The Thousand Brains Project

Numenta, 2024
Authors: Viviane Clay, Niels Leadholm, and Jeff Hawkins

Contents

Overview

We are developing a platform for building AI and robotics applications using the same principles as the human brain. These principles are fundamentally different from those used in deep learning, which is currently the most prevalent form of AI. Therefore, our platform represents an alternate form of AI, one that we believe will play an ever-increasing role in the future.

The Platform described herein represents a central component of the Thousand Brains Project (TBP), and consists of two key aspects, all of which will be open-source: a modular and configurable Architecture for intelligent interaction with the world, and the SDK infrastructure for developers to build AI applications that leverage the Architecture. The ultimate aim is to enable developers to build AI applications that are more intelligent, more flexible, and more capable than those built using traditional deep learning methods. Our implementation is based on the core principles of the Thousand Brains Theory and the computations of the mammalian neocortex.

One key differentiator between the TBP and other AI technologies is that the TBP is built with embodied, sensorimotor learning at its core. Sensorimotor systems learn by sensing different parts of the world over time while interacting with it. For example, as you move your body, your limbs, and your eyes, the input to your brain changes. In the Architecture, the learning derived from continuous interaction with an environment represents the foundational knowledge that supports all other functions. This contrasts with the growing approach that sensorimotor interactions are a sub-problem that can be solved by beginning with an architecture trained on a mixture of internet-scale language and multi-media data. In addition to sensorimotor interaction being the core basis for learning, the centrality of sensorimotor learning manifests in the design choice that all levels of processing are sensorimotor. As will become clear, sensory and motor processing are not broken

up and handled by distinct architectures, but play a crucial role at every point in the Architecture where information is processed.

A second differentiator is that our sensorimotor systems learn structured models, using *reference frames*, coordinate systems within which locations and rotations can be represented. The models keep track of where their sensors are relative to things in the world. They are learned by assigning sensory observations to locations in reference frames. In this way, the models learned by sensorimotor systems are structured, similar to CAD models in a computer. This allows the system to quickly learn the structure of the world and how to manipulate objects to achieve a variety of goals, what is sometimes referred to as a 'world model'. As with sensorimotor learning, reference frames are used throughout all levels of information processing, including the representations of not only environments, but also physical objects and abstract concepts - even the simplest representations in the Architecture are represented within a reference frame.

There are numerous advantages to sensorimotor learning and reference frames. At a high level, you can think about all the ways humans are different from today's AI. We learn quickly and continuously, constantly updating our knowledge of the world as we go about our day. We do not have to undergo a lengthy and expensive training phase to learn something new. We interact with the world and manipulate tools and objects in sophisticated ways that leverage our knowledge of how things are structured. For example, we can explore a new app on our phone and quickly figure out what it does and how it works based on other apps we know. We actively test hypotheses to fill in the gaps in our knowledge. We also learn from multiple sensors and our different sensors work together seamlessly. For example, we may learn what a new tool looks like with a few glances and then immediately know how to grab and interact with the object via touch.

One of the most important discoveries about the brain is that most of what we think of as intelligence, from seeing, to touching, to hearing, to conceptual thinking, to language, is created by a common neural algorithm. All aspects of intelligence are created by the same sensorimotor mechanism. In the neocortex, this mechanism is implemented in each of the thousands of cortical columns. This means we can create many different types of intelligent systems using a set of common building blocks. The Architecture we are creating is built on this premise. The SDK will provide the core components and developers will then be able to assemble widely varying AI and robotics applications using these components in different numbers and arrangements.

Any engineer will be able to create AI applications using the Platform without requiring huge computational resources or background knowledge.

1.1 Goals of the Thousand Brains Project

Long Term Goals

A central long-term goal is to build a universal Platform and communication protocol for intelligent sensorimotor systems. The common communication protocol can be used as an interface between different custom modules, and its universality is central to the ease of use of the SDK we are developing. For instance, one person may have modules optimized for flying drones using birds-eye observations, while another may be working with different sensors and actuators regulating a smart home. Those two are quite different modules but they should be able to communicate on the same channels defined here. Third parties could develop sensor modules and learning modules according to their specific requirements but they would be compatible with all existing modules due to a shared communication protocol.

A second goal of the Thousand Brains Project (TBP) is to be a catalyst for a whole new way of thinking about machine intelligence. The principles of the TBP differ from many principles of popular AI methodologies today and are more in line with the principles of learning in the brain. Most concepts presented here derive from the Thousand Brains Theory (TBT) (Hawkins et al., 2019) and experimental evidence about how the brain works. Modules in the SDK are inspired by cortical columns in the neocortex (Mountcastle, 1997). The communication protocol between modules relies on sparse location and reference frame-based data structures. They are analogous to long-range connections in the neocortex. In our implementation, we do not need to strictly adhere to all biological details and it is important to note that should an engineering solution serve us better for implementing certain aspects, then it is acceptable to deviate from the neuroscience and the TBT. In general, the inner workings of the modules can be relatively arbitrary and do not have to rely on neuroscience as long as they adhere to the communication protocol. However, the core principles of the TBP are motivated by what we have learned from studying the neocortex.

Third, this project aims to eventually bring together prior work into a single framework, including sparsity, active dendrites, sequence memory,

and grid cells (Ahmad and Scheinkman, 2019; Hawkins and Ahmad, 2016; Hawkins, Ahmad, and Cui, 2017; Hawkins et al., 2019; Lewis et al., 2019).

Finally, it will be important to showcase the capabilities of our SDK. We will work towards creating a non-trivial demo where the Architecture can be used to showcase some capability that would be hard to demonstrate any other way. This may not be one specific task but could play to the strength of this system to tackle a wide variety of tasks. We will also work on an easy-to-use open-source SDK that other practitioners can apply and test on their applications. We want this to be a platform for all kinds of applications and not just a specific technology showcase.

We have a set of guiding principles that steer the Thousand Brains Project. Throughout the life of the project there may be several different implementations and within each implementation there may be different versions of the core building blocks but everything we work on should follow these core principles:

- Sensorimotor learning and inference: We are using actively generated temporal sequences of sensory inputs instead of static inputs.
- Modular structure: Easily expandable and scalable.
- Common communication protocol: Inner workings of modules are highly customizable such that many different sensor modules (and modalities) and learning modules can work together seamlessly.
- Voting: A mechanism by which a collection of experts can use different information and models to come to a more robust and stable conclusion.
- Reference frames: The learned models should have inductive biases that make them naturally good at modeling a 4D world. The learned models can be used for a variety of tasks such as manipulation, planning, imagining previously unseen states of the world, fast learning, generalization, and many more.
- Rapid, continual learning where learning and inference are closely intertwined: supported by sensorimotor embodiment and reference frames, biologically plausible learning mechanisms enable rapid knowledge accumulation and updates to stored representations while remaining robust under the setting of continual learning. There is also no clear distinction between learning and inference. We are always learning, and always performing inference.

Short Term Goals

Our goal in the near term is to continue building an Architecture based on the principles listed above with a general set of abilities for modeling and interacting with the world. We want to understand and flesh out some of the key issues and mechanisms of learning in such a modular, sensorimotor setup. Two key issues we will focus on next are learning compositional objects using hierarchy, and using learned object models to enable sophisticated ('model-based') action policies.

In the current stage of building up the code framework, we are focusing on the two basic components; learning modules and sensor modules, and the communication between them. In the initial implementation, many components are deliberately *not* biologically constrained, and/or simplified, so as to support visualizing, debugging, and understanding the system as a whole. For example, object models are currently based on explicit graphs in 3D Cartesian space. In the future, these elements will be substituted with more powerful, albeit more inscrutable neural components.

Another goal for the coming months is to open-source and communicate our progress and achievements so far. We want to make it easy for others to join the project and contribute to the Platform. We will provide access to the simple SDK and examples to get started. We also want to spread the ideas of the Thousand Brains Theory and the corresponding Architecture to a wider audience. We aim to do this by writing blog posts, releasing videos, open-sourcing our code, and creating a community around the project.

1.2 Opening Statements

1.2.1 Document Structure

This document represents a high-level overview of the Platform, with a particular emphasis on the core principles informing the design of the Architecture that will power it. It does not yet represent a comprehensive description of the Platform, although we look forward to releasing updates to this document in the coming months, as well as additional resources.

The Architecture can be roughly divided into two key ideas: The individual modules as repeatable, semi-independent information processing units, and the communication protocol between modules. After a general high-level overview, those two aspects are described in more detail.

The conceptual overview is followed by an overview of our algorithmic implementation (to be released). This chapter starts out with an overview of how the abstract TBT concepts explained in the previous chapter are realized in the code base (to be released). It then goes into some detail on all of the main components of the system.

1.2.2 Challenging Preconceptions

Several of the ideas and ways of thinking introduced in this document may be counter-intuitive to people used to the way of thinking prominent in current AI methods, including deep learning. For example, ideas about intelligent systems, learning, models, hierarchical processing, or action policies that you already have in mind might not apply to the system that we are describing. We therefore ask the reader to try and dispense with as many preconceptions as possible and to understand the ideas presented here on their own terms. We are happy to discuss any questions or thoughts that may arise from reading this document. Please reach out to us at ThousandBrains@numenta.com.

Below, we highlight some of the most important differences between the system we are trying to build here and other AI systems.

- We are building a sensorimotor system. It learns by interacting with the world and sensing different parts of it over time. It does not learn from a static dataset. This is a fundamentally different way of learning than most leading AI systems today and addresses a (partially overlapping) different set of problems.

- We will introduce learning modules as the basic, repeatable modeling unit, comparable to a cortical column. An important thing to point out here is that none of these modeling units receives the full sensory input. For example in vision, there is no 'full image' anywhere. Each sensor senses a small patch in the world. This is in contrast to many AI systems today where the whole input is fed into a single model.

- Despite the previous point, each modeling system can learn complete models of objects and recognize them on its own. A single modeling unit should be able to perform all basic tasks of object recognition and manipulation. Using more modeling units makes the system faster and more efficient, and supports compositional and abstract representations, but a single learning module is itself a powerful system. In the

single model scenario, inference always requires movement to collect a series of observations, in the same way that recognizing a coffee cup with one of your fingers requires moving across its surface.

- All models are structured by reference frames. An object is not just a bag of features. It is a collection of features at locations. The relative locations of features to each other is more important than the features themselves.

1.2.3 Capabilities of the System

The system implemented in the Thousand Brains Project is designed to be a general-purpose AI system. It is not designed to solve a specific task or set of tasks. Instead, it is designed to be a platform that can be used to build a wide variety of AI applications. Like an operating system or a programming language does not define what the user applies it to, the Thousand Brains Project will provide the tools necessary to solve many of today's current problems as well as completely new and unanticipated ones without being specific to any one of them.

Even though we cannot predict the ultimate use cases of the system, we want to test it on a variety of tasks and keep a set of capabilities in mind when designing the system. The basic principle here is that **it should be able to solve any task the neocortex can solve**. If we come up with a new mechanism that makes it fundamentally impossible to do something the neocortex can do, we need to rethink the mechanism.

Following is a list of capabilities that we are always thinking about when designing and implementing the system. We are not looking for point solutions for each of these problems but a general algorithm that can solve them all. It is by no means a comprehensive list but should give an idea of the scope of the system.

- Recognizing objects independent of their location and orientation in the world.
- Determining the location and orientation of an object relative to the observer, or to another object in the world.
- Performing learning and inference under noisy conditions.
- Learning from a small number of samples.

- Learning from continuous interaction with the environment with no explicit supervision, whilst maintaining previously learned representations.
- Recognizing objects when they are partially occluded by other objects.
- Learning categories of objects and generalizing to new instances of a category.
- Learning and recognizing compositional objects, including novel combinations of their parts.
- Recognizing objects subject to novel deformations (e.g. Dali's 'melting clocks', or recognizing objects learned in 3D but seen in 2D).
- Recognizing an object independent of its scale, and estimating its scale.
- Modeling and recognizing object states and behaviors (e.g. if a stapler is open or closed; whether a person is walking or running, and how their body evolves over time under these conditions).
- Using learned models to alter the world and achieve goals, including goals that require decomposition into simpler tasks. The highest-level, overarching goals can be set externally.
- Generalizing modeling to abstract concepts derived from concrete models.
- Modeling language and associating it with grounded models of the world.
- Modeling other entities ('Theory of Mind').

1.3 Overview Of The Architecture

There are three major components that play a role in the Architecture: sensors, learning modules, and actuators[1]. These three components are tied together by a common communication protocol (CCP). Due to the unified communication protocol, the inner workings of each individual component can be quite varied as long as they leverage the appropriate interfaces.[2]

1.3.1 Three Core Components

Sensor Modules

Each sensor module receives information from a small sensory patch as input. This is analogous to a small patch on the retina (as in figure 1.6), or a patch of skin, or the pressure information at one whisker of a mouse. One sensor module sends information to one learning module which models this information. Knowledge of the whole scene is integrated through the communication between multiple learning modules that each receive different patches of the input space or through time by one learning module moving over the scene.

The sensor module contains a sensor and associates the input to the sensor with a feature representation. The information processing within the sensor module can turn the raw information from the sensor patch into a common representation (e.g. a Sparse Distributed Representation or *SDR*) which could be compared to light hitting the retina and being converted into a spike train, the pulses of electricity emitted by biological neurons. Additionally, the feature pose relative to the body is calculated from the feature's pose relative to the sensor and the sensor pose relative to the body. This means each sensor module outputs its current pose (location and rotation) as well as the features it senses at that pose. We emphasize that the available pose information is central to how the Architecture operates.

[1] Sensors may be actuators and could have capabilities to take a motor command to move or attend to a new location.

[2] In general, the learning modules in an instance of the Architecture will adhere to the concepts described herein, however it is possible to augment an Architecture with alternative learning modules. For example, we do *not* anticipate that the learning modules described herein will be useful for calculating the result of numerical functions, or for predicting the structure of a protein given its genetic sequence. Alternative systems could therefore be leveraged for such tasks and then interfaced according to the CCP.

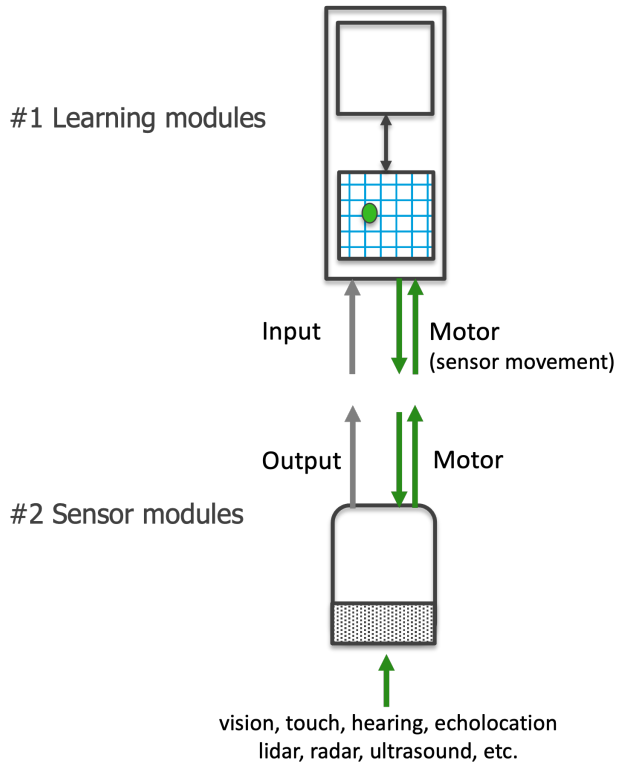

Figure 1.1: Sensor modules receive and process the raw sensory input. This is then communicated via a common communication protocol to a learning module which uses this to learn and recognize models of anything in the environment.

A general principle of the system is that **any processing specific to a modality happens in the sensor module**. The output of the sensor module is not modality-specific anymore and can be processed by any learning module. A crucial requirement here is that each sensor module knows the pose of the feature relative to the sensor. This means that sensors need to be able to detect features and poses of features. The system can work with any type of sensor (vision, touch, radar, LiDAR,...) and integrate information from multiple sensory modalities without effort. For this to work, sensors need to communicate sensory information in a common language.

Learning Modules

The basic building block for sensorimotor processing and modeling the output from the sensor module is a learning module. These are repeating elements, each with the same input and output information format. Each learning module should function as a stand-alone unit and be able to recognize objects on its own. Combining multiple learning modules can speed up recognition (e.g. recognizing a cup using five fingers vs. one), allows for learning modules to focus on storing only some objects, and enables learning compositional objects.

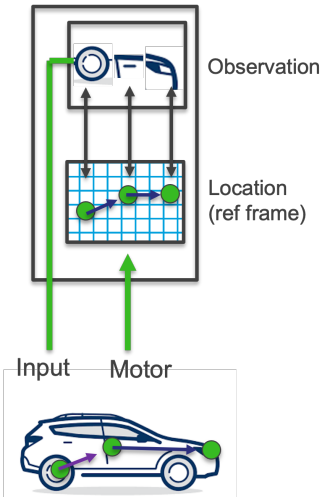

Figure 1.2: Learning modules learn structured models through sensorimotor interaction, using reference frames. They model how incoming features are arranged relative to each other in space and time.

Learning modules receive either feature IDs from a sensor or estimated object IDs (also interpreted as features) from a lower-level learning module[3]. The feature or object representation might be in the form of a discrete ID (e.g. the color red, a cylinder), or could be represented in a more high dimensional space (e.g. an SDR representing hue or corresponding to a fork-like object). Additionally, learning modules receive the feature's or object's pose relative to *the body*, where the pose includes location and rotation. In this way, pose relative to the body serves as a common reference frame for spatial computations, as opposed to the pose of features relative to each individual sensor. From this information higher level learning modules can build up graphs of compositional objects (e.g. large objects or scenes) and vote on the ID of the currently visible object(s).

The features and relative poses are incorporated into a model of the ob-

[3]By **object**, we mean a discrete entity composed of a collection of one or more other objects, each with their own associated pose. As such, an object could also be a scene or any other composition of sub-objects. At the lowest level of object hierarchy, an object is composed of 'proto-objects' (commonly thought of as features), which are also discrete entities with a location and orientation in space, but which are output by the sensor modules; as such, these cannot be further decomposed into constituent objects. Wherever an object (or proto-object) is being processed at a higher level, it can also be referred to as a **feature**.

ject. All models have an inductive bias towards learning the world as based in 3-dimensional space with an additional temporal dimension. However the exact structure of space can potentially be learned, such that the lower-dimensional space of a melody, or the abstract space of a family tree, can be represented. When interacting with the physical world, the 3D inductive bias is used to place features in internal models accordingly.

The learning module therefore encompasses two major principles of the TBT: Sensorimotor learning, and building models using reference frames (see figure 1.2). Both ideas are motivated by studies of cortical columns in the neocortex (see figure 1.3).

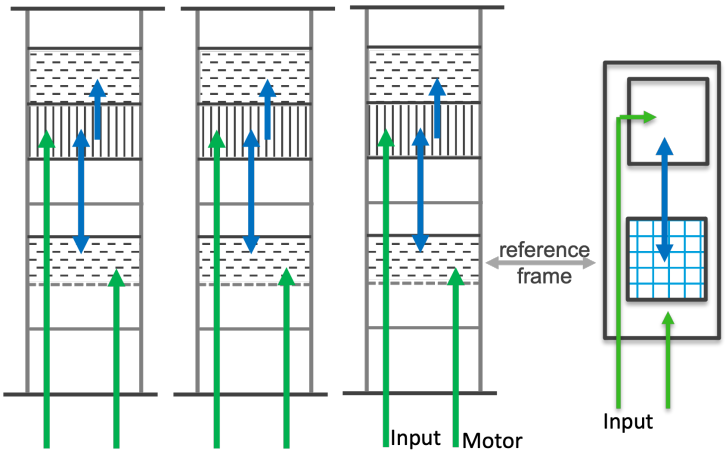

Figure 1.3: Conceptual sketch of how the learning module could be implementing possible mechanisms of cortical columns.

Besides learning new models, the learning module also tries to match the observed features and relative poses to already learned models stored in memory. In addition to performing such inference within a single LM, an LM's current hypotheses can be sent through lateral connections to other learning modules using the common communication protocol. We note again that the CCP is independent of modality, and as such, LMs that have learned objects in different modalities (e.g. vision vs. touch), can still 'vote' with each other to quickly reach a consensus. This voting process is inspired by the voting process described in Hawkins, Ahmad, and Cui (2017). Unlike when the CCP is used for the input and output of an LM, votes consist of multiple CCP-compliant messages, representing the union of multiple possible object hypotheses.

To generate the LM's output, we need to get the pose of the sensed object relative to the body. We can calculate this from the current incoming pose (pose of the sensed feature relative to the body) and the poses stored in the model of the object. This pose of the object can then be passed hierarchically to another learning module in the same format as the sensory input (features at a pose relative to the body).

Once the learning module has determined the ID of an object and its pose, it can take the most recent observations (and possibly collect more) to update its model of this object. We can therefore continually learn more about the world and learning and inference are two intertwined processes.

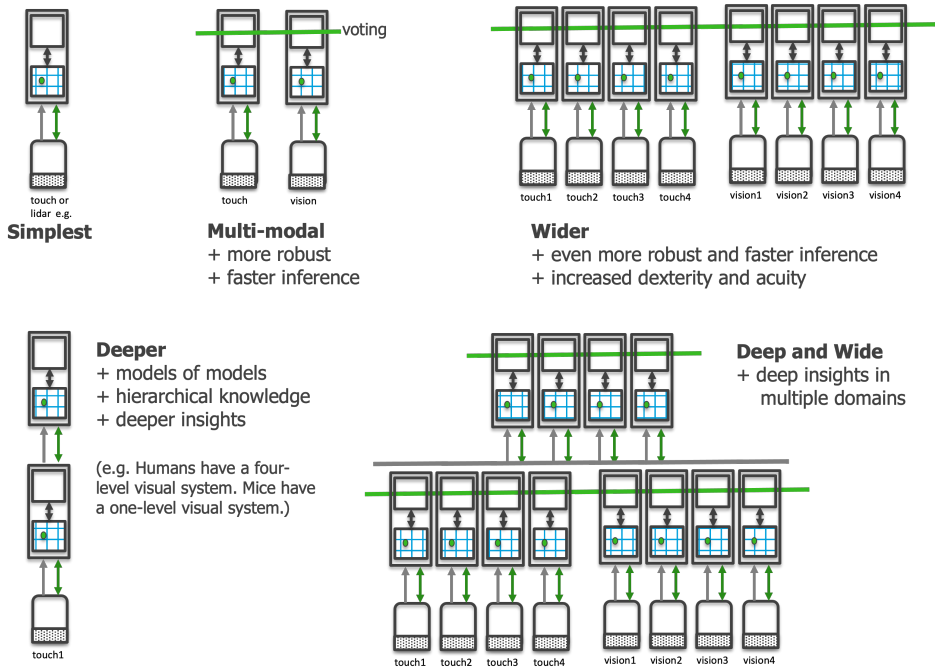

Figure 1.4: By using a common communication protocol between sensor modules and learning modules, the system can easily be scaled in multiple dimensions. This provides a straightforward way for dealing with multiple sensory inputs from multiple modalities. Using multiple learning modules next to each other can improve robustness through votes between them. Additionally, stacking learning modules on top of each other allows for more complex, hierarchical processing of inputs and modeling compositional objects.

Common Communication Protocol

We use a common communication protocol that all components (LMs, SMs, and motor systems) adhere to. This makes it possible for all components to communicate with each other and to combine them arbitrarily. The CCP defines what information the outputs of SMs and LMs need to contain.

In short, a CCP-compliant output contains *features at a pose*. The pose contains a location in 3D space (naturally including 1D or 2D space) and represents where the sensed features are relative to the body, or another common reference point such as a landmark in the environment. The pose also includes information about the feature's 3D rotation. Additionally, the output can contain features that are independent of the object's pose such as color, texture, temperature (from the SM) or object ID (from the LM).

Besides features and their poses, the standard message packages also include information about the sender's ID (e.g. the particular sensor module) and a confidence rating.

The inputs and outputs of the system (raw sensory input to the SM and motor command outputs from the policy) can have any format and do not adhere to any communication protocol. They are specific to the agents sensors and actuators and represent the systems interface with the environment.

The lateral votes between learning modules communicate unions of possible poses and objects. They do not contain any information about 'features' from the perspective of that learning module's level of hierarchical processing. In other words, while an LM's object ID might be a feature at higher levels of processing, lateral votes do not send information about the features which that learning module itself has received. We further note that the vote output from one LM can also include multiple CCP message packages, representing multiple possible hypotheses.

At no point do we communicate structural model information between learning modules. What happens within a learning module does not get communicated to any other modules and we never share the models stored in an LMs memory.

Communication between components (SMs, LMs, and motor systems) happens in a common reference frame (e.g. relative to the body). This makes it possible for all components to meaningfully interpret the pose information they receive. Internally, LMs then calculate displacements between consecutive poses and map them into the model's reference frame. This makes it

possible to detect objects independently of their pose.

The common reference frame also supports voting operations accounting for the relative displacement of sensors, and therefore LM models. For example, when two fingers touch a coffee mug in two different parts, one might sense the rim, while the other senses the handle. As such, 'coffee mug' will be in both of their working hypotheses about the current object. When voting however, they do not simply communicate 'coffee mug', but also *where* on the coffee mug other learning modules should be sensing it, according to their relative displacements. As a result, voting is not simply a 'bag-of-features' operation, but is dependent on the relative arrangement of features in the world.

1.3.2 Further Details

Below are additional details of the Architecture, including how the three components outlined above interact. Further details will be provided in future publications and resources that we share.

Learning on Different Spatial Scales (Hierarchy):

Learning modules can be stacked in a hierarchical fashion to process larger input patches and higher-level concepts. A higher-level learning module receives feature and pose information from the output of a lower-level module and/or from a sensor patch with a larger receptive field, mirroring the connectivity of the cortex. The lower-level LM never sees the entire object it is modeling at once but infers it either through multiple consecutive movements and/or voting with other modules. The higher-level LM can then use the recognized model ID as a feature in its own models. This makes it more efficient to learn larger and more complex models as we do not need to represent all object details within one model. It also makes it easier to make use of object compositionality by quickly associating different object parts with each other as relative features in a higher-level model.

Learning on Different Time-Scales

Additionally to learning on different spatial scales, modules can learn on different temporal scales. A low-level module may slowly learn to model general input statistics while a higher-level module may quickly build up

temporary graphs of the current state of the world, as a form of short-term memory. Of course, low-level modules may also be able to learn quickly, depending on the application. This could be implemented by introducing a (fixed or learnable) speed parameter for each learning module.

Voting/Consensus

Learning modules have lateral connections to each other to communicate their estimates of the current object ID and pose. For voting, we use a similar feature-pose communication as we use to communicate to higher-level modules. However, in this case we communicate a union of all possible objects and poses under the current evidence (multiple messages adhering to the CCP). Through the lateral voting connections between modules they try to reach a consensus on which object they are sensing at the moment and its pose (see figure 1.5). This helps to recognize objects faster than a single module could.

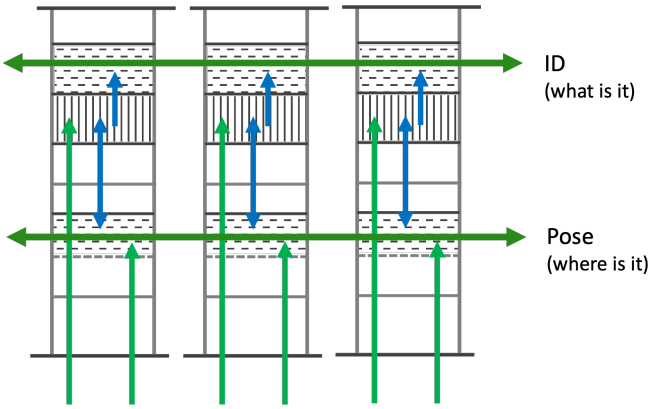

Figure 1.5: Voting between cortical columns in the neocortex as suggested by the Thousand Brains Theory.

Motor Input

The movement information (pose displacement) can be a copy of the selected action command (efference copy) or deduced from the sensory input. Without the efference copy, movement can for example be detected from optical flow or proprioception. Sensor modules use movement information to update

their pose relative to the body. Learning modules use it to update their hypothesized location within an object's reference frame.

Motor Output

Each learning module produces a motor output. The motor output is formalized as a goal state and also adheres to the common communication protocol. The goal state could for example be generated using the learned models and current hypotheses by calculating a sensor state which would resolve the most uncertainty between different possible object models. It can also help to guide directed and more efficient exploration to known features in a reference frame stored in memory. Different policies can be leveraged depending on whether we are trying to recognize an object or trying to learn new information about an object.

Hierarchy can also be leveraged for goal-states, where a more abstract goal-state in a high-level learning module can be achieved by decomposing it into simpler goal-states for lower-level learning modules. Importantly, the same learning modules that learn models of objects are used to generate goal-states, enabling hierarchical, model-based policies, no matter how novel the task.

Sub-Cortical Motor Areas

The Architecture is an entire sensorimotor system. Each learning module receives sensory input and an efference copy of the motor command and outputs a feature-at-pose along with a motor command. Since many modules may produce conflicting motor commands (e.g. different patches on the retina cannot move in opposite directions) they usually need to be coordinated in a motor area. This motor area contains an action policy that decides which action commands to execute in the world based on the motor outputs from all learning modules. It also needs to translate the goal state outputs of the learning modules into motor commands for the actuators. It then sends this motor command to the actuators of the body and an efference copy of it back to the sensor modules.

In the brain, a lot of this processing occurs subcortically. Therefore in our system, we also don't need to resolve these issues within a learning module but can do it within a separate motor area. However, we need to keep in mind that the motor area does not know about the models of objects that

are learned in the learning modules and therefore needs to receive useful model-based motor commands from the LMs.

Predictions

Learned models in the memory of the learning module can be used to make predictions about future observations. If there are multiple models that match the current observations, the predictions would have more uncertainty attached to them. The prediction error can be used as a learning signal to update models or as a criterion for matching during object recognition.

Currently there is no prediction in time, although in the future such capabilities will be added via the inclusion of a temporal dimension. This will help support encoding behaviors of objects, as well as predictions that can be used for motor-policy planning. For example, the long-term aim is for the Architecture to be able to predict how a simple object such as a stapler evolves as it is opened or closed, or to coarsely model the physical properties of common materials.

1.3.3 Bringing it Together

To consolidate these concepts, please see figure 1.6 for a potential instantiation of the system in a concrete setting. In the example, we see how the system could be applied to sensing and recognizing objects and scenes in a 3D environment using several different sensors, in this case touch and vision.

While this hopefully makes the key concepts described above more concrete and tangible, keep in mind that this is just one way in which the Architecture can be instantiated. By design, the Platform can be applied to any application that involves sensing and active interaction with an environment. Indeed, this might include more abstract examples such as browsing the web, or interacting with the instruments that control a scientific experiment.

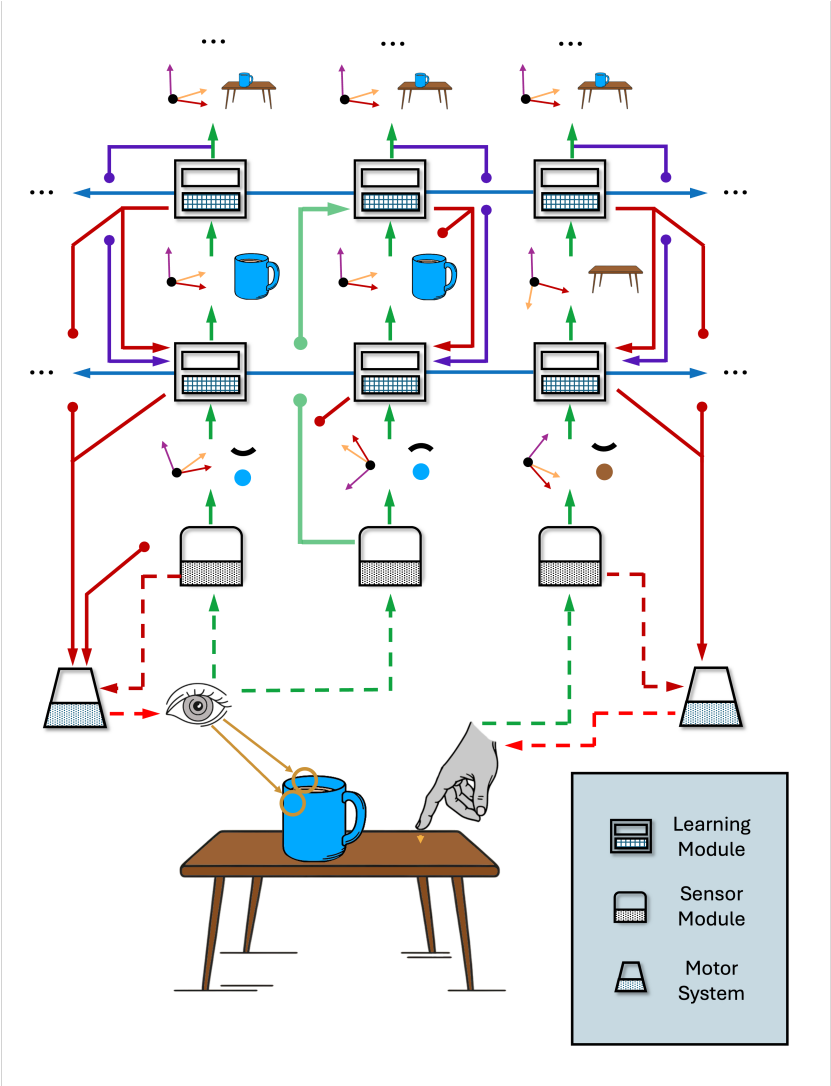

Figure 1.6: High-level overview the Architecture with all the main conceptual components mirroring figure 1.4 applied to a concrete example. Green lines indicate the main flow of information up the hierarchy. Purple lines show top-down connections, biasing the lower-level learning modules. Light blue lines show lateral voting connections. Red lines show the communication of goal states which eventually translate into motor commands in the motor system. Every LM has a direct motor output. Information communicated along solid lines follows the CCP (contains features and pose). Discontinuations in the diagram are marked with dots on line-ends. Dashed lines are the interface of the system with the world and subcortical compute units and do not need to follow the CCP. Green dashed lines communicate raw sensory input from sensors. Red dashed lines communicate motor commands to the actuators. The dark red dashed lines send sensory information directly to the motor system and implement a fast reflex loop for purely input-driven policies. The large, semi-transparent green arrow is an example of a connection carrying sensory outputs from a larger receptive field directly to the higher-level LM.

1.3.4 Conclusion

We have provided a high-level description of an in-development Platform, consisting of an Architecture for intelligent sensorimotor learning and action, and an SDK to support the development of applications that leverage that Architecture. We look forward to sharing further details about the Platform in the coming months, and hope that you will consider joining in its development and growth.

Resources

2.1 Glossary

This section aims to provide concise definitions of terms commonly used in the Thousand Brains Project.

reference frame: A specific coordinate system within which locations and rotations can be represented. For instance, a location may be represented relative to the body (body/ego-centric reference frame) or relative to some point in the world (world/allo-centric reference frame) or relative to an object's center (object-centric reference frame).

pose: An object's location and orientation (in a given reference frame). The location can for example be x, y, z coordinates and the orientation can be represented as a quaternion, Euler-angles or rotation matrix.

displacement: The spatial difference between two locations. In 3D space, this would be a 3D vector.

transformation: Applies a displacement/translation and a rotation to a point.

path integration: Updating an agent's location by using an estimate of its current location, together with an estimate of its own movement.

sensorimotor/embodied: Learning or inference through interaction with an environment using a closed loop between action and perception. This means observations depend on actions and in turn the choice of these actions depend on the observations.

policy: Defines the function used to select actions. Selected actions can be dependent on a model's internal state and on external inputs.

sparse distributed representation (SDR): A binary vector with significantly more 0 bits than 1 bits. Significant overlap between the bit assignments in different SDRs captures similarity in representational space (e.g.

similar features).

dendrites: Implement pattern recognizers to identify patterns such as a specific SDR. One neuron is typically associated with multiple dendrites such that it can identify multiple patterns.

In biology, dendrites of a postsynaptic cell receive information from the axons of other presynaptic cells. The axons of these presynaptic cells connect to the dendrites of postsynaptic cells at a junction called a "synapse". An SDR can be thought of as a pattern which is represented by a set of synapses that are collocated on a single dendritic segment.

sensor module: A computational unit that turns raw sensory input into a common communication protocol output. The structure of the output of a sensor module is independent of the sensory modality and represents a set of features at poses.

learning module: A computational unit that takes features at poses as input and uses this information to learn models of the world. It is also able to recognize objects and their poses from the input if an object has been learned already. Finally, it can output actions in the form of target goal-states for other modules.

reference frame graph: A set of nodes that are connected to each other with edges. Both nodes and edges can have features associated with them. For instance all graphs used in the Architecture have a location associated with each node and a variable list of features. An edge can, for example, have a displacement associated with it.

efference copy: A copy of the motor command that was output by the policy and sent to the actuators. This copy can be used by learning modules to update their states or make predictions.

inductive bias: An assumption about the nature of the world that is built into an algorithm/model. If the assumption holds, this can enable the model to learn more efficiently than without the inductive bias. However, it will result in limitations of the model if the assumption does not hold or is overly restrictive.

Bibliography

Ahmad, Subutai and Luiz Scheinkman (2019). "How Can We Be So Dense? The Robustness of Highly Sparse Representations". In: *ICML 2019 Workshop on Uncertainty and Robustness in Deep Learning*.

Hawkins, Jeff and Subutai Ahmad (2016). "Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex". In: *Frontiers in Neural Circuits* 10. ISSN: 16625110. DOI: 10.3389/fncir.2016.00023.

Hawkins, Jeff, Subutai Ahmad, and Yuwei Cui (2017). "A Theory of How Columns in the Neocortex Enable Learning the Structure of the World". In: *Frontiers in Neural Circuits* 11.October, pp. 1–18. ISSN: 1662-5110. DOI: 10.3389/fncir.2017.00081. URL: http://journal.frontiersin.org/article/10.3389/fncir.2017.00081/full.

Hawkins, Jeff et al. (2019). "A framework for intelligence and cortical function based on grid cells in the neocortex". In: *Frontiers in Neural Circuits*. ISSN: 16625110. DOI: 10.3389/fncir.2018.00121.

Lewis, Marcus et al. (2019). "Locations in the neocortex: A theory of sensorimotor object recognition using cortical grid cells". In: *Frontiers in Neural Circuits*. ISSN: 16625110. DOI: 10.3389/fncir.2019.00022.

Mountcastle, Vernon B. (1997). "The columnar organization of the neocortex". In: *Brain* 120.4. ISSN: 00068950. DOI: 10.1093/brain/120.4.701.