 - Y_i$$

  Two important properties of this operation are that
  - It is associative, commutative and distributive over the sum or grouping operation.
  - It produces a vector that is very different from the operands. Almost always orthogonal.

- The **grouping operation** is like finding a resultant vector of two vectors, symbolized by +. To describe more formally, for each dimension we place the values of that dimension for each operand as an equivalent vector on a circle as shown in Figure 3. For example, the equivalent vector for the value 0 is (0,1), 4 is (1,0), 8 is (0,-1), 12 is (-1,0). The sum operation for each dimension will be the resultant vector of the equivalent vectors of all operands for that dimension after normalization and rounding to the closest equivalent vector corresponding to an integer. For example in Figure 3, the normalized result of equivalent vectors for 3 and 7 is 5. Another example in Figure 3 demonstrates rounding. When we add the three equivalent vectors 0, 7 and 13 the result is 14 after rounding. The most important property of the grouping operation is that the resulting vector is similar to each of its operand vectors.

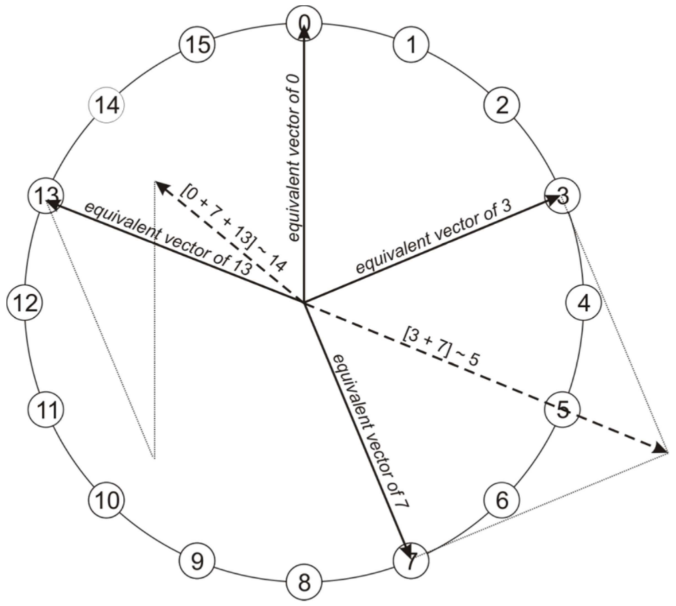

Figure 3. Finding the equivalent vector for a value of a dimension.

These properties of the binding and grouping operations allow Modular Composite Representation vectors to act as reduced descriptions. A reduced description can be created from a collection of attributes and their names. For example, from two attributes shape and color, we can create a reduced description of a "red circle". If we have vectors representing $Shape$, $Color$, red and $circle$, this can be done by using binding and grouping operations as follows:

$$F = [circle \otimes Shape + red \otimes Color]$$

This is a useful reduced description only if we can recover the components that created F. This can be done if we use an Integer Sparse Distributed Memory to store these vectors. We will continue this example to show how to recover vectors from a reduced description using an ISDM, after we introduce ISDM in the next section.

2.3.1 Integer Sparse Distributed Memory

Integer Sparse Distributed Memory (ISDM) is like Kanerva's Sparse Distributed Memory (SDM), but it is used to store high dimensional integer vectors, for example, Modular Composite Representation vectors. Like SDM it is an auto-associative memory. For details on the functioning of ISDM, please refer to Snaider (2012). ISDM consists of a fixed number of hard-locations where vectors can be stored. A hard location is a point in the multi-dimensional space representing an integer vector, a point of the kind that this memory will eventually store. These hard locations are distributed uniformly randomly over the vector space. A vector is written in all the hard-locations that are within a critical distance from that vector. Since a vector is written in multiple locations it is noise robust. Also, when we try to read a previously written vector from the memory we do not need the exact vector to effect the read. A noisy version of the vector can be given to ISDM, which works like a cleanup memory. Since the noisy vector is like the vector we want to read, ISDM will look in approximately the same vicinity as it used to store the non-noisy vector, to find hard-locations to read this vector from. ISDM performs a kind of polling from all the hard locations that it finds within the critical distance to produce the vector that was originally written. The resulting vector is less noisy because of the polling. ISDM can then repeat this process until convergence to obtain the cleaned up or non-noisy vector. This is how we can read the original vector using a noisy version of that vector.

2.3.2 Reduced Descriptions

In Section 2.3, we created a reduced description of a "red circle" using red, $circle$, $Shape$ and $Color$ vectors as:

$$F = [circle \otimes Shape + red \otimes Color]$$

A reduced description has a role filler type structure, where $Color$ and $Shape$ are the roles and red and $circle$ are corresponding fillers. For F to be a useful reduced description of "red circle" we need to be able to retrieve the attributes of the "red circle" from the vector F. We can do this using the method of probing as described in the following section.

2.3.3 Probing

Probing is the processes of checking to see if a reduced description vector contains the probed vector as a component.

For reduced descriptions created by the binding operation, this is done by binding the reduced description with the inverse of the role vector (unbinding). The result of this process is the filler and noise. This noise can be removed by using cleanup memory to retrieve the actual filler, if we have all the vectors stored in an ISDM. For example, if we want to check the shape of the "red circle" we could probe its reduced description with the inverse of $Shape$.

$$F \otimes Shape^{-1} = [circle \otimes Shape \otimes Shape^{-1} + red \otimes Color \otimes Shape^{-1}]$$

$Color$, $Shape^{-1}$ and red, when bound, produce noise, and $Shape$ and $Shape^{-1}$ when bound cancel out, so we get:

$$F \otimes Shape^{-1} = [circle + noise]$$

This noise can be removed by reading this resulting vector from a cleanup memory, so we come full circle. If F is probed with a role that was not a component, to begin with, then all we get is noise, which results in an unsuccessful read operation from the cleanup memory.

For reduced descriptions created by grouping operations, probing is done by checking the distance[1] of the probed vector from the group vector. If the distance between a group vector and the probe vector is less than a threshold, we can say that the group vector was grouped/composed using the probe vector. This is possible because the group vector is similar to all of its component vectors.

1. Distance between two Modular Composite Representation vectors is defined as an extended Manhattan metric which accounts for the modularity of dimensions.

Thus, the distance between the group vector and any of its components is less than the distance between any two random vectors. Formally, for a vector G with components A and B,

$$G = [A + B]$$
$$distance(G, A) < \textit{Threshold}$$
$$distance(G, B) < \textit{Threshold}$$
$$distance(G, X) \geq \textit{Threshold},$$

where,

X *is any random Modular Composite Representation vector.*

Snaider (2012) has this threshold as the *Indifference Distance* of this Modular Composite Representation space, where *Indifference Distance* is defined as the distance between any two random vectors in this Modular Composite Representation space.

2.4 Hierarchical Temporal Memory

Hierarchical Temporal Memory is a spatio-temporal pattern recognizer capable of online learning, proposed by Hawkins et. al. (2011). It is inspired by the computations in the layers of the neocortex in the brain (Hawkins et al., 2009). It uses activity of a region of columns to represent a given input. It employs sparse distributed representations for representing input. This means that for any given input only a few of the columns, out of thousands, in a region are active, typically 2% to 5%. Sparse representations provide a lot of benefits for representation of a domain of inputs. They are noise robust. They also allow a lot of freedom and flexibility with which to encode the similarity between inputs. Thus, it is very expressive. For more details please refer to Hawkins et al. (2011).

2.5 Problem Elaboration

We need to build a Sensory Memory for LIDA that will allow us to leverage the greater expressive power of the Modular Composite Representation vectors used in the Vector LIDA system, to replace the earlier simpler feature detectors used so far in the Sensory Memory. For future Vector LIDA based agents, we can use HTM Regions to build the Sensory Memory. Each HTM Region will be responsible for sensing an aspect of the environment. For example, one region can be used to sense the location of the agent from a GPS sensor, another region can be used to sense the motor commands of the gripper of an arm of the agent. All the benefits of using HTM Regions for this task is explained in the next section.

We face a few challenges in building the Sensory Memory out of HTM Regions. The first challenge is that the representations coming out of HTM Regions are sparse distributed representations, whereas the rest of the system in Vector LIDA is going to use Modular Composite Representation vectors. Therefore, we need to be able to translate sparse distributed representation vectors into Modular Composite Representation vectors. We need to do this in such a way that we do not violate the embodied cognition commitment of the LIDA model. We can do this only by preserving the meaning and expressiveness of the sparse distributed representations during translation so that the rest of the system can understand its environment effectively. Thus, the translation will not violate embodied cognition.

The second challenge is that we need to allow HTM Regions to employ online reinforcement learning. This is a challenge because if we want to allow HTM to learn during the lifetime of an agent we need to adhere to the Conscious Learning commitment of LIDA model, as mentioned in Section 2.1. To allow conscious learning we will only let HTM learn from the contents of the conscious broadcast. In Vector LIDA the conscious broadcast will consist of one or more Modular Composite Representation vectors. We need to be able to retrieve the lowest level sensory content from these vectors, and further translate them into sparse distributed representations, so that Sensory Memory Module can map these sparse distributed representations to the respective regions and columns, and perform reinforcement learning accordingly.

3. HTM as Sensory Memory

We want to build a Sensory Memory that can deliver information-rich representations to PAM so that Modular Composite Representation based systems of Vector LIDA can exploit Modular Composite Representation's power of representation, and thus allow the agent to better deal with complex and dynamic environments.

LIDA's Sensory Memory can use HTM Regions as feature detectors. We can envision an agent having multiple HTM Regions to sense a variety of aspects of its environment. Even though HTM is not the state of the art in image processing and computer vision, as it improves it can be used to directly sense the environment through a camera. It has been shown to work for a video game environment on ray cast image data by Sungur (2017). For now, it can still be used for simpler agents by doing edge detection and image segmentation in a pre-processing step and feeding in the segmented filtered image as input to an HTM Region to detect shapes. Another HTM Region can be employed to detect colors in the segmented filtered image. Benefits of doing this include that similar shapes and similar colors will have similar representations created by HTM Regions. A big advantage of using HTM based regions is that they are agnostic of the input modality. We do not need to hand-craft the feature detectors for every feature and every modality. Unlike many other neural network models, they employ online learning. Depending on the need of the agent, HTM can be setup to extract as much granular information from the input as needed. For example, it can be setup so that it can detect Red and Pink to be similar colors, or very different, as needed by the agent.

The current implementation of HTM employs Encoders to feed data into the HTM. Encoders are a preprocessing layer for the input. Numenta has created Encoders for various type of data like Scalar, Category, Date, Coordinate, Geospatial Coordinate etc[2]. An agent needing to understand its acceleration (a three-tuple Coordinate) from its accelerometer sensor, or its location (a Geospatial Coordinate) from the GPS sensor, can use these encoders to sense from its environment. Here too we can reap the benefits of similarity in input from the HTM's pattern recognition. An acceleration of similar magnitude or direction will have a similar representation by HTM. For example, an accel-

2. Available at `https://github.com/numenta/nupic/tree/master/src/nupic/encoders`

eration due North will have similar representation to an acceleration due Northwest, but completely different representation to one due West. A location near to another location will have a similar representation, and these are tunable by the parameters of these Encoders to suit the needs of the agent. In this way, we can have HTM sense a variety of attributes of the environment for an agent with a high degree of control on the expressiveness.

It is important to note that the LIDA model is a conceptual framework and that agents are specific implementations crafted from this framework, utilizing some or all the concepts available in the framework. An agent can use many different HTM Regions based on its need, and the HTM Regions will be configured so that it can deliver representations that are relevantly information-rich for its purposes.

The proposed HTM based Sensory Memory allows for flexibility between how much pre-learned structures are needed and how much can be learned. If we need some pre-learned structures, we can train an HTM Region before using it for the agent in the same, a simulated or a similar environment. If we need it to learn everything from scratch while living in the environment, we can use untrained HTM Regions.

4. Translation Problem

Sparse distributed representations produced by HTM Regions and Modular Composite Representations, used for representations in LIDA, are both high dimensional vectors but are very different in nature. Sparse distributed representations are sparse binary vectors, in which only a few dimensions have 1s and rest are 0s. Modular Composite Representations are integer modular vectors, in which each dimension can have an integer value between a specified range. HTM Regions will represent a stimulus in the form of sparse distributed representations. PAM needs to use these sparse distributed representations to start recognizing relevant objects/percepts in its environment. Since PAM is built out of Modular Composite Representation vectors, we need to create Modular Composite Representation vectors out of sparse distributed representations produced by HTM Regions, so that PAM can use them meaningfully for recognition. This means that similar sparse distributed representations should produce Modular Composite Representation vectors that are close in Modular Composite Representation vector space. Thus, their similarity, and therefore meaning, in Modular Composite Representation vector space is preserved.

We need to create these Modular Composite Representation vectors in such way that we can get the sparse distributed representation vectors back from the Modular Composite Representation vectors. We need to do this because of the conscious learning commitment of LIDA model. Since the HTM Region employs online learning, if we want the HTM Regions in Sensory Memory to learn during the lifetime of the agent, we need to adhere to the conscious learning commitment, as also mentioned in Section 2.5, and provide only the contents of conscious broadcast for its use for learning. The contents of the conscious broadcast from a Vector LIDA based agent will be in the form of one or more Modular Composite Representation vectors. We need to retrieve the sensory content (in the form of sparse distributed representations) from these Modular Composite Representation vectors, and give it to the respective HTM Region from whence it came, for the HTM Region to be able to learn from it.

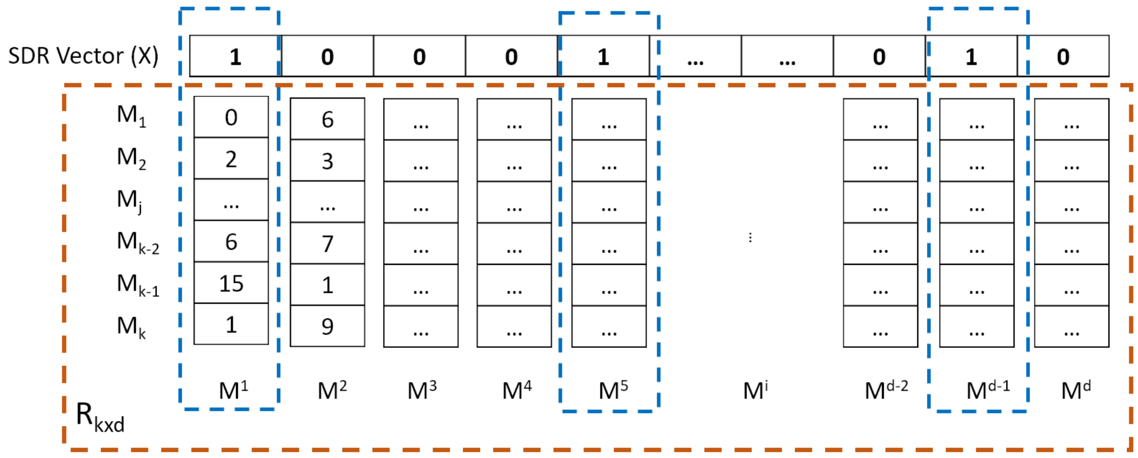

Figure 4. Projected vector of X is Y. Where Y is the sum (as defined by grouping operation) of all the Modular Composite Representation vectors M^i where sparse distributed representation vector X_i has a 1, as shown by blue boxes. The orange box shows the abstract random projection matrix $R_{k \times d}$.

Thus, the translation problem is a two-part problem of translating sparse distributed representation vectors to Modular Composite Representation vectors in such a way that they preserve their meaning upon translation, and they can be later retrieved back for learning in HTM Regions if needed. We first describe the solution to translating a sparse distributed representation vector to a Modular Composite Representation vector in the next Section 4.1. In the following Section 4.2, we describe how to get sparse distributed representations from Modular Composite Representation vector based conscious contents.

4.1 Sparse Distributed Representation to Modular Composite Representation

The inspiration for the solution of the problem of trying to convert sparse distributed representation vectors to Modular Composite Representation vectors comes from a technique in mathematics called Random Projection (Bingham & Mannila, 2001). Random projection gives us a method of projecting a high-dimensional vector space to a lower dimensional vector space, in a way which approximately preserves the distances between the points, as determined by the Johnson-Lindenstrauss lemma (Dasgupta & Gupta, 1999). In random projection, the original d-dimensional data is projected to a k-dimensional $(k \ll d)$ subspace through the origin, using a random $k \times d$ matrix R, where every column is orthogonal to all other columns. If the original data of N points is in matrix $X_{d \times N}$. Then we get the projection $P_{k \times N}$ as:

$$P_{k \times N} = R_{k \times d} \times X_{d \times N}$$

Here, we store our sparse distributed representations in the matrix $X_{d \times N}$, where d is the dimensionality of the sparse distributed representation and N is the number of sparse distributed representations we want to translate. We generate the random $R_{k \times d}$ matrix, per HTM Region, by generating d random Modular Composite Representation vectors, of the size defined to be used in the agent, here

k. Since it is a property of random Modular Composite Representation vectors that they are orthogonal to each other, we satisfy the requirement of generating an appropriate R. Thus, by multiplying the matrices and using the modular addition within a dimension as described in the grouping operation, we can get the resulting projected data in Modular Composite Representation vector space in $P_{k \times N}$, in the form of N k-dimensional Modular Composite Representation vectors.

Abstractly we can describe this process of translation as treating each dimension of a sparse distributed representation vector as a feature and generating a random Modular Composite Representation vector for each feature. This random Modular Composite Representation vector represents its feature in Modular Composite Representation space. Now, whenever we get a sparse distributed representation with some features active, we generate a Modular Composite Representation vector with the grouping of all the Modular Composite Representation vectors that represent all the active features, as shown in Figure 4. For a sparse distributed representation vector X we generate a projected vector Y using M, where M is a set of randomly generated Modular Composite Representation vectors one for each dimension in X indexed by i as M^i, as:

$$Y = \sum_{i \in I} M^i,$$

where,

I *is the set of indices of dimensions in* X *with an active bit and*

Σ *is the grouping operation.*

We need to be able to distinguish between Modular Composite Representation vectors generated by sparse distributed representations from two different HTM Regions co-existing in an agent. For this reason, we generate a random Modular Composite Representation vector and assign it to each region, as a Modular Composite Representation vector representing that region. Once we translate a sparse distributed representation we can bind (using the binding operation defined in Section 2.3) the resulting Modular Composite Representation vector with the Modular Composite Representation vector assigned to the region from which the sparse distributed representation came from. Now, the resulting vector is ready to be used by PAM. Optionally, we can also additionally bind it to a Modular Composite Representation vector that represents the sensory modality which the HTM Region is looking at. For example, we may have multiple HTM Regions employed for each of the various modalities of the agent like visual, auditory, motor, internal environment etc.

4.2 Modular Composite Representation to sparse distributed representation

A LIDA based agent will get a Modular Composite Representation vector or a group of Modular Composite Representation vectors from each conscious broadcast. A Modular Composite Representation vector will be a reduced description of some kind. We can use the unbinding probing operation to determine if the vector has any content from any of the HTM Regions in that agent. We retrieve the filler from any successful probes with any HTM Regions. Once we get the filler for an HTM Region we can then use the grouping probe operation using all M^i vectors as probes one by one. The M^i vectors that pass the probing indicate that the corresponding feature of the sparse

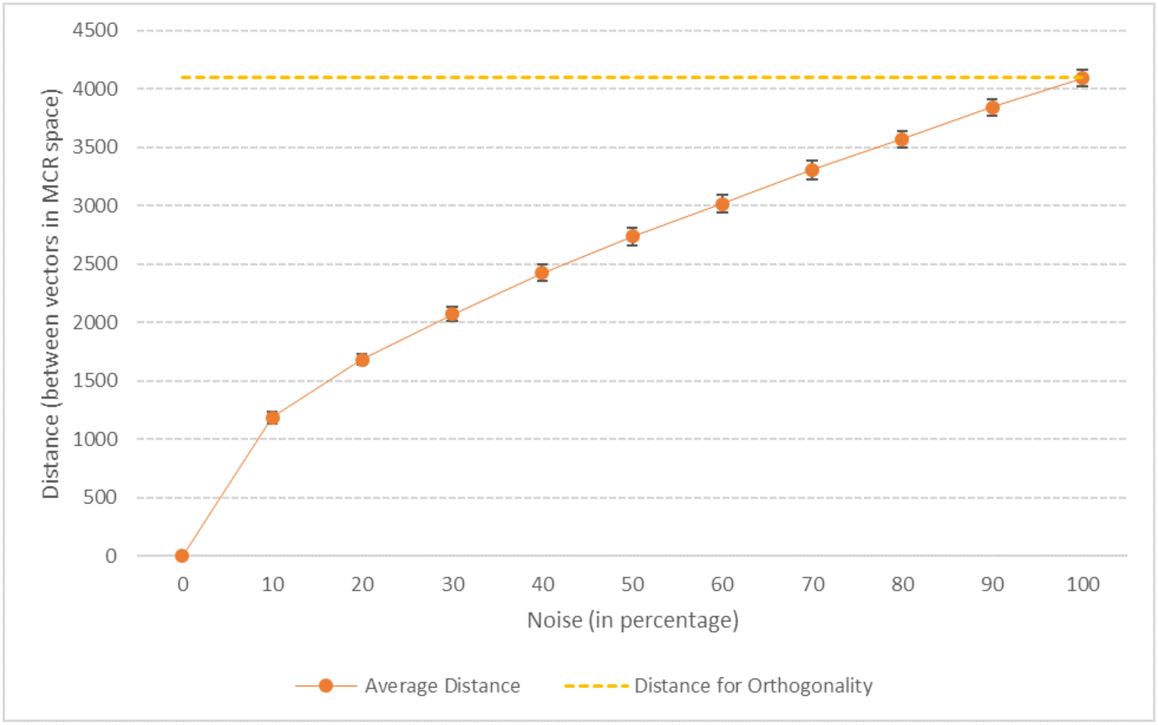

Figure 5. This graph shows the average distance between translated Modular Composite Representation vectors, before and after adding noise to the corresponding sparse distributed representation vectors. Standard deviation bars are shown on each point at the measured noise level.

distributed representation vector dimension was active. This way we can reconstruct the sparse distributed representation from the Modular Composite Representation vector. Once we reconstruct the sparse distributed representation vector X, it can be used by the corresponding HTM Region for learning.

Formally, to reconstruct each dimension i of X we probe the Modular Composite Representation vector Y with the corresponding Modular Composite Representation vector M^i.

$$X_i = \begin{cases} 1 & \text{if } distance(Y, M^i) < Threshold \\ 0 & \text{if } distance(Y, M^i) \geq Threshold \end{cases}$$

We use a stricter Threshold of grouping probe operation than used by Snaider (2012). Here we use $\mu_{id} - k * \sigma_{id}$ as the Threshold, where μ_{id} and σ_{id} are the mean and standard deviation of the Indifference Distance between the Modular Composite Representation vectors of the specified length, and k is a parameter of the method that can be varied to optimize the performance of reconstruction.

5. Experimental Studies

To establish the validity of our method of translation we need to prove two things. First, this method must preserve the distance between two vectors in translation, i.e. different sparse distributed representation vectors when projected should be orthogonal Modular Composite Representation vectors, and similar sparse distributed representation vectors, when projected, should result in Modular Composite Representation vectors that are very close in Modular Composite Representation space. Second, the reconstruction from a Modular Composite Representation vector back to a sparse distributed representation vector is of high fidelity and preserves most of the information. We can tolerate some information loss in the translation back to sparse distributed representation vectors because sparse distributed representation vectors are very noise robust.

For the first test, we generate 100 random sparse distributed representation vectors, 1024 dimensions long and 2% sparse, this is a typical dimensionality for sparse distributed representation vectors from HTM Regions. We then translate them, as described in Section 4.1, into corresponding Modular Composite Representation vectors of dimension 1024, also typical for Modular Composite Representation vectors. We call this set of Modular Composite Representation vectors a reference set. To obtain similar sparse distributed representation vectors and check the quality of translation we add some noise to the sparse distributed representation vectors. Less noise means that sparse distributed representation vectors are similar more noise means they are different to the point where 100% noise means completely different sparse distributed representation vectors. So, we add x% noise to the sparse distributed representation vectors by randomly changing x% of active bits to inactive and activating the same number of other randomly chosen inactive bits, thus maintaining the sparsity at the same level. In other words, adding x% noise to a 1024-dimensional sparse distributed representation vector creates a new sparse distributed representation vector that has a Hamming distance of $(2 * x/100 * 1024)$ from the original vector. We then use these noisy sparse distributed representation vectors to generate a new set of Modular Composite Representation vectors. We measure the distance of Modular Composite Representation vectors in the reference set to the ones generated by corresponding noisy sparse distributed representations. Figure 5 graphs the results of this measurement for the 100 vectors.

For the second test, we take the reference set of the 100 Modular Composite Representation vectors and translate them back to sparse distributed representation vectors using the method described in Section 4.2. For mod_{16} Modular Composite Representation vectors with 1024 dimensions μ_{id} was 4095 and σ_{id} was 74, computed empirically. Thus, the threshold used was $4095 - k * 74$, as mentioned in Section 4.2. We then compute precision and recall for the number of dimensions of a sparse distributed representation correctly retrieved for each of the Modular Composite Representation vectors. The statistics of the precision and recall for 100 vectors for different values of the parameter k are recorded in Table 1.

We can see from Table 1, that this system is able to reconstruct the sparse distributed representations with perfect precision and recall for $k = 5$. We have made the code that produced these results publicly available.[3]

6. Discussion

We needed the representations in the Vector LIDA model to be grounded and, we wanted to build a Sensory Memory that could do justice to the power of Vector LIDA by delivering an information-

3. Available at `https://github.com/pulinagrawal/cla-to-mcr/blob/master/cla-mcr.ipynb`

Table 1. This table shows the statistics in the form of mean (±standard deviation) of precision and recall of retrieval of dimension values of sparse distributed representation vectors from 100 Modular Composite Representation vectors for various values of the parameter k.

	k=0	k=1	k=2	k=3	k=4	k=5
Precision	0.038 (±0.001)	0.110 (±0.007)	0.461 (±0.052)	0.943 (±0.050)	0.999 (±0.006)	1.0 (±0.0)
Recall	1.0 (±0.0)	1.0 (±0.0)	1.0 (±0.0)	1.0 (±0.0)	1.0 (±0.0)	1.0 (±0.0)

rich representation of the environment to the model. We demonstrated how HTM Regions can be used to build high-quality feature detectors for Vector LIDA model. These HTM Regions can be configured to be as expressive in capturing the information from the environment as needed by the particular agent. We also demonstrated how the HTM based Sensory Memory can facilitate successful symbol grounding by achieving high-quality translation from HTM representations (sparse distributed representations) and Vector LIDA representations (Modular Composite Representation vectors) that preserve the information (therefore meaning) captured by the HTM Regions.

This preservation of information happens in both the directions of translation. In the forward translation from sparse distributed representations to Modular Composite Representation vectors, we can see that similar sparse distributed representation vectors remained similar when translated to Modular Composite Representation space. This is demonstrated by Figure 5, where a sparse distributed representation vector with less noise, therefore considered similar to the original sparse distributed representation, had a small distance to the translated Modular Composite Representation vector of the original sparse distributed representation after translation. A translated sparse distributed representation vector with more noise, and its corresponding Modular Composite Representation vector in the reference set are far apart in Modular Composite Representation space. In fact, adding 100% noise to a sparse distributed representation, i.e. creating a completely different sparse distributed representation, and translating it to a Modular Composite Representation vector, made it orthogonal to its corresponding reference vector in Modular Composite Representation space. In the backward translation from Modular Composite Representation vectors to sparse distributed representations, we were able to perfectly reconstruct a sparse distributed representation vector that created that Modular Composite Representation vector as evidenced by the perfect precision and recall with zero standard deviation for k=5, as shown in Figure 6. This result depends on the sparsity of the sparse distributed representation vectors because the sparsity directly affects the number of Modular Composite Representation vectors that are grouped. For a given Modular Composite Representation specification we can only group a certain number of Modular Composite Representation vectors at once. This limit can be increased by increasing the integer range of values that one dimension can hold. For this typical specification of Modular Composite Representation and sparse distributed representation vectors, we were able to show that all the meaning is preserved in the backward translation as well.

Since the meaning is preserved we can say that the symbols used by Vector LIDA in the form of Modular Composite Representation vectors are grounded in the environment using the HTM based Sensory Memory. This will allow us to build agents out of Vector LIDA that have grounded symbols. Also, HTM based Sensory Memory allows the agent to represent its environment richly and meaningfully using Sparse Distributed Representations and Modular Composite Representation vectors. Thus, we propose a way to build Sensory Memory within Vector LIDA with HTM Regions and a method of translation, that we show provides a faithful representation of the environment, as evidenced by our lossless translation, to compliment the augmented capabilities of Vector LIDA.

Acknowledgements

We thank the reviewers for their useful comments. We also thank Sean Kugele, Steve Strain for improving the quality of work with useful discussions. Dr. Bernie Daigle provided useful insights in improving the readability of the content.

References

Aggleton, J. P., & Pearce, J. M. (2001). Neural systems underlying episodic memory: Insights from animal research. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *356*, 1467–1482.

Baars, B. J. (1993). *A cognitive theory of consciousness*. Cambridge, MA: Cambridge University Press.

Barsalou, L. W. (1999). Perceptions of perceptual symbols. *Behavioral and Brain Sciences*, *22*, 637–660.

Bingham, E., & Mannila, H. (2001). Random projection in dimensionality reduction: Applications to image and text data. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 245–250). New York: ACM.

Cutsuridis, V., Hussain, A., & Taylor, J. G. (Eds.). (2011). *Perception-action cycle: Models, architectures, and hardware*. New York: Springer.

Dasgupta, S., & Gupta, A. (1999). *An elementary proof of the Johnson-Lindenstrauss lemma*. Technical report, International Computer Science Institute, Berkeley, California, USA.

de Vega, M., Glenberg, A., & Graesser, A. (Eds.). (2012). *Symbols and embodiment: Debates on meaning and cognition*. New York: Oxford University Press.

Dijkstra, T., Schöner, G., & Gielen, C. (1994). Temporal stability of the action-perception cycle for postural control in a moving visual environment. *Experimental Brain Research*, *97*, 477–486.

Franklin, S. (1997). *Artificial minds*. Cambridge, MA: MIT Press.

Franklin, S., Madl, T., Strain, S., Faghihi, U., Dong, D., Kugele, S., Snaider, J., Agrawal, P., & Chen, S. (2016). A LIDA cognitive model tutorial. *Biologically Inspired Cognitive Architectures*, *16*, 105–130.

Franklin, S., Strain, S., McCall, R., & Baars, B. (2013). Conceptual commitments of the LIDA model of cognition. *Journal of Artificial General Intelligence*, *4*, 1–22.

Freeman, W. J. (2002). The limbic action-perception cycle controlling goal-directed animal behavior. *Proceedings of the 2002 International Joint Conference on Neural Networks* (pp. 2249–2254). Honolulu, HI: IEEE.

Fuster, J. M. (2002). Physiology of executive functions: The perception-action cycle. In *Principles of frontal lobe function*, 96–108. New York: Oxford University Press.

Fuster, J. M. (2004). Upper processing stages of the perception-action cycle. *Trends in Cognitive Sciences*, *8*, 143–145.

Hawkins, J., Ahmad, S., & Dubinsky, D. (2011). Hierarchical temporal memory including HTM cortical learning algorithms, 0.2. From `https://numenta.org/resources/HTM_CorticalLearningAlgorithms.pdf`.

Hawkins, J., George, D., & Niemasik, J. (2009). Sequence memory for prediction, inference and behaviour. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, *364*, 1203–1209.

Kanerva, P. (1988). *Sparse distributed memory*. Cambridge, MA: MIT Press.

Kanerva, P. (1994). The spatter code for encoding concepts at many levels. *Proceedings of the International Conference on Artificial Neural Networks* (pp. 226–229). Sorrento, Italy: Springer.

Neisser, U. (1976). *Cognition and reality: Principles and implications of cognitive psychology*. New York: WH Freeman.

Snaider, J. (2012). *Integer sparse distributed memory and modular composite representation*. Doctoral dissertation, University of Memphis, Memphis, TN.

Snaider, J., & Franklin, S. (2014a). Modular composite representation. *Cognitive Computation*, *6*, 510–527.

Snaider, J., & Franklin, S. (2014b). Vector LIDA. *Procedia Computer Science*, *41*, 188–203.

Snaider, J., Franklin, S., Strain, S., & George, E. O. (2013). Integer sparse distributed memory: Analysis and results. *Neural Networks*, *46*, 144–153.

Sun, R. (1999). Accounting for the computational basis of consciousness: A connectionist approach. *Consciousness and Cognition*, *8*, 529–565.

Sungur, A. K. (2017). *Hierarchical temporal memory based autonomous agent for partially observable video game environments*. Master's thesis, Middle East Technical University, Ankara, Turkey.

Winston, P. H. (1992). *Artificial intelligence*. Reading, MA: Addison-Wesley.