Exploring Models of Cortical Columns for Biologically Constrained AI in Object Recognition

By

SANTIAGO ENRIQUEZ

Biomedical Engineering BS, Polytechnic University of Madrid, 2023

Thesis

Submitted in partial fulfillment of the requirements for the Degree of Master of Science in the Graduate Program of Biomedical Engineering at Brown University

PROVIDENCE, RHODE ISLAND

MAY 2025

AUTHORIZATION TO LEND AND REPRODUCE THE THESIS

As the sole author of this thesis, I authorize Brown University to lend it to other institutions for the purpose of scholarly research.

Date_____ _____

Santiago Enriquez, Author

I further authorize Brown University to reproduce this thesis by photocopying or other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Date_____ _____

Santiago Enriquez, Author

This thesis by Santiago Enriquez is accepted in its present form by the Center of Biomedical Engineering, School of Engineering, and Division of Biology and Medicine as satisfying the thesis requirements for the degree of Master of Science.

Date_____ Signature: _____

Dr. Mohamed Sherif, Advisor

Date_____ Signature: _____

Dr. Wael Asaad, Reader

Date_____ Signature: _____

Dr. Jonghwan Lee, Reader

Approved by the Graduate Council

Date_____ Signature: _____

Dr. Marissa Gray, Biomedical Engineering Master's Program Director

Acknowledgements

I would like to thank my family for their constant support and encouragement throughout my career, and for allowing me to have the opportunity to study abroad at a highly ranked university. Furthermore, I would like to acknowledge my thesis advisor, Dr. Mohamed Sherif, for taking me into this fascinating project, providing advice, a work environment that made me feel welcomed, and positivity, as we walked through this project on a theory and implementation that is still being developed. Finally, I would like to express my gratitude to the new friends I made at Brown for making my time at Providence something to remember. With you, these two years' experience have gone by so fast and yet still made so many memories that everyone back home will be hearing of for a while.

Table of Contents

Table of Figures

Table of Tables

Abstract of Exploring Models of Cortical Columns for Biologically Constrained AI in Object Recognition, by Santiago Enriquez, ScM, Brown University, May, 2025.

Objective: This thesis explores the viability of biologically constrained artificial-intelligence architecture based on Numenta, Inc.'s Thousand Brains Theory (TBT) for object recognition, and initial steps into hierarchy.

Methods: Learning modules (LMs) emulating layers II/III, IV, and VI were implemented with Monty v0.0.3 and embedded in physics-enabled Habitat-Sim scenes. Each LM received sensations from either a distant "eye" agent or a surface "finger" agent and encoded features as distributed graphs. Four supervised architectures were trained on six Yale–CMU–Berkeley objects for one epoch (14 canonical views) and 500 sensorimotor steps per view: (1) single-LM distant; (2) single-LM surface; (3) two horizontally connected LMs; and (4) five horizontally connected LMs, making decisions from 3 LMs agreeing through a voting system. Complementary experiments evaluated a two-level hierarchy and an unsupervised agent that updated its memory after every encounter.

Results: After one epoch on each object, a single LM learned a graph model of each object and later re-identified learned items; though thin objects, like cutlery, caused ambiguities in both learning and inference. Surface agents produced smoother graphs and faster inference than distant agents. Although it was expected that surface agents would model full transparent objects and not just the visible parts, this did not occur. Lateral connections (a voting system) delivered benefits: faster inference depending on the number of LMs that have to agree, and more robustness from multiple LMs agreeing. Extending episodes to 1,000–1,500 steps refined graphs, providing slightly faster inference but no accuracy improvement. The unsupervised agent progressively updated its models over three encounters, and the hierarchical architecture produced sparser models on the higher level LM, hinting at abstraction.

Conclusions: These experiments show that sensory-motor architectures grounded on the Thousand Brains Theory pose a promising direction for advancing AI into real-world interactions, being able to create accurate object models from minimal experience and refine them online. It also showed advantages of a voting system, as well as what could be the early stages of abstraction when applying hierarchy structures.

TBT motivates further investigation into learning and inference of transparent objects, as well as compositional objects and scenes.

Chapter 1: Introduction

From the moment you are born until the day you die, everything you think happens to you is just an interpretation of the world and experiences made by the brain. Since this organ is the one responsible for making sense of what is happening as well as developing a response, it poses an intriguing field of study that has been attracting scientists from many different disciplines [1]. These disciplines range from an explicit investigation to discover brain functioning (such as cognitive neuroscience) to drawing inspiration from the brain to develop machines and programs with the ability to replicate intelligent behavior (like the field of AI [2]). Seeing the broad range of functions of the brain, we will focus on the neocortex and how it processes information from the outside world.

Neocortex

The neocortex is one of the most fascinating and distinctive parts of the human brain [3]. How the neocortex has evolved in mammals is considered to be a key step in higher-order functions, as it is central to sensory perception, cognition, language, and motor control [4]. The anatomical structure of the mammalian neocortex is formed by six layers and is organized in a columnar architecture, which is perhaps the only agreed-upon framework with which to explore models of cortical circuits [5].

The neocortex is between 2 - 4 mm thick and composed of 6 defined layers horizontally and in columns vertically, being the basic unit of this structure, the *mini column* [6,7]. This structure is preserved across all regions, which may suggest that it underlies all the functions of the neocortex [7]. The layers are named I – VI, from the surface inward [3,7], and each of them contains specific cell types and connectivity patterns. The structure of the layers are as follows:

Layer I is the outermost layer containing very few neurons but a high density of dendrites and axons from deeper layers, acting as a hub for the collection and processing of widespread information [8]. Layers II/III are composed of pyramidal neurons and receive sensory information from layer IV, which integrates with other inputs [9]. Pyramidal cells from layers II/III project to other cortical areas. Layer IV is densely packed with stellate neurons and receives the primary sensory inputs from sensory thalamic nuclei, making it very developed in sensory areas and less developed in motor areas [10,11]. Layer V is the main output layer of the mini columns, its pyramidal cells are divided into two groups, one that projects to cortical structures and the other that projects to subcortical structures [12]. Finally, Layer VI is the deepest layer and is home to different types of cells, as with layer IV, which also receives direct sensory input, and like layer V, its cells also project to both cortical and subcortical regions [13]. However, its neurons especially project to layer IV, probably playing an important role in modulating the responses of this layer [13]. Figure 1 shows a sketch of the laminar structure of the neocortex.

Despite the advances in recording technology, the functions of each layer are still unclear, leaving the layers and cortical columns (CCs) mechanisms in the hypothetical world [14].

Computational Neuroscience

Even though there have been a significant number of discoveries about the brain's anatomy and cellular composition, there is still little information on how this system works for us to be able to interpret the world [1,15]. This investigation has been divided into many areas of study, where there are plenty of models that differ abundantly in form and content [5]. This broad discrepancy from different models arises from the difficulty in correlating structure and function [16] as well as from the specific experiments that try to explain a specific situation [5]**.** One field dedicated to this

investigation is Computational Neuroscience. In computational neuroscience, in general, there are two approaches on understanding the brain's behaviour: the first, bottom-up, where you take the cellular architecture of the brain and try to develop a theory on how to make the behaviour you are seeing [17]; or top-down, where through the relationship between brain and behaviour you develop a theory [18]. Any way you choose to approach the study, there has been a need for the use of computational models to shed some light on how the different parts of the brain may play their part in the behavioural experiment [19,20].

Computational neuroscience has arisen to explain how electrical and chemical signals in the brain represent and process information, bridging molecular/cellular mechanisms with systems-level behavior [1]. This field employs theoretical analysis, mathematical models, and abstractions of the brain to develop computational models of the theory that governs the behaviour being investigated, and with them perform simulations of the cognitive task to provide feedback on the results acquired [1920]. Overall, the computational approach to neuroscience investigation provides a method that tries to step away from the subjective, using quantifiable data to abstract concepts such as the roles played by different structures in tasks such as learning[17].

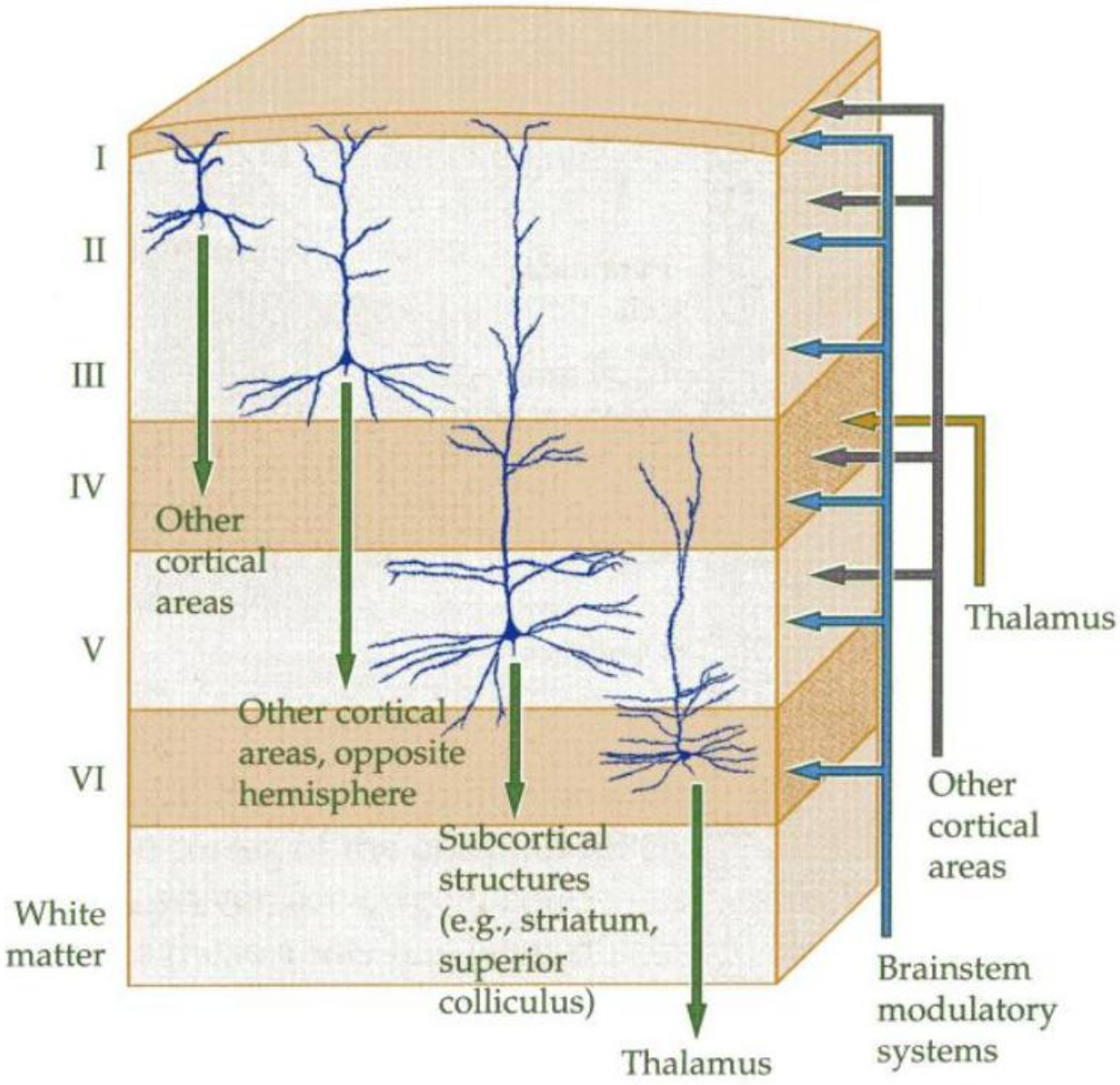

Figure 1 Diagram of the arrangement of dendrites and axons of pyramidal neurons in the different layers. Green arrows indicate projection to structures not shown in the diagram. Arrows on the right represent inputs of each layer coming from structures not shown [21]

Deep Neural Networks

As mentioned, not only have people in the field of neuroscience been interested in the brain's architecture and function. The world of computer science has also been greatly inspired by the brain for building programs and machines. Artificial neural networks (ANN) are an example of

this. ANN draws inspiration from the principle of information processing in biological systems, using as processing units mathematical representations of the neurons [22]. These networks aim to solve problems mimicking human cognition [22,23].

The artificial neuron works by receiving multiple binary inputs—analogous to how a biological neuron receives signals through its dendrites from synaptic connections with other neurons—and, through linear transformations followed by an activation function (non-linear function), produces a single output [24]. The activation function resembles the activation threshold of a neuron (it decides if the information is propagated or not). This output is then propagated to one or more subsequent neurons, depending on the network's connectivity [25]. A diagram of an artificial neuron can be seen in Figure 2.

Furthermore, ANNs are organized in layers of these artificial neurons, and the networks composed of various layers connected with each other are called Deep Neural Networks (DNN) [23]. DNN layers are arranged hierarchically, with each layer learning features from the output data of the previous one. As the depth of the network increases, successive layers are capable of capturing progressively more abstract and high-level representations [23].

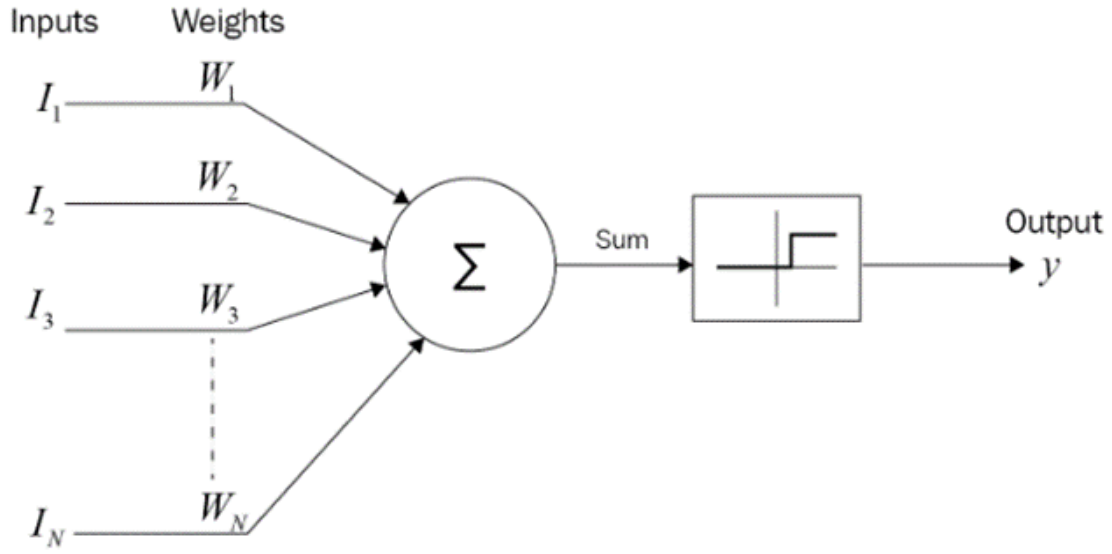

Figure 2 Diagram of an Artificial Neuron used for ANN.

Here is where inspiration from the neocortex architecture stops. Computer scientists either mimic the behavior of the brain (like the attention mechanism of transformers) or expand on mechanisms that can be added to these networks to achieve better performance. Especially on the task of image recognition, DNN architectures that have proven to be very capable are the convolutional neural networks (CNN).

CNNs are deep learning architectures formed by multiple processing layers, where the main function used is the convolution, followed by some fully connected layers of artificial neurons [26,27]. One advantage of these architectures, and the reason for them being so widely used, is that they don't need as many parameters as other DNNs since the number of fully connected layers is much smaller, giving most of the work to the convolution [27]. An example of a common CNN is shown in Figure 3.

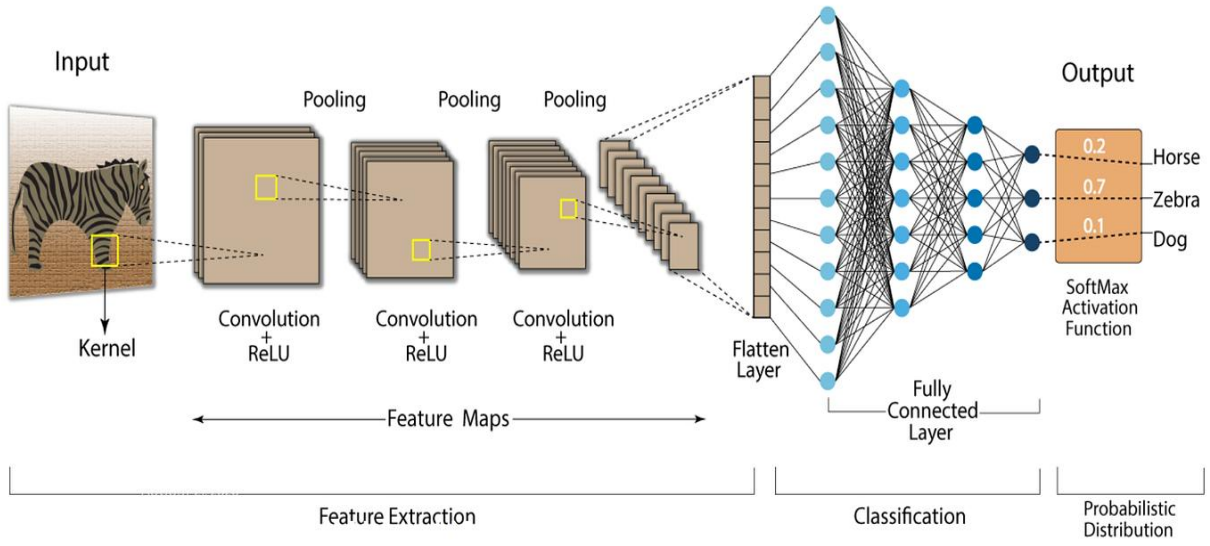

Figure 3 Example of a CNN architecture

The convolution operation of these architectures is an example of trying to mimic brain behavior, especially the visual cortex [27]. Instead of processing the entire image at once, this operation divides the 2D input into smaller patches. Each patch is individually processed to produce a localized output, and as the data flows through the network, eventually reaching the fully connected layers, the results from all these patches are combined and processed [26,27]. As mentioned, this mimics the human visual cortex as the different patches of the retina can only see a portion of what's in front of them, and later the brain interprets the image from all the different patches.

One of the main problems of DNNs, especially when used in fields that greatly impact human lives, is the lack of interpretability or transparency in how neural networks make decisions [22,27,28]. This is commonly called a “Black box” or “Black box models”. It means we can see the

inputs and the outputs, but we don't fully understand what's happening in between, how exactly the network processes the data to arrive at a particular output [29].

DNNs are "black boxes" for several reasons that make it very difficult to retrieve the "How" on their predictions. Essentially, DNNs consist of multiple layers of interconnected artificial neurons. Each neuron performs a mathematical operation on its inputs (weighted sum and activation function) and passes the result to the next layer [29]. With many layers, the initial input data undergoes numerous complex transformations. Adding so many non-linear operations makes it difficult to trace a direct, understandable path from input to output[29].

The Thousand Brains Theory (TBT)

As mentioned, the brain is responsible for multiple tasks (perception, learning, memory, decision-making…), and though its anatomy is very well documented, there are a lot of discrepancies on how it performs its different functions [19]. Overall, discrepancies between discoveries and studies make for a continuous rewriting or reinterpreting of results from past experiments and theories. One of the most recent theories being challenged is the localizationist framework, where each brain function is performed by one given discrete and isolated cortical area [30]. This framework, though very well accepted, is being challenged due to advances in neuroimaging and brain mapping techniques that have provided new insights into the function-structure organization [30]. Having this framework reviewed can also be supported by cases where people who underwent hemispherectomy can return to their regular activities with no impairment [31]. In this scene, Numenta Inc., a neuroAI company, is developing a new theory, called Thousand Brains Theory (TBT), that proposes a new framework for understanding the function of the neocortex. TBT's framework presents a full sensory-motor system (sensory information is the

input, and motor signals are the output) where the CC is the processing unit. Contrary to other frameworks, all CCs in TBT learn full models of the world based on the information they receive, instead of functioning as a step of data integration. With this new framework, Numenta aims to develop a new AI system that can operate in real-world environments with human-like capabilities, and is derived from the operating principles of the neocortex [32,33]. This challenge involves a fusion of computational neuroscience and AI.

Current AI (mostly DNNs) has achieved superhuman performance in certain domains, but it is limited by its inability to generalize across tasks or adapt dynamically to new environments without retraining [32]. Numenta's solution is to implement insights from neocortical neurobiology into AI agents[32]. These constraints include:

- Sparse representations: Unlike dense vectors used in deep learning, TBT models use sparse distributed representations (SDRs), which are robust to noise and are more efficient in data processing [32,34]. This resembles sparse firing of pyramidal neocortical neurons.
- Realistic neuron models: Models that mimic the complexity of biological neurons (like dendritic processing zones)
- Reference frames: In contrast with the brute force memorization of the DNNs, the neocortex stores knowledge in reference frames that allow predictions based on movements and sensory inputs [32].

The first version of this AI was the Hierarchical Temporal Memory (HTM) and served as the building block for the Thousand Brains Project AI called Monty, after Vernon Mountcastle [33,35]. HTM focuses on building predictive models of objects and concepts, enabling intelligent behavior in dynamic environments [7,32]. The architecture of HTM is formed by CCs, organized in

6 layers, with the input layers being composed of many mini columns (Figure 4), as the neocortex is. Each of these mini columns is composed of HTM neurons. HTM neurons try to mimic biological neurons more realistically, e.g., involving dendritic computations, unlike DNN neurons seen before. HTM neurons have three integration zones: apical and basal dendrites that process contextual information from different sources, and proximal dendrites that activate the neuron from sensory input, in a feed-forward fashion [32,36]. HTM neurons reside in one of three states: inactivated, default state (output 0); activated, fires an output signal when input matches prediction (output 1); and predictive, where contextual input predicts later activation (output 0), which maps biologically to neuronal subthreshold depolarization. Figure 5 shows an HTM neuron structure. Its working mechanism goes as follows: When a novel input is sensed, all neurons in a few given mini columns fire, and one neuron in each of the mini columns activated is chosen to be the representation of that novel input. The aggregation of the chosen neurons will form the SDR that represents the novel input, and only they will all activate when this input is received again. The predictive state of a neuron is seen when the CC is receiving input that generally preludes the state that activates said neurons (its apical and basal dendrites are receiving this input, but the proximal dendrites haven't received feed-forward sensory input yet) [7,34]. In TBT, these neurons are arranged into layers that map to neocortical laminae and will be explained later in the Methods section.

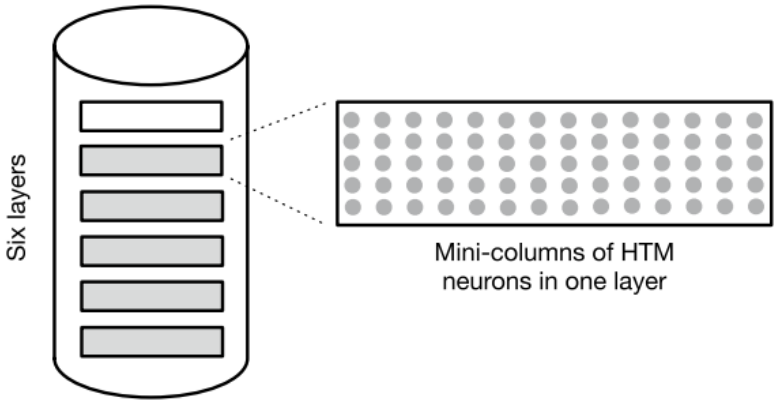

Figure 4 Organization of a CC [32]

Another key concept, which will be elaborated later, is hierarchy. In the neocortex, all CCs are organized hierarchically, which means that the output of a CC becomes the input of another. This is hypothesized to allow the neocortical hierarchy to process more complex and abstract concepts, similar to deeper layers of the CNNs. A key advantage of hierarchy is that it allows for learning of compositional objects or scenes without having to learn everything from scratch. For example, if you already know what a wheel, a motor, and windows are when you infer them because you saw a car, instead of learning what a car is from scratch, the higher order CC can integrate the different parts the lower CC is seeing to form the object of a car. This also allows for efficient allocation of models since the model of the wheel learned can be used in other objects (such as bikes) instead of learning a new wheel each time you see a different vehicle [32,36].

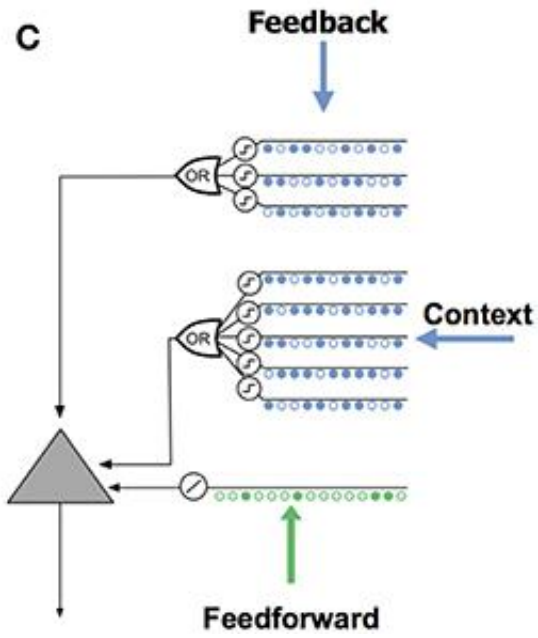

Figure 5 Sketch of an HTM neuron. Feedback and Context are the inputs to apical and basal dendrites, and the feedforward is the input to proximal dendrites. [34]

Chapter 2: Materials and Methods

In this section, we are going to introduce the space where all our models have been trained and evaluated, as well as the Dataset of objects that have been used. Furthermore, we will describe the structure, organization, and mechanism of the Python package Monty, as the code's current state of TBT.

Habitat-Sim

As we stated, the goal of TBT is to create an AI agent that can operate and interact with the real world, including in novel situations. For this reason, the agent needs a training environment that can simulate the real world to test the abilities of Monty. The software used to simulate the world and create this environment for Monty is Habitat-sim [37].

Habitat-sim is a high-performance, physics-enabled 3D simulator designed for research in AI. It provides a platform for simulating realistic interactions between agents and virtual environments that allow complex tasks like navigation or object manipulation. Habitat environments support CAD models and objects like the YCB Database [38] to be loaded for realistic simulations. Another advantage that Habitat provides is a set of highly customizable agents that can be equipped with various sensors, like RGB cameras or depth sensors, and are able to manipulate objects through physical interactions. From these sensors, we will use two: a distant agent which resembles an eye, and a touch-based agent which resembles a finger. The sensors will extract features and their locations for the model to compute for its learning (the information transmitted by these sensors acts as the encoded signals from the sensory organs). All of this provides Monty with an ideal space for training and testing before going into real-world tasks.

YCB Database

The objects that Habitat is going to load for the models to learn come from the Yale-CMU-Berkeley Object and Model set (YCB Dataset) [38].

The YCB Dataset is a comprehensive data set designed for robotic manipulation and research. It is formed by 77 high-quality digital models of objects that can be encountered in daily life, like kitchen utensils (e.g., cutlery, plates, mugs), different foods, or tools. These objects are made from high-definition RGB images as well as textured 3D mesh models (from 16k to 512k polygons for resolution) that allow for different interactions between the agents and the model. Figure 6 shows some objects of the YCB dataset in a Habitat environment.

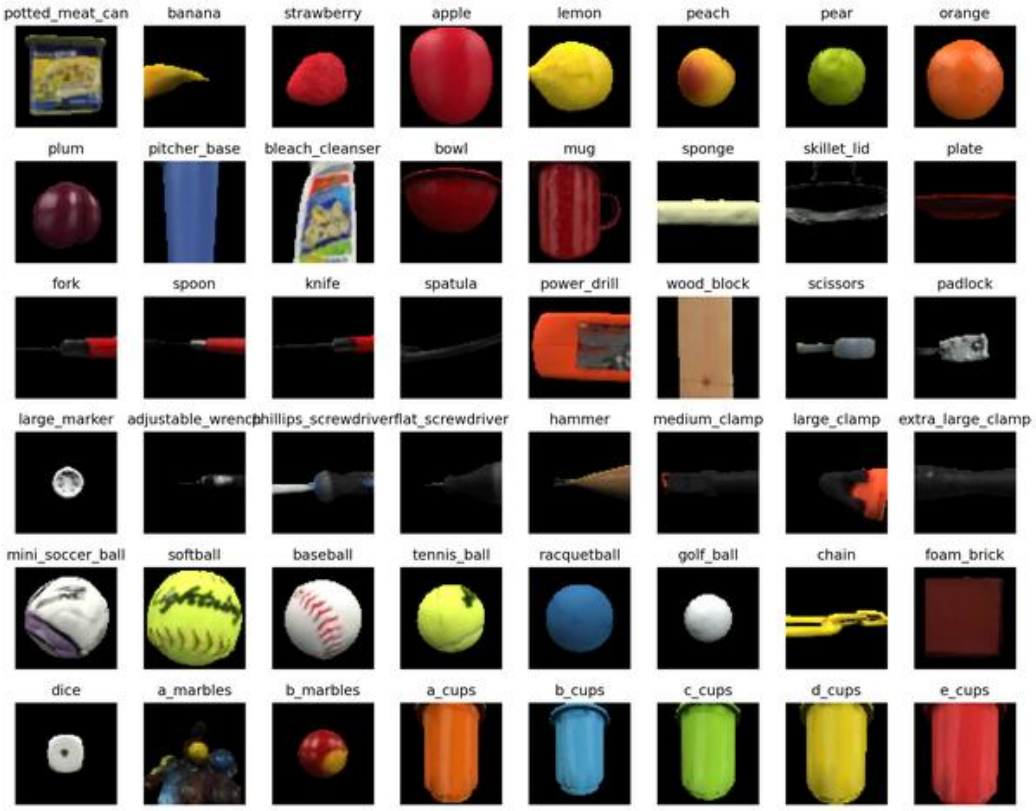

Figure 6 Objects from the YCB Dataset in a Habitat environment

Models

Now let us take a closer look at TBT and how the model encodes and learns to recognize objects [33]. On TBT, as well as in the neocortex, computations are performed by cortical columns (CCs). CCs in the TBT process sensory-motor information through the layered circuits to build structured models of objects and concepts. CCs inside the neocortex, as well as TBT, interact in two ways: horizontally or vertically. TBT proposes that horizontal connections compose a voting system where all CCs inferring the same object agree on what they are seeing from their already learned objects. Voting between columns allows the models to come to a faster and more robust decision in comparison to a single CC. On the other hand, vertical connections between CC aid in understanding more complex objects (like compositional objects or a compositional scene, which in turn are made out of other objects), and possibly in understanding and learning abstract concepts.

As we stated, CCs are composed of 6 layers; though this is well known, their function remains uncertain. TBT proposes the following functions of each layer and uses them to build the CCs on their models. Layer I communicates with higher-level CCs that pass a goal state for the current CC to achieve. The goal state is what the model aims to achieve, which can be translated into an interaction with the object. Layer II/III integrates feed-forward sensory inputs with feedback from higher regions and lateral connections. Voting takes place in layer II/III. On this layer, models of the object are made and stored; in the current version of TBT, these models are graphs in 3D Cartesian space. Layer IV receives sensory information about the features of an object, e.g., the edge of a cube, and distributes it across the layers. It acts as the entry point for sensorimotor data, which is then mapped to locations in an object-centric coordinate system. Layer V generates motor commands and communicates predictions to subcortical regions. This

can be an action proposal for the sensors to resolve the uncertainty of the object. For example, when differentiating between a fork and a knife, it will propose to go to the head where the differentiating features lie. Layer VI calculates an allocentric location signal, representing where a sensed feature is positioned relative to the object being explored. During movement, Layer VI updates the spatial coordinates to stay consistent in object-centric models. For example, when a finger moves across a mug's surface, Layer VI ensures the mug's handle remains mapped to the same relative location despite sensor displacement. The location signal is passed to Layer IV for integration with the features sensed, allowing for spatial context. The idea of a location signal generated by this layer comes from grid-cell-like mechanisms, a key point in this theory. The notion of having grid-cell-like mechanisms in the neocortex comes from analogies to entorhinal grid cells and their role in path integration; Mountcastle's principle of cortical uniformity, which states similar computational mechanisms across cortical regions; and observations of L6's connectivity that align with reference frame maintenance.

Even though the functions of the different layers are theorized, some are not fully developed or polished, so in the current version of Monty (Version 0.0.3), for object recognition, only Layers II/III, IV, and VI are implemented (figure 7).

In this Project, we ran different experiments that asses some of the learning and inferring capabilities of Monty in the task of object recognition. All models used a sensor module (SM), called the viewfinder, which helps the model stay on the object when inferring. No learning was

done through the viewfinder. Figure 8 shows different views from various SMs.

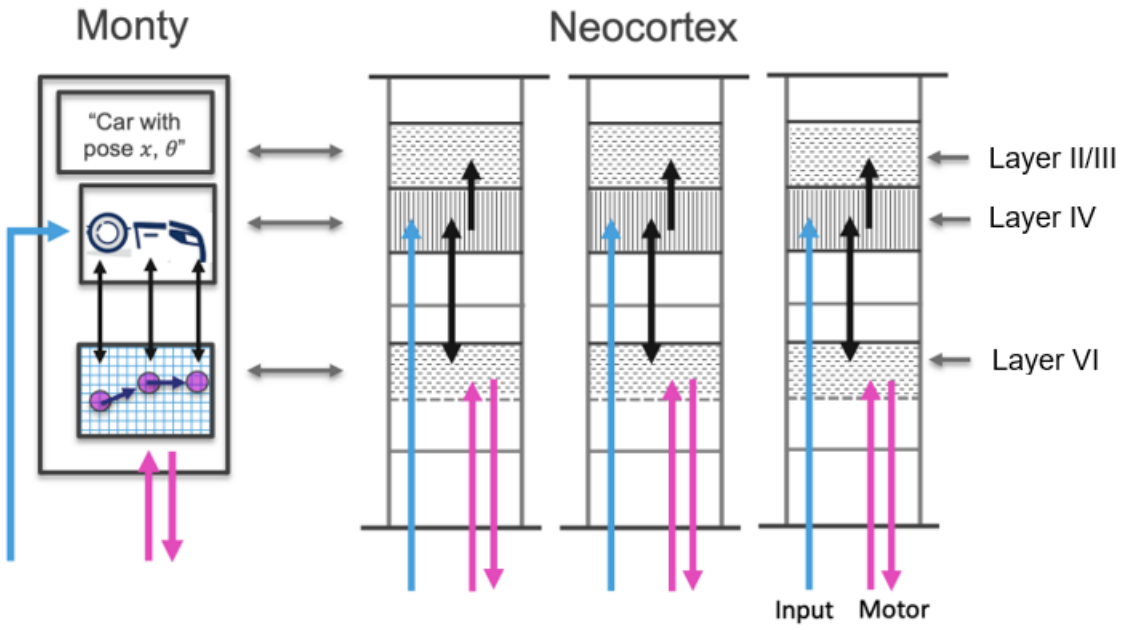

Figure 7 Properties from the TBT implemented in Monty [33]

The experiments done were the following:

- To establish that our simulation setup and implementation are correct, we built two models composed of one SM and one learning module (LM, they are Monty's CCs). One model received information from a Distant agent (simulating information coming from the eye) and the other from a Surface agent (simulating information coming from a finger). These two models were trained on six objects, for six epochs (one epoch per object), 14 episodes per epoch (they were trained on all the views of a cube, meaning its 6 faces and 8 vertexes), and 500 steps per episode (each agent took 500 samples of the object in each view). The action policy used in the models was a *Curvature-Informed surface* (where the agent follows the principal curvature of its starting point in the object) for the surface agent and a *Naïve Scan Spiral* (where the agent takes random

samples of the object) for the distant agent. To evaluate the models, the models had to infer all the objects learned with 2 distractor objects around and some added noise. We used objects that are different, e.g., a mug and a skillet lid, as well as objects that are similar to each other (like the fork, knife, and spoon).

- To assess the capabilities of the voting system, we implemented another two models. One model had two SMs and two LMs. Each SM is connected to an LM, and the two LMs are connected laterally to each other through layerII/III. The second model had five SMs and five LMs, where each SM feeds sensory information to an LM, and the five LMs are connected together horizontally. Both models used distant agents, as well as the same training parameters as the first model. For the 5-LM model, when evaluating an object, once 3 out of the 5 LMs agree on an object through voting, it is consider that the model knows what the object is.
- The third experiment used two models with the same properties as the 2-LM before, but with 1000 and 1500 steps per episode this time. Here, we aimed to assess the level of confidence the model exhibited when trained for a longer period.
- The last experiment we did was a 2 LM in a hierarchy structure, where the top LM takes input from the lower LM. This experiment had similar parameters to the ones before and also received information from a distant agent.

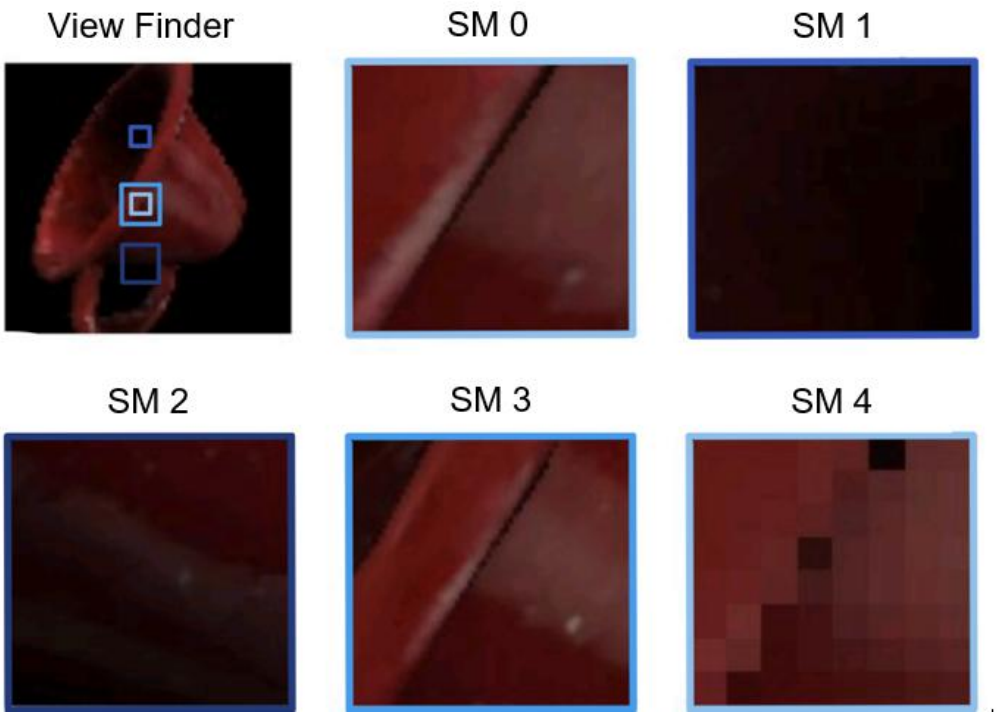

Figure 8 Different Views of the multiple SMs and the Viewfinder of a mug [33]

All experiments are of a supervised fashion to make it simpler for us to control what the model learns. However, we still did a simple unsupervised experiment to get a grasp of how this system is aimed to perform (as a continuous learning AI). This model has one SM and one LM with a surface agent. The model was trained on 2 objects that were spawned in the environment with a random rotation. Every time the model encounters an object, it will take 100 steps to infer it. If this object has already been learned by the model, it will take another 1000 steps to update this object in its memory. On the other hand, if the model does not recognize the object, it will store it as a novel object in its memory and then take 1000steps to be more accurate.

Chapter 3: Results

To address the performance and capabilities of AI based on the TBT framework of the neocortex, we created four different architectures and trained them using various experiment parameters. The architectures are:

1- Two models with one LM: a model with a distant agent, and a model with a surface agent.

2- A model with a Distant sensing agent with 2 LMs horizontally connected.

3- A model with a Distant sensing agent and five LMs horizontally connected.

4- A model with a Distant sensing agent and two LMs that are vertically connected.

All models were trained on detailed 3D models of everyday objects, as described in the Methods section. Below, we present the learned object models and their performance in environments with distractor objects.

Training and Models

According to the TBT framework, a single CC should be able to build its model of an observed object and, with it, perform inference by itself. Therefore, we evaluated whether a model with one LM could learn and recognize multiple objects without relying on vertical or horizontal connections. This applies to both types of agents the SM can be connected to (Distant or Surface). Figure 9 shows the model of a mug made by both one LM architectures after the training for 1 epoch and 14 episodes on the mug only. This model is what the LM will use to infer a mug if encountered again. The color of the locations in the graph models is the confidence of the model in that point. Here we see how LMs were able to make this graph model by seeing the object once from all 14 views of a cube.

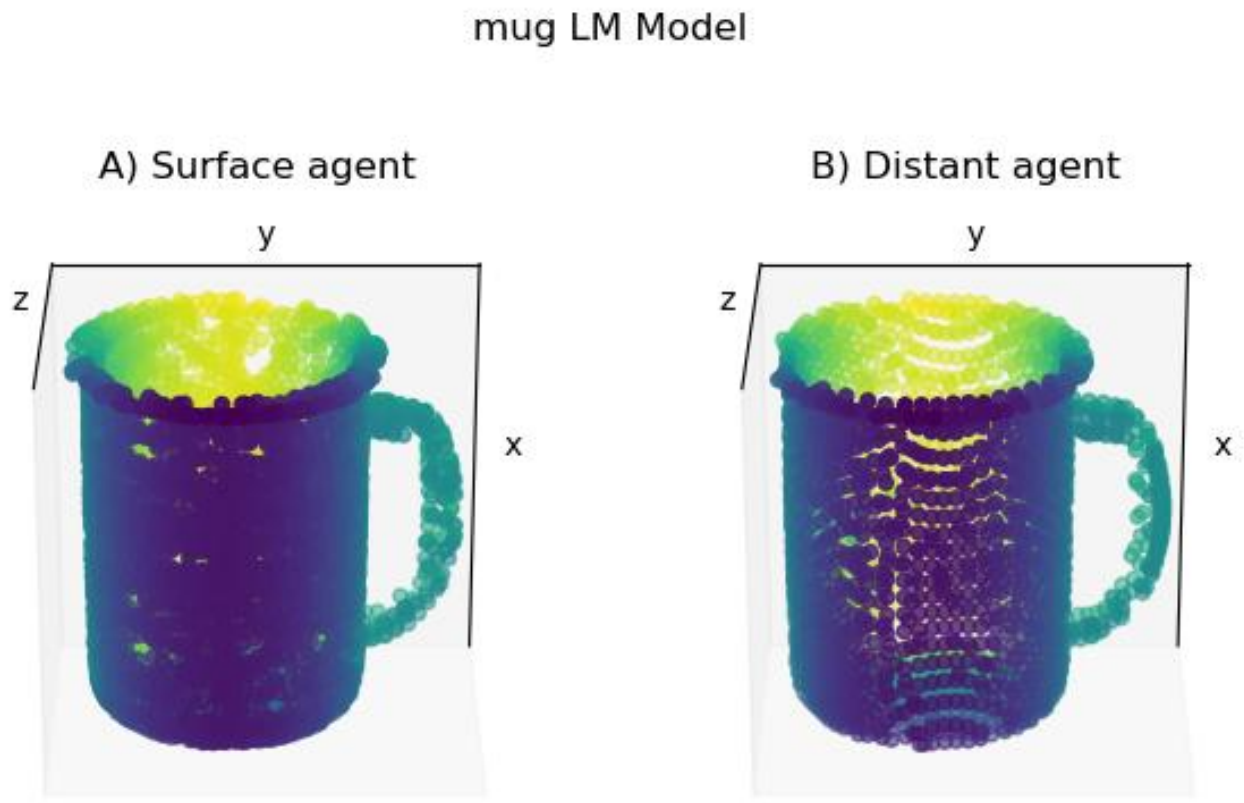

Figure 9 Graph model of a mug made by A) one LM and one SM connected to a surface agent and B) one LM and one SM connected to a distant agent. Both trained over 14 episodes (six faces + 8 vertices of a cube) and 500 sensory steps in each episode. One episode will be one view of the object.

One observation to point out is that surface agents give more information to the SMs and are better at computing distances on the object. In addition, the surface agent model used the *Curvature-Informed surface* action policy (only available for surface agents). This is why the surface agent model looks smoother.

Though the single LM showed no problem in learning more than one object, it showed some confusion when learning thinner objects like cutlery, giving them some extra thickness in their graphs. However, from this model, you can still perform inference. Figure 10 illustrates the models of a fork made by both models. Moreover, given that the distant agent is not in contact with the object, it can lose some inference steps resulting in a sparser model (Figure 10. B)

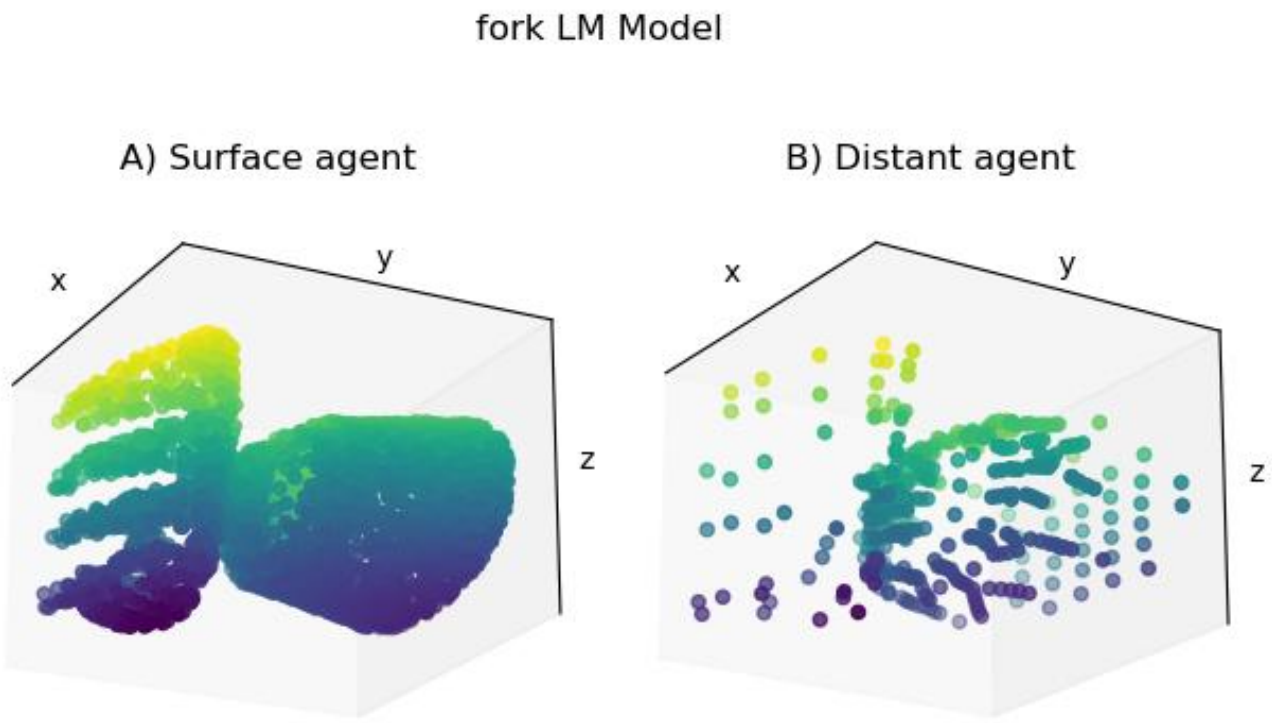

Figure 10 Graph model of a fork made by A) one LM and one SM connected to a surface agent and B) one LM and one SM connected to a distant agent.

One object that we saw in the YCB database that seemed interesting to investigate to grasp the scope of differences between distant and surface agents was the skillet lid. Since most of the object was transparent, except for the metal ring that delimited the object and the handle, we thought that models trained on the different agents would produce different results. We hypothesized that the model with the distant agent would only model the metal ring and the handle, whereas the model with the surface agent would graph the whole object (even if it is transparent or not). To our surprise, both LMs graph the skillet lid the same way, only depicting the handle and the metal ring. This similarity in the models is most likely due to the viewfinder, which we will discuss in the next section. Figure 11 shows both models of the skillet lid.

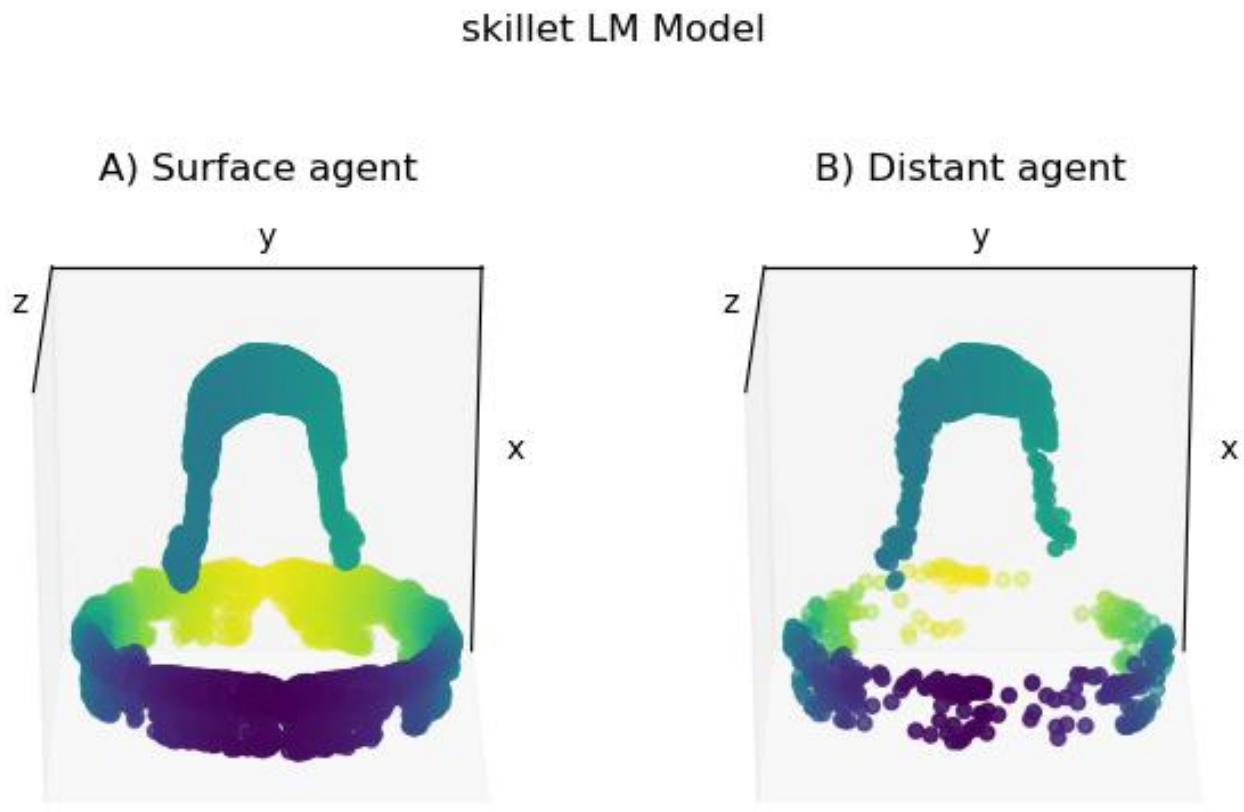

Figure 11 Graph model of a skillet made by A) 1 LM and 1 SM connected to a distant agent and B) 1 LM and 1 SM connected to a surface agent.

Before jumping into models with multiple LMs and SMs, we trained one model with one SM and one LM in an unsupervised fashion to examine the continuous learning capability of Monty models while performing sensory motor tasks (just like humans do). This experiment was performed in 3 different time steps (3 epochs), where in each epoch the model needed to infer the object, and once it knew what it was, it would explore the object a bit further to update the model of the object to a more precise one. We can see the evolution of the graph model of the object in Figure 12, where epoch 0 represents the model of the bowl after seeing it for the first time, and epochs 2 and 3 represent the updates of that model after encountering it again.

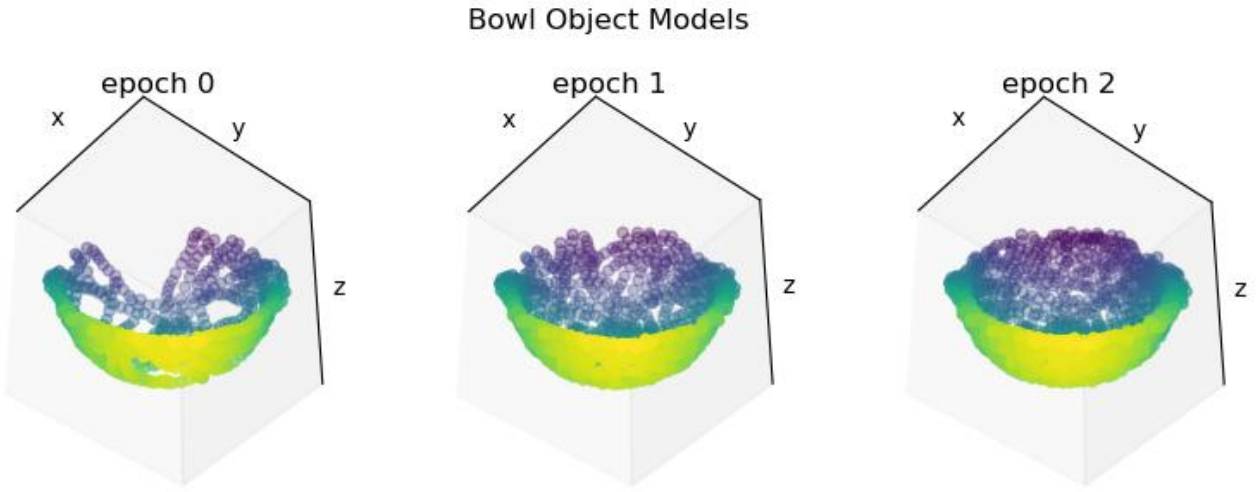

Figure 12 Graph model of a bowl at the end of each epoch. Epoch 0 would be the first time the model sees the object, recognizing it as a novel object and saving the new object in memory for it to be updated when it is encountered again (Epochs 1 and 2)

After assessing what one CC can do on its own, we proceeded to train 2 models, one with two LMs and another with five LMs. With these models, we sought to see the efficiency of the voting system, as well as later on, the difference between having CCs horizontally or vertically connected. In the 2-LM and the 5-LM models, each LM learns its own model of the object. In TBT, CCs don't integrate pieces of the scene between each other (to then pass it to the next level of the hierarchy). Instead, they vote on what object is being seen. For a voting system to happen, each CC needs to identify the object they are seeing. Figure 13 shows the object seen by each of the LMs in a 5-LM architecture. The object seen by each of the two LMs in the 2-LM architecture is depicted in Figure 14.

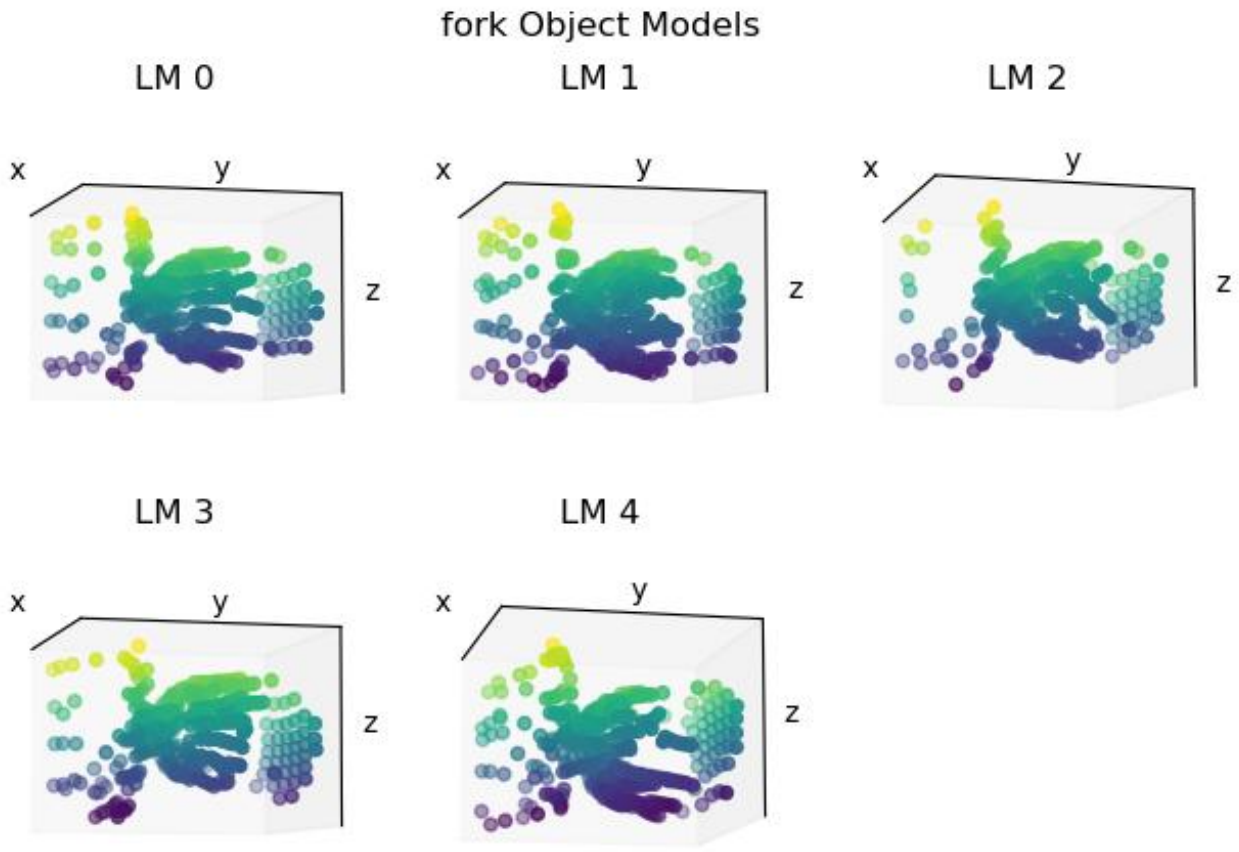

Figure 13 Graph models of a fork from each LMs in the 5-LM architecture. See how not all graph models are exactly the same even though they learned from the same object at the same time.

The 2-LM and 5-LM models were able to correctly infer the objects after 500 steps per episode. However, we wanted to examine how adding more training steps would affect performance. In the training models, having more steps in the episodes translated into more detailed graph models, as was expected (Figure 14).

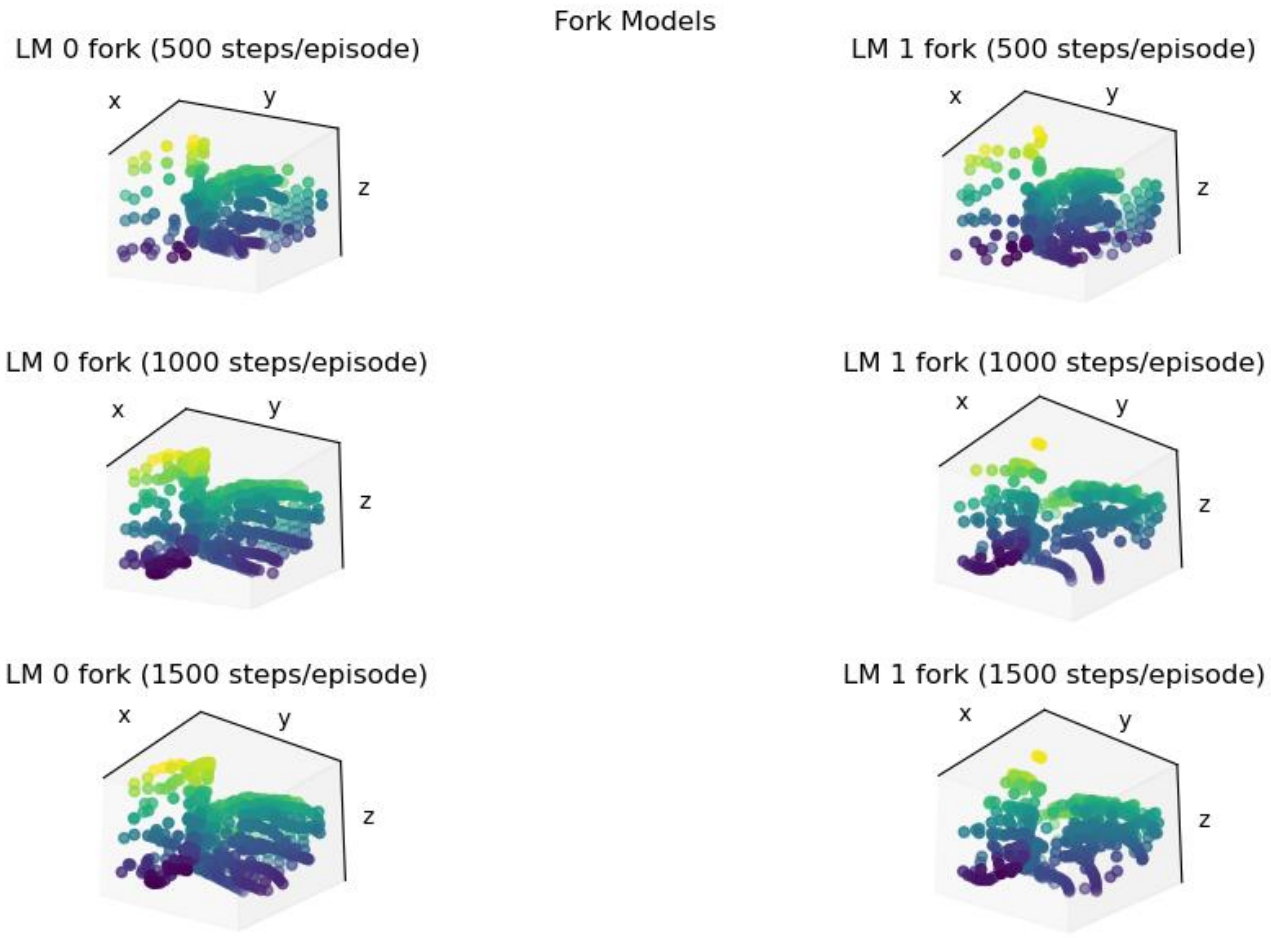

Figure 14 Graph models of each LM in a 2-LM architecture, trained with 500 steps per episode (top), 1000 steps per episode (middle), and 1500 steps per episode (bottom).

Testing

Now that we know the models have learned the objects, it is time to put them to the test. The test consists of seeing if the models can identify the objects they have learned, even in the presence of noise and distraction objects (objects it has never seen before).

In Table 1, we have the results of the evaluation of the 1 LM surface model. Here we see that the model can correctly infer most of the objects in few steps. This model performs very well on the objects whose graphs had no irregularities; however, on the thin, smaller objects, like the cutlery, it gets more confused, sometimes not being able to identify them in the step limit we used (1000 steps). This limitation was observed across all models and warrants further investigation to

improve performance on such objects. In addition, surface agents' more detailed graphs perform better on these confusing objects than distant agents, even in the more robust models (5-LM model Table 2). Moreover, the more information the surface agents gives makes models with this agent faster in the inference than models with distant agents even when the latter has a higher number of LMs (shown in the steps column of tables 1 and 2).

Column 1	primary_performance	num_steps	result	most_likely_object	primary_target_object
LM_0	correct	26	mug	mug	mug
LM_0	correct	33	c_toy_airplane	c_toy_airplane	c_toy_airplane
LM_0	correct	27	fork	fork	fork
LM_0	confused	29	mug	mug	knife
LM_0	correct	35	spoon	spoon	spoon
LM_0	correct	30	skillet	skillet	skillet
LM_0	confused	33	skillet	skillet	mug
LM_0	correct	26	c_toy_airplane	c_toy_airplane	c_toy_airplane
LM_0	confused	54	mug	mug	fork
LM_0	confused	324	spoon	spoon	knife
LM_0	correct	247	spoon	spoon	spoon
LM_0	correct	21	skillet	skillet	skillet
LM_0	correct	26	mug	mug	mug
LM_0	correct_mlh	14	['c_toy_airplane']	c_toy_airplane	c_toy_airplane
LM_0	correct	51	fork	fork	fork
LM_0	correct	87	knife	knife	knife
LM_0	confused	21	mug	mug	spoon
LM_0	correct	26	skillet	skillet	skillet

Table 1 Model of 1 LM with a surface agent performance. The first column shows the LM that is giving the rows results, in this case is always LM_0 since there is only one LM in this architecture. The second column shows if the output of the LM was correct or if model confused the object with another. Here, mlh means most likely hypothesis. Whenever mlh is shown means that the model didn't reach a full decision in the steps given in the training but it had one preferred candidate. The third column illustrates how many steps it took for the LM to reach the result. The forth column is the result given by the LM. If the result is in brackets is because the model didn't reach a result. Inside the brackets will be the objects that the model hasn't discarded as possible candidate. The fifth column shows the most likely result at the end of the evaluation. It is the same as the results except when there are many objects in the brackets, in this situation it will just show the most likely object of the list. The last column is just the object that the LM is trying to infer.

Now we step onto the evaluation of the models with more than 1 LM. With the voting system architectures with many LMs are able to achieve either higher speed or robustness to their predictions. In our examples, the 5-LM model can only make an inference if 3 out of the 5 LMs

agree upon the object they are seeing, which makes the model more robust. On the other hand, the 2-LM model decides as soon as 1 LM recognizes the object (even if it's wrong), which adds speed to the model. Table 2 shows the evaluation of the 5-LM model and Table 3 of the 2-LM model.

Column 1	primary_performance	num_steps	result	most_likely_object	primary_target_object
LM_0	correct	37	mug	mug	mug
LM_1	correct	33	mug	mug	mug
LM_2	correct	34	mug	mug	mug
LM_3	correct_mlh	34	['mug']	mug	mug
LM_4	correct_mlh	30	['mug']	mug	mug
LM_0	correct	24	banana	banana	banana
LM_1	correct	13	banana	banana	banana
LM_2	correct_mlh	28	['banana', 'skillet']	banana	banana
LM_3	correct	23	banana	banana	banana
LM_4	correct_mlh	26	['banana', 'skillet']	banana	banana
LM_0	confused	40	knife	knife	fork
LM_1	confused_mlh	2	['fork', 'knife', 'spoon']	spoon	fork
LM_2	confused	31	knife	knife	fork
LM_3	correct_mlh	27	['fork', 'knife']	fork	fork
LM_4	confused	18	spoon	spoon	fork
LM_0	confused	100	fork	fork	knife
LM_1	confused_mlh	71	['fork', 'knife']	fork	knife
LM_2	confused	61	fork	fork	knife
LM_3	confused	73	fork	fork	knife
LM_4	confused_mlh	33	['fork']	fork	knife
LM_0	correct	50	spoon	spoon	spoon
LM_1	correct	11	spoon	spoon	spoon
LM_2	confused	15	mug	mug	spoon
LM_3	correct_mlh	39	['fork', 'spoon']	spoon	spoon
LM_4	correct_mlh	16	['spoon']	spoon	spoon
LM_0	confused	60	spoon	spoon	skillet
LM_1	confused_mlh	25	['spoon']	spoon	skillet
LM_2	confused	43	spoon	mug	skillet
LM_3	correct	32	skillet	skillet	skillet
LM_4	confused_mlh	30	['spoon']	spoon	skillet

Table 2 Model of 5 LM with a distant agent performance. Notice how in column 4 some LMs have various hypotheses for the object (all these hypotheses are inside the brackets as described before), but in colmn 5 it is only shown the most likely for that LM.

One interesting finding is that the models trained for 1,000 (Table 4) and 1,500 steps per episode (Table 5) perform slightly worse on objects that have posed challenges in previous models compared to the model trained for 500 steps per episode (Table 3), though the difference is minimal. However, they did achieve faster processing on correctly inferred objects, which is the expected outcome. Speed in these evaluations is measured by how many steps it took the model to reach a decision.

Column1	primary_performance	num_steps	result	most_likely_object	primary_target_object
LM_0	correct	41	knife	knife	knife
LM_1	correct_mlh	14	['knife', 'fork']	knife	knife
LM_0	correct	49	spoon	spoon	spoon
LM_1	correct_mlh	15	['spoon']	spoon	spoon
LM_0	correct	69	fork	fork	fork
LM_1	correct_mlh	30	['knife', 'fork']	fork	fork
LM_0	correct	26	mug	mug	mug
LM_1	correct	26	mug	mug	mug
LM_0	correct	26	bowl	bowl	bowl
LM_1	correct_mlh	26	['bowl']	bowl	bowl
LM_0	correct	47	skillet	skillet	skillet
LM_1	confused_mlh	20	['spoon', 'skillet']	spoon	skillet

Table 3 Model of 2 LM with a distant agent trained for 500 steps per episode performance

Column1	primary_performance	num_steps	result	most_likely_object	primary_target_object
LM_0	correct_mlh	41	['knife']	knife	knife
LM_1	correct	35	knife	knife	knife
LM_0	correct_mlh	37	['spoon']	spoon	spoon
LM_1	correct	10	spoon	spoon	spoon
LM_0	correct	48	fork	fork	fork
LM_1	correct_mlh	20	['knife', 'fork']	fork	fork
LM_0	correct	23	mug	mug	mug
LM_1	correct	23	mug	mug	mug
LM_0	correct	26	bowl	bowl	bowl
LM_1	correct_mlh	25	['bowl']	bowl	bowl
LM_0	confused_mlh	55	['spoon', 'skillet']	spoon	skillet
LM_1	confused	28	spoon	spoon	skillet

Table 4 Model of 2 LM with a distant agent trained for 1000 steps per episode performance

Column1	primary_performance	num_steps	result	most_likely_object	primary_target_object
LM_0	correct	41	knife	knife	knife
LM_1	correct_mlh	21	['knife', 'fork']	knife	knife
LM_0	correct	32	spoon	spoon	spoon
LM_1	correct_mlh	7	['spoon']	spoon	spoon
LM_0	confused	65	knife	knife	fork
LM_1	correct_mlh	12	['knife', 'fork']	fork	fork
LM_0	correct	21	mug	mug	mug
LM_1	correct	19	mug	mug	mug
LM_0	correct_mlh	30	['bowl']	bowl	bowl
LM_1	correct	30	bowl	bowl	bowl
LM_0	confused_mlh	30	['spoon', 'fork', 'bowl', 'skillet']	spoon	skillet
LM_1	confused	20	spoon	spoon	skillet

Table 5 Model of 2 LM with a distant agent trained for 1500 steps per episode performance

Hierarchy

Finally, to start examining the effects of hierarchy, our last model was one with two vertically stacked LMs. Although hierarchy is not needed for simple objects like the ones we have seen, this first experiment can still shed some light on how stacked CCs could work to achieve abstraction or learn compositional objects and scenes. In Figure 15, we see how the higher-level LM has a sparser model of the object, which can mean some level of abstraction that we plan on addressing in future work.

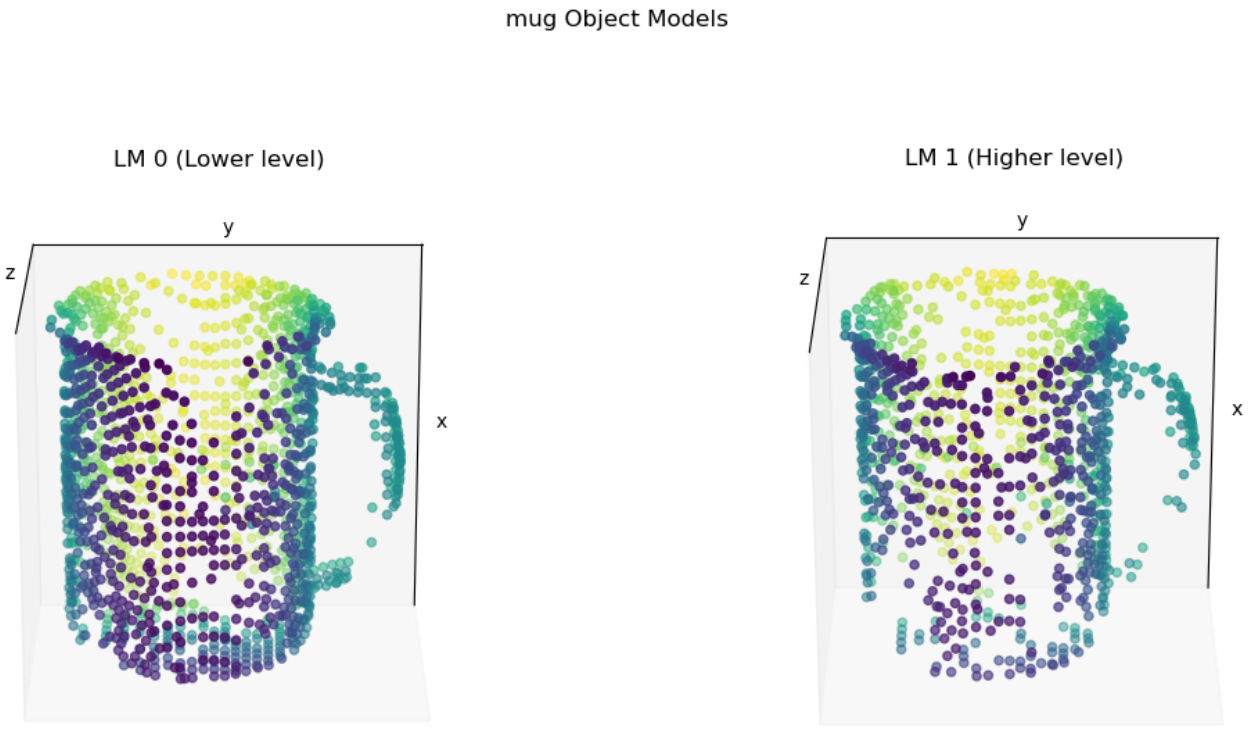

Figure 15 Graph models of the Lower-Level LM (left) and the Higher-Level LM (right). Here, it can be seen that the Higher-Level LM has a sparser view of the object.

Chapter 4: Discussion and Conclusions

Our study has reviewed the current state of AI as well as the current knowledge of the neocortex structure and functions, and with it, we arrived at TBT, intending to examine how it performs as a biologically constrained AI, especially capable of continual learning and interacting with the world. At this scenario, we used Monty (the repository for creating the models with TBT) [35] to run different experiments that may asses the capabilities of different elements and models that can be used, as well as first look at continual learning and hierarchy.

Our experiments demonstrated that a single CC is sufficient for learning and inferring multiple simple objects. However, the model currently struggles with thin and small objects, indicating areas for further improvement. The objects learned and stored by the CCs are used to interpret whether the sensory information coming in is a known object or a novel one. We also found that in comparison to Distant agents, Surface agents make better/smoother models of the objects (figure 10), are more often correct, and perform a faster inference, especially on objects that posed more challenges in both training and evaluation (shown in comparison between tables 1 and 2). Further evaluation of both agents led us to investigate objects that are mostly transparent (like the skillet lid). We hypothesized that these objects would be completely modeled by the architecture with the surface agent and only the non-transparent parts by the architecture with the distant agent. Contrary to our hypothesis, both architectures only modeled the non-transparent parts; we theorized that this is likely due to the viewfinder. The viewfinder is a type of agent, from which no learning is done, that has a general view of the object, which is used to put the other agents on the object and not outside of it. This is useful to not waste steps and views of our training finding the object, but since this agent is a camera-like agent (like distant agents), it

can't see the transparent parts of the object, making the surface agent never sense the transparent part.

Our training on multiple architectures with multiple LMs further showed that every CC sees the world through its own lens, which is demonstrated by each having an individual model of the object. Thus, architectures with multiple LMs connected horizontally do not aid each other during training, but they do provide other benefits in evaluation. During evaluation, multiple CCs talk to each other through the connections in Layer II/III, voting on the object they are inferring. Thanks to this voting system, horizontally connected CCs add speed and robustness to the architecture. Robustness is added because a certain number of CCs need to agree on what they are seeing, which makes misclassification harder. Another way to achieve some degree of robustness is to train with higher exploratory steps; this way, you get more detailed models; nonetheless, if the learning is not performed correctly, this could turn into misclassification. On the other hand, speed is achieved by having different agents start in different places of the object; therefore, they have a higher probability that some of them have landed on or near the unique features of the object.

Finally, we did two additional experiments to grasp what could be Monty's ability for continual learning and a first view into how hierarchy can achieve abstraction. The unsupervised task showed Monty's continual learning capability, where the model gets better performance the more it interacts with the world, setting a precedent towards an ability to perform in multiple fields. The second experiment was a 2-LM vertically connected architecture. This experiment aims to shed some light on the abstraction capabilities of Monty. Comparing the lower-level LM with the higher-level LM, we observed that the higher-level module made sparser models of the object, which supports possible abstraction through hierarchies, and possibly the first step for

learning compositional objects or scenes. Learning a compositional object under TBT could come from the lower-level CCs knowing the components the scene is made from, and the higher-level CCs understanding the full object. For this to happen, the higher-level CCs need not learn the specifics of the simpler objects but just understand where they are organized and how they relate to each other. This way, the models of simpler objects can be recycled into different objects, eliminating the need to learn from scratch. This will be the focus of future work.

Taking all the experiments into account, TBT rises as a framework to be further studied and developed for the neocortex mechanisms. Though it needs further advancements in the theory to explain more complex features, and in Monty, to solve the current problems and keep up with their theory's advancements. Furthermore, it also poses an alternative path of study for AI systems.

In conclusion, TBT presents a promising approach for advancing AI by emphasizing sensorimotor integration, where machines learn through interaction with their environment, which is a scarce set of training data. This scarce dataset contrasts with the need for current AIs to train on large databases to achieve high performance. Even if TBT's neuroscience proves to be incomplete, its core ideas can inspire more robust and adaptable AI systems. Sensory-motor systems are essential for developing machines that can understand and operate effectively in the real world.

References

1. Sejnowski TJ, Koch C, Churchland PS. Computational Neuroscience. *Science*. 1988;241(4871):1299-1306. doi:10.1126/science.3045969

2. Hirani R, Noruzi K, Khuram H, et al. Artificial Intelligence and Healthcare: A Journey through History, Present Innovations, and Future Possibilities. *Life*. 2024;14(5):557. doi:10.3390/life14050557

3. Creutzfeldt OD. Generality of the functional structure of the neocortex. *Naturwissenschaften*. 1977;64(10):507-517. doi:10.1007/BF00483547

4. Lui JH, Hansen DV, Kriegstein AR. Development and evolution of the human neocortex. *Cell*. 2011;146(1):18-36. doi:10.1016/j.cell.2011.06.030

5. Douglas RJ, Martin KAC. Neuronal circuits of the neocortex. *Annu Rev Neurosci*. 2004;27:419-451. doi:10.1146/annurev.neuro.27.070203.144152

6. Mountcastle VB. The columnar organization of the neocortex. *Brain*. 1997;120(4):701-722. doi:10.1093/brain/120.4.701

7. Hawkins J, Ahmad S, Cui Y. A Theory of How Columns in the Neocortex Enable Learning the Structure of the World. *Front Neural Circuits*. 2017;11. doi:10.3389/fncir.2017.00081

8. Huang S, Wu SJ, Sansone G, Ibrahim LA, Fishell G. Layer 1 neocortex: Gating and integrating multidimensional signals. *Neuron*. 2024;112(2):184-200. doi:10.1016/j.neuron.2023.09.041

9. Guy J, Staiger JF. The Functioning of a Cortex without Layers. *Front Neuroanat*. 2017;11. doi:10.3389/fnana.2017.00054

10. Welker C, Woolsey TA. Structure of layer IV in the somatosensory neocortex of the rat: Description and comparison with the mouse. *J Comp Neurol*. 1974;158(4):437-453. doi:10.1002/cne.901580405

11. Scala F, Kobak D, Shan S, et al. Layer 4 of mouse neocortex differs in cell types and circuit organization between sensory areas. *Nat Commun*. 2019;10(1):4174. doi:10.1038/s41467-019-12058-z

12. Moberg S, Takahashi N. Neocortical layer 5 subclasses: From cellular properties to roles in behavior. *Front Synaptic Neurosci*. 2022;14. doi:10.3389/fnsyn.2022.1006773

13. Thomson AM. Neocortical layer 6, a review. *Front Neuroanat*. 2010;4. doi:10.3389/fnana.2010.00013

14. Horton JC, Adams DL. The cortical column: a structure without a function. *Philos Trans R Soc B Biol Sci*. 2005;360(1456):837-862. doi:10.1098/rstb.2005.1623

15. Kruggel F. The macro-structural variability of the human neocortex. *NeuroImage*. 2018;172:620-630. doi:10.1016/j.neuroimage.2018.01.074

16. Gazzaniga MS. Principles of human brain organization derived from split-brain studies. *Neuron*. 1995;14(2):217-228. doi:10.1016/0896-6273(95)90280-5

17. Buzsáki G. The Brain–Cognitive Behavior Problem: A Retrospective. *eNeuro*. 2020;7(4). doi:10.1523/ENEURO.0069-20.2020

18. Krakauer JW, Ghazanfar AA, Gomez-Marin A, MacIver MA, Poeppel D. Neuroscience Needs Behavior: Correcting a Reductionist Bias. *Neuron*. 2017;93(3):480-490. doi:10.1016/j.neuron.2016.12.041

19. Kriegeskorte N, Douglas PK. Cognitive computational neuroscience. *Nat Neurosci*. 2018;21(9):1148-1160. doi:10.1038/s41593-018-0210-5

20. Khaleghi A, Mohammadi MR, Shahi K, Nasrabadi AM. Computational Neuroscience Approach to Psychiatry: A Review on Theory-driven Approaches. *Clin Psychopharmacol Neurosci*. 2022;20(1):26-36. doi:10.9758/cpn.2022.20.1.26

21. AliceD. Answer to "Folding (wrinkles) in cortex: Why is surface area more important than volume?" Psychology & Neuroscience Stack Exchange. December 10, 2014. Accessed April 9, 2025. https://psychology.stackexchange.com/a/8811

22. Janiesch C, Zschech P, Heinrich K. Machine learning and deep learning. *Electron Mark*. 2021;31(3):685-695. doi:10.1007/s12525-021-00475-2

23. Rusk N. Deep learning. *Nat Methods*. 2016;13(1):35-35. doi:10.1038/nmeth.3707

24. Forghani R. *Machine Learning and Other Artificial Intelligence Applications, An Issue of Neuroimaging Clinics of North America, E-Book: Machine Learning and Other Artificial Intelligence Applications, An Issue of Neuroimaging Clinics of North America, E-Book*. Elsevier Health Sciences; 2020.

25. Chang CH. Deep and Shallow Architecture of Multilayer Neural Networks. *IEEE Trans Neural Netw Learn Syst*. 2015;26(10):2477-2486. doi:10.1109/TNNLS.2014.2387439

26. Campanini García DA. Detección de objetos usando redes neuronales convolucionales junto con Random Forest y Support Vector Machines. Published online 2018. Accessed April 8, 2025. https://repositorio.uchile.cl/handle/2250/167863

27. Alzubaidi L, Zhang J, Humaidi AJ, et al. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *J Big Data*. 2021;8(1):53. doi:10.1186/s40537-021-00444-8

28. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat Mach Intell*. 2019;1(5):206-215. doi:10.1038/s42256-019-0048-x

29. Blazek PJ. Why we will never open deep learning's black box. Towards Data Science. March 2, 2022. Accessed April 8, 2025. https://towardsdatascience.com/why-we-will-never-open-deep-learnings-black-box-4c27cd335118/

30. Herbet G, Duffau H. Revisiting the Functional Anatomy of the Human Brain: Toward a Meta-Networking Theory of Cerebral Functions. *Physiol Rev*. 2020;100(3):1181-1228. doi:10.1152/physrev.00033.2019

31. Höller Y, Versace V, Trinka E, Nardone R. Functional connectivity after hemispherectomy. *Quant Imaging Med Surg*. 2020;10(5):1174-1178. doi:10.21037/qims.2020.03.17

32. Hole KJ, Ahmad S. A thousand brains: toward biologically constrained AI. *SN Appl Sci*. 2021;3(8):743. doi:10.1007/s42452-021-04715-0

33. Clay V, Leadholm N, Hawkins J. The Thousand Brains Project: A New Paradigm for Sensorimotor Intelligence. Published online December 24, 2024. doi:10.48550/arXiv.2412.18354

34. Hawkins J, Ahmad S. Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex. *Front Neural Circuits*. 2016;10. doi:10.3389/fncir.2016.00023

35. thousandbrainsproject/tbp.monty. Accessed April 14, 2025. https://github.com/thousandbrainsproject/tbp.monty

36. Hawkins J, Lewis M, Klukas M, Purdy S, Ahmad S. A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex. *Front Neural Circuits*. 2019;12. doi:10.3389/fncir.2018.00121

37. facebookresearch/habitat-sim. Accessed April 11, 2025. https://github.com/facebookresearch/habitat-sim

38. Object Set | YCB Benchmarks – Object and Model Set. Accessed April 11, 2025. https://www.ycbbenchmarks.com/object-set/