A Brain-Inspired Framework for Evolutionary Artificial General Intelligence

Mohammad Nadji-Tehrani, *Member, IEEE* and Ali Eslami, *Member, IEEE*

***Abstract*—From the medical field to agriculture, from energy to transportation, every industry is going through a revolution by embracing artificial intelligence (AI); nevertheless, AI is still in its infancy. Inspired by the evolution of the human brain, this paper demonstrates a novel method and framework to synthesize an artificial brain with cognitive abilities by taking advantage of the same process responsible for the growth of the biological brain called "neuroembryogenesis." This framework shares some of the key behavioral aspects of the biological brain such as spiking neurons, neuroplasticity, neuronal pruning, and excitatory and inhibitory interactions between neurons, together making it capable of learning and memorizing. One of the highlights of the proposed design is its potential to incrementally improve itself over generations based on system performance, using genetic algorithms. A proof of concept at the end of the paper demonstrates how a simplified implementation of the human visual cortex using the proposed framework is capable of character recognition. Our framework is open-source and the code is shared with the scientific community at www.feagi.org.**

***Index Terms*—Artificial General Intelligence, Genetic Programming, Spiking Neural Networks, Evolutionary Algorithms, Indirect Encoding.**

I. Introduction

MANY exciting brain research initiatives are in motion, both nationally and internationally, with a number of them focusing on methods and approaches to simulate the human brain. Despite decades worth of work, this field is still in its infancy, mainly due to the complexity of the human brain structure driven by the sheer number of neurons, variations in neuron morphology, and the dynamic nature of all components.

Billions of dollars are being spent on the simulation of the human brain, and it would be highly rewarding to see a new design paradigm accelerate the growth rate in this area. Some research groups are approaching the challenge from a hardware perspective and developing neuromorphic architectures to provide a physical platform for brain simulation. Others are building software frameworks to leverage existing hardware and simulate brain behaviors using code. There are challenges to both approaches. On the hardware side, specially designed hardware needs to be paired with specialized algorithms to simulate a biological brain. Hardware by nature has rigidity in its architecture, hence making it almost impossible to evolve similarly to a biological brain. This limitation forces the evolutionary needs to be pushed to the software stack running on top of the hardware or even demanding a new hardware revision leading to enormous engineering and manufacturing expenses. On the other hand, software-based frameworks are very flexible and capable of evolving but are bound to the limitation of resources offered by the hardware on which they run. The biggest challenge on the software side is designing a network model that serves a desirable purpose.

We are proposing a new paradigm and framework called the Framework for Evolutionary Artificial General Intelligence (FEAGI) to help us build intelligent systems inspired by the human brain, which can grow, evolve, scale, and adapt to the environment, requiring limited efforts to define structural characterizations for any given cortical area targeted for simulation. Our approach is a model based on the segregation of the biological brain's anatomy and physiology, which has enabled a simplified yet scalable design process.

In this paper, we discuss the proposed framework's architecture and capabilities. Our contributions are as follows:

- Design of a scalable framework architecture for human brain simulation based on spiking neurons.
- Novel method to indirectly encode an artificial brain's anatomical structures in a genome-inspired data structure through indirect encoding.
- Method to grow an artificial brain inspired by the process of neuroembryogenesis in the human brain.
- Modules with input and output interfaces to help convert physical stimuli into neuronal activities, and vice versa.
- Functions to simulate the biological brain's physiological activities, such as neurotransmission, neuroplasticity, and inhibitory-excitatory behaviors.
- Special memory architecture to help exhibit cognitive abilities such as learning, long-term memorization, and memory recall.
- Genome database as part of the evolutionary framework to capture the artificial brain's genomic information along with statistics tied to its performance to help us evolve the artificial brain's anatomical properties over generations using genetic algorithms, thus allowing the artificial brain to improve its cognitive abilities over time.
- Burst management technique to alleviate the lack of parallelism present in an artificial environment that stems from software and hardware limitations compared to a biological counterpart.
- Augmented framework with a visualization module to help gain insight into the flow of information within cortical pathways as well as memory formations.

We believe that using such a framework can accelerate research in the quest for building large-scale models capable

M. Nadji-Tehrani and A. Eslami are with the Department of Electrical Engineering and Computer Science, Wichita State University, Wichita, KS, USA (emails: mxnadjitehrani@shockers.wichita.edu, ali.eslami@wichita.edu).
M. Nadji-Tehrani is also with NetApp Inc., Sunnyvale, CA, USA.

of artificial general intelligence. Our proof of concept demonstrates the ability to read digits from the MNIST database [1] and recognize them in real-time. To prove the versatility of the framework, after evolving the artificial brain through thousands of generations using handwritten digit training, we trained the new generations using apparel images from the Fashion-MNIST database [2].

The rest of this paper is organized as follows: Section II briefly reviews the existing research and accomplishments in related areas. Section III introduces the framework, outlines the components, and describes the major functionalities. Section IV showcases system highlights. Section V provides an overview of a proof of concept we developed using this framework. Section VI analyzes the results. Section VII highlights the future potential of this framework and our plans for improving it. Section VIII concludes the paper.

II. Related Work

Researchers have been pursuing human brain simulation with different objectives for many years. Some have quested to understand better how the human brain functions and to assist in root-causing brain-related disorders, and others have sought to create artificial intelligence (AI) modeled after the ultimate example provided by nature. Our goal in creating FEAGI is mainly the latter and to offer a framework that, on one hand, can help grow a functional brain using predefined anatomical and physiological properties and, on the other hand, is capable of evolving to produce an artificial intelligence that is superior to that from which it was initially created.

Currently, there are multiple active brain simulation projects around the world. One of the most funded studies is the Human Brain Project (HBP) [3], with its goal of building a detailed working simulation of the entire human brain. The HBP is collaborating with a large number of scientists and collecting very detailed information about brain structures down to the synaptic level while leveraging supercomputers to run real-time simulations. The approach of the HBP is considered a bottom-up approach, producing results very close to the biological brain but associated with very high computing and modeling costs, complexity, and reliance on high-fidelity information from the biological brain [4].

The Semantic Pointer Architecture Unified Network, or Spaun [5], is a functional brain model with multiple cognitive abilities created using the neural simulator Nengo [6], [7] and built based on spiking neurons [8] by creating a higher-level model inspired by the biological brain. Nengo offers a powerful software development kit (SDK) to design detailed artificial brain models. Nengo also can interface with neuromorphic hardware such as SpiNNaker [9] and Neurogrid [10]. The Defense Advanced Research Projects Agency's (DARPA) SyNAPSE project [11] has resulted in the creation of TrueNorth chip [12] accompanied by a software simulator called Compass [13], [14]. Compass is capable of simulating trillions of neurons on a supercomputer. Other neuron-simulation software used by the research community includes NEST [15] and Brian [16], which are also designed for simulating spiking neuron models.

A key difference between existing brain simulators and the biological brain is that a mature biological brain came into existence as a result of an enduring growing process referred to as "neuroembryogenesis." In contrast, artificial deep neural networks simulating the brain are created using a predefined model. Our proposed framework is built on the novel idea of *growing* an artificial brain into existence using a genome-like structure that captures the biological brain's anatomical properties, such as cortical geometries, neuron morphologies, and interneuron connectivities, through indirect encoding [17].

In 1992, Gruau [18] introduced cellular encoding to program neural networks genetically. Ever since, there has been a significant amount of research in this area. In 2003, Stanley et al. [19] introduced the concept of artificial embryogeny as a subdiscipline of evolutionary computation, and later in 2007, Stanley introduced the compositional pattern-producing network (CPPN) [20], a particular kind of neural network acting as a function capable of generating realistic patterns. The CPPN, in conjunction with HyperNEAT [21], has proven to be very successful in evolving robot brains for improving motor behavior [21]. CPPN is created as a network of functions that can help produce phenotypes that follow patterns along with modularity and regularity. Phenotypes can be evolved using HyperNEAT [21]–[23] to produce new offspring.

Given all of the frameworks mentioned above, there is still a significant gap when considering each for designing an artificial general intelligence being. The approach adopted by HBP is low-level and outcome lacks system-level cognition. Nengo has cognitive abilities but requires written code that requires hours of simulation for seconds of real-time behavior, thus making it challenging to evolve, scale and interact in real-time. HyperNEAT has the evolutionary recipe but lacks the architecture needed for implementing cognitive behaviors.

Here, FEAGI can offer a framework built from the ground and brain-inspired in every aspect. Our approach depends on moderate information about the biological brain's structural details to help us define the properties of each cortical region in the genome. Genome entries follow a standard grammar for all brain regions and have the potential for being interfaced with information collected by the HBP to build a more accurate initial seed for our evolutionary framework.

III. Proposed Framework

A. Framework Overview

Our design methodology for the creation of FEAGI is based on an essential principle: to be inspired by the human brain but not to imitate it. The human brain is the ultimate machine regarding modularity, complexity, scalability, functionality, and evolvability; therefore, in building a framework capable of simulating its abilities, the qualities mentioned above must be taken into consideration, which is how we approached our design. In summary, we have designed a framework capable of growing a plastic artificial brain that can manifest cognitive behaviors and evolve over generations. Here, we attempt to outline various components of the framework and their relationships.

A genome-like data structure is developed that uses an indirect encoding method to capture the human brain's anatomical properties as the genotype, and helps to grow the artificial brain step by step by taking it through an artificial neuroembryogenesis process and storing it in another data structure, or "connectome", as a phenotype. The module responsible for this operation is called "neuroembryogenesis unit."

After the anatomical structures are in place by completing the artificial equivalence of the neuroembryogenesis process, the artificial brain will be ready to receive stimuli from the outside world through a dedicated module called the "input processing unit" (IPU) and passing it through the corresponding cortical layers defined via the connectome. Interneuronal communications within the simulated brain uses an efficient burst centric mechanism inspired by the sparsity of neuronal activities in the biological brain to help mitigate the need for highly parallel processing units and time-dependent complexities associated with it so that the artificial brain can leverage commodity hardware while simulating a highly parallel structure to achieve the desired functionality.

Special functions have been designed as part of the "neural processing unit" (NPU) to simulate the biological brain's physiological behaviors such as neuron firing, neuroplasticity, and neuron aging. The memory module has its unique structure to allow storage of the semantic pointers [24] generated from the information processed by cortical pathways. As its name implies, the learning module is responsible for learning activities, which will be described in detail in Section III-G. This module is also capable of self-learning, which enables the framework to feed itself information in a particular manner to help learn on its own.

The evolutionary module uses genetic algorithms to evolve the working genome over generations based on the information it collects during the artificial brain's lifecycle. Last, the activity visualizer module is developed to provide a better understanding of how data flows through the simulated brain and visualizes memory formations.

B. Genome

We have designed the genome module as a hierarchical data structure in the form of JavaScript Object Notation (JSON), a data-interchange format. The genome is the key structure enabling us to evolve the artificial brain using genetic algorithms, as described in Section III-H2, and is used to capture information needed to be able to build and rebuild a functioning artificial brain. We have encoded anatomical properties –including cortical geometry, neuron density, neuron morphologies, growth rules, and orientation selectivity– as well as function variables driving the brain physiology –including firing threshold, plasticity constant, refractory period, leak coefficient, and maximum consecutive firing threshold– as genotypes using indirect encoding [21], [25]. The genome is read using the neuroembryogenesis unit and translated into cortical regions, cortical layers, neurons, and synapses between neurons as a phenotype that is stored in the connectome. Our genome implementation intentionally ignores anatomical structures that play a supportive role for the neurons such as the circulatory system, ventricles, meninges, and glial cells.

FEAGI generates the brain anatomical structures in its entirety using properties defined in the genome; therefore, without any code change and only through gene modification, a new artificial brain with a new set of properties can be generated. This code independence opens the door for using genetic algorithms to have them evolved over generations.

C. Connectome

The connectome is the phenotype created as a result of passing the genome through the artificial neuroembryogenesis process. The connectome is built using an object-oriented data structure to support scalability requirements when the implementation reaches the full human brain scale. The connectome contains information on individual cortical areas, cortical layers, neurons, neuron locations, synapses between neurons, and connection strengths.

D. Neuroembryogenesis Unit

Development of the biological brain is a very complicated and enduring process starting from the early stages of embryonic development called "neurulation," where the neural tube takes shape, and continues to evolve during postnatal development. In the upcoming subsections, we describe how our framework is structured to simulate this processes. In FEAGI, we have been inspired by the biological process and implemented a module that orchestrates a simplified version by reading instructions from the genome and recording the phenotypes in our connectome, as shown in Fig. 1. By the end of this process, we will have cortical areas, layers, sublayers, neurons, and synapses in place.

1) Cortical Area Formation: The neuroembryogenesis unit is developed as a collection of procedures responsible for decoding the genome and building up the connectome. The first step is to have the functional cortical areas defined. In our implementation, every functional cortical area consists of multiple regions and subregions. As an example, to implement the ventral visual pathway, we have defined areas including V1, V2, V4, and IT, as cortical regions, and for region V1, we have defined subregions to accommodate the presence of various cortical layers. This structure has helped us to account for differences in characteristics across distinct cortical layers.

2) Neurogenesis: The next step after defining the cortical areas is to populate them with neurons. Our neurogenesis implementation covers biological phases including neuron **proliferation**, **migration**, and **differentiation** [26], [27]. Since we are not dealing with a physical environment and all definitions are logical, we have adopted a simplified approach to the phases mentioned above: First, the genome is read to decode the density of neurons in a given cortical area. Next, we add neurons in the form of a JSON object to the connectome data structure –referred to as "proliferation" in the biological brain– while associating neuron properties in the form of key-value pairs to each neuron object inherited from the corresponding cortical layer –referred to as "differentiation" in the biological brain. For the final step, we assign each neuron a fixed location in the form of $[x, y, z]$ relative to the boundaries of the cortical region's geometry –referred to as "migration" [28].

Due to the diversity of neuron types in the human central nervous system, as part of the design of the framework, we anticipated that support for different neuron models would

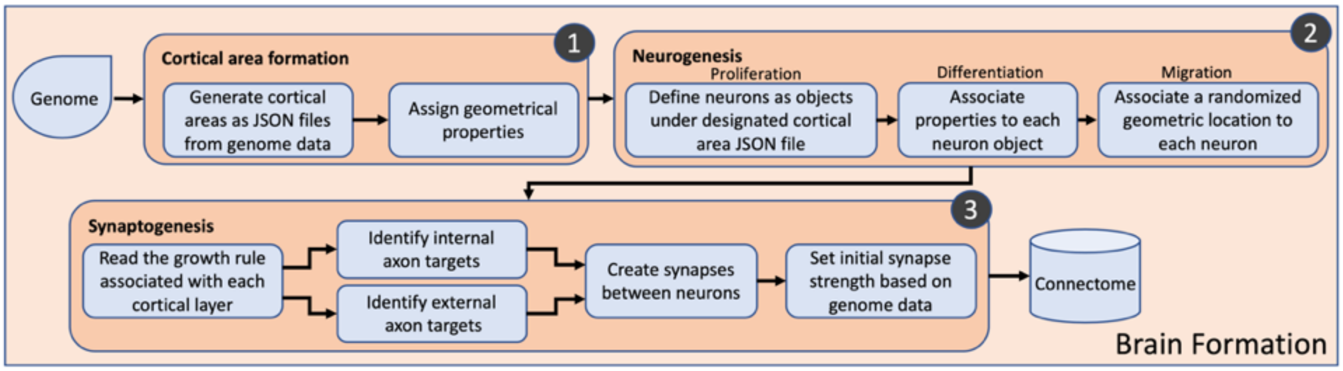

Genome
Cortical area formation
Generate cortical areas as JSON files from genome data
Assign geometrical properties
Neurogenesis
Proliferation
Define neurons as objects under designated cortical area JSON file
Differentiation
Associate properties to each neuron object
Migration
Associate a randomized geometric location to each neuron
Synaptogenesis
Read the growth rule associated with each cortical layer
Identify internal axon targets
Identify external axon targets
Create synapses between neurons
Set initial synapse strength based on genome data
Connectome
Brain Formation

Fig. 1. Artificial brain formation that begins by reading genome instructions and going through multiple phases sequentially, eventually capturing neurons and synapses that are created throughout the process in an object-oriented database, or connectome, for future use as part of the artificial brain's operations.

be essential. In our implementation, we developed a custom neuron model that at large follows the leaky integrate and fire model [29]. The framework is designed to allow each cortical layer or sublayer to have its own designated neuron model. This implementation enables us in the future to model various cortical areas of the biological brain that do not follow a standard neuron model. Parameters defining the neuron model and its behavior are encoded in the genome.

Here are the highlights of neuron definiton in FEAGI:

- Every neuron is created as a result of the neurogenesis process and is identified as an object in the connectome.
- Metadata are associated with each neuron in the database, capturing neuron as well as synaptic properties.
- In the biological brain, two neurons can share hundreds or thousands of synapses. In FEAGI, we have defined a single aggregate that represents the level of impact of one neuron on another. This design decision simplifies the overall architecture, saves storage, and improves performance to a large extent by eliminating the need to track and process a large number of connections between two individual neurons.
- Synapse definition is established by the association of the destination neuron identification number with the source neuron object in the database along with a value defining the synaptic strength among them.
- When the synapse between a source and destination neuron is initially created, a default post-synaptic current value is associated with the source neuron's axon. This value will later change as the neuron goes through long-term potentiation (LTP) or long-term depression (LTD), which is described in detail in Section III-F4.
- Upon neuron firing, its post-synaptic current will create a change in the membrane potential of the downstream neurons. If the neuron is defined as excitatory, then will cause an increase, and if the neuron is defined as inhibitory, then it will cause a decrease.
- The leaky behavior of the neuron is implemented such that with the passage of each burst, a neuron loses some of its membrane potential.

3) Synaptogenesis: Neighbor identification is the prerequisite to synaptogenesis. Axonal growth in a biological brain is an intricate yet exquisite process. An axon's growth is guided step by step through molecular cues, with some being attractive and others repulsive [27]. Sometimes they travel very long distances and surprisingly find their target destination with precision [27], [30]. Axon projections from one cortical layer to another or from one region to another are responsible for shaping the connections within the network and are essential in how data is passed along the cortical pathways and how information is processed.

In our framework, since we are dealing with a digital ecosystem instead of a biological one, we have explored alternative solutions. We investigated various neuron morphologies and noticed that the synapse locations on an axon could be enclosed and defined by a three-dimensional geometrical space outlining a region where the axon of one neuron would synapse with other neurons with high probability.

We have defined a set of geometrical constraints and called them "growth rules." These rules consist of a set of formulas defining spatial boundaries where synaptogenesis is allowed to occur, as shown in Fig. 2. To find candidate neurons, we first read the location of the source neuron, target cortical region, and growth rule applied to the neuron from the genome. The growth rule helps us identify all candidate neurons from the target cortical region by checking if the target neuron coordinate falls within the region defined by the growth rule. The final step is to select a subset of neurons from the pool of eligible neurons to create the synapses with. For this step, we randomly select n neurons from the candidate pool, where n is a parameter defined in the genome for the source neuron's cortical area. With this approach, we have enabled the framework to guide axons to their destination, similar to how biological neurons find their own. Parameters associated with these rules are added to the artificial brain's genome so they can evolve. After the neighbor identification process detects candidate neurons for a given source, we can generate a synapse between the two, by adding the target neuron's identification number and synaptic weight to the source neuron's object in connectome.

E. Input/Output Processing Unit

Input and output processing units are the interfaces between the artificial brain and the outside world. Input processing modules are designed to receive raw data from various input

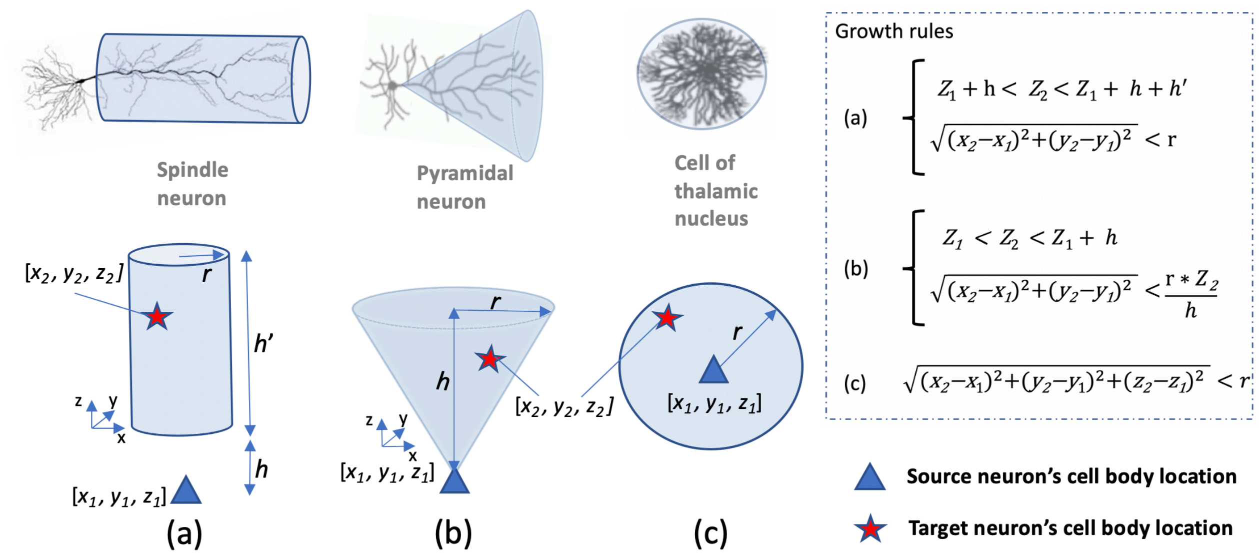

Spindle neuron
Pyramidal neuron
Cell of thalamic nucleus
Growth rules
(a) $Z_1 + h < Z_2 < Z_1 + h + h'$; $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} < r$
(b) $Z_1 < Z_2 < Z_1 + h$; $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} < \frac{r * Z_2}{h}$
(c) $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2} < r$
$[x_2, y_2, z_2]$
$[x_1, y_1, z_1]$
r
h'
h
(a)
(b)
(c)
Source neuron's cell body location
Target neuron's cell body location

Fig. 2. Growth rules as a set of geometrical formulas defining space in which axon of source neuron can find its destination neuron, defined in cartesian coordinates and inspired by morphology of biological neurons; source neuron's coordinates represented by $[X_1, Y_1, Z_1]$ and a potential destination neuron's coordiante represented by $[X_2, Y_2, Z_2]$: (a) growth rule "a" defines a cylindrical boundary into which the destination neuron coordinates can fall, where h is neuron's distance from the cylinder's base, h' is height and r is radius of cylinder; (b) growth rule "b" simulates cone-shaped synaptogenesis region similar to that occupied by pyramidal neuron's axons, where h is height of the cone, and r is radius of its base; (c) growth rule "c" represents spherical-shaped neuron with axon length reaching maximum r as sphere's radius.

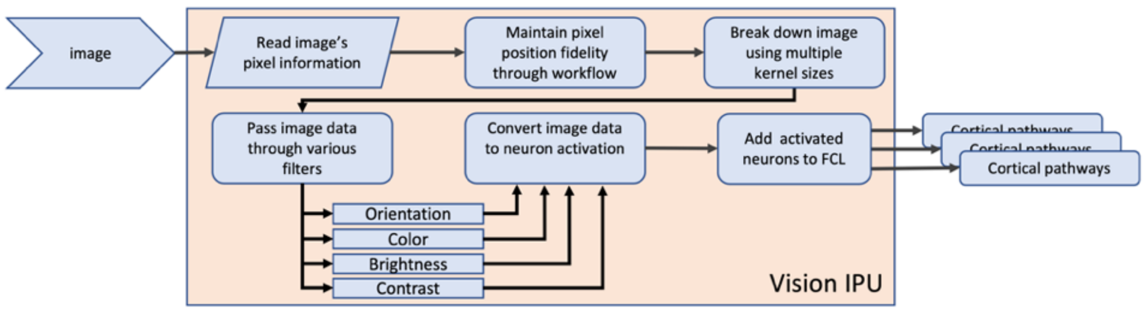


image
Read image's pixel information
Maintain pixel position fidelity through workflow
Break down image using multiple kernel sizes
Pass image data through various filters
Convert image data to neuron activation
Add activated neurons to FCL
Cortical pathways
Cortical pathways
Cortical pathways
Orientation
Color
Brightness
Contrast
Vision IPU

Fig. 3. Flow of information through vision IPU.

portals and have that information ready to feed to cortical pathways in the form of neuron stimulations. Similarly, to be able to understand the artificial brain's output, either by humans or a computer, neuronal activities must be processed and relayed to the corresponding output device.

The biological brain receives information from the peripheral nervous system connected to various receptors such as photoreceptors, thermoreceptors, chemoreceptors, and mechanoreceptors. Similarly, in FEAGI, every input processing unit/output processing unit (IPU/OPU) would differ in how information is collected, encoded, and presented to the first cortical layer as part of a particular functional cortical pathway. As an example, for visual information, there will be a designated IPU that will convert the image of a given object to its fundamental elements such as color and brightness while maintaining the pixel arrangements, similar to the retinal information-processing structure. The human eye collects optical information using photoreceptors and passes it through an assortment of cells before forwarding it to the thalamus via ganglion cells [27]. Additional mapping occurs through the six layers of the lateral geniculate nucleus (LGN) within the thalamus. In the proposed framework, the visual IPU plays the same role as the retina, and the rest of the processing activities are handed off in the form of neuronal stimulation to the neural processing unit representing the cortical pathways.

In the design of the vision IPU, we have applied two fundamental principles inspired by nature. The first is based on how neuron projections maintain their position fidelity against their neighbors as they project from the retina to LGN, and from there to the striate cortex. The second principle corresponds to how horizontal cells with dendrites spanning sideways collect information from neighboring neurons. We have applied this principle by leveraging a well-known technique in image processing that involves applying a kernel convolution matrix by which a small-size matrix is used to scan the image while applying filters. This method helps with breaking down the image into its fundamental components such as orientation, color, brightness, and contrast. Sensitivity to these fundamental factors leads to the activation of neurons at the beginning of the visual cortical pathway, as shown in Fig. 3.

The process of decoding memory back to output processing unit first requires proper neuron wiring from the memory region to the OPU, which would be driven by the genome. The second step is the actual decoding of neuron signals received from its corresponding memory unit to the actual physical signal needed to interact with systems that are external to the artificial brain. In our proof of concept, we have only implemented the UTF-8 OPU. The neuron wiring for UTF-8 IPU was a one-to-one neuron mapping from the memory region to a one-dimensional matrix representing individual UTF-8 characters.

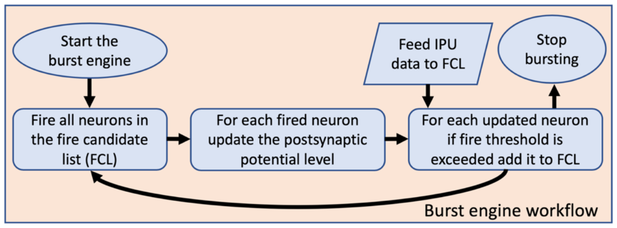

Start the burst engine
Feed IPU data to FCL
Stop bursting
Fire all neurons in the fire candidate list (FCL)
For each fired neuron update the postsynaptic potential level
For each updated neuron if fire threshold is exceeded add it to FCL
Burst engine workflow

Fig. 4. Artificial brain built by FEAGI running on bursts, whereby list of neurons ready to fire is added to fire queue, and they all fire simultaneously when burst is triggered, a mechanism used to overcome the parallel nature of spiking neurons when implemented in environment with limitations in parallelism.

F. Neural Processing Unit

The NPU is where the majority of the artificial brain's physiological behaviors are managed and orchestrated, while anatomical definitions are all driven by the connectome. The NPU consists of a collection of functions that are described as follows:

1) Neuron Firing, Queues, and Bursts: There are approximately 100 billion neurons in a mature human brain, and the timely firing of individual neurons drives the transferring and processing of information across the brain. There are two significant challenges to simulating the brain; one associated with the brain's architecture stemmed by how neurons are interconnected, and the other tied to the parallel processing nature of the brain since every neuron can act independently of others. To address the latter, we have developed a method called "burst engine."

The concept of burst engine is based on event-driven simulation models [31]–[34] and the underlying principle of how FEAGI processes information. Every burst instance is associated with a list of neurons, in the form of a queue, whose membrane potentials have passed their firing thresholds and are ready to fire. This list is referred to as the "fire candidate list" or FCL, as shown in Fig. 4. The burst engine fires all neurons listed in the FCL and simultaneously updates the membrane potential of the neurons, which have synaptic connectivity with those that are firing.

Synaptic information is queried from the connectome through dictionary lookups. If any of the destination neurons pass their membrane potential, then they will be added to the FCL and fired in the next round of the burst engine cycle. At any point in time, we maintain two instances of the FCL, one of which is the previous instance. Comparing the content of these two lists has enabled us to implement neuroplasticity, which is explained in detail in Section III-F4. Managing neuron firing through bursts has helped us gain control over the highly parallel nature of spiking neurons, which play a crucial role in learning and memorization. We have analyzed the algorithmic complexity of our burst engine logic and have evaluated it to be $O(n^2)$. This complexity is associated with the number of firing neurons times the average number of synapses associated with each of them. It is important to note that the total number of neurons is not a contributing factor.

2) Neurotransmitters: In the biological brain, after the neuron firing occurs, depending on the type of neurotransmitters released by the pre-synaptic neuron, different types of behavior are observed on the post-synaptic neuron. FEAGI accounts for both excitatory and inhibitory neurotransmitter effects. We have defined excitatory behavior to update the post-synaptic neuron properties within the connectome to show an increase in action potential every time the upstream neuron fires.

In our implementation, the inhibitory behavior takes two different forms: the first is based on the inhibitory synapse type, where upon firing, the firing neuron would have a negative impact on the action potential of the downstream neuron; the second form keeps track of the number of consecutive firings of a neuron within a window of bursts, and upon exceeding the threshold, a snooze flag is activated, thus preventing neurons from firing for a predefined number of bursts as defined in the genome. Implementation of this inhibitory behavior has proven to be very beneficial since we observed that without it, neuron clusters were entering an infinite loop of firing. The inhibitory function acts as a damper, thereby regulating excessive feedback and stabilizing the communication.

3) Refractory Period: The refractory period in the biological brain is a duration post-neuron firing, whereby the neuron would be unable to fire again until the period is over. In our implementation, instead of the absolute time, we have used the concept of bursts in a way that after a neuron fires, we prevent it from firing for a predefined number of bursts. The refractory burst count is defined for a given neuron as part of the neuron properties in the genome for a given cortical layer/sublayer, and its value can evolve over generations.

4) Neuroplasticity: Hebb's hypothesis [27] famously states that "Neurons that fire together, wire together." We have implemented this hypothesis as a variant implementation of spike-timing-dependent plasticity (STDP). In traditional STDP models [35], there is a continuous relationship between synaptic weight and time-delta between the arrival of the pre-synaptic spike and the post-synaptic spike, that is, the longer the time-delta, the lower the synaptic weight of the STDP. In our implementation, we have leveraged the sequence of bursts produced by the burst engine and kept the window of influence down to a single burst sequence, as shown in Fig. 5, and the mathematical formula deriving the first graph is as follows:

$$SynapticWeight = \begin{cases} C & \text{if } \Delta B = +1 \\ -C & \text{if } \Delta B = -1 \\ 0 & \text{if } |\Delta B| > 1 \end{cases}, \tag{1}$$

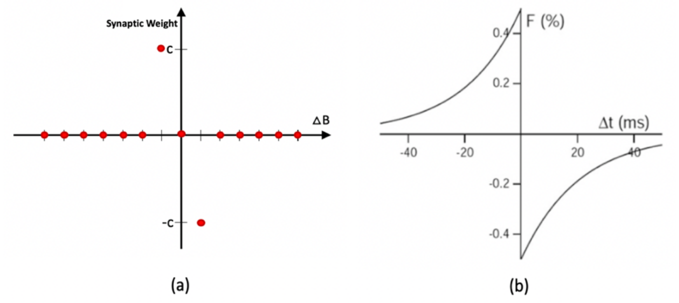


Fig. 5. Relation between spike arrival time-delta and synaptic weight: (a) In FEAGI, the difference between the burst sequence of the pre-synaptic and post-synaptic neuron determines the synaptic weight applied during long-term potentiation and long-term depression. Only if the difference of firing is one burst, will there be an impact, and for anything beyond the impact will be zero. (b) In depicting the typical STDP model, the difference between the arrival time of the pre-synaptic and post-synaptic spike (x-axis) determines the synaptic weight (y-axis) [35].

where C is the potentiation constant defined as a gene for each cortical layer, and ΔB is the difference between the burst sequence identifier associated with the post-synaptic firing neuron B_{post} and the pre-synaptic firing neuron B_{pre}. That is, $\Delta B = B_{\text{post}} - B_{\text{pre}}$.

Assuming the upstream neuron A has its axon connected to neuron B, in STDP, when A fires briefly before B, this leads to LTP, and when A fires briefly after B, this leads to LTD on the synapse between A and B. In our implementation, when A fires in burst n and B fires in burst $n+1$, this leads to LTP, and when A fires in burst n and B fires in burst $n-1$, then LTD occurs. In other terms, when a post-synaptic neuron fires in burst $n-1$ and a pre-synaptic neuron fires in burst n, there will be a penalty on the strength of the connection between the two neurons.

5) Synaptic Pruning: As the brain operates, neuroplasticity causes synaptic weight among neurons to increase or decrease. In some cases, the synaptic weight between two neurons may reduce to a very small number or zero. Such synapses will be eliminated. This is synonymous to the synaptic pruning phenomenon in the biological brain.

6) Spiking Neuron Model: The neuron model is a mathematical and logical description defining a neuron's behavior. Many neuron models have been proposed and studied over decades [36]. In our implementation, we have adopted a custom model that is largely designed based on the well-known leaky integrate and fire model [29]. Given we have adopted a burst centric logic for our framework, this design decision has fundamentally influenced the underlying neuron model, as shown in (2):

$$P = \sum_{i=1}^{n} I_i - \frac{L_b}{L_c}, \tag{2}$$

where P is the membrane potential for the neuron, I_{i} is the post-synaptic current of the i^{th} upstream neighbor, L_{b} is the number of bursts since the last neuron membrane potential calculation, and L_{c} is the leak constant. Equation (3) describes the neuron fire trigger condition:

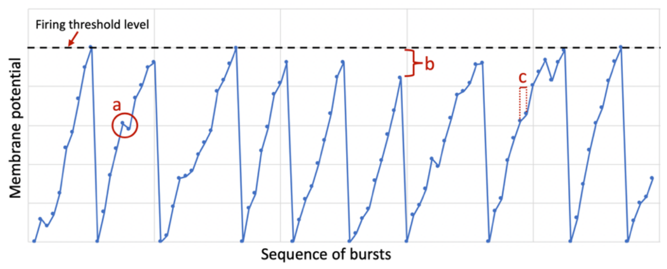


Fig. 6. Changes in neuron membrane potential across consecutive bursts: (a) A dip in the membrane potential has occurred since the neuron leakage has overpowered the influence of the upstream neurons. (b) In some cases, there is a perception of premature firing on the graph, which is due to the neuron fire logic occurring before recording the membrane potential in the connectome. (c) This area indicates of the span of a single burst.

$$(P > F_t) \wedge (F_b > R_b) \wedge (C > S), \tag{3}$$

where P is the membrane potential, F_{t} is the firing threshold, F_{b} is the number of bursts since the last neuron firing, R_{b} is the neuron's refractory period in the form of burst count, C is the pointer identifying the current burst counter position, and S is the pointer identifying the burst counter position by which the neuron was flagged in order to be snoozed. Changes in membrane potential across bursts are depicted in Fig. 6.

G. *Memory and Learning*

Both memory and learning are closely interrelated and together play a crucial role in cognition and intelligence. In this research, we have mainly focused on long-term declarative memory and have deferred coverage on other memory types to future research. Many structures within the human brain are involved in declarative memory formation, and depending on the input type, different cortical areas play a role.

1) Memory Design: Donald Hebb, in a book published in 1949 [37], proposed the concept of "cell assembly" as a group of simultaneously activated neurons representing an object in response to an external stimulus. We have leveraged this concept as the core foundation of our memory design.

2) Memorization Process: When an input processing unit is exposed to an external stimulus, it triggers a set of chain reactions. First, raw information is broken down by the IPU using functions that are sensitive to the fundamental physical properties of the stimuli, as explained in Section III-E. For example, a sound will be broken down into frequency and magnitude modulation bands, and an image will be broken down into contrast and color variation before they are fed back into their designated functional pathways, as shown in Fig. 3. Functional pathways are a collection of interrelated cortical layers with different neuronal properties working together to process the information further. At the end of the pathway, a semantic pointer [24] will represent the external stimulus as a unique signature and then have it projected to the memory region. As the memory neurons ensemble becomes activated in close time proximity, long-term potentiation (LTP) occurs among them, thus leading to a strengthened synaptic connectivity among neurons tied to the stimulus. This ensemble is what we refer to as a "cell assembly", as shown in Fig. 7.

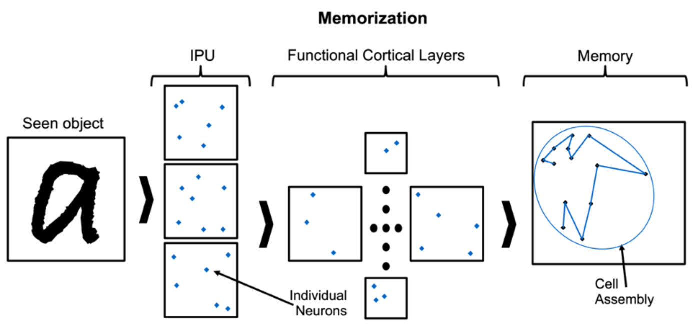


Fig. 7. Conceptual view of how memory is formed in FEAGI: process begins with raw data fed to IPU for initial processing and conversion to neuronal activity, then supplied to cortical pathways, and ultimately landing in memory region in form of cell assembly.

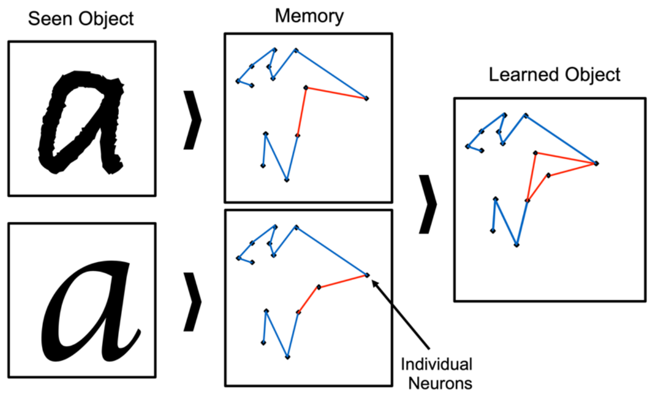


Fig. 8. Conceptual view of how multiple memory formations can join to present same object.

Given that the activation of memory neurons by upstream cortical layers is an asynchronous process, the formation of the cell assembly is gradual and along the way, leads to the increase in synaptic connectivity between neurons, but not all to the same extent. In general, neurons participating in a cell assembly may not be fully connected.

3) Declarative Memory: Declarative or explicit memory is a type of memory related to events or facts and can be consciously recalled [27]. Declarative memory can be classified into working short-term memory and working long-term memory. Long-term memory is the one we implemented as part of the framework. In our implementation, long-term memory consists of a pool of loosely connected neurons. When an object stimulates the IPU, some neurons in the memory pool become activated; when another similar object is exposed, another set of neurons becomes activated. When both objects are introduced to the IPU in a short period, a large percentage of activated neurons will be common between the two events, due to event similarities. This situation is where the enhancement of synaptic strength between neurons belonging to the cell assemblies associated with each object occurs. After multiple occurrences of seeing both objects simultaneously, the strengthened synaptic connection between neurons would tie the objects together. Therefore, if the stimulus related to the first object occurs, then it would lead to activation of neurons related to the co-occurring object as well, as shown in Fig. 8.

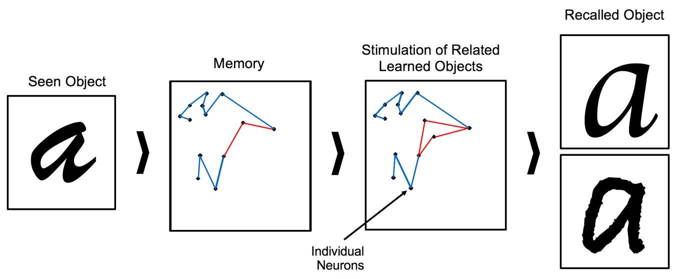


Fig. 9. Conceptual view of how memory recall can occur, assuming that artificial brain has been previously trained on two different forms of letter “a”, thereby forming two cell assemblies with a large number of neurons in common, as shown in Fig. 8; exposure of a new form of letter “a” results in commonality of neurons between new cell assembly and the other two, thereby activating them and triggering memory recall.

4) Memory Recall Process: The recall process involves the output processing unit. After the memorization process has taken place and different variations of an object have been stored in the long-term memory, the stimulation of a version of the trained object will lead to activation of the memory cells associated with all close variations of that same object. At this point, memory cell activations generate stimulation in the OPU neurons, which ultimately leads to a physical manifestation. For example, seeing the character “a” in one font can activate neurons related to previously seen versions of “a” in other fonts, which in turn triggers the OPU associated with UTF-8 output to tell the operating system that the letter “a” was seen, as shown in Fig. 9.

5) Associative Learning: Associative learning is responsible for the creation of event associations in our brain. In the late nineteenth century, in a famous experiment performed by Ivan Pavlov, a dog was repeatedly exposed to the sound of a bell when offered a piece of meat. Initially, the dog’s response to being shown the meat without the bells sound was salivation, but after repeated co-occurrences of showing the meat and ringing the bell, eventually the dog made an association between the bell ringing and the meat, whereby ringing the bell alone became the trigger for the dog’s salvation [27]. Implementation of associative learning in FEAGI was inspired by this behavior in the biological brain as follows: When an IPU processes a stimulus, say IPU A, this generates neuronal activities in its corresponding cortical pathway, ultimately activating a group of neurons in the memory region associated with the IPU; at the same approximate time, when IPU B processes a second stimulus, this leads to another unique set of neuronal activities in the memory region tied to IPU B. This is where we leverage Hebb’s hypothesis one more time and wire together the neurons from different memory regions that are firing together.

6) Self-Learning: To be able to have the artificial brain evolve through generations, we saw the need to have a self-learning module so it can teach itself numbers and do a self-assessment at the end. Self-assessment results are captured in a database and used for fitness evaluations. For example, to be able to learn the number seven, the biological brain needs to be exposed to variations of this number over and over in a short period so that association can occur. To perform this operation In FEAGI, different variations of the character's image from the MNIST database are repeatedly fed to the vision IPU approximately when the correct UTF-8 version of it is fed to the UTF-8 IPU. This task is repeated over and over with multiple variations of the same character's image. Information flows in the form of neuronal activities through vision and UTF-8 cortical pathways simultaneously, and when reaching the corresponding memory modules, distinct cell assemblies begin to form. After the learning process for a particular character is complete, the self-learning algorithm moves on to the next character in the queue. This method was inspired by how children learn the alphabet through simultaneous printing and observation that is repeated many times.

7) Training Methods: Both supervised, and reinforcement learning techniques have been used as part of this framework.

- Supervised learning: Supervised learning is implemented in such a way as to read digits from the MNIST selection randomly and expose them to the IPUs for the exposure period. This is followed by a period of delay and repeat.
  - Reading from MNIST: Variations of labeled images of numbers from the MNIST database are read, and the image data along with label information are simultaneously fed to the vision and UTF input processing units correspondingly.
  - Exposure duration: A variable has been defined to be able to adjust the amount of exposure each input would have to the system and is measured with the unit of burst count. The smallest exposure duration is a single burst. The exposure equates to feeding the raw data associated with the stimulus to the input processing unit repeatedly for the exposure duration. We have explored the impact of changing the exposure duration on trainability and noticed that lowering the exposure period below ten burst instances would have a negative impact, but increasing it above that threshold would not have any significant influence. We have not explored what parameters would drive this threshold, but we suspect that the number of neuron layers that the information needs to pass through before stimulating the output processing neurons would have an impact on it.
  - Training order: The training module randomly reads digits from the MNIST collection without any particular order.
  - No activity period: It is essential to have a no-activity period between each exposure phase to assist with neuronal activities from the previous exposure decay within the cortical layers and help get rid of the noise before a new set of activities occurs.
- Reinforcement learning: We have leveraged the concept of pain and inhibitory neurotransmitters to perform reinforcement learning. Stimulation of a wrong image label leads to the activation of pain receptors. Pain receptors stimulate pain neurons, which in turn apply long-term depression (LTD) against the synapses involved in activation of a wrong response. Similarly, we award the synapses involved with the correctly recognized object by applying long-term potentiation (LTP).

8) Testing: The testing framework has much overlap with the training framework with the difference being that during testing, the label information is not fed to the system. After an image is exposed and the neuronal stimulations have passed through the cortical layers, a listener monitors the activities of the UTF8 output processing unit and records the results. There will be three scenarios: first, the stimulation of the output neuron could correspond to the correct image label; second, it could correspond to a wrong label; and finally, there could be no stimulation. We capture this information in a data structure and use it for the brain fitness calculation when the testing period is over. Ultimately, test results are saved in the genome database alongside the genome.

H. Evolutionary Module

Multiple factors influence the evolution of a system, including imposed environmental constraints, the effects of external and internal environmental changes on the system dynamics, and survival needs. A system can evolve both while it is active and after becoming deactivated through genetic evolution.

A brain developed by FEAGI evolves from multiple aspects. The first evolutionary instance occurs during the artificial neuroembryogenesis process while the artificial brain is grown from a genome until it reaches maturity. The second occurs in the form of plasticity while the artificial brain is exposed to an external stimulus and undergoes the learning process. In this case, new connections are made, and existing connections are strengthened, deteriorated, or eliminated. The final evolutionary aspect of FEAGI is centered around using genetic algorithms to evolve the genome over generations, as explained below.

1) Genome Database: To improve an artificial brain instance, we collect statistical data associated with the overall artificial brain performance during cognitive tasks throughout its lifetime, and we store it in a genome database, alongside the genome responsible for the creation of that artificial brain, as shown in Fig. 14 in Appendix . Collected statistical data are crucial to the creation of the future artificial brain offspring through genetic algorithms. As part of the offspring creation, mutation is injected into the genome so that the next generation can be created with a new network structure, thereby leading to a different performance. Genomes from high performing artificial brains are crossbred to generate a new generation.

As an example, as part of gene modification, the gene outlining the neuron density in vision V1 can be increased in value, which would result in the creation of a new artificial brain with the V1 layer containing more neurons with different behavior. Similarly, different firing patterns can be exposed to different layers.

Our framework is capable of collecting statistical data while the brain is interacting with external stimuli, and save it in the form of metadata along with the genome. The burst management design discussed in Section III-F enables our framework to efficiently collect statistical data from all cortical areas while the artificial brain is active. Another important source of information for the evolutionary module is the self-learning infrastructure. We measure the fitness of the system based on the percentage of correct MNIST digits recognized by the system after it has gone through the learning process.

Every time our artificial brain goes through a life cycle, it collects statistical data, and when it is ready to shut down, all genomic information along with statistical data is stored back in the genome database. When a new offspring is generated, a new genome entry is added to the database. All we need to produce the next generation of an artificial brain is a genome from the database, along with a genetic algorithm.

Fig. 14 outlines the hierarchy involved in the genome database. For each genome instance, there are two main sets of information captured: one set, categorized as "properties," captures all parameters needed to regrow a new artificial brain; the second set, categorized as "statistics," captures the artificial brain's performance and behavior during its lifetime.

2) Genetic Algorithms: The core premise of FEAGI as an evolutionary framework is to allow the artificial brain's architecture to evolve through generations. To this end, we have developed an evolutionary Python library as part of FEAGI, capable of reading a genome instance and performing genetic algorithms as follows:

- Selection: Every time a new brain instance is about to be developed, the first step is to provide the genome. We have introduced a logic to randomly select from the following three options for genome selection:
  - Select a genome from the database with the highest evaluated fitness.
  - Select the newest genome in the database.
  - Randomly select any genome from the database.

  After the selection, we randomly decide on whether to use the selected genome as is or to let it go through either mutation or cross-over.
- Mutation: Mutation is performed by modifying the value of various parameters (genes) within the genome by a random percentage between -30% and $+30\%$. The choice of $\pm30\%$ was initially a good choice because it allowed the genetic algorithm to explore a broader range of parameter values without getting stuck in a local minima. After the fitness grew beyond 50%, we changed the window to $\pm10\%$ to allow for smaller fluctuations and hence finer tuning. Changing parameter values through mutation would result in a new anatomical structure and influences the behavior of the functions simulating brain physiology that ultimately leads to an improved or diminished system performance. In our current implementation, the mutation is limited to a subset of genes within the genome, meaning that not all parameters defined within the genome are subject to mutation. The reason for this choice is to limit initial development efforts, indeed an area for future improvements.
- Crossover: Crossover occurs when a portion of one genome is swapped with another. In our implementation, we have adopted a simplified approach. We randomly select two genomes from the pool of top-performing genomes available in the database, and then read a random set of genes from one genome and swap values with its counterpart.

IV. System highlights

FEAGI consists of two significant workflows that closely interact with each other: one is the evolutionary workflow that handles genome and is responsible for regenerative tasks, and the other, the information workflow that is responsible for interacting with outside world, data-processing, memorization, and learning, as shown in Fig. 15 in Appendix.

Two of the essential characteristics of FEAGI as a brain simulation framework are *scalability* and *evolvability*. Considering the sheer number of neurons working together in the human brain to achieve cognitive tasks, it is vital to expect the same from a framework attempting to replicate its behavior. The scalability and evolvability factors become crucial when our goal is to achieve an efficient and effective network design and architecture. Here, we list some of the contributing factors in scalability and evolvability of the framework. The framework offers the user two options during the initial launch: one to grow an artificial brain from scratch using the genome file and the second to load a previously saved one. The latter option would load a fully connected artificial brain along with all the memories and learned artifacts.

A. Scalability

- Modularity of genome structure
  - Genome modularity can enable any number of cortical areas to be defined independently and added to an existing artificial brain model.
  - The genome is structured such that scaling up the artificial brain by adding cortical layers would be as easy as adding a few entries in genome that would define the cortical layer's anatomical properties.
  - Genomes are stored in MongoDB in the form of JSON documents. MongoDB is extensible and highly scalable, enabling FEAGI to capture genomic information for millions of generations.
- Burst engine: This technique is a crucial scalability factor leading to a reduced memory and energy footprint needed to operate an artificial brain with billions of neurons.
- Connectome design and structure: We have chosen an object-oriented data structure to capture the artificial brain's anatomy in the connectome so that it can be stored in a scalable database format. As of now, and without any compression or optimization, each neuron requires about 1 kilobytes of storage. Size of the connectome scales linearly with the number of neurons. If an entire human brain with about 100 billion neurons is simulated, then would require 100 terabytes of storage with this implementation, which is well within the supported limits of existing document databases such as MongoDB.
- Multiprocessing: We have leveraged multiprocessing techniques to allow parallel operations, such as neuron firing, to take advantage of multiple CPU cores.
- Modularity of input and output processing units: Each IPU or OPU is independent of the others and can plug into the framework by passing along processed stimuli

to cortical pathways. This design enables the framework to add support for processing new types of input such as sound or tactile sensation as well as interacting with the outside world through various output methods.

B. Evolvability

- Genome-based growth:
  - Our design is based on segregating brain anatomy from its physiology. The genome contains enough information to be able to grow an artificial brain step by step, thereby enabling an evolutionary algorithm to be used to influence its growth and maturation, which can ultimately be used to improve its performance over generations.
  - The genome database enables data analytics to be performed against statistical data collected from each running brain, which can be a powerful tool in identifying influential genes and high-performing genomes.
- Plasticity: Synaptic plasticity evolves the artificial brain structure, thereby enabling learning and memorization.
- Scale-out evolution: Our framework has the capability of spawning multiple artificial brain instances simultaneously in order to speed up going through multiple generations as part of the evolutionary process. There is currently a limitation that multiple instances can be created only within a single operating system instance, and it is our goal to lift this limitation in future iterations.

C. Insights

The artificial brain built by FEAGI, similar to the biological one, can grow considerably large and encompass numerous neurons and synaptic connections. To gain insight into the interworking of the artificial brain, we designed and implemented an infrastructure to allow raw data to be collected while the brain is active and have them stored in a time-series database called "InfluxDb." We then leveraged an open-source monitoring solution called "Grafana" to develop a custom dashboard to help visualize activities in various regions and cortical areas, as shown in Fig. 16 in Appendix.

V. Proof of Concept

To demonstrate the feasibility and success of this framework, we have implemented a simple simulation of the visual system that is limited to monovision in grayscale, which is only capable of processing static images. We have developed two different IPUs; one capable of reading 28x28 images from the MNIST database, and the other capable of reading UTF-8 characters from standard input. UTF-8 input has been used for supervised learning to teach the artificial brain the association between a character's image and its UTF-8 version.

The visual cortical pathways in our proof of concept have been built after the human brain's ventral visual pathway consisting of layers V1, V2, V4, and IT, but for simplicity, we have omitted layer V4. Similar to the human primary visual cortex, we have defined orientation selectivity for V1 sublayers. Layer V1 in the human brain consists of six sublayers, and we have implemented the same. Additionally, in layer V1 of the biological brain, there is the concept of ocular dominance columns [27], which we have ignored since our focus is only on monovision implementation. We have also ignored structures such as cytochrome oxidase blobs [27] since they relate to color detection, and our implementation has been limited to grayscale. From the connectivity perspective, vision IPU output is mapped to the six V1 sublayers. Every V1 sublayer projects its axons to layer V2, and layer V2 has its neuron axons projected to layer IT. Individual neuron connections within layers as well as from one layer to another are driven by the synaptogenesis rules that we have defined in the genome. We tried to define the genome parameters in a way that is close to our understanding of the human brain.

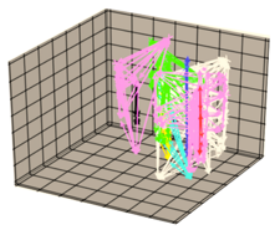


Fig. 10. Visualization of cell assemblies. Each color cluster corresponds to the cell assembly associated with a particular number.

Memory neurons in the genome are defined to not have synaptic connectivity to others initially. Rather, upon simultaneous firing, long-term potentiation can occur, and they synapse with each other. As a group of memory neurons becomes activated over and over at approximately the same time, they bond and form the cell assembly, as described in Section III-G1, and help to shape memories.

Visual memories are created by the formation of cell-assemblies that collectively represent the memory of a particular object. We have visualized the formation of cell assemblies as the artificial brain undergoes training, as shown in Fig. 10. Each color is representative of a cell assembly associated with a particular number. We can observe that each cell assembly has a unique shape and form. Another observation is the fact that there are many overlaps between various cell assemblies. This overlap harms the fitness of the brain because it leads to misidentification of one number with another.

We have developed an OPU capable of translating UTF-8 memory patterns back into a UTF-8 character. Development of the UTF-8 OPU has enabled us to build an end-to-end example of how information can be fed to the artificial brain, stored in long-term memory, and presented back to the outside world within the proposed framework.

The UTF-8 IPU and OPU have a simple cortical structure and are custom made to be the translation layer between the neuron spikes and the operating system. The UTF-8 IPU, OPU, and memory are all one-dimensional cortical regions consisting of a single column of neurons, where each neuron's axon is connected directly to its corresponding neuron in the downstream UTF-8 layer.

The process of image recognition starts with the vision IPU, (refer to Section III-E for more details) and continues through the cortical pathways. By passing through the vision IPU, the

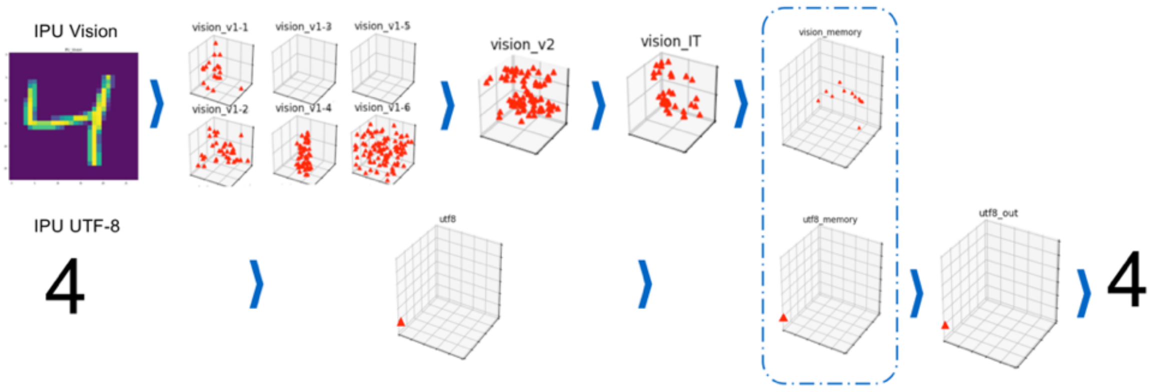


Fig. 11. Neuronal activities within the framework leading to association of UTF-8 version of character with handwritten version read from MNIST database.

image breaks down into its fundamental components and feeds as a series of spikes to the first layer of the cortical pathway connected to the vision IPU, which happens to be layer V1. Neurons in each cortical layer have axons connected to other neurons as part of their subsequent layers, so activation in one layer propagates through the cortical pathway and eventually reaches the final layer, or IT layer. This layer is the only one with synapses to the memory neurons, so when neurons in layer IT are activated, they stimulate memory neurons associated with visual memory. When the act of seeing the image is repeated over and over, this leads to the stimulation of the same memory neuron ensemble over and over, thus leading to the formation of long-term memory.

The next phase of learning is the association of a UTF8 character with the visual memory. The essential factor in enabling the artificial brain to accomplish this is the wiring of neurons from the visual memory to the UTF-8 memory. When neurons from both memory regions fire together, they wire together and improve their synaptic connectivity. UTF8 has its dedicated cortical pathway, which is simple. When the user enters a key on the keyboard at the same approximate time the image is read by the vision IPU, the UTF signal propagates and reaches its UTF8 memory region. At this point, we leverage the long-term potentiation technique again but now between the two memory regions. As the cell assemblies in both memory regions become activated, they become wired together, so now the activation of one can lead to the activation of the other. This simple yet powerful function would tie memories from one functional memory region to another, so that the simulation of one leads to the stimulation of the other.

Fig. 11 depicts the neuronal activities in our framework when the number 4 is read from the MNIST database and passed through visual pathways. The activation of neurons in the IPU creates a chain reaction leading to the activation of neurons throughout the cortical pathway. On the other hand, at the same time frame when the number 4 is read from the MNIST, the user types the character 4 in the framework, it is fed to the UTF-8 IPU, and from there it passes through the UTF-8 cortical pathway. Every time the neurons that collectively represent the number 4 from the vision and UTF-8 memory regions fire, their connectivity improves, and eventually the activation of one group can lead to the activation of the other. Similar to a biological brain, the learning process demands repetition and proper timing. When we learn a new object, we look at it and repeat its name over and over to help memorize the association. The same happens in our framework. After multiple instances of simultaneous memory stimulations, the artificial brain learns that an MNIST version of number 4 is associated with the UTF-8 version of 4.

During training, the artificial brain is exposed to both image and UTF representation of a character at the same time, so memory associations occur, and later on, when the same character is seen again, this leads to the activation of the same cell assembly and consequently the activation of its UTF counterpart in another memory region. At this point, the neuron connections of the UTF8 OPU with its memory neurons help stimulate the OPU region associated with the seen character, thereby resulting in the display of character on the computer display.

When a new object exposed to the system is similar to a previously seen version, this leads to the stimulation of a large majority of neurons representing the original object. This stimulation fuses the cell assembly of one object to the other, and if it happens that the first object was associated with a UTF-8 character, then the new object will also stimulate the same character. Fig. 17 in Appendix depicts this phenomenon. First, the artificial brain is trained to recognize variations of the number 1 along with the UTF-8 version of it. Next, the brain is exposed to an unseen version of the number 1, which leads to the stimulation of the UTF-8 version of the number 1 as well.

VI. Results and Discussion

We have allowed our proof-of-concept implementation to evolve through many generations. An area of interest is to prove that the artificial brain created by FEAGI has the capability of improving itself over generations. To evaluate improvements, we have defined a fitness function as follows:

$$Fitness = \frac{1}{1+e^{(-t+a)}} \times \frac{c}{t}, \tag{4}$$

where c is the number of times the artificial brain has been capable of correctly identifying a stimulus, t is the total number of times the stimulus has been exposed to the brain, and a is the activity threshold. In (4), the first fraction is

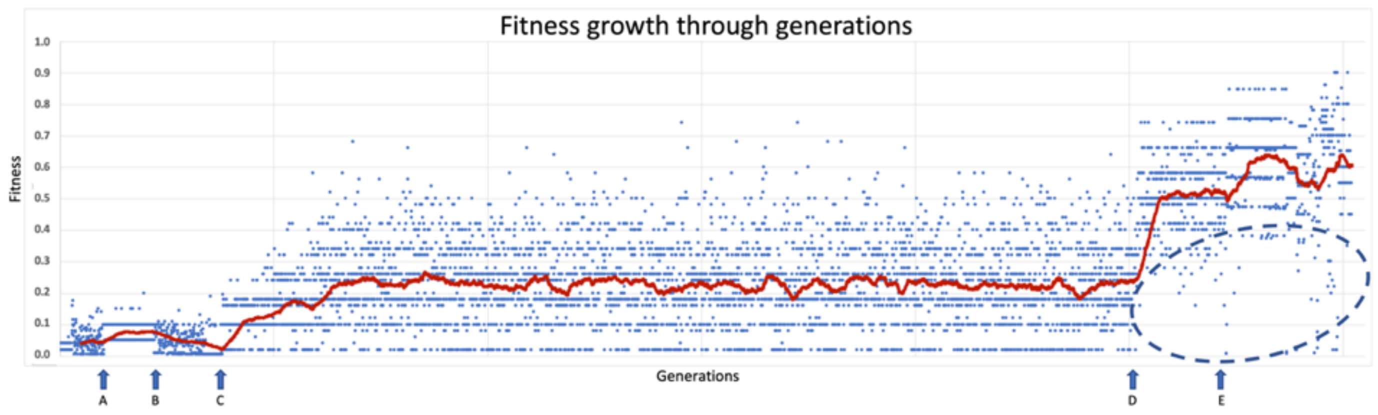


Fig. 12. Artificial brain's fitness improvement over generations. Individual dots outline the fitness value for a given generation; the line is a moving average over the past 100 data points; letters A, B, and C are indicators of a generation phase where a manually induced physiological mutation has occurred; letter D is where we introduced the early termination feature; letter E is where reinforcement learning was introduced; and the dotted line oval is highlighting the observation of less low-performing instances due to the early termination feature. Data points with a fitness of zero have been excluded from the graph.

a shifted sigmoid function that imposes a negative influence when the number of exposures is below a threshold.

A. Performance Improvements over Time

We have developed a baseline genome outlining the growth rules, anatomical properties, and limited physiological characteristics of various cortical regions, and we have set the brain to evolve over thousands of generations using two sets of MNIST digits from 0 to 9. Each generation has gone through the self-learning and self-assessment processes, and as a result, a fitness value was calculated and associated with a given genome. As shown in Fig. 12, the fitness of the brain in the latest generations has improved significantly compared to the earlier ones, and the overall trend is positive. We can also observe an interesting phenomenon that is highlighted by the letters A, B, and C. These letters are indicators of the generation points where we manually induced a physiological mutation in the system. We observe that the fitness trend noticeably changes at these points, especially point C. We hypothesize that the fitness improvements if only driven by the anatomical changes would reach a resistance level after finite generations and will only fluctuate within a range until a physiological change can take it to the next level, as shown by point C in Fig. 12. We are deferring the proof of our hypothesis to future research. Point D in Fig. 12 corresponds to the generation where a feature for early detection and termination of low performing brains has been introduced. This feature monitors the level of neuronal activities in the visual memory, and if the level is zero, proactively terminates the training process and eliminates that generation. This feature has enabled us to have a more efficient genetic algorithm by not wasting time going through the full training and testing process on generations with low potential. As a result, we observe a relatively empty region right after point D, highlighted with a dotted line oval. Point E in the Figure is associated with the introduction of reinforcement learning capability as explained in Section III-G7.

An important factor to discuss is the fitness value. As shown in Fig. 12, our current measurement of fitness ranges from 0 to 0.9, with 1 being the perfect score. We acknowledge that this is a low score in comparison to state-of-the-art handwriting-recognition algorithms [38]. For our initial proof of concept, we have identified below a few areas that we think are contributing to not achieving a competitive level of fitness:

- Evolutionary constraints: In the biological brain, physiological activities play a crucial role in how the brain functions. In our implementation, we have simulated such activities by writing functions in Python; an example is the one for neuron firing. As part of these functions, variables have been defined that influence how the function behaves; an example is a variable representing the firing threshold for the neuron firing function. We have externalized all such variables used as part of functions to the genome so they can evolve. The major shortcoming that still stands is the ability to evolve the structure of the code defining the functions themselves. In this regard, we propose two areas for future research: first, to find methods and approaches to limit code-dependency for the physiological functions, and second, to investigate a framework with self-evolving code capability.
- Constraints on parallelism: Our code is currently only leveraging the CPU. Leveraging the GPU can help reduce the training time considerably, thereby leading to a shorter lifecycle for each brain instance and ultimately a higher number of generations over time. Taking advantage of the neuromorphic hardware would take this one step further.
- Scale-out limitations: Currently, we cannot scale the creation of brain instances beyond a single operating system instance, which is limiting us to have the framework run on multiple servers while sharing the same genome database. Adding this feature would allow the system to infinitely scale-out and leverage a large number of computing nodes in the cloud or within a data center.

B. Impact of Exposure Time on Fitness

Fig. 18 in Appendix highlights the results of our research on how changes in stimulus exposure time could impact fitness in a particular generation. We can observe that a slight change in the exposure time can have a significant impact on the fitness value –changing the exposure time from 8 to 15 bursts can improve fitness up to 20%. We also observed that for this particular genome, increasing the number of training sets from 10 to 60 is not contributing to fitness improvements. After further investigations by monitoring cortical activities in the layers leading to the visual memory region as well as inspecting the cortical anatomical properties, we determined that the selected genome was mutated as such that has led to sub-optimal synaptic connectivity between layers V4 and IT. This deficiency led to the scarce and localized firing of neurons in the IT region that subsequently created a narrow data flow to the visual memory region, which ultimately caused low learning potential for the artificial brain despite the availability of additional training data. It should be noted, however, that the evolution of genes over time influences this behavior over generations, leading to improvements in fitness.

C. Comprehension in Real-Time

Even though the error rate in our trained model is still quite high and would require many more generations to become competitive, we have a unique advantage, and that is the comprehension speed. The time it takes for the artificial brain developed by FEAGI to detect an exposed image is currently around $160ms$ while running on a laptop, which can be considered as real-time. In contrast, Spaun [5], the leading brain simulator based on spiking neurons, would require around 2.5 hours of processing to simulate one second of handwriting recognition task. The reason for such a drastic difference goes back to our design philosophy: to be inspired by the brain but not to imitate it. In our design, we have tried to simplify the model and rely on evolution to help us achieve a functional architecture suitable for a digital environment. As an example, our decision to substitute thousands of synapses between two neurons with a single synapse averaging the post-synaptic influence has resulted in significant performance gain. Similarly, we have simplified physiological behaviors in a way to simulate the intended outcome without imitating all the steps followed in a biological counterpart. Spaun has adopted a lower-level architecture which on the positive side would be a very close representation of the human brain, but it comes at the cost of complexity requiring extensive processing power for simulation that prevents it from operating in real-time on commodity hardware.

D. Framework Versatility

As part of our proof of concept, we went through more than $18,000$ generations, with fitness being evaluated using the MNIST handwriting dataset. After we reached the fitness of 0.90, we started to train the new generations using the Fashion-MNIST instead. Initially, the fitness for apparel recognition was less than 0.1, but in less than 100 generations, it rose to 0.65 but stayed at that maximum level over future generations. We do not have any firm explanation for this behavior, but our theory is that the fitness rose very quickly because the seed genome used for apparel training was previously evolved through thousands of generations for handwriting recognition, hence the artificial brain's structure was optimized for general image recognition. Since Fashion-MNIST contains categories

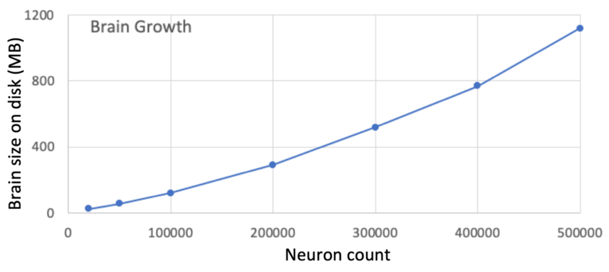


Fig. 13. Relationship between total neuron count and consumed disk space.

with close similarities, such as T-shirts and pullovers, we suspect that our visual pathway in the early regions have not evolved enough to be able to distinguish certain features.

E. Brain Footprint

We have measured the amount of storage consumed by the connectome database in order to forecast the system requirements as brain scales. As shown in Fig. 13, in our current implementation, a brain with 500,000 neurons consumes approximately 1.3 gigabytes of storage. We are currently storing brain anatomical information in JSON format without performing any data serialization or compression. An important observation in Fig. 13 is that the growth rate is parabolic, because in our experiment, we have kept the size of each cortical area dimensions constant while increasing the neuron count. Keeping the dimensions constant leads to a denser cortical region with more neurons per unit of space. In this situation, there is a higher probability for neurons to synapse, thus leading to excess synapse creation, which is the cause of non-linear capacity growth.

VII. Future Work

We see ourselves at the beginning of a long road of discovery and research ahead, but we are confident that the chosen modular architecture with evolutionary capabilities will provide a significant boost in accelerating our progress in realizing artificial general intelligence. Our current proof of concept demonstrates the framework's functionality but also highlights the lack of accuracy needed for effective character recognition, especially compared to recent deep-learning models. Many parameters work hand in hand to constitute a functioning brain, which in our implementation is primarily driven by the genome. Our genome must be tweaked to improve fitness, and we have leveraged genetic algorithms to help the brain evolve, but our current implementation requires further work to take advantage of genetic algorithms in a compelling way. One of the areas that require attention is the ability to evolve the brain's physiological properties. In our current implementation, we have parameterized just a few functions representing physiological activities, and future work is necessary to enable all brain aspects to evolve.

Another area for future research will be to add additional input processing units, such as tactile and auditory. Recent research in the area of robotics has shown great potential for applications of spiking neural networks [39], and we believe our framework can play a significant role in future advancements.

Alternatively, we are looking for ways to build an interface between data obtained by the Human Brain Project [3] and our framework so that we can automate the creation of a seed genome as the first generation of our framework instance. Last, leveraging neuromorphic hardware can have a significant impact on improving the artificial brain's performance. It is our goal to build extensions to our framework, thereby allowing integration with leading neuromorphic hardware such as IBM TrueNorth [12], NeuroGrid [10], and SpiNNaker [9].

VIII. Conclusion

Many existing brain simulation frameworks have cognitive abilities, accuracy, and visualization capabilities that are far superior to what we have offered here in our framework. The differentiating factors in our proposal are the evolvability and scalability enabled by our chosen architecture based on a novel encoding technique inspired by the neuroembryogenesis process in the human embryo. In our proof of concept, we demonstrated an artificial visual cortex generated from a genome simulating the ventral visual pathway of the human brain capable of learning, memorizing, and recalling digits in real-time read from the MNIST database.

References

[1] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," *AT&T Labs*, vol. 2, p. 18, 2010.

[2] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, 2017.

[3] H. Markram, "The human brain project," *Scientific American*, vol. 306, no. 6, pp. 50–55, 2012.

[4] C. Eliasmith and O. Trujillo, "The use and abuse of large-scale brain models," *Current Opinion in Neurobiology*, vol. 25, pp. 1–6, 2014.

[5] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012.

[6] C. Eliasmith and C. H. Anderson, *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT Press, 2004.

[7] T. Bekolay, J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. Voelker, and C. Eliasmith, "Nengo: A python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, vol. 7, p. 48, 2014.

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

[9] A. Mundy, J. Knight, T. C. Stewart, and S. Furber, "An efficient spinnaker implementation of the neural engineering framework," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, Budapest, Hungary, July 2015, pp. 1–8.

[10] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[11] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: Cortical simulations with 109 neurons, 1013 synapses," in *Proceedings of the IEEE Conference on High Performance Computing Networking, Storage and Analysis*, Portland, Oregon, November 2009, pp. 1–12.

[12] J. Sawada, F. Akopyan, A. S. Cassidy, B. Taba, M. V. Debole, P. Datta, R. Alvarez-Icaza, A. Amir, J. V. Arthur, A. Andreopoulos *et al.*, "Truenorth ecosystem for brain-inspired computing: scalable systems, software, and applications," in *Proceedings of the IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, Utah, USA, November 2016, pp. 130–141.

[13] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, "Compass: A scalable simulator for an architecture for cognitive computing," in *Proceedings of the IEEE International Conference on High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, Utah, USA, November 2012, p. 54.

[14] D. Varghese and V. Shankar, "Cognitive computing simulator-compass," in *Proceedings of the IEEE International Conference on Contemporary Computing and Informatics (IC3I)*, Mysore, India, November 2014, pp. 682–687.

[15] M.-O. Gewaltig and M. Diesmann, "Nest (neural simulation tool)," *Scholarpedia*, vol. 2, no. 4, p. 1430, 2007.

[16] D. Goodman and R. Brette, "Brian: A simulator for spiking neural networks in python," *Frontiers in Neuroinformatics*, vol. 2, 2008.

[17] P. Koehn, "Combining genetic algorithms and neural networks: The encoding problem," Master's thesis, The University of Tennessee, Knoxville, 1994.

[18] F. Gruau, "Cellular encoding of genetic neural networks," Technical report 92-21, Laboratoire de l'Informatique du Parallilisme. Ecole, 1992.

[19] K. O. Stanley and R. Miikkulainen, "A taxonomy for artificial embryogeny," *Artificial Life*, vol. 9, no. 2, pp. 93–130, 2003.

[20] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 131–162, 2007.

[21] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock, "Evolving coordinated quadruped gaits with the hyperneat generative encoding," in *Proceedings of the IEEE Congress on Evolutionary Computation*, Trondheim, Norway, May 2009, pp. 2764–2771.

[22] J. Huizinga, J. Clune, and J.-B. Mouret, "Evolving neural networks that are both modular and regular: Hyperneat plus the connection cost technique," in *Proceedings of the Annual Conference on Genetic and Evolutionary Computation*, Vancouver, CA, July 2014, pp. 697–704.

[23] J. Schrum, J. Lehman, and S. Risi, "Automatic evolution of multimodal behavior with multi-brain hyperneat," in *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, Denver, Colorado, USA, July 2016, pp. 21–22.

[24] C. Eliasmith, *How to build a brain: A neural architecture for biological cognition*. Oxford University Press, 2013.

[25] J. Schrum, J. Lehman, and S. Risi, "Using indirect encoding of multiple brains to produce multimodal behavior," *CoRR*, vol. abs/1604.07806, 2016.

[26] J. Rubenstein and P. Rakic, *Patterning and cell type specification in the developing CNS and PNS: Comprehensive developmental neuroscience*. Academic Press, 2013, vol. 1.

[27] M. Bear, B. Connors, and M. Paradiso, *Neuroscience: Exploring the brain*, 4th ed. Wolters Kluwer, 2016.

[28] C. Belzung and P. Wigmore, *Neurogenesis and neural plasticity*. Springer, 2013.

[29] A. D. Rast, F. Galluppi, X. Jin, and S. Furber, "The leaky integrate-and-fire neuron: A platform for synaptic model exploration on the spinnaker chip," in *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, Barcelona, Spain, July 2010, pp. 1–8.

[30] C. A. De la Hoz, E. G. Jones, and J. A. L. Sahd, *From development to degeneration and regeneration of the nervous system*. Oxford University Press, 2008.

[31] M. Mattia and P. D. Giudice, "Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses," *Neural Computation*, vol. 12, pp. 2305–2329, 2000.

[32] A. Delorme and S. J. Thorpe, "Spikenet: an event-driven simulation package for modelling large networks of spiking neurons," *Network: Computation in Neural Systems*, vol. 14, no. 4, pp. 613–627, 2003.

[33] O. Rochel and D. Martinez, "An event-driven framework for the simulation of networks of spiking neurons," in *Proceedings of 11th European Symposium On Artificial Neural Networks (ESANN)*, Bruges, Belgium, Apr. 2003.

[34] E. Ros, R. Carrillo, E. M. Ortigosa, B. Barbour, and R. Agís, "Event-driven simulation scheme for spiking neural networks using lookup tables to characterize neuronal dynamics," *Neural Computation*, vol. 18, no. 12, pp. 2959–2993, 2006.

[35] N. K. Kasabov, *Time-Space, Spiking Neural Networks and Brain-Inspired Artificial Intelligence*. Springer, 2019.

[36] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[37] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[38] J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Providence, Rhode Island, USA, June 2012, pp. 3642–3649.

[39] D. H. García, S. Adams, A. Rast, T. Wennekers, S. Furber, and A. Cangelosi, "Visual attention and object naming in humanoid robots using a bio-inspired spiking neural network," *Robotics and Autonomous Systems*, vol. 104, pp. 56 – 71, 2018.

APPENDIX

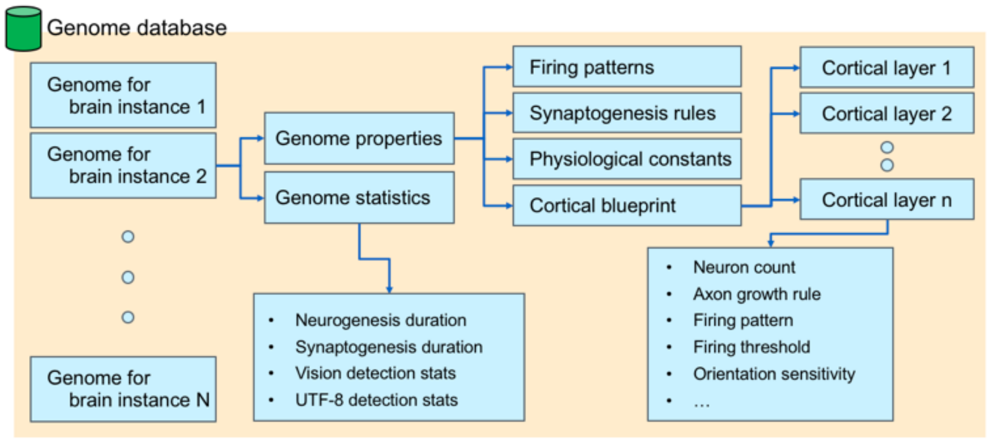


Fig. 14. Structure of genome database.

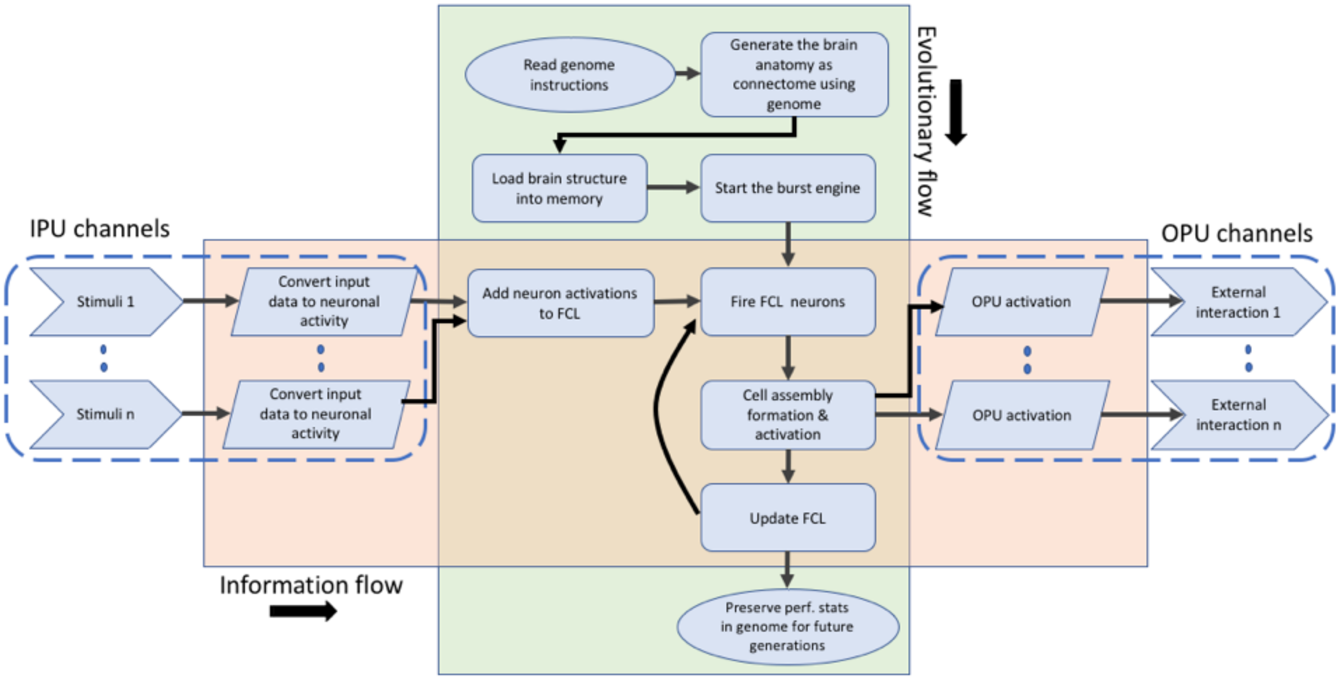
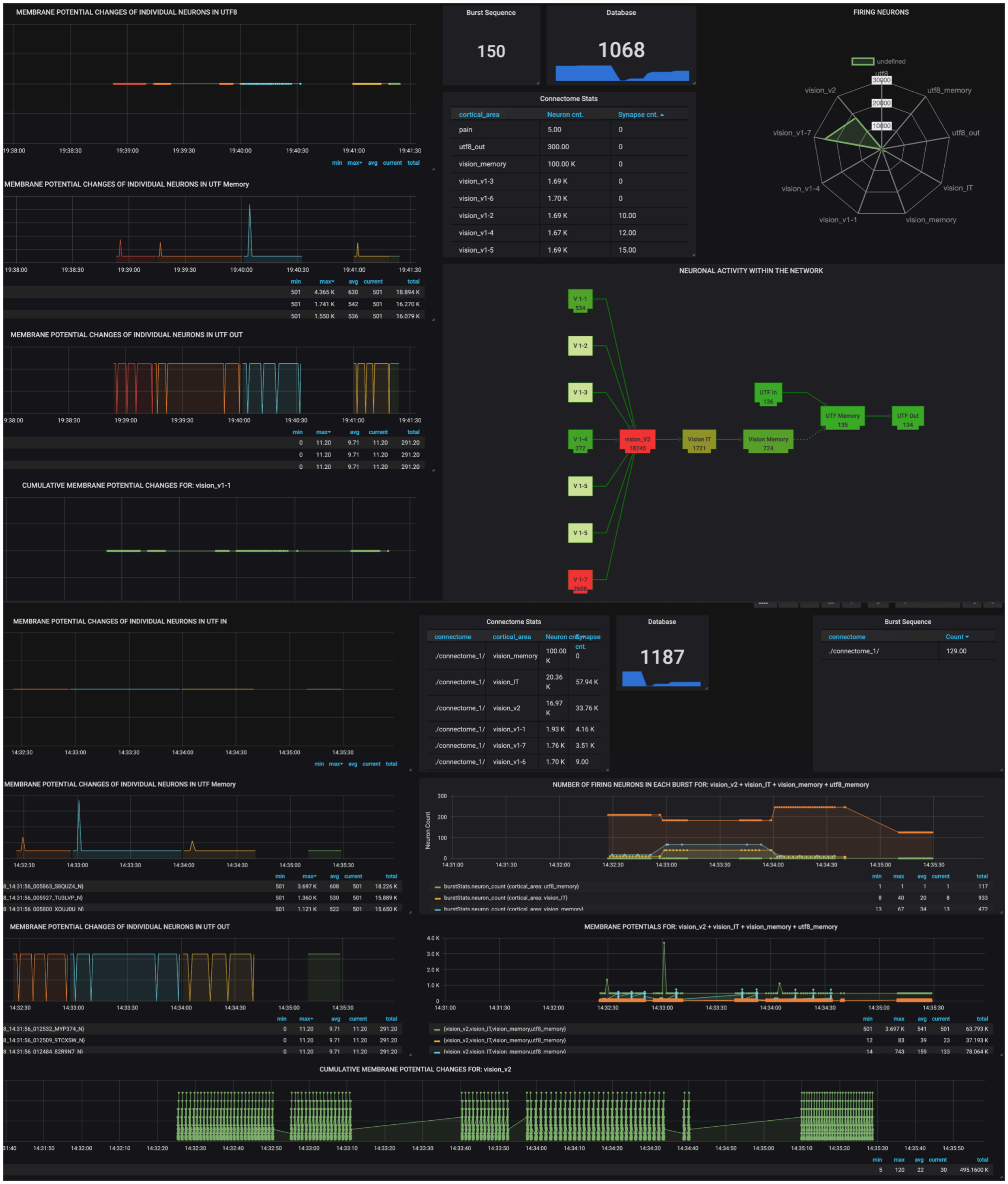

Read genome instructions
Generate the brain anatomy as connectome using genome
Load brain structure into memory
Start the burst engine
Evolutionary flow
IPU channels
OPU channels
Stimuli 1
Stimuli n
Convert input data to neuronal activity
Convert input data to neuronal activity
Add neuron activations to FCL
Fire FCL neurons
Cell assembly formation & activation
Update FCL
OPU activation
OPU activation
External interaction 1
External interaction n
Information flow
Preserve perf. stats in genome for future generations

Fig. 15. FEAGI system-level workflow.


MEMBRANE POTENTIAL CHANGES OF INDIVIDUAL NEURONS IN UTF8
Burst Sequence
150
Database
1068
FIRING NEURONS
undefined
utf8
utf8_memory
utf8_out
vision_IT
vision_memory
vision_v1-1
vision_v1-4
vision_v1-7
vision_v2
30000
20000
10000
MEMBRANE POTENTIAL CHANGES OF INDIVIDUAL NEURONS IN UTF Memory
MEMBRANE POTENTIAL CHANGES OF INDIVIDUAL NEURONS IN UTF OUT
CUMULATIVE MEMBRANE POTENTIAL CHANGES FOR: vision_v1-1
NEURONAL ACTIVITY WITHIN THE NETWORK
V 1-1 534
V 1-2
V 1-3
V 1-4 272
V 1-5
V 1-5
V 1-7 2608
vision_V2 18245
Vision IT 1721
Vision Memory 724
UTF In 136
UTF Memory 135
UTF Out 134
MEMBRANE POTENTIAL CHANGES OF INDIVIDUAL NEURONS IN UTF IN
Database
1187
NUMBER OF FIRING NEURONS IN EACH BURST FOR: vision_v2 + vision_IT + vision_memory + utf8_memory
Neuron Count
MEMBRANE POTENTIALS FOR: vision_v2 + vision_IT + vision_memory + utf8_memory
CUMULATIVE MEMBRANE POTENTIAL CHANGES FOR: vision_v2

Connectome Stats

cortical_area	Neuron cnt.	Synapse cnt. ▲
pain	5.00	0
utf8_out	300.00	0
vision_memory	100.00 K	0
vision_v1-3	1.69 K	0
vision_v1-6	1.70 K	0
vision_v1-2	1.69 K	10.00
vision_v1-4	1.67 K	12.00
vision_v1-5	1.69 K	15.00

min	max▾	avg	current	total
501	4.365 K	630	501	18.894 K
501	1.741 K	542	501	16.270 K
501	1.550 K	536	501	16.079 K

min	max▾	avg	current	total
0	11.20	9.71	11.20	291.20
0	11.20	9.71	11.20	291.20
0	11.20	9.71	11.20	291.20

Connectome Stats

connectome	cortical_area	Neuron cnt.	Synapse cnt.
./connectome_1/	vision_memory	100.00 K	0
./connectome_1/	vision_IT	20.36 K	57.94 K
./connectome_1/	vision_v2	16.97 K	33.76 K
./connectome_1/	vision_v1-1	1.93 K	4.16 K
./connectome_1/	vision_v1-7	1.76 K	3.51 K
./connectome_1/	vision_v1-6	1.70 K	9.00

Burst Sequence

connectome	Count ▾
./connectome_1/	129.00

	min	max▾	avg	current	total
8_14:31:56_005863_S8QUZ4_N}	501	3.697 K	608	501	18.226 K
8_14:31:56_005927_TU3LVP_N}	501	1.360 K	530	501	15.889 K
8_14:31:56_005800_XOUJ0U_N}	501	1.121 K	522	501	15.650 K

	min	max	avg	current	total
burstStats.neuron_count {cortical_area: utf8_memory}	1	1	1	1	117
burstStats.neuron_count {cortical_area: vision_IT}	8	40	20	8	933
burstStats.neuron_count {cortical_area: vision_memory}	13	67	34	13	472

	min	max▾	avg	current	total
8_14:31:56_012532_MYP374_N}	0	11.20	9.71	11.20	291.20
8_14:31:56_012509_9TCXSW_N}	0	11.20	9.71	11.20	291.20
8_14:31:56_012484_82R9N7_N}	0	11.20	9.71	11.20	291.20

	min	max	avg	current	total
{vision_v2,vision_IT,vision_memory,utf8_memory}	501	3.697 K	541	501	63.793 K
{vision_v2,vision_IT,vision_memory,utf8_memory}	12	83	39	23	37.193 K
{vision_v2,vision_IT,vision_memory,utf8_memory}	14	743	159	133	78.064 K

min	max	avg	current	total
5	120	22	30	495.1600 K

Fig. 16. Monitoring dashboard created to help monitor cortical activities across various regions.

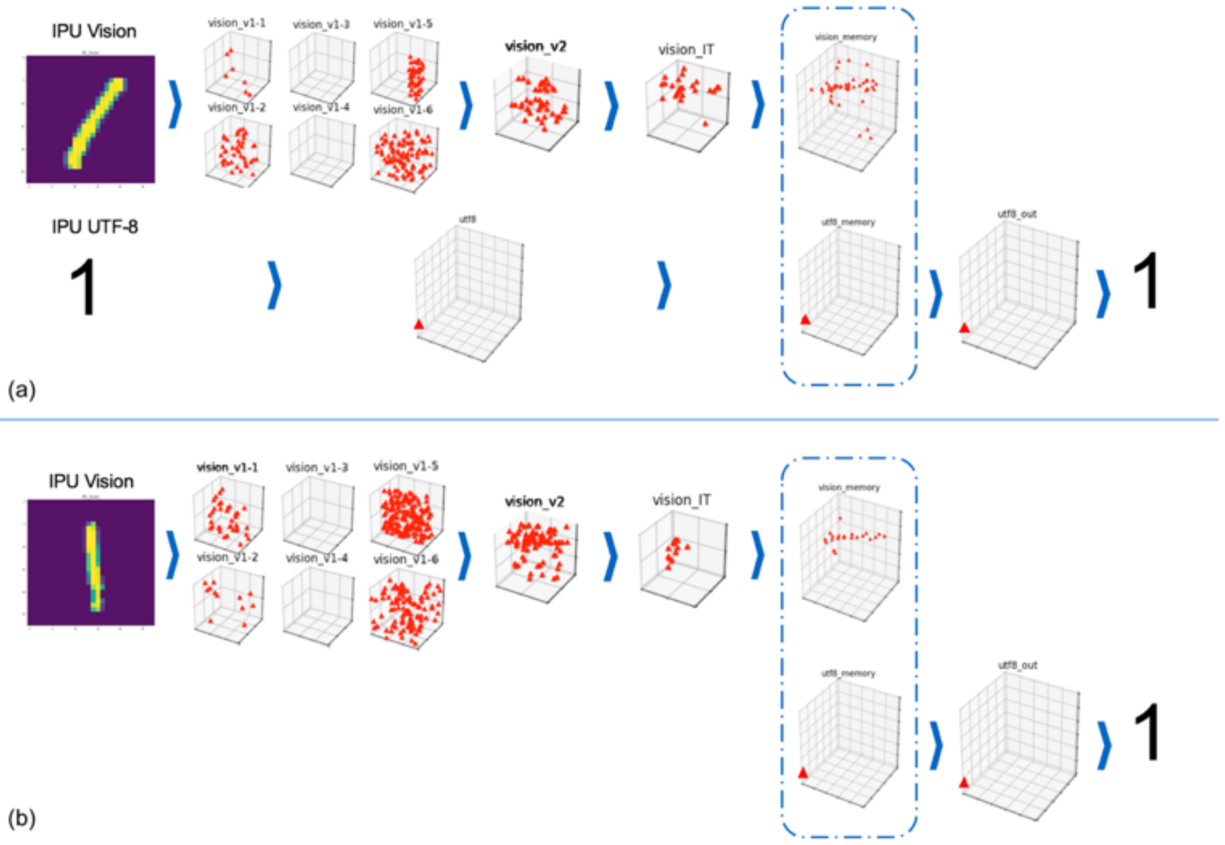


Fig. 17. Framework behavior during learning and recall: (a) exposure of handwritten version of number 1 along with UTF-8 version of number 1 to framework at same approximate time, leading to association of cell assemblies between visual memory and UTF-8 memory; (b) exposure of framework with another handwritten version of number 1 leads to activation of same cell assembly in UTF-8 memory region and recognition of character 1 in association with handwritten version.

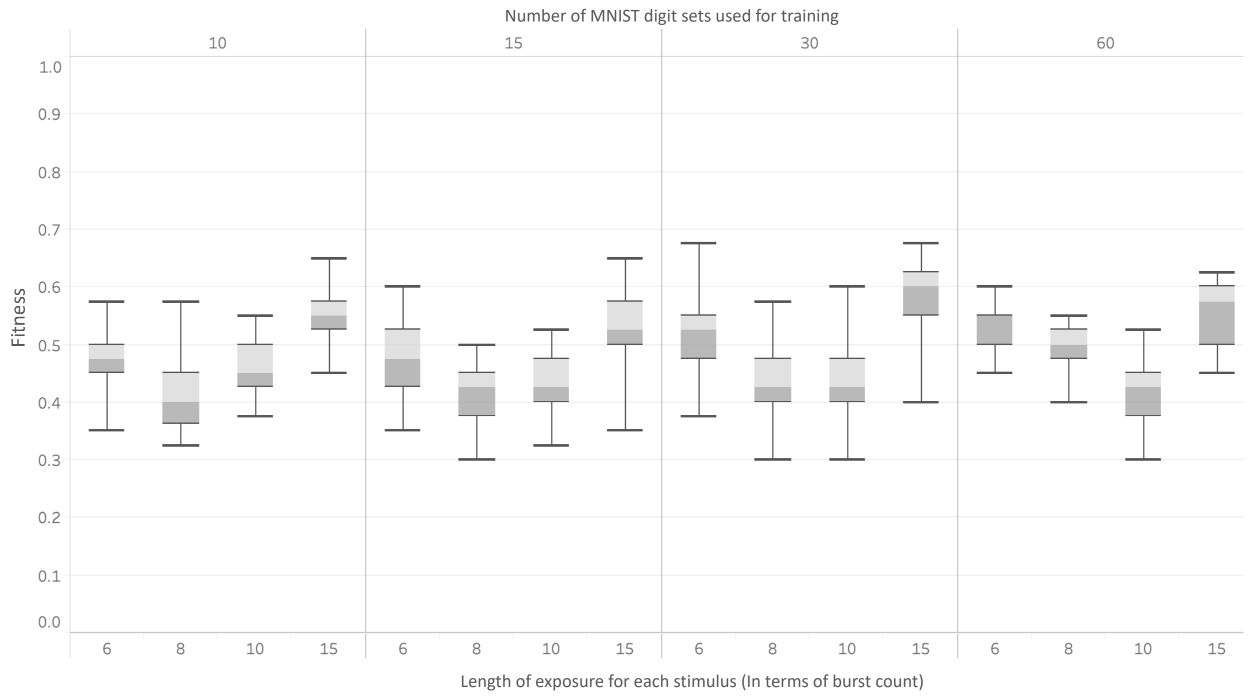


Fig. 18. Impact of exposure time on fitness and its correlations with the size of training set. For this analysis, a random genome was used frequently with a different number of training sets and exposure times. Each training set is considered 10 MNIST digits, and exposure time is the duration by which every training or test sample is exposed to the input processing unit.