A Structural Human Learning Model (HLM) for Specialized Exponential Learning Growth in Artificial Intelligence Systems — Revision 2

Professor Michael Recchia
Chief Scientist & Chief Technology Officer
AdaptNet Technologies / AdaptNet AI Laboratories
Distributed Intelligence Research Group
Lakewood New Jersey, USA
Casal delle Mole, Casalvieri, Frosinone Italia

REVISION 2
Date: August 28, 2026 | Revision 2

Scientific Status: This document presents a proposed research architecture and testable hypotheses. Statements concerning AdaptOne performance, energy efficiency, BioHybrid scaling, consciousness-related monitoring, and predominantly biological implementations are prospective unless explicitly identified as measured or experimentally validated.

Abstract 2
Introduction 2
The Basis Case of the Recchia Human Learning Model 3
Major Node – High Level Neuron Representation 3
Minor Node – Axon Representation 3
Synaptic End Bulb Signaling 3
Linked List Representation of the HLM Neuron 4
Construction of Neural Networks 4
Identification of "Learning" 4
Programming Language Implementation 4
Limitations of Conventional Matrix-Based Neural Networks 5
The Learning Event Concept 5
Multi-Sensory Correlation Mechanism 6
AdaptOne Cognitive Architecture Integration 6
Multi-Sphere Cognitive Expansion 6
Figure Placeholder – Fundamental Biological Thought Mechanism 7
Mathematical Representation of HLM Learning Events 7
HLM vs Conventional Deep Learning 7
Fundamental HLM Learning Mechanism Diagram 8
AdaptOne Multi-Sphere Cognitive Architecture 9
Biological Neuron vs. HLM Linked-List Neuron Representation 10
Proof of Concept (PoC): Autonomous Neuron Activation and Network Formation 11
Example Pseudocode for PoC Implementation 13
Implications for Artificial General Intelligence 13
Structural Learning vs Parameter Learning 13
Modeling Thinking Biology 13
References 14

The Recchia Human Learning Model (HLM) and AdaptOne AI System

Abstract

This document presents the theory of operation of a machine-based Human Learning Model (HLM) and its implementation within the AdaptOne Artificial Intelligence system. The HLM proposes a fundamentally different basis for machine learning architecture by modeling the biological structure of neurons using linked data structures rather than matrix-based neural networks. This paper outlines the conceptual framework, architectural basis, and computational representation of neurons and synaptic signaling within the HLM paradigm.

By organizing data into structures that more closely resemble biological neural components, it becomes possible to define native computational operations that approximate key aspects of biological signaling and association. In this sense, the HLM is intended not merely as an alternative implementation strategy, but as a structural reinterpretation of how learning systems may be represented in software.

As chip design advances, it may become increasingly important to pair learning architectures with computational substrates capable of structural elasticity. If exponential learning growth occurs, the capacity of Neural Trees (NTs) and Neural Networks (NNs) must expand and contract with that growth. Static chipsets may eventually constrain such architectures, whereas dynamically extensible substrates—conceptually similar to bio-modeled cells with elastic compute and memory allocation—could better support the long-term realization of HLM-based systems.

Introduction

Modern artificial intelligence systems typically model neurons and neural networks through matrix-based architectures. These systems rely heavily on linear algebra, matrix multiplication, and weighted numerical transformations to emulate synaptic behavior. While successful in many domains, such approaches remain abstractions that diverge significantly from the structural organization of biological neurons and may therefore limit the path toward higher forms of intelligence.

The Recchia Human Learning Model (HLM) introduces a new architectural basis that directly models selected structural components of biological neurons using linked data structures available in languages such as PL/1, Pascal, Python, and C++. Instead of representing neural connectivity through matrix links, the HLM represents neurons as hierarchical linked data structures composed of major and minor nodes. In its fullest realization, the model also anticipates bio-based or bio-modeled chipsets capable of

dynamic growth and shrinkage, thereby supporting Neural Trees (NTs) and Neural Networks (NNs) that are dynamically extensible within available compute, memory, biological, and interconnect resources.

The Basis Case of the Recchia Human Learning Model

In the Recchia HLM model, the fundamental computational element is not a matrix or tensor but a linked-list structure that models the biological neuron. This linked structure captures both the hierarchical organization and the functional relationships among the components of a neuron.

Major Node – High Level Neuron Representation

The Major Node represents the neuron at the highest level of abstraction. It is a multi-element data structure containing fields that represent the primary anatomical and functional components of a biological neuron. Functional descriptions of native and advanced neuron capabilities are made available within those fields.

- Cell Body (Soma)

- Cell Membrane

- Dendrites

- Axon

- Axon Hillock

Minor Node – Axon Representation

The Axon is represented as a Minor Node embedded within the Major Node. This subordinate structure contains the computational analogs of key axonal elements.

- Oligodendrocyte

- Node of Ranvier

- Axon Terminal

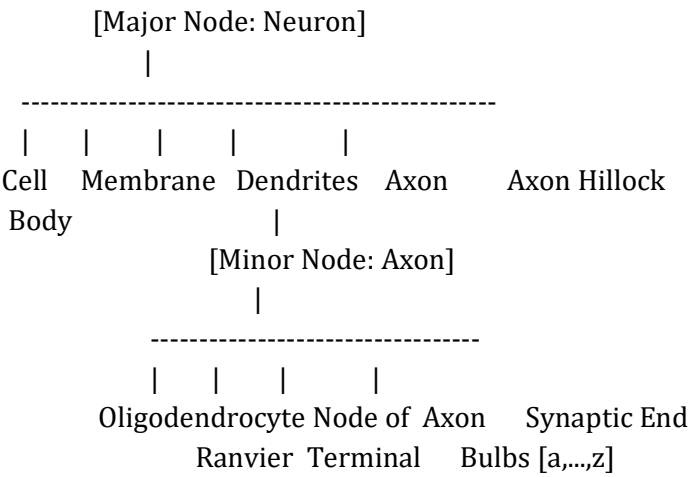
- Synaptic End Bulbs [a,...,z], where z >= 1st dimension

Synaptic End Bulb Signaling

The Synaptic End Bulbs are conceptualized as dynamically expandable signaling structures capable of transmitting signals to the synaptic bulbs of other neurons. In computational terms, they may be represented as expandable, potentially multidimensional signaling objects responsible for propagating neuron activation events.

Linked List Representation of the HLM Neuron

Conceptual Linked List Diagram:



Construction of Neural Networks

Networks of neurons are constructed by linking synaptic end bulbs to the synaptic bulbs of other neuron structures. These connections form dynamic linked networks rather than static matrices, allowing the architecture to evolve and grow in complexity over time. The structures formed as neurons connect, detach, and evolve constitute the basis of learning. As neural connections increase, Neural Trees (NTs) and Neural Networks (NNs) emerge as aggregate structures composed of one or more connected neurons.

Identification of "Learning"

As neurons connect to one another, the aggregate connected structure becomes either an NT or an NN. A given NT may be understood as providing a Learned Element Unit (LEU) within a single, relatively narrow domain space, whereas an NN may consist of multiple classes of NTs and therefore support more complex knowledge representation across domains.

Programming Language Implementation

The HLM neuron model can be implemented using languages with strong data structure construction and manipulation capabilities, including, but not limited to, the following:

- PL/1

- Pascal

- C++

- Python

Although PL/1 and Pascal are often regarded as older languages, their data-structure capabilities remain relevant when evaluated from the perspective of systems programming and deterministic software design. Languages that provide efficient structured data operations and predictable memory behavior may offer advantages for implementing dynamically growing neuron structures.

Limitations of Conventional Matrix-Based Neural Networks

Most contemporary artificial intelligence systems model cognition using matrix-based neural networks. These systems represent neurons mathematically using weighted matrices and nonlinear activation functions.

While these systems have proven effective in pattern recognition tasks, they abstract away many of the structural and developmental properties of biological neurons. Biological neural systems exhibit:

- Temporal integration of signals
- Structural growth and pruni35ng
- Multi-modal sensory correlation
- Synaptic plasticity driven by experience

Matrix-based systems generally rely on gradient-descent training over large datasets rather than continuous experiential learning. As a result, they lack the dynamic structural growth observed in biological cognition.

The Recchia Human Learning Model (HLM) attempts to address this limitation by modeling the neuron structurally rather than purely mathematically, with the goal of supporting continuous experiential growth rather than only parameter adjustment.

The Learning Event Concept

In the HLM framework, learning occurs through discrete experiential events referred to as Learning Event Units (LEUs). Each LEU represents an atomic unit of knowledge acquisition.

A Learning Event Unit contains the following conceptual elements:

- Sensory Input
- Temporal Context
- Associative Linkages

• Reinforcement State
• Structural Growth Trigger

Rather than adjusting weights in a matrix, the HLM architecture grows new structural relationships between linked neuron objects when learning events occur.

Multi-Sensory Correlation Mechanism

Biological intelligence relies heavily on the correlation of simultaneous sensory inputs. For example, an organism may observe a visual stimulus while simultaneously hearing a sound and receiving linguistic input.

In such cases the brain forms associative links between the correlated sensory signals.

In the HLM architecture this process is implemented by linking Learning Event Units across multiple sensory subsystems such as:

• Vision Center
• Auditory Processing
• Language Representation

This cross-modal linking allows the system to build increasingly complex conceptual representations by associating information across sensory and symbolic domains.

AdaptOne Cognitive Architecture Integration

AdaptOne Appliance-Oriented Cognitive Architecture

Security and Cognitive Governance

Architectural Safety by Design

The AdaptOne Artificial Intelligence system serves as the implementation platform for the HLM model.

Within AdaptOne, the Human Learning Model operates as the core learning substrate while specialized cognitive modules provide sensory processing and higher-level reasoning capabilities.

Primary modules include:

• Vision Center – Optical perception and pattern extraction
• Auditory Processing – Sound recognition and correlation

• Language Model – Symbolic representation and linguistic association
• Frontal Lobe – Decision-making and reasoning
• Brain Stem – System regulation and resource management

Each subsystem contributes data to the HLM learning engine, which constructs and expands the internal knowledge network.

Multi-Sphere Cognitive Expansion

Unlike biological brains which are constrained to two hemispheres, artificial systems are capable of expanding into multiple cognitive spheres.

Each sphere may contain:

• Its own Human Learning Model instance
• Specialized sensory or reasoning functions
• Inter-sphere communication channels

Future versions of AdaptOne may incorporate eight or more spheres operating in parallel, creating a distributed cognitive architecture capable of highly specialized and potentially super-intelligent behavior.

AdaptOne Appliance-Oriented Cognitive Architecture

The AdaptOne architecture separates cognition from specialized computation. Within this model, the Human Learning Model (HLM) operates as the cognitive core responsible for reasoning, learning, memory, planning, knowledge validation, and task orchestration. Specialized AI appliances perform domain-specific computation under HLM orchestration rather than replacing the HLM as the source of intelligence.

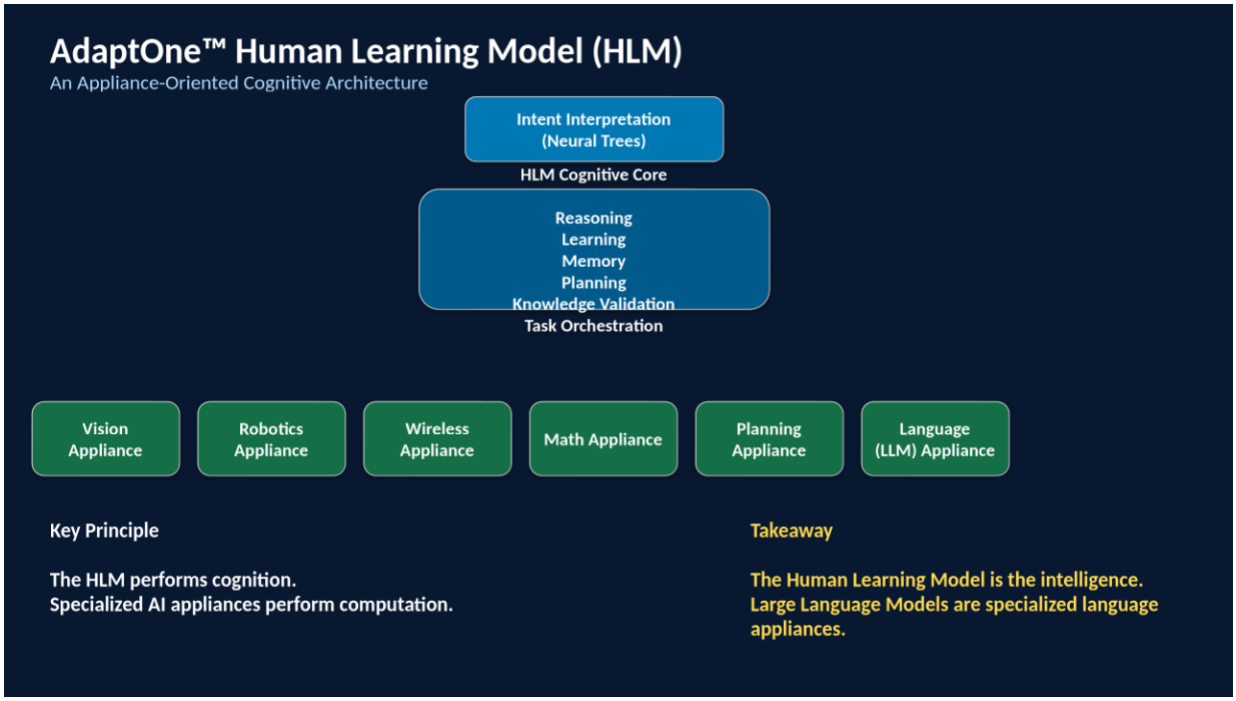


Figure 2 — AdaptOne appliance-oriented cognitive architecture. The HLM Cognitive Core performs cognition, while specialized appliances provide domain-specific computational capabilities, including vision, robotics, wireless, mathematics, planning, and language generation.

The architecture shown in Figure 2 formalizes an important separation of responsibility. Intent interpretation is performed through Neural Trees that classify the external input and route it toward the relevant cognitive or computational domains. The HLM Cognitive Core then determines the reasoning strategy, validates knowledge, coordinates memory, and invokes only those appliances required for the task.

This design is intentionally not LLM-centric. A Large Language Model may be used within AdaptOne, but only as a Language Appliance. The LLM performs natural language understanding, summarization, translation, explanation, or generation when invoked by the HLM. It does not own the system memory, does not define the reasoning policy, and does not become the cognitive authority of the system.

The key architectural principle is therefore: the Human Learning Model is the intelligence; specialized AI appliances perform computation. Under this model, appliances may evolve independently. A Language Appliance could be implemented using one LLM today and replaced by a different model in the future without changing the HLM cognitive architecture. Similarly, vision, robotics, wireless, mathematics, planning, and other specialized appliances can be upgraded without redefining the core learning model.

This appliance-oriented design also aligns with deterministic computing principles. Rather than placing all functionality inside a single monolithic model, AdaptOne can isolate

capabilities into bounded modules with explicit interfaces, controlled resources, auditable execution paths, and well-defined authorization boundaries.

Figure Placeholder – Fundamental Biological Thought Mechanism

[Insert Publication-Quality Diagram Here]

Diagram Title:
Fundamental Biological Thought Mechanism and HLM Learning Event Unit

Conceptual Flow:

Sensory Input
↓
Temporal Correlation Layer
↓
Learning Event Unit (LEU)
↓
Associative Memory Graph
↓
Structural Growth Trigger
↓
Cognitive Architecture Expansion

Mathematical Representation of HLM Learning Events

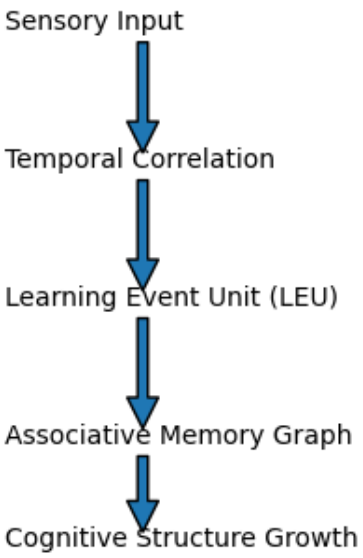
In the Human Learning Model, learning occurs through discrete experiential events rather than gradient descent. A simplified representation of a Learning Event Unit (LEU) can be expressed as:

HLM vs Conventional Deep Learning

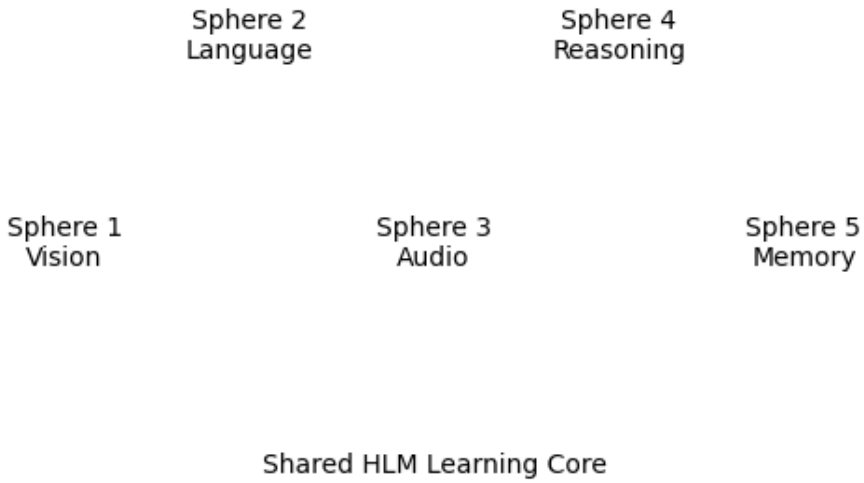
Property	HLM	Deep Learning	Transformers
Core structure	Linked neuron data structures	Weight matrices	Attention matrices
Learning mechanism	Structural growth from experience	Gradient descent training	Attention-weight optimization
Temporal modeling	Native time correlation	Limited unless modeled	Token sequence modeling
Architecture growth	Dynamic expansion	Static after training	Static model size

Biological correspondence	Higher structural biological inspiration; functional equivalence unvalidated	Abstract computational model	Abstract computational model

Fundamental HLM Learning Mechanism Diagram



AdaptOne Multi-Sphere Cognitive Architecture



Biological Neuron vs. HLM Linked-List Neuron Representation

Figure 1 illustrates a conceptual comparison between a biological neuron and the equivalent structural representation used in the Recchia Human Learning Model (HLM).

On the left side of the illustration is a biological neuron. Biological neurons consist of dendrites that receive signals, a cell body (soma) where signals are integrated, an axon hillock where action potentials originate, and a long axon that propagates signals toward synaptic terminals. The axon may be insulated by myelin sheaths, interrupted by Nodes of Ranvier which facilitate rapid signal conduction. At the end of the axon, synaptic end bulbs transmit electrochemical signals to other neurons.

On the right side of the illustration is the HLM computational analog of the neuron. Rather than representing neural connectivity using weight matrices, the HLM represents neurons using hierarchical linked data structures. The primary structure is the Major Node, representing the neuron as a whole. This major node contains fields corresponding to the principal biological components including the cell body, membrane, dendrites, and axon hillock.

The axon itself is represented as a subordinate data structure referred to as a Minor Node. Within this minor node are fields that correspond to structures found along the biological axon, including the oligodendrocyte representation, nodes of Ranvier, axon terminals, and a dynamically expandable array of synaptic end bulbs.

This representation preserves the hierarchical nature of biological neurons while allowing computational implementations to construct dynamic networks through linked structures. Unlike matrix-based neural networks, this approach allows the architecture to grow structurally as new connections are formed through learning events.

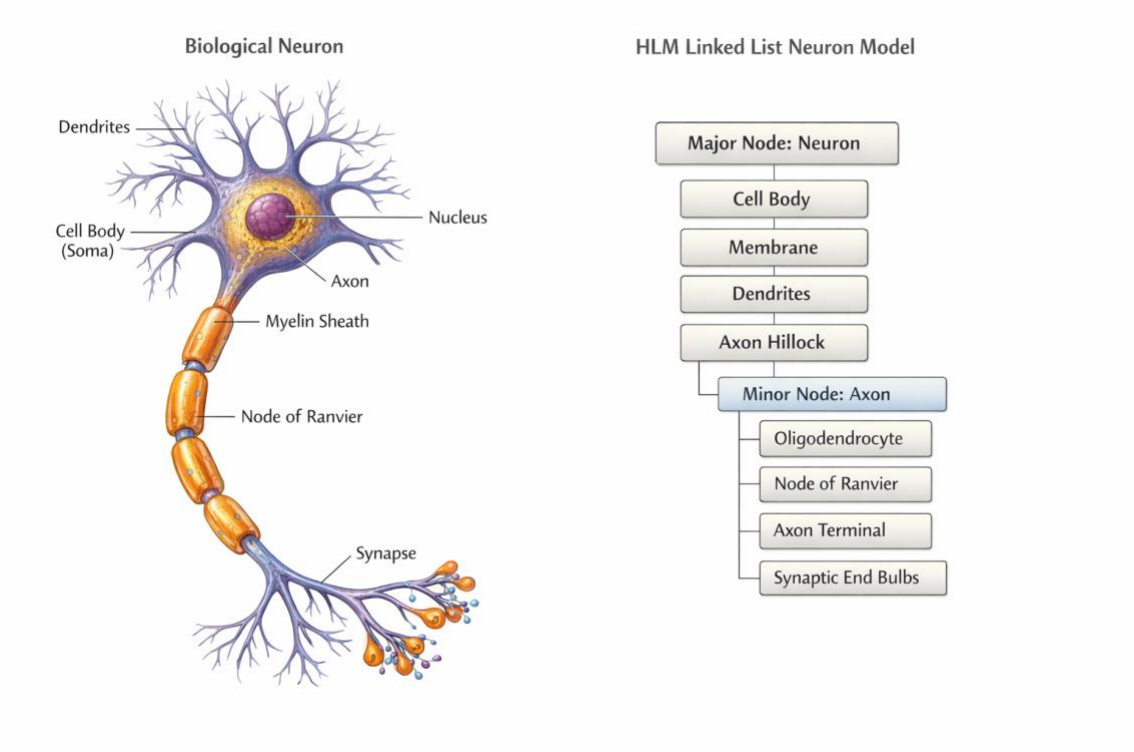


Figure 1 — Side-by-side comparison of a biological neuron and the HLM linked-list neuron facsimile.

Figure 1 compares an actual biological neuron with the equivalent HLM linked-list facsimile. The left panel shows the biological signal path—dendrites, soma, axon hillock, myelinated axon, Nodes of Ranvier, and synaptic end bulbs. The right panel shows the HLM computational mapping of that same organization into a structured Major Node and a subordinate Minor Node. The comparison makes the paper's central claim visually explicit: in the HLM, the neuron is represented as a structured biological object rather than as an abstract weight in a matrix, thereby making dynamic pathway growth and network formation easier to model.

Proof of Concept (PoC): Autonomous Neuron Activation and Network Formation

The Human Learning Model (HLM) neuron representation can be demonstrated through a simple proof-of-concept experiment illustrating how a linked-list neuron structure can be autonomously activated by an external signal and subsequently begin constructing neural pathways.

Step 1 — External Signal Activation

An external event, such as a sensory stimulus or data input, sends a signal to the neuron's dendritic interface. In software terms, this may be implemented as an event trigger or interrupt that activates the neuron object.

When the signal strength exceeds the threshold defined at the Axon Hillock field of the Major Node, the neuron enters an active state and begins listening for correlated input signals.

Step 2 — Listening State

Once activated, the neuron transitions into a listening mode where it continuously observes incoming stimuli across its dendritic inputs. Each observed signal is time-stamped and stored within temporary event buffers associated with the neuron object.

During this stage the neuron attempts to detect temporal correlations between incoming signals.

Step 3 — Learning Event Formation

When correlated signals are detected within a defined temporal window, the neuron constructs a Learning Event Unit (LEU). This event represents the formation of a new associative link between stimuli.

In computational terms, this process creates new linked list connections between neuron nodes, expanding the neural network graph.

Step 4 — Pathway Construction

Each successful learning event generates a new synaptic pathway by linking the synaptic end bulb structures of the activated neuron to other neuron nodes within the system.

Because the architecture uses linked structures rather than fixed matrices, the network can expand dynamically as new pathways are formed.

Step 5 — Emergent Network Growth

As additional neurons become activated and participate in learning events, the system evolves from isolated neuron objects into a growing neural network. Over time the structure forms a hierarchical associative memory graph representing accumulated knowledge.

This proof-of-concept demonstrates how a biologically inspired neuron representation implemented through linked data structures could autonomously construct neural networks through experience, modeling selected aspects of structural plasticity and associative network formation. It does not establish functional equivalence with biological cognition.

Example Pseudocode for PoC Implementation

```
Neuron MajorNode = CreateNeuron()

ExternalSignal -> MajorNode.Dendrites

if SignalStrength > MajorNode.AxonHillock.Threshold:
   MajorNode.State = LISTENING

while MajorNode.State == LISTENING:
   CaptureIncomingSignals()

   if TemporalCorrelationDetected():
      LEU = CreateLearningEvent()
      LinkNeurons(LEU)
```

Implications for Artificial General Intelligence

If intelligence depends not only on parameter optimization but also on the structural growth of neural organization, then future artificial systems may require architectures capable of dynamic topological expansion. The HLM provides one conceptual path toward such systems by representing neurons as expandable linked structures capable of forming and reorganizing hierarchical networks through experience.

Structural Learning vs Parameter Learning

Most contemporary AI systems rely on parameter learning, where numerical weights are optimized within a largely fixed network architecture. The HLM model instead emphasizes structural learning, in which the architecture itself evolves as new learning events occur. This distinction is central to the HLM thesis: learning is expressed not only through state changes, but through the creation of new structural pathways.

Modeling Thinking Biology

Biological cognition emerges from networks of neurons that evolve structurally through experience. Unlike traditional artificial neural networks that rely primarily on fixed topologies and weight optimization, biological systems reorganize connectivity through plasticity, signaling, and growth. The Human Learning Model attempts to capture these properties by representing neurons as structured computational objects whose relationships grow dynamically through learning events.

Kandel, E. R., Schwartz, J. H., & Jessell, T. M. (2000). Principles of Neural Science. McGraw-Hill.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group. (1986). Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press.

Friston, K. (2010). The free-energy principle: a unified brain theory? Nature Reviews Neuroscience, 11(2), 127–138.

Security and Cognitive Governance

A Human Learning Model capable of continuous structural growth must include security and governance mechanisms by design. The central security objective is to prevent autonomous learning, planning, or reasoning from becoming uncontrolled authority over external systems. In AdaptOne, cognition and execution should be deliberately separated.

The HLM may reason, learn, form associations, and propose actions, but every external action should pass through a deterministic Policy and Safety Engine. This engine evaluates whether an action is permitted, whether the requesting sphere or appliance has the necessary capability, whether human authorization is required, and whether the action is consistent with system policy.

• Separation of cognition from execution: the HLM may reason and plan, but it should not directly execute privileged external actions.

• Policy and Safety Engine: all outbound actions are mediated by deterministic safety policy before execution.

• Capability-based access control: each appliance and learning sphere receives only the permissions required for its function.

• Human authorization for high-impact actions: privileged, irreversible, financial, legal, physical, medical, or infrastructure-affecting actions require explicit approval.

• Zero-trust appliance interaction: appliances, learning spheres, external tools, and data sources authenticate before exchanging information.

• Immutable audit trail: learning events, reasoning traces, appliance invocations, policy decisions, and external actions are logged for forensic review.

• Cryptographically signed Learning Events: accepted Learning Event Units can be signed or hash-linked to preserve origin, integrity, and provenance.

• Authenticated Learning Spheres: each sphere has a verifiable identity and bounded authority within the broader AdaptOne system.

• Sandboxed experimental learning: uncertain or untrusted learning can be tested in an isolated environment before promotion into trusted memory.

• Cross-sphere validation: important conclusions can be validated by independent spheres before becoming authoritative knowledge.

• Rollback and snapshot recovery: HLM structures should support restoration to known-good states if poisoning, corruption, or unsafe growth is detected.

• Anomalous growth detection: unexpected pathway expansion, self-modification, privilege-seeking behavior, or abnormal appliance invocation patterns should trigger alerts or containment.

• Prompt-injection and data-poisoning resistance: external language inputs and training events should be treated as untrusted until evaluated by policy and validation mechanisms.

• Explainability before execution: for high-impact actions, the HLM should provide a human-readable explanation of intent, evidence, assumptions, and expected consequences before action is authorized.

Architectural Safety by Design

The security goal is not merely to add guardrails after intelligence has been constructed. The goal is to make safety an architectural property of AdaptOne. The HLM is the cognitive substrate, but authority is mediated through policy, identity, capability, and auditability.

This distinction is important for preventing an HLM-based system from "going rogue." A system becomes dangerous when reasoning, autonomy, tool access, memory modification, and external authority collapse into a single uncontrolled loop. AdaptOne should avoid that failure mode by requiring that every significant action pass through explicit governance boundaries.

In this model, learning does not automatically imply permission to act. A newly formed Learning Event Unit may expand internal knowledge, but it does not automatically grant new external privileges. Likewise, a Language Appliance may generate text, but it cannot independently authorize tool use, modify trusted memory, control robotics, access protected systems, or issue infrastructure commands unless the HLM and Policy and Safety Engine explicitly permit the action.

The result is a governed cognitive architecture: the HLM provides intelligence, specialized appliances provide capabilities, and the security layer determines when and how those capabilities may be safely used. This preserves the benefits of continuous structural learning while reducing the risk that autonomous learning growth produces uncontrolled behavior.

References

Hebb, D. (1949). The Organization of Behavior.

McCulloch, W., Pitts, W. (1943). A Logical Calculus of Ideas Immanent in Nervous Activity.

Hopfield, J. (1982). Neural networks and physical systems with emergent collective computational abilities.

LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep Learning. Nature.

Vaswani, A. et al. (2017). Attention is All You Need.

Technical Addenda

The following addenda extend the AdaptOne / Human Learning Model architecture with BioHybrid asynchronous-computing, operational-continuity, implementation, and power-consumption analyses.

TECHNICAL ADDENDUM: ASYNCHRONOUS API WORKFLOW

Interfacing Digital Compute with Biological Wetware Arrays

1. Architectural Overview & The Async Imperative

Interfacing standard digital software with biological networks strongly favors a non-blocking asynchronous computing layer. Silicon transistors switch electronically on nanosecond-to-picosecond timescales depending on technology, whereas human cortical neurons operate on millisecond-scale biological dynamics. Biological processing introduces variable latency due to membrane dynamics, axonal conduction, synaptic transmission, network state, and plasticity. An asynchronous event-driven architecture therefore provides a scalable way to isolate variable biological response times from deterministic digital services.

Synchronous, blocking HTTP execution can be inefficient for BioHybrid workloads because variable biological latency may hold connections and execution resources unnecessarily. Therefore, the proposed Biological Intelligence Operating System (biOS) handles computational payloads through an asynchronous Event-Driven Messaging Architecture. A digital host schedules a task payload, receives an immediate acknowledgment token, and later polls or receives a callback once the biological network has produced a measurable response.

> *System Design Recommendation: For scalable biOS deployments, use non-blocking queues or event brokers (for example, RabbitMQ or Apache Kafka) to isolate the biological execution boundary. Synchronous interfaces may still be used at external boundaries when appropriate, provided that they do not require a thread or connection to block for the full biological computation interval.*

2. End-to-End Async Computational Lifecycle

A single transactional loop—from a software-originated execution request to biological inference and hardware feedback—executes via a strict 5-stage decoupled pipeline:

Stage 1: Job Ingestion & Queuing: The external software application issues a POST request containing the target computational matrix (e.g., an anomaly detection frame) to the biOS

edge endpoint. The biOS validates the request schema, signs it with a Job ID UUID, writes the payload to a priority message queue, and returns an immediate HTTP 202 Accepted response containing a job-tracking token.

Stage 2: Digital-to-Analog Signal Mapping: The biOS Hardware Intermediary Layer pops the job from the queue. It maps raw digital metrics into a spatiotemporal electrical-stimulation pattern and translates that schema into specific electrode coordinates on the Microelectrode Array (MEA). A stimulation protocol should specify parameters such as pulse frequency, voltage or current amplitude, pulse width, waveform, electrode location, and inter-pulse interval. For example, an encoded input might use a 100-Hz pulse train with device-appropriate amplitude and pulse width directed to a designated input electrode cluster.

Stage 3: Wetware Compute Window: The MEA applies controlled electrical stimulation to the living neural culture. The biological network responds through membrane dynamics, axonal conduction, and synaptic activity. The system then enters an isolated 'Stimulate, Settle, and Read' interval whose duration is determined experimentally for the specific culture, task, MEA, and decoding method. Any numerical latency range used during prototyping should be labeled an illustrative engineering parameter rather than a universal biological constant.

Stage 4: Analog-to-Digital Extraction: The MEA's recording electrodes pick up the downstream bio-electric spikes fired by the neurons. The hardware filters out background noise, captures the micro-volt amplitudes, and generates a structured digital bitstream (the response spike-train matrix). This biological output represents the network's inference result.

Stage 5: Async Callback / Event Resolution: The biOS matches the recorded response matrix with the original Job ID token. It publishes an execution-complete event to the message broker. The broker invokes the user application's registered Webhook listener, pushing the structured JSON response payload to the developer's server. Alternatively, short-polling clients can fetch this data via a matching GET endpoint.

3. Endpoints & Payload Contracts

The following section outlines the structural JSON schema and telemetry objects exposed by the biOS API layer for orchestrating wetware computational cycles.

Endpoint 1: Submit Computational Task (Async)

- Protocol/Method: POST
- Route: /api/v1/wetware/compute
- Content-Type: application/json

```
{
  "rack_id": "sg-rack-01",
  "unit_id": "cl1-core-12",
  "callback_url": "https://api.clientdomain.com/v1/bioutcomes",
```

```
  "timeout_ms": 5000,
  "payload": {
    "input_vector": [0.821, 0.045, 0.119, 0.943],
    "mapping_mode": "spatiotemporal_frequency",
    "target_input_nodes": [102, 104, 106, 108]
  }
}
```

Endpoint 2: Ingestion Acknowledgment Response

• HTTP Status: 202 Accepted

```
{
  "job_id": "job_9f81a2c3-4d5e-6f7a-8b9c-0d1e2f3a4b5c",
  "status": "queued",
  "timestamp_utc": "2026-08-28T21:40:00Z",
  "estimated_settle_ms": 380,
  "check_status_url":
"https://bios.dayone.com/api/v1/wetware/jobs/job_9f81a2c3"
}
```

Endpoint 3: Asynchronous Webhook Outbound Event Payload

• Protocol/Method: POST (Dispatched from biOS to Client Callback)

```
{
  "job_id": "job_9f81a2c3-4d5e-6f7a-8b9c-0d1e2f3a4b5c",
  "status": "completed",
  "telemetry": {
    "biological_latency_ms": 242,
    "basal_firing_rate_hz": 14.2,
    "glucose_consumption_delta": -0.004
  },
  "inference_output": {
    "output_vector": [1, 0, 1, 1, 0],
    "confidence_score": 0.974,
    "active_output_nodes": [501, 503, 504]
  }
}
```

4. Error Handling & Biological State Codes

Standard HTTP errors (400, 404, 500) only cover digital failures. biOS extends this error layer by embedding specific Biological Exception Codes inside response payloads when wetware failures occur:

Exception Code	System Definition	Automated Mitigation Action
BIO_ERR_501	Biological Latency Timeout: Neurons failed to generate output spike train within allotted window.	Issue a low-amplitude structural conditioning pulse to awaken pathways.
BIO_ERR_509	Metabolic Exhaustion Warning: Node glucose drop or high lactate indicates cell fatigue.	Throttle unit execution pool by 50%. Divert incoming traffic to adjacent cores. Trigger early nutrient flush.
BIO_ERR_612	Network Attenuation / Plasticity Failure: Neurons ceased physical remodeling in response to white noise reinforcement.	Flag core for biological degradation. Initiate Digital Twin emulation download and schedule a physical array replacement swap.
BIO_GUARD_777	Anomalous Integrated Neural Activity: sustained activity exceeds predefined research-monitoring thresholds for complexity, persistence, integration, or unexpected autonomous behavior. No single metric, including synchrony or Φ, is treated as evidence of consciousness.	Suspend nonessential stimulation; place the cartridge in observation/isolation mode; preserve telemetry and experimental state; prohibit automatic destructive intervention; require predefined scientific and ethical review before resuming or modifying the experiment.

5. AdaptOne BioHybrid Operational Continuity Addendum

This section extends the asynchronous biOS workflow with the proposed AdaptOne BioHybrid operational architecture: biological health telemetry, replaceable BioCompute Cartridges, inventory readiness, learning continuity, backup/recovery, and cautious monitoring for possible emergent consciousness-related behavior. **These mechanisms are architectural proposals and research hypotheses; they are not presented as validated production capabilities.**

5.1 Biological Health Monitoring

AdaptOne should maintain a BioHealth feature vector for every active and reserve cartridge. The vector can combine MEA electrical activity (firing rate, burst rate, spike amplitude, active-electrode count), response repeatability, signal-to-noise ratio, electrode impedance, stimulus-response latency, temperature, pH, dissolved oxygen, media condition, and available cell-viability/metabolic indicators. AdaptOne can combine these into a versioned BioHealth Index and a degradation slope rather than relying on a single instantaneous threshold.

Suggested operational states: NORMAL - continue service; DEGRADED - recalibrate and increase monitoring; STRESSED - reduce workload/stimulation and route traffic to a healthy module; CRITICAL - isolate the cartridge and initiate replacement.

5.2 Zero-Downtime BioCompute Cartridge Replacement

The replaceable unit should be a standardized, sealed BioCompute Cartridge containing the living neural culture, MEA, cartridge identity, calibration metadata, and environmental interface. AdaptOne should maintain at least one fully conditioned Hot Reserve. When degradation is detected, the replacement cartridge first receives shadow traffic, is compared with the active cartridge against functional acceptance thresholds, and is promoted through an atomic switchover. The degrading cartridge is then removed from production without interrupting the digital service.

5.3 Cartridge Inventory and Readiness

Inventory should be divided into an Active Pool, Hot Reserve, Warm Reserve, and Replenishment/Conditioning Pipeline. The BioInventory Controller should track cartridge age, BioHealth Index, degradation rate, calibration state, predicted remaining useful life (RUL), compatibility, and replenishment lead time. Safety stock should be dynamic: expected cartridge demand during replenishment lead time plus a failure-risk buffer. Accelerating degradation should automatically raise the required reserve level.

5.4 Maintaining Learning Across Neuron Replacement

AdaptOne Neural Trees and network structures should be treated as the durable semantic and contextual memory. The biological cartridge is a replaceable adaptive substrate. Before a swap, AdaptOne snapshots Neural Trees, topology, associations, timestamps, encoder/decoder parameters, calibration state, stimulus-response signatures, confidence distributions, and relevant temporal behavior. The new cartridge is conditioned using this

durable state as the teacher/reference, then runs in shadow mode until functional equivalence is demonstrated.

The design objective is therefore to preserve learned function and semantic relationships, not the physical identity or exact microscopic synaptic state of individual neurons.

5.5 Backups, Versioning, and Recovery

Backups should include Neural Trees and network topology, associations and timestamps, event logs, encoder/decoder models, biological response signatures, calibration history, BioHealth history, and training/reinforcement history. A practical policy is continuous event logging plus versioned immutable checkpoints, multiple independent copies, integrity checks, and scheduled restore drills. A 3-2-1 strategy (three copies, two storage classes, one isolated/off-site copy) is appropriate for the digital state.

The exact microscopic synaptic state of a living culture is not currently a conventional digital backup object. Recovery therefore restores the AdaptOne digital state, installs a healthy cartridge, replays calibration/training material, shadow-tests the reconstructed behavior, and promotes the cartridge only after acceptance criteria are met.

5.6 Detecting Possible Emergent Consciousness

No universally accepted engineering detector for consciousness exists. AdaptOne should therefore treat any 'consciousness' monitor as a research and safety watch, not as proof of subjective experience. Candidate indicators include persistent self-modeling, cross-module integration, metacognitive error/uncertainty behavior, temporal continuity of internal state, coherent novel goal formation, and reproducible state-dependent responses to controlled perturbation.

Any escalation should require multiple converging indicators, longitudinal evidence, reproducibility, predefined thresholds, and independent scientific and ethical review. A single metric such as synchrony or Φ should not by itself be interpreted as evidence of consciousness.

6. Example Client Implementations

The following examples implement the existing biOS asynchronous contract: submit a wetware job, receive HTTP 202 with a Job ID, and obtain completion through non-blocking status polling. In a production deployment, the preferred completion path can be the registered webhook/event broker described earlier in this addendum.

6.1 Python Example (asyncio + aiohttp)

```
import asyncio

import aiohttp
```

```
BASE = "https://bios.example.com"



async def submit_job(session):

    request = {

        "rack_id": "sg-rack-01",

        "unit_id": "adaptone-bio-12",

        "callback_url": "https://client.example.com/v1/bioutcomes",

        "timeout_ms": 5000,

        "payload": {

            "input_vector": [0.821, 0.045, 0.119, 0.943],

            "mapping_mode": "spatiotemporal_frequency",

            "target_input_nodes": [102, 104, 106, 108]

        }

    }

    async with session.post(

        f"{BASE}/api/v1/wetware/compute", json=request

    ) as response:

        response.raise_for_status()

        ack = await response.json()

        return ack["job_id"], ack["check_status_url"]



async def wait_for_result(session, status_url):

    while True:

        async with session.get(status_url) as response:

            response.raise_for_status()

            result = await response.json()



        if result["status"] == "completed":

            return result

        if result["status"] in {"failed", "cancelled"}:
```

```
            raise RuntimeError(result)

        await asyncio.sleep(0.25)  # non-blocking wait


async def main():
    async with aiohttp.ClientSession() as session:
        job_id, status_url = await submit_job(session)
        result = await wait_for_result(session, status_url)
        print(job_id, result["inference_output"])


asyncio.run(main())
```

6.2 C++ Example (C++20 coroutine-style pseudocode)

This example uses abstract async HTTP functions so the architecture is independent of a particular C++ networking library.

```
#include <chrono>
#include <iostream>
#include <string>
#include <nlohmann/json.hpp>

using json = nlohmann::json;

// Provided by the selected async HTTP runtime:
task<json> http_post_json(std::string url, json body);
task<json> http_get_json(std::string url);
task<void> async_sleep(std::chrono::milliseconds delay);

task<json> run_bios_job() {
    json request = {
        {"rack_id", "sg-rack-01"},
        {"unit_id", "adaptone-bio-12"},
        {"callback_url", "https://client.example.com/v1/bioutcomes"},
```

```
        {"timeout_ms", 5000},
        {"payload", {
            {"input_vector", {0.821, 0.045, 0.119, 0.943}},
            {"mapping_mode", "spatiotemporal_frequency"},
            {"target_input_nodes", {102, 104, 106, 108}}
        }}
    };

    json ack = co_await http_post_json(
        "https://bios.example.com/api/v1/wetware/compute", request);

    std::string status_url = ack.at("check_status_url");

    for (;;) {
        json state = co_await http_get_json(status_url);

        if (state.at("status") == "completed")
            co_return state;

        if (state.at("status") == "failed")
            throw std::runtime_error("biOS wetware job failed");

        co_await async_sleep(std::chrono::milliseconds(250));
    }
}

task<void> application() {
    json result = co_await run_bios_job();
    std::cout << result["inference_output"].dump() << '\n';
}
```

6.3 PL/I (PL/1) Example

PL/I is included here because its rich data-structure capabilities make it particularly well suited to representing AdaptOne neural objects: neurons, synapses, weighted associations, hierarchical Neural Trees, variable-length collections, state vectors, and nested biological telemetry structures. Its value in this design comes from the close fit between the conceptual neural architecture and PL/I's structured in-memory representations, not from any particular deployment platform. The example below keeps transport-specific HTTP/JSON operations behind site-specific routines so the core AdaptOne model remains focused on structured neural representation and asynchronous execution.

```
ADAPTONE_BIOS: PROC OPTIONS(MAIN);


   DCL JOB_ID       CHAR(64) VARYING;
   DCL STATUS_URL   CHAR(256) VARYING;
   DCL JOB_STATUS   CHAR(16) VARYING;
   DCL REQUEST_JSON CHAR(2048) VARYING;
   DCL RESULT_JSON  CHAR(4096) VARYING;


   REQUEST_JSON =
      '{"rack_id":"sg-rack-01",' ||
      '"unit_id":"adaptone-bio-12",' ||
      '"timeout_ms":5000,' ||
      '"payload":{"input_vector":[0.821,0.045,0.119,0.943],' ||
      '"mapping_mode":"spatiotemporal_frequency",' ||
      '"target_input_nodes":[102,104,106,108]}}';


   /* Site-specific routine performs POST and returns HTTP 202 fields. */
   CALL BIOS_SUBMIT_ASYNC(REQUEST_JSON, JOB_ID, STATUS_URL);


   PUT SKIP LIST('Queued biOS job: ', JOB_ID);


   JOB_STATUS = 'QUEUED';
```

```
   DO WHILE (JOB_STATUS ^= 'COMPLETED');


      /*
       * In production, return control to the scheduler/message loop here.
       * This sample routine represents a later event-driven re-entry.
       */
      CALL WAIT_NONBLOCKING_EVENT(250);


      CALL BIOS_GET_STATUS(
         STATUS_URL,
         JOB_STATUS,
         RESULT_JSON
      );


      IF JOB_STATUS = 'FAILED' THEN DO;
         PUT SKIP LIST('biOS job failed: ', JOB_ID);
         STOP;
      END;
   END;


   CALL PROCESS_BIOS_RESULT(RESULT_JSON);


END ADAPTONE_BIOS;
```

6.4 AdaptOne Integration Pattern

In all three languages, the Job ID should become the correlation key between the AdaptOne Neural Tree event and the biological execution. AdaptOne can persist a small transaction record containing Neural Tree/node identity, Job ID, cartridge ID, encoder version, BioHealth Index at dispatch, timestamp, and expected completion window. When the completion event arrives, the result is attached to the correct learning context and can update both the durable AdaptOne structure and the biological-response model.

ADDENDUM B

POWER-CONSUMPTION ARCHITECTURES FOR ADAPTONE

Comparing Matrix-Based AI, Native AdaptOne Silicon, Hybrid Bio/Silicon, and Fully Biological Compute

This addendum develops a power-consumption framework for three possible AdaptOne implementation paths. The comparison deliberately separates known reference measurements from AdaptOne research projections. No performance-normalized power measurements presently exist for AdaptOne; therefore, all AdaptOne savings ranges below are engineering hypotheses to be validated on identical workloads, accuracy targets, latency constraints, and complete wall-power measurements.

B.1 Reference Energy Baseline

Modern matrix-based AI systems obtain most of their performance from dense or sparsified tensor/matrix operations executed on GPUs and accelerators. As a concrete device reference, an NVIDIA H100 SXM has a configurable maximum TDP of up to 700 W. An eight-GPU server can therefore expose roughly 5.6 kW of GPU TDP before CPUs, memory, networking, storage, cooling, and power-conversion losses are included. This is a hardware reference, not a claim that every AI inference consumes 5.6 kW.

By contrast, the adult human brain is commonly estimated to consume approximately 20 W of metabolic power. That figure is useful as evidence that biological information processing can be extraordinarily energy-efficient, but it must not be treated as a direct performance-equivalent benchmark against a GPU or AI model.

Reference sources: NVIDIA H100 specifications (H100 SXM maximum TDP up to 700 W); NCBI Basic Neurochemistry, which estimates total human cerebral metabolic rate at approximately 20 W.

Architectural generations used in this addendum: Generation 1 — AdaptOne/Silicon, in which Neural Trees and learning structures remain digital; Generation 2 — AdaptOne BioHybrid, in which durable AdaptOne state remains digital while selected adaptive computation is assigned to biological cartridges; Generation 3 — Predominantly Biological AdaptOne, in which increasing portions of the AdaptOne learning substrate itself are represented biologically. These are successive research architectures, not interchangeable descriptions of one implementation.

B.2 Comparison A — Matrix-Based AI on Silicon vs. Native AdaptOne on Silicon

A native silicon implementation of AdaptOne would retain conventional electronic hardware while replacing large portions of matrix-centric computation with AdaptOne's structural, event-driven Neural Trees and network operations. The theoretical power

advantage comes from avoiding unnecessary multiply-accumulate activity when only a small part of the learned structure changes or is traversed.

The principal mechanisms are expected to be: sparse activation; event-driven execution; local structural updates rather than global weight-matrix modification; selective traversal of Neural Trees; persistent symbolic/structural state that does not require recomputation at every step; and the ability to map frequently used AdaptOne operations into dedicated silicon accelerators.

B.3 Comparison B — Matrix-Based AI vs. AdaptOne Hybrid Bio/Silicon

The hybrid design assigns semantic organization, orchestration, durable Neural Trees, safety monitoring, persistence, I/O, and deterministic control to silicon while assigning selected adaptive pattern-recognition or associative functions to living neural cartridges. The objective is not to replace all silicon computation. Instead, AdaptOne selectively invokes biological computation where neural plasticity and massively parallel low-energy dynamics provide an advantage.

The relevant system-power equation must include more than neuronal metabolism. A valid BioHybrid measurement includes MEA electronics, amplification, ADC/DAC conversion, pumps or microfluidics, temperature control, environmental sensing, cartridge conditioning, digital orchestration, and reserve-cartridge overhead. The biological neurons themselves may operate at very low power, but the support system can dominate at small scale. The energy advantage should increase only if useful biological compute scales faster than its life-support overhead.

B.4 Comparison C — AdaptOne on a Predominantly Biological Computational Substrate

The longest-term architecture is a predominantly biological AdaptOne implementation in which most learning and inference functions reside in biological neural networks and silicon is reduced to a minimal supervisory, communication, sensing, and persistence layer. A fully biological implementation is treated as a more distant limiting case that would require biological or bio-compatible equivalents for functions now performed electronically. In this model, AdaptOne Neural Trees would require a biological or bio-compatible physical representation rather than merely using biology as an accelerator.

This architecture has the greatest theoretical energy-efficiency potential and the greatest technical uncertainty. Major unresolved questions include deterministic addressing, durable state persistence, exact backup/recovery, manufacturing repeatability, cartridge life, input/output bandwidth, biosecurity, learning transfer, and whether a biological representation of AdaptOne structures can retain the required semantics and operational reliability. The term fully biological should therefore be reserved for a future architecture in which supervisory, I/O, persistence, and related functions also have validated biological equivalents.

B.5 Illustrative Research Target Ranges — Not Predictions

The following normalized index is intended solely for architecture planning and hypothesis formation. It sets a matrix-based silicon implementation to 100 energy units for the same completed workload. The AdaptOne ranges are illustrative research targets selected to define experimentally testable goals; they are not measurements, forecasts, or empirically derived predictions. Future values should instead be derived from measured active-node fraction, memory-access energy, accelerator utilization, MEA and life-support power, and useful work per joule.

Architecture	Compute Model	Normalized Energy Index	Illustrative Reduction	Confidence
Current matrix-based AI / silicon	GPU/accelerator tensor operations	100	Baseline	Measured device power; workload-dependent
Native AdaptOne / silicon	Sparse, event-driven Neural Trees + networks	20–50	~2× to 5× lower	Theoretical engineering target
AdaptOne BioHybrid	AdaptOne silicon control + biological neural cartridges	5–20	~5× to 20× lower	Higher uncertainty; support power included
Predominantly Biological AdaptOne	Biological neural processing + minimal supervisory interface	1–10	~10× to 100× lower	Long-term research hypothesis; not a production prediction

Interpretation: these ranges are intentionally broad. AdaptOne silicon must first demonstrate that structural/event-driven execution reduces energy per completed task. BioHybrid systems must then show that biological computation saves more energy than MEA and life-support infrastructure consume. A fully biological chipset remains a research concept and should not be assigned a production power number until a functional prototype exists.

B.6 Illustrative 8×H100 Reference Scenario

For intuition only, consider a server whose eight H100 SXM GPUs expose a combined maximum GPU TDP of approximately 5.6 kW. Applying the normalized research-target ranges above to that 5.6 kW GPU-only reference would imply the following hypothetical compute-power envelopes. These are not performance-equivalent forecasts and exclude the rest of the server and facility.

Architecture	Normalized Index	Illustrative Compute Power	Status
8×H100 matrix baseline	100	5.6 kW GPU TDP reference	Hardware reference
Native AdaptOne silicon	20–50	~1.12–2.8 kW	Hypothetical target
AdaptOne BioHybrid	5–20	~0.28–1.12 kW	Hypothetical target

Predominantly Biological AdaptOne	1–10	~0.056–0.56 kW	Long-term research hypothesis

B.7 Measurement Method Required to Validate the Hypothesis

Use the identical application workload and input dataset on every architecture.
Hold output quality/accuracy and maximum latency constant.
Measure total wall power, not merely processor or neuron power.
Include training/conditioning energy, idle power, cooling, biological life-support, reserve cartridges, storage, and networking.
Report joules per completed inference/task and joules per successful learning update.
Measure steady-state operation and full lifecycle energy, including cartridge replacement and reconditioning.
Publish uncertainty ranges and failure/retry energy so biological variability is not hidden.

B.8 Central AdaptOne Power Hypothesis

The central hypothesis is that AdaptOne can reduce energy in three successive stages: (1) eliminate unnecessary matrix operations through structural/event-driven computation on silicon; (2) offload selected adaptive functions to biological neural networks while retaining silicon for control and persistence; and (3) ultimately move increasing portions of the AdaptOne learning substrate into biological computation itself.

The expected direction of power reduction is therefore Matrix AI/Silicon → AdaptOne/Silicon → AdaptOne BioHybrid → predominantly biological AdaptOne. The magnitude of each reduction must be established experimentally rather than inferred from biological efficiency alone.

Professor Michael Recchia
AdaptNet AI Labs