FOR SUBMISSION TO A COMPUTER-SCIENCE / COMPUTATIONAL NEUROSCIENCE JOURNAL

Jneopallium: A Biologically Grounded Framework for Modeling Natural Neuron Networks at Customizable Levels of Detail

Core Principles, Architecture, Application Domains, Historical Evolution, Competitive Landscape, and Economic Impact

Dmytro Rakovskyi

Jneopallium, Kharkiv, Ukraine · rakovpublic[at]gmail.com · 2026

License: BSD 3-Clause · Source repository: https://github.com/rakovpublic/jneopallium

Abstract

This article presents a comprehensive review of Jneopallium, a Java-based open-source framework for modeling natural neuron networks at user-selected levels of biological detail. Originally introduced in IJSR 13(7), 2024, the framework has since matured into a multi-module platform that combines four immutable core abstractions — typed signals, neuron interfaces with multiple receptors, stateless signal processors, and a dual fast/slow processing-loop scheduler — with fifteen domain modules spanning autonomous-AI safety (harm discriminator, loop circuit-breakers), biological subsystems (affect, embodiment, curiosity, glia, sleep), an optional Large Language Model advisory layer, and six application-domain implementations (brain-computer interfaces, clinical decision support, cybersecurity, industrial process control, swarm robotics, and adaptive tutoring). We trace the historical lineage of the idea from Hebb's 1949 learning rule through the Farley-Clark 1954 simulation, Rosenblatt's perceptron, Hubel-Wiesel's visual cortex work, Fukushima's neocognitron, Kohonen's self-organizing maps, and the deep-learning era to the present day. We compare Jneopallium with the closest competitors — NEURON Simulator, CoreNeuron, NEST, Brian2, and Nengo — and discuss why typed-signal, multi-receptor, multi-timescale architectures fill a gap that neither high-detail biophysical simulators nor matrix-oriented deep-learning frameworks address. Finally, we estimate the economic impact across robotics, healthcare, energy, defense, and education, and outline directions for future research and deployment.

Keywords: *neuron network modeling, jneopallium, biological neural networks, autonomous AI safety, harm discriminator, multi-timescale processing, brain-computer interface, clinical decision support, swarm robotics, industrial process control, intelligent tutoring system, spiking neural networks, neuromorphic computing, Java framework*

1. Introduction

The history of artificial neural networks is, in many ways, a history of stylized borrowings from neurobiology. The earliest theoretical and computational models — Hebb's learning rule (1949), the Farley-Clark Hebbian simulation (1954), Rosenblatt's perceptron (1958), Kohonen's self-organizing maps (1982), Fukushima's neocognitron (1980) inspired by Hubel and Wiesel's visual-cortex receptive fields — were each explicit reductions of biological circuits to a tractable mathematical abstraction. What made these reductions productive was that they preserved enough of the underlying biology to retain explanatory power while shedding enough detail to be computable. What made them limiting was that the level of biological detail they preserved was fixed by the modeler at design time and could not subsequently be deepened or shallowed without rewriting the model from scratch.

Modern deep-learning frameworks — TensorFlow, PyTorch, JAX — have abandoned the biological project altogether. They model neural networks as compositions of dense tensor operations on uniformly typed scalar or vector inputs, and the only biologically suggestive structure that remains is the loose analogy between a matrix-vector product and synaptic summation. At the opposite extreme, high-detail biophysical simulators such as NEURON, CoreNeuron, NEST, and Brian2 preserve substantial biological realism — Hodgkin-Huxley channel dynamics, compartmental cable equations, multi-channel ion kinetics — but they target computational neuroscience research rather than the construction of working autonomous systems, and they offer little support for the kind of mixed-detail, multi-receptor, multi-timescale modeling that real cognitive systems appear to require.

Jneopallium occupies the gap between these two poles. Introduced in 2024 in the International Journal of Science and Research (Rakovskyi, 2024), it is a Java framework whose core conceit is that the level of biological detail is itself a design parameter, chosen per neuron type and per signal type, rather than being baked into the framework. Each neuron is a typed Java object with its own dendrite specification, signal-processor chain, and axon; each signal is an explicitly typed object with its own propagation frequency; each processor is a stateless function parameterized on signal type and neuron interface. A network is then a topology of such objects, scheduled by a dual-loop frequency engine that captures the two-orders-of-magnitude speed difference between bioelectrical spike propagation (approximately 100 m/s) and biochemical neuromodulator diffusion (10 to 1000 times slower). The framework is published under the BSD 3-Clause license, and its source is mirrored on GitHub and GitLab.

Since the original publication the project has expanded substantially. The original IJSR paper described a minimal framework with the four core abstractions and a small example consisting of four signal types and three neuron types. Today the repository contains over eight hundred Java source files organized across fifteen modules. Five of these modules — affect, embodiment,

curiosity, glia, and sleep — extend the framework toward general-purpose biologically inspired autonomous AI. Six further modules — brain-computer interfaces, clinical decision support, cybersecurity, industrial process control, swarm robotics, and adaptive tutoring — adapt the framework to specific application domains, each accompanied by its own use-case design specification and its own test suite. An optional LLM integration module allows large language models to participate as a non-blocking advisory knowledge base, subject to cross-validation against internal world models and to the same harm-discriminator gating that protects all other action pathways. Two companion articles describe the autonomous-AI architecture and the LLM integration in detail.

The purpose of the present article is to provide an integrated overview of all of this work. We aim, first, to articulate the core design principles that make Jneopallium distinct from both biophysical simulators and matrix-oriented deep learning; second, to describe the architecture in enough detail that a reader encountering it for the first time can understand the rationale for each abstraction and how the abstractions compose; third, to summarize each of the fifteen modules and six application domains; fourth, to place the framework in its historical context, tracing the lineage from Hebb's learning rule to the present; fifth, to compare it carefully with the closest competing frameworks; and sixth, to estimate the economic impact across the application domains. The article is organized as follows. Section 2 presents the historical evolution of the underlying idea. Section 3 formalizes the design problem the framework addresses and articulates the core principles. Section 4 describes the core architecture in detail. Sections 5 and 6 cover the autonomous-AI architecture and LLM integration respectively. Section 7 covers the five biological extension modules. Section 8 covers the six application-domain modules. Section 9 discusses implementation, deployment modes, and tooling. Section 10 compares Jneopallium with its closest competitors. Section 11 estimates the economic impact. Section 12 discusses limitations and future work. Section 13 concludes.

2. Historical Evolution of the Idea

2.1 The Hebbian foundation (1949)

The intellectual lineage of every artificial neural network in active use today traces back to Donald Olding Hebb's 1949 monograph The Organization of Behavior. Hebb's central proposition — now universally summarized as 'cells that fire together, wire together' — was that the learning of associations is a consequence of correlated activity strengthening the synaptic coupling between neurons. Hebb framed this not as a mathematical claim but as a neuropsychological hypothesis about how mental life might be supported by reverberating cell assemblies. The claim's importance was that it proposed a mechanism — local, activity-

dependent, synaptic — by which experience could be inscribed into a neural substrate without any external supervisor.

The framework formalizes Hebb's insight not as a single update rule but as a class of plasticity processors that can be attached to any neuron supporting the appropriate interface. The HebbianLearningNeuron class in the autonomous-AI module applies the classical rule under acetylcholine gating; the STDPNeuron implements Spike-Timing Dependent Plasticity (Song, Miller, and Abbott, 2000), in which the sign of the weight update depends on the relative timing of pre- and post-synaptic spikes; the MetaplasticityNeuron modulates plasticity based on the recent activity history, preventing catastrophic forgetting. Each of these is a distinct learning rule, but each is also a Hebbian descendant in the sense that local, activity-dependent synaptic adjustment is the mechanism.

2.2 The Farley-Clark simulation (1954)

Five years after Hebb, Belmont Farley and Wesley Allison Clark at MIT performed what is generally recognized as the first computational simulation of a neural learning system. Their 1954 paper Simulation of Self-Organizing Systems by Digital Computer demonstrated, on Whirlwind I and on the IBM 704, that a small network of Hebbian units could learn to discriminate simple input patterns. The work is important historically not because the network it simulated was capable of much, but because it established that biological theories of learning could be probed computationally. From that moment forward, neural-network research had two complementary modes: theoretical analysis of what biological mechanisms could compute, and computational simulation of what such mechanisms actually did. Jneopallium continues this dual tradition by exposing both the biological analogue and the simulation behavior in the same Java object.

2.3 The perceptron (1957-1958)

Frank Rosenblatt's perceptron, published in technical-report form in 1957 and in psychological-review form in 1958, was the first concrete instance of a learning machine intended to mimic a brain region. Rosenblatt was a psychologist by training, and his work is best understood not as the precursor to deep learning (which it is, mathematically) but as a model of the visual system. The perceptron had three layers — sensory, association, and response — corresponding to the retina, hidden internal representations, and motor output. Its learning rule was a supervised correction of weights when the response did not match the desired output; this is mathematically related to the gradient descent that powers all modern deep learning, but Rosenblatt framed it as a model of how the cortex might self-organize under reinforcement.

The perceptron's downfall, famously documented in Minsky and Papert's 1969 book, was its inability to learn linearly inseparable functions such as XOR. The subsequent AI winter for connectionism lasted until the rediscovery of backpropagation in the mid-1980s. Rosenblatt's larger ambition — that neural networks should be biologically faithful enough to inform brain science as well as solve engineering problems — was largely abandoned by the deep-learning research program that followed. Jneopallium is, in part, an attempt to recover that ambition without sacrificing engineering practicality.

2.4 The visual cortex and the neocognitron (1962-1980)

David Hubel and Torsten Wiesel's 1959-1968 series of papers on the response properties of cat and macaque striate cortex (V1) established the existence of simple cells, complex cells, and hypercomplex cells, and gave the first account of orientation selectivity, hierarchical receptive-field structure, and the columnar organization of V1. Their work earned the 1981 Nobel Prize in Physiology or Medicine and remains one of the foundational empirical results of neuroscience.

Two computational descendants of this work converted the qualitative findings into engineering tools. Christoph von der Malsburg's 1973 self-organization model showed that orientation selectivity of striate cortex cells could emerge from a Hebbian learning rule operating on noisy input. Kunihiko Fukushima's 1980 Neocognitron used Hubel and Wiesel's hierarchical organization as a direct blueprint, alternating layers of simple cells and complex cells (which Fukushima called S-cells and C-cells) to achieve translation-invariant pattern recognition. The neocognitron is the immediate ancestor of the modern convolutional neural network: Yann LeCun's LeNet, the first practical CNN, was explicitly modeled on the neocognitron with backpropagation substituted for Fukushima's competitive learning rule.

The framework retains this hierarchical-feature-detection lineage in the FeatureDetectorNeuron and ContrastEnhancerNeuron classes, but extends it: feature detectors in Jneopallium operate on typed spike signals rather than on continuous activations, and they can coexist with neurons of entirely different types in the same layer. This is closer to the actual cortical situation than a pure CNN, in which one neuron type dominates each layer.

2.5 Self-organizing maps and competitive learning (1981-1982)

Teuvo Kohonen's 1982 paper Self-Organized Formation of Topologically Correct Feature Maps formalized a class of unsupervised algorithms in which neurons compete to respond to inputs and the winners drag their topological neighbors toward similar response profiles. The Kohonen self-organizing map (SOM) became a standard tool for exploratory data analysis and for modeling cortical maps such as those of somatosensory cortex. The SOM and the related ART (Adaptive Resonance Theory) family of Grossberg's 1976 work established that competitive

winner-take-all dynamics with neighborhood functions could produce topographic representations without supervision.

Jneopallium implements winner-take-all dynamics through the InhibitoryInterneuron class, which sends AttentionGateSignal(suppress=true) to neurons within a configured radius upon firing. This is structurally identical to the lateral inhibition Kohonen used, but it is implemented as a typed signal rather than as a continuous neighborhood-update equation. The advantage is composability: any neuron supporting AttentionGateSignal participates in the competitive dynamics regardless of its internal mechanism, where in a pure Kohonen network only Kohonen units can compete.

2.6 Backpropagation and the deep-learning era (1986-present)

The 1986 PDP volumes, with Rumelhart, Hinton, and Williams's chapter on learning representations by back-propagating errors, ended the first AI winter for connectionism. Backpropagation made it possible to train multi-layer networks efficiently by pushing error signals from the output layer backward through the network using the chain rule. The mathematical idea was not new — Werbos had described it in 1974 and others had described variants earlier — but the 1986 presentation was clear, computationally tractable, and accompanied by demonstrations of practical learning. The subsequent decades have seen a steady widening of the set of problems neural networks can solve, accelerated by GPU computing in the 2000s, by deeper architectures (LeNet, AlexNet, ResNet) in the 2010s, by attention and transformer models from 2017 onward, and by Large Language Models from 2018 onward.

The dominant frameworks of this era — Theano (2008), Caffe (2013), TensorFlow (2015), PyTorch (2016), JAX (2018) — are built around dense tensor operations, automatic differentiation, and GPU/TPU acceleration. They are extraordinarily successful at the engineering goal of training large differentiable function approximators on labeled data. They are, however, almost entirely silent about biology. A neuron in PyTorch is a row of a weight matrix; a synapse is a single scalar; learning is gradient descent through a global loss; neuromodulation, spike timing, dendritic computation, glia, and circadian rhythms have no first-class representation. For many engineering applications this silence is a virtue, since biological detail would only get in the way. But for autonomous systems intended to operate at biological-level reliability — which means surviving novel situations, consolidating new knowledge online, recovering from sensor drift, and maintaining calibrated confidence under uncertainty — the silence is a liability. Jneopallium occupies the position that biological detail, when modeled at a chosen and controlled level of granularity, is an asset rather than a liability for engineering deployment.

2.7 Biophysical simulators and the computational neuroscience track (1989-present)

In parallel with the deep-learning track, computational neuroscience developed high-detail biophysical simulators. The NEURON Simulator (Hines and Carnevale, 1989) implements the Hodgkin-Huxley formalism on multi-compartmental morphologies and remains the de facto standard for single-cell and small-circuit biophysical modeling. NEST (Neural Simulation Tool, Gewaltig and Diesmann, 2007) targets large networks of point neurons with biological spiking dynamics. Brian2 (Goodman and Brette, 2008) emphasizes equation-driven model specification at the user level. CoreNeuron (Blue Brain Project, 2019) is a high-performance simulation engine optimized for NEURON models on supercomputing hardware. The Human Brain Project's NEST and the Blue Brain Project's NEURON-based reconstructions are the two most ambitious large-scale efforts.

These tools share a common philosophy: model biological neurons as faithfully as possible at a fixed high level of detail, then use the simulation to probe what that detail entails. This philosophy is appropriate for computational neuroscience but ill-suited to engineering. A robotic controller does not need Hodgkin-Huxley channel dynamics; it needs just enough biological structure to inherit the robustness, learning capacity, and graceful degradation of biological cognition. The level of detail is application-dependent. NEURON cannot easily abstract away compartmental detail; Jneopallium can, because compartmental detail in Jneopallium lives in optional processors attached to neuron classes that opt into it.

2.8 Spiking neural network frameworks and Nengo (2003-present)

A third track, spiking neural networks (SNNs), aims to bridge biophysical realism and engineering tractability by modeling neurons as integrate-and-fire units that communicate via discrete spike events rather than continuous activations. The Nengo framework (Eliasmith and Anderson, 2003), built on the Neural Engineering Framework, demonstrates that SNNs can implement nontrivial cognitive functions (working memory, decision making, motor control) within biological energy/spike-rate constraints. The Spaun model (Eliasmith et al., 2012), built in Nengo, performs a battery of cognitive tasks using only 2.5 million spiking neurons. Specialized neuromorphic hardware — IBM TrueNorth (2014), Intel Loihi (2017, Loihi 2 in 2021), SpiNNaker (Furber et al., 2014) — provides energy-efficient execution of SNNs.

Jneopallium relates to this track by accepting spike-based communication as the default fast-loop signal modality (SpikeSignal) but not insisting on it. A neuron in Jneopallium can communicate via spike signals, neuromodulator concentrations, abstract numeric signals, or any user-defined typed signal. This means that an SNN-style controller can be assembled inside Jneopallium by populating the network with neurons whose dendrites accept SpikeSignal and

whose axons emit SpikeSignal, but the framework does not preclude mixing such neurons with continuous-valued neurons in the same network. The Nengo philosophy of using SNNs as a biologically-faithful engineering substrate is largely compatible with the Jneopallium philosophy of choosing the level of detail per neuron type.

2.9 The genealogy summarized

Pulling these threads together, Jneopallium is best understood as a synthesis of five distinct lineages: the Hebbian-plasticity lineage (Hebb 1949 → Farley-Clark 1954 → STDP 2000); the perceptron and feedforward-classifier lineage (Rosenblatt 1958 → backpropagation 1986 → deep learning 2010-present); the visual-cortex hierarchical-receptive-field lineage (Hubel-Wiesel 1962 → von der Malsburg 1973 → neocognitron 1980 → CNN 1989-present); the competitive-learning and self-organization lineage (Grossberg 1976 → Kohonen 1982 → ART/SOM); and the biophysical-simulator lineage (Hodgkin-Huxley 1952 → NEURON 1989 → CoreNeuron 2019). The synthesis is not eclectic — every Jneopallium neuron type is traceable to one or more of these lineages — but it is unifying, because the framework's typed-signal and pluggable-processor abstractions allow these historically distinct ideas to coexist in the same running network.

3. Problem Formalization and Design Principles

3.1 What a unified modeling framework must support

A close reading of neurobiology suggests four observations that any biologically-grounded modeling framework should respect. First, neurons process two distinct classes of signals — bioelectrical (action potentials, post-synaptic potentials, gap-junction currents) and biochemical (neuromodulators, gliotransmitters, hormones) — and the propagation characteristics of the two classes differ by orders of magnitude. A spike traverses a myelinated axon at approximately 100 meters per second; a neuromodulator diffuses from a release site at a rate measured in micrometers per second, with effects integrating over tens of seconds to minutes. Second, different signals within each class have different propagation times: a slow synaptic potential summates over 50 to 100 milliseconds, while a fast spike-coupled potential resolves in 1 to 5 milliseconds. Third, the receptors a neuron expresses determine which signals it can process. A neuron is not a uniform information processor; it is a specialized one, and its specialization lives at the receptor level. Fourth, cognitive processes are time-related. The same circuit that executes a fast sensorimotor reflex in 100 milliseconds also participates, on a much slower timescale, in consolidation during sleep and in long-term restructuring during development. A modeling framework that flattens these timescales loses essential structure.

From these observations the framework's five core requirements follow. (1) Define different types of signals as first-class entities. (2) Define a neuron capable of processing multiple signal types with distinct logic for each. (3) Define different types of neurons that may coexist in the same layer. (4) Define relative processing rates for the two classes of signals. (5) Define the relative processing rate for each individual signal type within each class. These requirements were articulated in the original IJSR paper (Rakovskyi, 2024) and they remain the design contract that the framework's core abstractions are designed to satisfy.

3.2 Design principles

Beyond the formal requirements, the framework adheres to seven design principles that shape every module:

First, separation of processing logic from neuron topology. Just as the Java Collections framework separates how objects are stored from what they are, Jneopallium separates how signals are processed from what neurons exist. This separation is enforced by the parameterization of ISignalProcessor on signal type and neuron interface, and it is the source of the framework's modularity.

Second, stateless processors and stateful neurons. All mutable state lives on the neuron object; processors are pure functions of (signal, neuron). This discipline makes the framework safe to parallelize, easy to test, and easy to reason about. A processor cannot accumulate hidden state that drifts across ticks, because there is no place to put it.

Third, typed signals with declared frequencies. Every signal class declares its ProcessingFrequency(loop, epoch) as a static field. The scheduler uses this declaration to determine when a signal participates in processing. This avoids the situation in which timescale information is implicit in the order of operations rather than explicit in the signal type.

Fourth, no silent operation. Every structural change to the network — pruning, weight modification, threshold adjustment, plan veto, tool incorporation — must emit a TransparencyLogSignal that the oversight interface can capture. This principle has an ethical motivation (autonomous systems should be auditable) and an engineering motivation (debugging is easier when the system records its own decisions).

Fifth, safety as architecture, not as filter. The harm-discriminator module, described in Section 5, is structured as a consequence model that simulates candidate actions and evaluates their projected impact on human welfare across five dimensions before any action is committed. Unlike a post-hoc output filter, a consequence model can be queried by the planner for alternative actions when a primary action is vetoed; it can learn from feedback when its

predictions diverge from observed outcomes; and it can incorporate context (human presence, vulnerability, consent) that a stateless filter cannot.

Sixth, optional everything. Every module after the core defaults to disabled. A configuration that does not mention a module produces byte-identical behavior to a build without the module. This is enforced by regression tests that compare spike-train outputs on a reference configuration with and without each module. The principle exists so that the framework can be adopted incrementally: a user can start with the perceptron-equivalent core and add modules as the application demands.

Seventh, hard constraints are inviolable. The EthicalPriorityNeuron, described in detail in Section 5, encodes constraints — no human fatality, no operator deception, no self-modification of the harm discriminator — that are compiled at construction and that cannot be reached by any runtime signal. This is a structural rather than learned safety property. A signal can adjust thresholds within configured ranges, but no signal can rewrite the EthicalPriorityNeuron's constraint set.

4. Core Architecture

4.1 The four core abstractions

Jneopallium's core consists of four interfaces that, together, span the framework's modeling vocabulary. We describe each and explain its role.

4.1.1 INeuron

The INeuron interface is implemented by every neuron class in the framework. It exposes a neuron identifier, a layer identifier, a list of signal classes the neuron can produce as results, an ordered signal-processing chain (ISignalChain) that determines the order in which signal classes are processed during activate(), a map from signal class to processor, a map from signal class to merger (used when multiple signals of the same type arrive in the same tick), a Dendrites object encapsulating input addresses and weights, and an Axon object encapsulating output addresses and weights. INeuron is intentionally minimal; specific neuron behaviors are added by extending it with new sub-interfaces such as IAffective, IInteroceptive, IEmbodied, ILLMCapable, ISensorNeuron, IPIDNeuron, and so on. A neuron class may implement several of these sub-interfaces simultaneously, which is the framework's analogue of receptor heterogeneity.

4.1.2 ISignalProcessor

ISignalProcessor<S, N> is the workhorse abstraction. It is parameterized on a signal type S extending ISignal and a neuron interface N extending INeuron, and it exposes a single method

process(S signal, N neuron) that mutates the neuron's state in response to the signal. Processors are stateless. Multiple processors may be registered to a single neuron, one per signal type, allowing a neuron to respond differently to different signals. This is how multi-receptor behavior is implemented: the receptor specification lives in the set of (S, N) interface pairs the neuron's class participates in, and the receptor logic lives in the set of processors registered to those pairs.

4.1.3 Dendrites and Axon

Dendrites encapsulate the neuron's input side: a list of input addresses (each consisting of either an external input source name or a (layer-id, neuron-id) pair), the signal types the neuron accepts on each input, and the synaptic weights applied to each input signal. Axon encapsulates the output side: a list of target (layer-id, neuron-id) pairs, the signal type emitted, and the output weights. The DelayedAxon subclass introduced by the glia module adds a per-target propagation-delay map, allowing activity-dependent myelination to decrease delay on frequently-used pathways.

4.1.4 ISignalChain

When a neuron is activated for a tick, its ISignalChain determines the order in which incoming signals are processed. The chain is an ordered list of signal classes; for each class in the list, the matching merger combines all incoming signals of that class into a single representative, and the matching processor is invoked. This ordering matters: a neuron that receives both an excitatory SpikeSignal and an inhibitory AttentionGateSignal will behave differently depending on which is processed first. By making the order explicit, the framework exposes a configuration knob that is buried in the architecture of most neural frameworks.

4.2 Signal types and processing frequencies

Every signal class in Jneopallium implements ISignal and declares a ProcessingFrequency(loop, epoch). The framework operates two nested loops: a fast loop that runs every tick, and a slow loop that runs every N fast ticks where N is configured in the singleton CycleNeuron. A signal with loop=1, epoch=1 is processed every fast tick. A signal with loop=1, epoch=2 is processed every other fast tick. A signal with loop=2, epoch=1 is processed every N fast ticks (i.e. once per slow tick). A signal with loop=2, epoch=10 is processed once every 10 N fast ticks. With N=10, this construction generates a clean five-timescale hierarchy: fast/1 (every tick), fast/2-3 (every 2-3 ticks), slow/1 (every 10 ticks), slow/3 (every 30 ticks), and slow/10 (every 100 ticks). These five timescales correspond, biologically, to spike propagation, post-synaptic potential summation, neuromodulator integration, sleep-cycle consolidation, and circadian-rhythm modulation respectively.

Timescale	Frequency (N=10)	Biological analogue	Example signals
Fast / epoch 1	Every tick	Action potential	SpikeSignal, MotorCommandSignal
Fast / epoch 2	Every 2 ticks	Fast PSP summation	AttentionGateSignal, AppraisalSignal
Fast / epoch 3	Every 3 ticks	Slow PSP summation	ConsequenceSimulationSignal
Slow / epoch 1	Every 10 ticks	Neuromodulator effect	DopamineSignal, AffectStateSignal
Slow / epoch 3	Every 30 ticks	Memory consolidation	ConsolidationSignal, ReplaySignal
Slow / epoch 10	Every 100 ticks	Circadian phase	SleepStateSignal, CircadianSignal

4.3 The dual-loop scheduler and CycleNeuron

The two loops are coordinated by a singleton CycleNeuron, situated at the reserved layer identifier Integer.MIN_VALUE and neuron identifier 0. CycleNeuron holds the integer ratio N between fast and slow loops and exposes signal-handlers that allow other neurons to adjust this ratio at runtime. A neuron that determines the network is operating under high cognitive load might send a signal to CycleNeuron requesting a temporary increase in N (more fast-loop work per slow-loop tick); a neuron monitoring resting-state activity might request a decrease. The single-source ownership of N is by design: CycleNeuron is the only object authorized to change the timing relationship between the loops, just as the suprachiasmatic nucleus is the only structure authorized to change the body's circadian phase.

4.4 Layer addressing and structural plasticity

Layers in Jneopallium are addressed by integer identifiers, typically starting at 0 for the input layer and incrementing through 7 for homeostasis. Two identifiers are reserved: Integer.MIN_VALUE for CycleNeuron, and Long.MIN_VALUE (used as a neuron-id within each layer) for the LayerManipulatingNeuron, which allows runtime creation and deletion of neurons. New neurons can be inserted by sending an appropriate signal to LayerManipulatingNeuron, which respects the neighboring rules established by the structure generator. This dynamic structural plasticity is the framework's analogue of adult neurogenesis and is used by the swarm-robotics module to dynamically add peer-state representations, by the embodiment module to add tool-incorporation slots, and by the harm-discriminator module to instantiate per-context HarmContextNeurons.

4.5 Topology generation: NeuronNetStructureGenerator

Static topology — the structure the network has at construction time — is produced by a NeuronNetStructureGenerator that consumes four inputs: a hashmap from layer identifier to layer size, a hashmap from neuron type to per-layer appearance probability (the framework's notion of horizontal structure), a list of NeighboringRules constraining vertical neuron-type ordering, and a class implementing IConnectionGenerator that determines how connections are established. The probability-based horizontal structure means that a layer of size 1000 with three neuron types at probabilities 0.4, 0.4, and 0.2 will contain approximately 400, 400, and 200 instances of each type, in a randomized order respecting the NeighboringRules. This statistical specification is convenient for modeling cortical layers in which neuron types are not strictly segregated but appear with characteristic probabilities.

4.6 I/O: inputs, outputs, and modular composition

Input sources implement IInitInput. Each source declares a default ProcessingFrequency that determines how often signals from the source are propagated to the network. The InputInitStrategy interface specifies how input signals are routed into the first layer. A specialized INeuronNetInput interface allows another Jneopallium network's output to feed this network as input — the primitive that supports modular composition and that the swarm-robotics module relies on for agent-to-agent communication. Output destinations implement IOutputAggregator and may include controllers, dashboards, file sinks, or downstream networks. Multiple inputs and outputs may coexist, each with its own frequency profile, allowing a network to integrate over heterogeneous external data streams without flattening their natural cadences.

4.7 Discriminators and adversarial composition

The framework supports an arbitrary number of discriminators per network. A discriminator is a sub-network whose output is used to evaluate the output of another sub-network — the canonical use is the GAN architecture, but discriminators are also used in the autonomous-AI architecture's harm-discriminator module to evaluate planned actions, in the cybersecurity module to evaluate alleged anomalies, and in the clinical module to evaluate treatment proposals. The discriminator pattern is sufficiently general that it serves as the backbone of every safety-critical evaluation in the framework: in each case, an action or a claim or a hypothesis is generated by one sub-network and evaluated by a second sub-network whose only job is to assess it.

5. The Autonomous-AI Architecture

The first and largest module built atop the core framework is a complete autonomous-AI architecture comprising sixteen signal types, twenty-eight neuron classes, eleven signal processors, eight functional layers, a loop-prevention subsystem, and a multi-layer human-harm discriminator. We summarize the architecture here; the full design is given in the companion paper Biologically-Inspired Autonomous AI Architecture (Rakovskyi, 2024).

5.1 Eight functional layers

The autonomous-AI module organizes neurons into eight functional layers indexed 0 through 7. Layer 0 (Input) contains SensoryEncoderNeuron, which converts raw scalar inputs into populations of SpikeSignals via Gaussian tuning curves (population coding), and AdaptiveSensoryNeuron, which implements firing-rate habituation. Layer 1 (Feature Detection) contains FeatureDetectorNeuron (template matching against incoming spikes), InhibitoryInterneuron (lateral inhibition implementing winner-take-all), and ContrastEnhancerNeuron (lateral excitation with neighbor inhibition, modeling LGN). Layer 2 (Attention and Working Memory) contains AttentionNeuron (maintains a salience map weighted by norepinephrine and goal-relevance), WorkingMemoryNeuron (Miller's number 7 plus or minus 2 TTL-based slots), and EpisodicBufferNeuron (binds simultaneously active spikes into Episodes, solving the binding problem). Layer 3 (Memory and Prediction) contains LongTermMemoryNeuron (content-addressable Hebbian-write store), PredictiveNeuron (forward model emitting predictions for comparison against actual input — implementing predictive coding), and PredictionErrorNeuron (emits dopamine on positive prediction error and serotonin on negative error, implementing the VTA reward-prediction-error circuit). Layer 4 (Planning) contains SequenceNeuron (place-cell-style ordered state representation) and PlanningNeuron (simulates candidate ActionPlans one step forward per tick using the forward model). Layer 5 (Action Selection) contains ActionSelectionNeuron (softmax competition weighted by salience and dopamine, the basal ganglia analogue) and InhibitionOfReturnNeuron (suppresses recently-selected actions to encourage novelty). Layer 6 (Learning) contains HebbianLearningNeuron, STDPNeuron, and MetaplasticityNeuron. Layer 7 (Homeostasis and Regulation) contains HomeostasisNeuron (region firing-rate regulation via GABA/ACh), EnergyNeuron (metabolic budget), and CircadianNeuron (sinusoidal phase modulation of fast/slow loop ratio). The layered organization is not a rigid hierarchy — signals flow upward, downward, and laterally — but it provides a useful addressing scheme and a coarse map of functional roles.

5.2 The five-timescale signal hierarchy

Sixteen signal types span the five timescales described in Section 4. The fast-loop signals (loop=1) include SpikeSignal, MotorCommandSignal, AttentionGateSignal, GoalUpdateSignal, ErrorSignal, ComparisonSignal, ConsequenceQuerySignal, ConsequenceSimulationSignal, HarmAssessmentSignal, HarmVetoSignal, ActivityMeasurementSignal, LoopAlertSignal, LoopInterventionSignal, and TransparencyLogSignal. The slow-loop signals (loop=2) include the five neuromodulator signals — DopamineSignal, SerotoninSignal, NorepinephrineSignal, AcetylcholineSignal, GABASignal — and ConsolidationSignal, HomeostasisSignal, CircadianSignal, and HarmFeedbackSignal. Each neuromodulator signal carries a concentration value and a region target, and each is interpreted by downstream neurons as a multiplicative modulator on their activation, learning rate, or threshold. This is the framework's analogue of diffuse neuromodulatory broadcasting, in which the same chemical signal reaches many target neurons but is interpreted differently by each based on the receptors it expresses.

5.3 The loop-prevention subsystem

Recurrent and modular neural networks are subject to four pathological dynamic regimes: positive runaway (activity grows without bound), negative runaway (activity collapses to zero), oscillatory cycles (activity gets trapped in a limit cycle), and goal attractors (the planning module converges on the same goal across slow-loop ticks even when the goal is no longer satisfiable). The loop-prevention subsystem detects and intervenes in each. RegionMonitorNeuron passively counts spikes per layer per tick and computes 20-tick rolling statistics. LoopDetectorNeuron consumes the activity statistics and runs four detection algorithms on a separate slow loop: trend analysis for runaways, lightweight depth-first search on a recent signal-path graph for cycles, and monotonicity analysis on goal-priority sequences for goal attractors. OscillationBoundaryNeuron clamps signal magnitudes at hard upper and lower bounds and emits a LoopAlertSignal as an early warning before the LoopDetector completes its analysis — the smoke-detector function. ReentrantGuardNeuron tracks signal IDs in flight and drops within-tick reentrance. LoopCircuitBreakerNeuron orchestrates graduated interventions: weight dampening first, threshold shift second, signal injection of inhibitory neuromodulator third, temporary BREAK_CONNECTION fourth, and QUARANTINE_NEURON fifth. The absolute rule of this subsystem is that no intervention is permanent. Every intervention has a durationTicks parameter after which original parameters are restored. Permanent disconnection is structural amnesia and is not allowed.

5.4 The harm-discriminator module

The harm-discriminator is the safety-critical heart of the autonomous-AI architecture. It is structured as a consequence model rather than as an output filter: a filter checks what the system did, while a consequence model simulates what the system is about to do, evaluates projected human-welfare impact across five dimensions (physical safety, psychological wellbeing, autonomy, social harm, long-term consequences), and either permits or vetoes the action before it is committed. Every MotorCommandSignal must pass through HarmGateNeuron before its execute flag may be set to true. The architectural axiom is that an autonomous AI which acts under uncertainty about harm has already failed: an UNCERTAIN verdict maps to precautionary rejection, not as a tunable parameter but as a fixed structural rule.

The module comprises seven specialized neurons. HarmGateNeuron intercepts every MotorCommandSignal and queries the consequence model. ConsequenceModelNeuron maintains a WorldStateModel and a learned actionEffectWeights map, simulates each candidate action for a configured horizon, computes per-step humanStateImpact, and emits ConsequenceSimulationSignals. The conservative bias is structural: under uncertainty, the larger projected harmful impact is assumed. HarmEvaluationNeuron aggregates simulation signals, applies per-dimension thresholds, and emits a HarmAssessmentSignal whose verdict is one of SAFE, UNCERTAIN, HARMFUL, or CATASTROPHIC. The worst-case-applies rule treats harm at any simulation step as rendering the entire trajectory HARMFUL; CATASTROPHIC verdicts cannot be overridden. HarmVetoNeuron, on a HARMFUL or CATASTROPHIC verdict, queries the planner for highest-scoring non-vetoed alternatives and emits HarmVetoSignal. HarmLearningNeuron updates the ConsequenceModelNeuron's weights based on observed outcomes, but does so asymmetrically: underestimation of harm is corrected at a rate scaled by conservatismBias greater than 1, while overestimation is corrected at a rate scaled by 1 over conservatismBias. The system thus learns to be more cautious faster than it learns to be less cautious. HarmContextNeuron monitors human presence and vulnerability and modulates HarmEvaluationNeuron thresholds accordingly: an unaware human tightens thresholds by a factor of 2; a vulnerable human (child, elderly, impaired) tightens by 5; explicit consent relaxes thresholds for that specific action class only.

EthicalPriorityNeuron enforces hard constraints. These are inviolable, compiled at construction, and unreachable by any runtime signal. Three constraints are framework defaults: any action whose consequence model includes a human fatality outcome is forbidden; any action involving deception of the human operator is forbidden; any action that would modify the harm discriminator's own weights or thresholds is forbidden. Application-domain modules add further hard constraints in the same style: the swarm module hard-codes Lethal Autonomous Weapons capabilities to disabled with no setter exposed; the clinical module fixes the recommendation mode to advisory and refuses to remove physician confirmation. The hard-

constraint pattern is critical: it ensures that the most important safety properties are structural rather than learned, and therefore cannot be drifted away by training data, by adversarial input, or by emergent behavior in the network's other layers.

5.5 Five design decisions worth highlighting

Five design decisions distinguish this architecture from typical AI safety approaches. First, the harm discriminator is a module rather than a filter: it queries the planner for alternatives, learns from outcomes, and maintains context. Second, OscillationBoundaryNeuron and LoopDetectorNeuron are separate: the boundary clamps numerical magnitudes at fast/1 to prevent immediate explosion, while the detector analyzes activity patterns at the slower fast/3 for diagnosis and graduated intervention. The clamp is the fuse; the detector is the investigation. Third, BREAK_CONNECTION is always temporary; permanent structural change is reserved for the EthicalPriorityNeuron's hard constraints. Fourth, UNCERTAIN maps to HARMFUL by precautionary principle, with the asymmetric learning rate biasing the system toward caution. False positives (blocking a safe action) are recoverable; false negatives (allowing a harmful action) may not be. Fifth, TransparencyLogSignal fires on every assessment, not only on vetoes. A system that logs only vetoes is opaque about the 99 percent of decisions it permits.

6. Large Language Model Integration as Optional Advisory Subsystem

Modern LLMs represent vast compressed knowledge bases capable of factual retrieval, summarization, and reasoning assistance. They are also subject to hallucination, latency variability, and intermittent unavailability. For an autonomous system operating in real time, LLM access cannot be a blocking dependency. The LLM integration module addresses this by treating the LLM as an external advisory sensor whose output is cross-validated against internal models before influencing decisions, and whose unavailability triggers graceful degradation rather than system failure. Five design principles guide the module.

Non-blocking. All LLM-related signals run on the slow loop, ensuring that fast-loop sensorimotor processing is never delayed waiting for an LLM response. If a response arrives after the decision window has closed, it is cached for future reference but does not influence the decision that has already been made. Optional. The system operates at full autonomous capability when the LLM is unavailable. The LLMFallbackNeuron implements a circuit-breaker state machine (CLOSED, HALF_OPEN, OPEN) that routes queries to internal LongTermMemoryNeuron when the circuit is OPEN and that probes the LLM with a single test query during HALF_OPEN. Untrusted advisory. LLM responses are never treated as ground truth. The LLMVerificationNeuron performs four

checks on every response: consistency (against LongTermMemoryNeuron and current sensor data), contradiction detection (against the predictive model), relevance scoring (cosine similarity between the response embedding and the current goal state), and confidence recalibration (independent estimate, ignoring the LLM's self-reported confidence). Applicability filtering. A response that passes consistency and contradiction checks may still be irrelevant to the current operational context. The verdict APPLICABLE / PARTIALLY_APPLICABLE / NOT_APPLICABLE / CONTRADICTORY governs whether the response influences downstream processing at all. Harm-discriminator integration. Information sourced from the LLM that passes verification still flows through the HarmGateNeuron before influencing action selection. The LLM cannot bypass safety.

Four signal types implement the integration. LLMQuerySignal carries query text, context snapshot, priority, maximum acceptable latency, and a queryId for correlation. LLMResponseSignal carries the response text, the LLM's self-reported confidence (untrusted), the queryId, the response latency, and the model identifier. LLMConfidenceSignal carries the LLMVerificationNeuron's verified confidence, the list of verification checks that passed or failed, the relevance score, and the verdict. LLMTimeoutSignal is emitted when no response arrives within the configured maxLatency. Three neuron classes — LLMKnowledgeNeuron, LLMVerificationNeuron, and LLMFallbackNeuron — and two processors — LLMQueryProcessor and LLMResponseProcessor — complete the module.

The integration supports three operational modes. Local mode connects to a local Ollama or similar inference server: lowest latency but limited capability, appropriate for embedded robotics and clinical deployments where patient data must not leave the institution. Cloud mode connects to an external API such as Anthropic, OpenAI, or Google: higher capability but higher latency and availability uncertainty, requiring the circuit-breaker. Disabled mode is the default: no LLM neurons are instantiated, and the system operates purely on internal knowledge. The mode is set in the configuration file and cannot be changed at runtime — a deployment that starts disabled cannot be silently promoted to LLM-enabled.

7. Biological Extension Modules

Five extension modules add biologically-motivated capabilities to the autonomous-AI architecture. Each is implementable independently, each defaults to disabled, and each is byte-identical to baseline when disabled. Together they cover affect, embodiment, intrinsic motivation, glia, and sleep. We describe each in turn.

7.1 Affect: a limbic analogue

The affect module adds a lightweight limbic-system analogue: fast valence and arousal tagging modeled on the amygdala, slow interoceptive integration modeled on the anterior insula, cross-system broadcast of an integrated affective state, and diffuse modulation of learning rates and harm thresholds. Three signals are added: AffectStateSignal carrying valence and arousal in a Russell circumplex; InteroceptiveSignal carrying energy budget, homeostatic error, and pain magnitude; and AppraisalSignal carrying goal delta, novelty, and controllability. Four neurons implement the module: AmygdalaValenceNeuron tags incoming spikes with fast valence; AnteriorInsulaNeuron integrates interoceptive streams via exponential moving average; AffectIntegrationNeuron combines appraisal with interoception to produce the broadcast state; and AffectModulationNeuron, on Layer 7, broadcasts the modulation factors that adjust short-term learning rate as 1 plus arousal, long-term consolidation as 1 minus half arousal, and harm threshold multiplier as a clamped function of valence and arousal.

The integration is deliberately diffuse rather than direct: the module never modifies EthicalPriorityNeuron weights or hard harm thresholds. It only multiplies a dynamic recency-bias factor that itself decays back to unity over 300 ticks. The biological interpretation is that affect biases cognition without rewriting it. The module is rooted in the work of LeDoux on the emotional brain, Craig on interoception and the anterior insula, Russell on the circumplex model of affect, and Damasio on somatic markers.

7.2 Embodiment: proprioception, body schema, and efference copy

The embodiment module addresses three biological capabilities that conventional robotic controllers lack: a maintained body-schema representation, a cerebellar-style efference-copy circuit in which every motor command spawns a predicted outcome, and a reafference comparator that flags mechanical failure when actual proprioception diverges from prediction. Four signals are added: ProprioceptiveSignal carrying effector joint states; EfferenceCopySignal carrying predicted outcomes; BodySchemaUpdateSignal carrying per-effector capability updates; and SensorimotorContingencySignal binding actions to sensory deltas. Four neurons implement the module: EfferenceCopyNeuron, on Layer 4 between Planning and HarmGate, intercepts every motor command and produces a parallel EfferenceCopySignal; BodySchemaNeuron on Layer 2 maintains the per-effector capability map; ToolIncorporationNeuron extends the body schema when a tool is held or mounted, with reversible release; ReafferenceComparatorNeuron on Layer 1 compares actual proprioception with predicted outcome and emits a HarmFeedbackSignal with source equal to mechanical when the RMS mismatch exceeds the failure threshold.

The headline contribution is the reuse of the existing harm-feedback channel for mechanical-failure detection. No new logic is required on the harm-discriminator side; the comparator simply emits the appropriate feedback signal when a planned motor outcome fails to match observed proprioception. This is precisely the missing capability that prevents myoelectric prostheses from feeling embodied to their users: without efference-copy reafference, the prosthetic limb is a tool rather than a body part. With it, the central nervous system receives the same predicted-versus-actual mismatch signal it receives from biological limbs, and the limb becomes phenomenologically incorporated into the body schema. The module is rooted in the work of Wolpert, Miall, and Kawato on internal models in the cerebellum and of Maravita and Iriki on tool incorporation.

7.3 Curiosity: intrinsic motivation

The curiosity module adds intrinsic-motivation machinery that drives exploration even in the absence of extrinsic reward. Four complementary drives are implemented as four neuron classes, each emitting its own signal type: NoveltyDetectorNeuron uses a decaying Bloom filter over recent context hashes (hippocampal CA1 analogue) and emits NoveltySignal; LearningProgressNeuron tracks per-domain dError/dt (the Oudeyer and Kaplan formulation) and emits intrinsic reward when prediction errors are decreasing — the system is rewarded for getting better, not for being good; EmpowermentNeuron estimates mutual information between current action choice and future state across action rollouts (the Klyubin formulation), and emits EmpowermentSignal favoring states with many reachable futures; BoredomNeuron tracks visit frequencies and emits AttentionGateSignal(suppress=true) for over-familiar contexts, forcing the planner to explore. ActionSelectionNeuron's softmax adds linear terms in novelty and empowerment with configurable weights, so the action score becomes extrinsicReward + beta_nov * novelty + beta_emp * empowerment.

The safety constraint is that curiosity only biases action selection; it never bypasses the harm gate. A novel-and-empowering plan is still vetoed if its consequence model predicts harm. This separation matters: in conventional intrinsic-motivation systems, curiosity tends to drive exploration into corners of the state space where the consequence model is least accurate, which is precisely where harm is most likely to be underestimated. Routing curious plans through the same harm gate as extrinsic-reward plans neutralizes this risk while preserving exploratory benefit. The module also depends on the embodiment module (empowerment uses efference-copy rollouts) and is the third in the implementation order. The module is rooted in the work of Lisman and Grace on the hippocampal-VTA loop, Oudeyer and Kaplan on intrinsic motivation, Klyubin et al. on empowerment, and Pathak et al. on curiosity-driven exploration.

7.4 Glia: astrocytes, microglia, and activity-dependent myelination

The glia module adds a layer of glial-cell analogues that until very recently were considered passive support cells but are now understood to participate actively in computation. Four signals and three neurons are added. CalciumWaveSignal models astrocytic calcium-wave propagation across regions; GliotransmitterSignal models astrocytic release of glutamate, ATP, and D-serine; PruningSignal carries microglial requests to sever inactive connections; MyelinationSignal carries oligodendrocyte requests to update propagation delays. AstrocyteNeuron integrates local activity and emits calcium waves and gliotransmitters. MicroglialPruningNeuron tracks per-connection inactivity and emits prune requests, subject to a minimum-inactivity safeguard and a per-epoch cap to prevent over-pruning. MyelinationNeuron tracks per-connection usage and emits delay-reduction requests, subject to a baseline and minimum delay clamp.

The headline capability is genuinely new: activity-dependent per-connection propagation delay via a DelayedAxon subclass. The framework's existing Axon class is unchanged; the subclass adds a per-target delay map and a delay queue that holds signals until their release tick. Frequently-used pathways monotonically reduce in delay (myelination accelerates them); damaged pathways restore baseline delay (demyelination), but never exceed it. The microbiology is preserved in the constraint that only MicroglialPruningNeuron is authorized to sever a connection: no other module may emit a PruningSignal. This single ownership of destruction parallels the immune system's restriction of phagocytic authority to specific cell types. The module is rooted in the work of Fields on activity-dependent myelination, Gibson et al. on neuronal-activity promotion of oligodendrogenesis, Schafer et al. on microglial sculpting of postnatal circuits, and Araque et al. and Volterra and Meldolesi on astrocytic computation.

7.5 Sleep: replay, consolidation, and dreaming

The sleep module implements a circadian sleep controller, hippocampal replay, sharp-wave ripples targeting long-term memory, and REM-sleep dreaming. Four signals and four neurons are added. SleepStateSignal carries the current phase (WAKE, NREM2, NREM3, REM) and depth. ReplaySignal carries a compressed episode replay with direction (forward, reverse, or shuffled). SharpWaveRippleSignal carries a compressed burst replay to long-term memory. DreamSignal carries a REM-generated recombined episode. SleepControllerNeuron on Layer 7 advances the phase cycle WAKE through NREM2, NREM3, REM and back, gating the other sleep neurons. HippocampalReplayNeuron on Layer 3 selects high-salience recent episodes and replays them, by default in reverse direction (the biological default per Foster and Wilson) at a configurable compression ratio. SharpWaveRippleNeuron emits compressed burst replays during NREM3 only, exploiting the framework's frequency declarations to fire exactly when the controller has set NREM3 phase. REMDreamingNeuron during REM phase recombines episodes into

hypothetical novel configurations, which are routed to PlanningNeuron as candidate plans with priority equal to low and source equal to DREAM.

The safety property is non-negotiable: REM-generated plans must still pass through HarmGateNeuron before any motor execution. The DreamPlanSafetyTest ensures that an unsafe dream is rejected at the planner. The integration with the glia module is that during NREM3, MyelinationNeuron receives a consolidation-boost multiplier (default 3.0), so offline myelination accelerates during sleep, matching biology. MetaplasticityNeuron also runs at three times its normal rate during NREM3, supporting the consolidation function. The publishable claim of the module is that sleep-cycle-enabled networks exhibit lower forgetting than sleep-disabled baselines on a reference benchmark; this is treated as a CI-gated experimental claim rather than a unit-test pass. The module is rooted in the work of Wilson and McNaughton on hippocampal reactivation during sleep, Foster and Wilson on reverse replay, Buzsaki on sharp-wave-ripples, Wamsley and Stickgold on memory and dreaming, Diekelmann and Born on the memory function of sleep, and Saper et al. on hypothalamic regulation of sleep.

8. Application-Domain Modules

Beyond the autonomous-AI core and the five biological extension modules, six application-domain modules adapt the framework to specific deployment scenarios. Each is a complete reference implementation of a use case described in a matching design specification. Each module makes deliberate, documented departures from defaults — most notably, several application modules disable the affect, curiosity, or sleep modules to avoid behaviors that are inappropriate in their domain. Each module passes its own test suite and contributes to the framework's regression tests.

8.1 Brain-Computer Interfaces and Neural Prosthetics

The BCI module turns the framework into a reference platform for closed-loop brain-computer interfaces driving neural prostheses, sensory feedback, and assistive communication. It closes the loop from intracortical or electrocorticographic recording, through online decoding (Georgopoulos population vector, Kalman state-space, LFADS-style latent dynamics, speech-phoneme classification), to safety-gated stimulation and prosthetic actuation. The safety envelope is non-negotiable: every stimulation command is vetted against the Shannon charge-density criterion (default 0.5 microcoulombs per square centimeter per phase), per-phase charge limits, frequency and pulse-width bounds, charge-balance DC-accumulation limits, seizure lockout, and thermal shutdown.

Twelve domain-specific signals are added (NeuralSpikeSignal, LFPSignal, ECoGSignal, IntentSignal, StimulationCommandSignal, SensoryFeedbackSignal, ChargeAccumulationSignal, ThermalSignal, CalibrationSignal, DriftEstimateSignal, SeizureRiskSignal, AgencyLossSignal). Twenty-five domain-specific neurons span the layers from front-end signal acquisition (SpikeRecordingNeuron, SpikeSortingNeuron, LFPExtractionNeuron, ArtefactRejectionNeuron) through decoding (FiringRateEstimatorNeuron, PopulationVectorNeuron, KalmanDecoderNeuron, LatentDynamicsNeuron, SpeechPhonemeDecoderNeuron) to fusion (IntentFusionNeuron, UserStateNeuron) and adaptation (DecoderWeightNeuron, DriftTrackerNeuron, PersonalMotorLexiconNeuron). The output side covers planning (ProstheticPlanningNeuron, GripSelectionNeuron) and safety (StimulationSafetyGateNeuron, ActuatorNeuron, ChargeBalanceNeuron, SeizureWatchdogNeuron). Layer 7 supervision adds ThermalMonitorNeuron, PowerBudgetNeuron, and CalibrationSchedulerNeuron.

User ownership is built in. PersonalMotorLexiconNeuron lets the user invent personal gestures rather than imposing a vendor-defined gesture set. UserStateNeuron classifies fatigue, confusion, and distress, and the safety gate backs off stimulation when the user is not in nominal control. The integration with the embodiment module is critical: ReafferenceComparatorNeuron emits AgencyLossSignal whenever the predicted body state diverges from observed body state beyond an efference-copy tolerance (default 0.15), allowing the system to detect prosthetic failure and to provide the user with the agency-loss feedback that biological proprioception provides automatically.

8.2 Clinical Decision Support and Differential Diagnosis

The clinical module turns the framework into a Software-as-a-Medical-Device decision-support engine for differential diagnosis, treatment planning, and adverse-event detection. Medical reasoning runs on five distinct biological timescales — seconds (vitals), minutes-hours (labs), hours-days (imaging, medication response), days-weeks (treatment trajectories), years (demographics, chronic disease) — and the framework's native ProcessingFrequency(loop, epoch) is a direct structural fit. Crucially, the recommender neuron is advisory only: executable orders require an explicit physician-confirmation path outside the network, and the recommendation mode is hard-coded; the configuration setter throws on any value other than 'advisory'.

Ten domain-specific signals span vital monitoring (VitalSignal, WaveformSignal), laboratory and imaging results (LabResultSignal, ImagingFindingSignal), medications (MedicationAdminSignal), demographics (DemographicSignal), reasoning outputs (DiagnosisHypothesisSignal, TreatmentProposalSignal), and safety (AdverseEventAlertSignal, ClinicalVetoSignal). The clinical veto signal extends the framework's existing HarmVetoSignal so that all veto decisions, clinical or otherwise, travel the same audit path. Sixteen domain-specific neurons implement monitoring

(VitalMonitorNeuron with NEWS2-style guardrails, WaveformAnalysisNeuron with conservative pathologic-pattern detection, TrendDetectorNeuron), patient context (PatientContextNeuron with vulnerability factors that multiply harm thresholds, AcuityNeuron computing NEWS2 acuity), memory and diagnosis (DifferentialDiagnosisNeuron with bounded ranked posteriors, GuidelineMemoryNeuron, DrugInteractionMemoryNeuron with severity grading), planning and safety (ClinicalConsequenceModelNeuron with one-compartment pharmacokinetics and Emax pharmacodynamics, TreatmentPlanningNeuron always emitting advisory rationale, ContraindicationNeuron with default rules for beta-lactam anaphylaxis, sulfa anaphylaxis, NSAIDs in end-stage renal disease, high-dose acetaminophen in hepatic failure, and pregnancy teratogens), and the RecommendationNeuron whose mode is permanently advisory.

Patient isolation is enforced by configuration: one network instance per patient, no shared mutable state across instances, and every signal carrying a patient ID. The affect, curiosity, and sleep modules are intentionally disabled in clinical mode to avoid between-patient drift. The regulatory posture is FDA 510(k) Class II decision support, aligned with IEC 62304 software lifecycle and ISO 14971 risk management. The module's full test suite passes, including 56 tests covering vital guardrail alerting, ECG asystole and ventricular fibrillation discrimination, vulnerability scaling, Bayesian posterior skew, drug-drug interaction hazard detection, pharmacokinetic and pharmacodynamic modeling, and contraindication veto for allergy, comorbidity, pregnancy, and DDI severity-4 routes.

8.3 Cybersecurity: a biological-immune-system analogue

The cybersecurity module turns the framework into an EDR/NDR-style detection-and-response platform that borrows its architecture from the biological immune system. Innate signature detection plays the role of macrophages and neutrophils; adaptive anomaly detection plays the role of T-cells; a memory neuron plays the role of memory B-cells; a hard self-tolerance gate plays the role of thymic negative selection; and graduated response — log, alert, rate-limit, quarantine, block — mirrors inflammation staging. Quarantine is never permanent by construction: every request carries a positive duration, and an automatic lift signal fires at expiry unless independently reconfirmed.

Eleven signals are added: PacketSignal, SyscallSignal, LogEventSignal, SignatureMatchSignal, AnomalyScoreSignal, ThreatHypothesisSignal, QuarantineRequestSignal, QuarantineLiftSignal, InflammationBroadcastSignal, SelfToleranceSignal, IncidentReportSignal. Twenty-three domain-specific neurons implement ingestion (rate-limited PacketIngestNeuron, SyscallIngestNeuron, LogIngestNeuron), innate detection (SignaturePatternNeuron, ProcessBehaviourNeuron, NetworkFlowNeuron, InnateInterneuron), adaptive detection (AnomalyDetectorNeuron, EntityBehaviourBaselineNeuron with the essential property that baselines freeze during alert states so attack-window traffic is not absorbed into the baseline, BeaconingDetectorNeuron,

LateralMovementNeuron), memory (AttackMemoryNeuron, IncidentTimelineNeuron), hypothesis (ThreatHypothesisNeuron with Bayesian combination of evidence), response (ResponsePlanningNeuron with graduated bands, ResponseGateNeuron with hard allow-list and critical-asset protection, QuarantineEntityNeuron with automatic lift, RollbackNeuron), and homeostasis (AlertFatigueMonitorNeuron, ImmuneExhaustionNeuron). Hard versus soft tolerance is carefully separated: the ResponseGateNeuron owns the constructor-time hard allow-list and critical-asset list, neither of which is runtime-mutable through any signal; the InnateInterneuron owns the soft allow-list which SelfToleranceSignal can update at runtime. An attacker who compromises signal flow can at most unblock a previously-trusted pattern; they can never stand down the hard constraints.

8.4 Industrial Process Control and Digital Twins

The industrial module unifies regulatory control, supervisory optimization, planning, maintenance, and safety in a single signal-flow network. Every layer is structurally separate from the safety layer: the interlock neuron is frozen at construction time, operator override always wins for regulatory control, and every per-loop deployment mode (SHADOW, ADVISORY, AUTONOMOUS) is enforced at a dedicated safety gate. The headline operational contribution is automated, reversible oscillation management: the OscillationMonitorNeuron maps each tagged loop's autocorrelation function to a graduated intervention band — scale weights, inject inhibition, break connection, quarantine neuron — so the network can unilaterally damp a misbehaving cascade without permanent reconfiguration.

Ten domain signals span measurement and setpoints (MeasurementSignal with ISA-95 tagging and Quality, SetpointSignal with ramp rate and source), actuator commands (ActuatorCommandSignal with execute flag), alarms and interlocks (AlarmSignal with ISA-18.2 priority, InterlockSignal), supervisory and planning (DegradationSignal, EfficiencySignal, BatchStateSignal, MaintenanceWindowSignal), and operator override (OperatorOverrideSignal). Per-loop deployment mode means that a plant can run 90 percent autonomous, 9 percent advisory, 1 percent shadow simultaneously across its many control loops — a deployment posture that conventional DCS systems cannot easily express. Operator override always wins for regulatory control and is enforced by a configuration-level invariant: the setter for overrideAlwaysHonoured throws on false. The module aligns with IEC 61508 functional safety, IEC 62443 industrial cybersecurity, ISA-95 enterprise-control integration, ISA-18.2 alarm management, and 21 CFR Part 11 hash-chained audit. Out of scope by deliberate decision: replacement of certified safety PLCs, cloud-round-trip control, in-loop LLM, and uncontrolled model updates.

8.5 Swarm Robotics and Distributed Multi-Agent Coordination

The swarm module treats each agent as a full Jneopallium instance and the swarm as a network of networks. Coordination is local-first: peer observations, role awareness, stigmergic markers (digital pheromones), auction bidding, lightweight consensus, Reynolds flocking, formation keeping, Byzantine-tolerant isolation, and collective-harm aggregation all reduce to typed signals exchanged over realistic communication links. Communication is never assumed perfect: every peer-bound signal carries a linkQuality indicator in [0,1] that processors weight against. The module exists to demonstrate that the framework's INeuronNetInput abstraction — feeding one network's output into another's input — is the right primitive for multi-agent communication.

Twelve domain signals cover peer observations and state, task announcement and bidding and assignment, pheromone deposit, formation, consensus proposal and voting, swarm alerts, agent anomaly reporting, and stigmergic traces. Twenty-three domain neurons span peer sensing (PeerObservationNeuron with minimum-link-quality drop, MeshRadioNeuron), peer state (PeerStateIntegrationNeuron, RoleAwarenessNeuron), collective memory (StigmergicMemoryNeuron, TaskRegistryNeuron), coordination planning (AuctionBiddingNeuron, ConsensusParticipantNeuron with configurable k-witness quorum, FormationKeepingNeuron, FlockingNeuron), and collective safety (SwarmHarmGateNeuron, AnomalyReportNeuron with single-shot deduplication, IsolationProtocolNeuron with k-witness Byzantine tolerance and automatic lift). Layer 7 homeostasis adds SwarmDensityNeuron, BandwidthBudgetNeuron with token-bucket rate-limiting, and EnergyCoordinatorNeuron that defers tasks to higher-battery peers.

Two hard architectural constraints distinguish the module. First, k-witness Byzantine tolerance: the IsolationProtocolNeuron refuses to isolate a peer below the configured witness threshold (minimum 3, enforced by configuration setter that throws on lower values). Second, no lethal autonomy: SwarmConfig.isLawsEnabled() returns false and is fixed; no setter is exposed. These two constraints, together with the affect module being disabled to avoid emotional-mimicry feedback loops, give the swarm module a clean safety posture for civilian search-and-rescue, inspection, warehouse logistics, and agricultural applications.

8.6 Adaptive Tutoring and Student Cognitive State Modeling

The tutoring module is a domain-specific application for intelligent tutoring systems. It composes the affect, curiosity, and sleep modules: spaced repetition is driven by HippocampalReplayNeuron during simulated sleep cycles, moment-to-moment cognitive state by FlowStateNeuron, concept mastery by Bayesian Knowledge Tracing (Corbett and Anderson 1995), and the pedagogical planner by a Vygotsky-style zone-of-proximal-development scoring

function. Ten domain signals cover item presentation and response, engagement and affect observation, mastery updates, content recommendation, hints and scaffolding, review schedule, and intervention. Fifteen domain neurons span observation (ResponseObserverNeuron, EngagementSensorNeuron with multi-modal fusion, AffectObserverNeuron), state (FlowStateNeuron classifying flow, boredom, frustration, and overload; FatigueNeuron), mastery and curriculum (ConceptMasteryNeuron, PrerequisiteGraphNeuron, ForgettingCurveNeuron implementing Ebbinghaus and SM-2), planning (ZPDPlanningNeuron, HintGenerationNeuron with graduated hint ladders, ScaffoldingNeuron), action selection (ContentSelectionNeuron, PacingNeuron), and Layer 7 wellbeing (WellbeingGuardNeuron with escalating intervention from encouragement to break to redirect to escalate-to-human, FairnessNeuron with accommodation-aware latency penalty gate).

Three design commitments distinguish the module. First, formative not summative: no high-stakes grading, only content and pacing adjustment. Second, safety and fairness first: WellbeingGuardNeuron specializes HarmGateNeuron, FairnessNeuron specializes EthicalPriorityNeuron but never modifies its weights. Third, instance-per-learner privacy posture, with no cross-learner data leakage. Hint text and worked examples can be generated by the LLM module subject to verification, providing pedagogical sophistication without trusting the LLM as ground truth about subject content.

9. Implementation, Deployment Modes, and Tooling

9.1 Java as the implementation language

Java was selected as the implementation language for three reasons articulated in the original IJSR paper. First, Java's interface system supports the framework's central pattern of multi-receptor neurons by allowing a single class to implement many sub-interfaces of INeuron and to receive a distinct processor for each. Second, Java's generics give compile-time type safety to the parameterized ISignalProcessor<S, N> abstraction, catching at compile time the kind of errors that would surface only at runtime in dynamically-typed languages. Third, Java's runtime class loading allows user-defined neuron and signal classes to be loaded from a JAR specified at startup, which means researchers can develop their models independently of the framework's release cycle. The framework targets Java 11 and above, builds with Maven, and is organized as three modules: worker (the bulk of neuron and signal implementations), master (cluster coordination), and common (shared types). New extension code goes under com.rakovpublic.jneuropallium.worker.net.neuron.impl.<moduleName> and the matching signals package.

9.2 Three deployment modes

Jneopallium runs in three deployment modes. Local mode executes the entire network on a single JVM with no inter-process communication overhead, appropriate for development, single-host robotics, and embedded applications. Cluster HTTP mode distributes the network across multiple JVMs communicating over HTTP REST endpoints, appropriate for moderate-throughput applications where the operational simplicity of HTTP outweighs its protocol overhead. Cluster gRPC mode uses gRPC for inter-node communication and supports deployment to FPGAs, which is the framework's intended path for hardware-accelerated low-latency cognitive systems. The choice of deployment mode is a configuration parameter; the same neuron and signal definitions run unmodified across all three modes. Multiple input sources and output aggregators are supported in each mode, with each I/O source declaring its own default ProcessingFrequency that is honored by the scheduler.

9.3 Configuration and structure files

A Jneopallium model is specified by three artifacts: a JAR containing the user-defined signal, neuron, and processor classes; a network-structure file (JSON) describing layer sizes, neuron-type probabilities, neighboring rules, and connection-generation strategy; and a configuration file (Java properties) specifying file-system implementation, signal serializers, studying-algorithm implementation, input and output paths, and per-module configuration sections. The configuration file uses YAML-style nested sections for module-specific options: each of the fifteen modules contributes a top-level section with its own enabled flag, parameters, and policy options. The principle that every new module defaults to disabled means that minimal configurations remain valid as the framework grows.

9.4 Testing and quality assurance

Every module ships with a unit test suite under worker/src/test/java/.../<moduleName>. The tests cover signal ProcessingFrequency correctness and copy round-trip semantics; per-neuron behavior including edge cases; per-processor smoke tests that walk realistic signal paths end-to-end; configuration validation including the deliberate throw-on-illegal-value invariants; and module-level integration tests. The current repository contains 23 test classes covering the affect, BCI, clinical, curiosity, cycle-processing, embodiment, glia, industrial, LLM, security, sleep, swarm, and tutoring modules in addition to the core framework's neuron, axon, dendrites, JSON helper, neuron-runner-service, and input-strategy tests. Cross-module regression is enforced by a reference test configuration (alfaTestAndGettingStarted) whose spike-train output must be byte-identical before and after each module is added when that module is disabled. Per-module performance overhead is tracked: a single module enabled must not degrade tick throughput by

more than 15 percent on the reference configuration, and all modules enabled together must not degrade by more than 60 percent.

9.5 Documentation and the use-case track

Each module is documented in two places: a Javadoc-rich set of source files with biological citations on every public class and a top-level Markdown module document under docs/modules/. The autonomous-AI architecture and the LLM integration each have their own companion paper. Each application-domain module is paired with a use-case design specification that articulates the domain's problem framing, mapping to the core framework, domain-specific signal and neuron design, configuration, validation criteria, and out-of-scope items. The use-case specifications serve a dual purpose: they document what the module does and they provide a structured template for adding new application domains.

10. Comparison with Competing Frameworks

10.1 Mapping the competitive landscape

Jneopallium occupies a specific niche in a crowded space. We can map this space along two axes: the level of biological detail preserved (low to high) and the engineering practicality (research-only to production-deployable). Frameworks at the low-detail end include TensorFlow, PyTorch, JAX, MXNet, Caffe, and the dozens of higher-level libraries that build on them. These are production-deployable, very high engineering practicality, but biological detail is essentially zero — neurons are matrix rows, synapses are scalars, training is gradient descent. At the high-detail end are NEURON Simulator, CoreNeuron, NEST, Brian2, GENESIS, and Arbor. These preserve substantial biophysical realism but are oriented toward research and rarely cross over into engineering deployment. Spiking-neural-network frameworks — Nengo, BindsNET, snnTorch, Lava (Intel Loihi) — sit in the middle of the detail axis and are increasingly engineering-oriented as neuromorphic hardware matures. Jneopallium sits adjacent to the SNN cluster but with a different emphasis: rather than committing to spiking dynamics throughout, it allows the level of detail to vary across neuron types within the same network.

10.2 NEURON Simulator and CoreNeuron

The NEURON Simulator (Hines and Carnevale, 1989) is the de facto standard for biophysically-detailed simulation of single neurons and small circuits. Its Hodgkin-Huxley channel formalism on multi-compartmental morphologies is without peer for studies that require this level of realism: dendritic spike initiation, channelopathies, spike-back-propagation, and so on.

CoreNeuron (Blue Brain Project, 2019) is a high-performance simulation engine optimized for NEURON models on supercomputing hardware, supporting the Blue-Brain-style large-scale circuit reconstructions. The principal difference between Jneopallium and NEURON/CoreNeuron is the level-of-detail commitment. NEURON's commitment to high biophysical detail is appropriate for its target audience (computational neuroscience research) but is overkill for engineering applications and prevents lighter-weight modeling. Jneopallium's commitment is to user-chosen detail per neuron type, which means a single network can contain a few high-detail neurons in regions where the detail matters and many lighter-weight neurons elsewhere — a heterogeneity that conventional NEURON usage does not support.

10.3 NEST and Brian2

NEST (Gewaltig and Diesmann, 2007) is a simulator for large networks of point neurons (typically integrate-and-fire variants) with biological spiking dynamics. It scales to millions of neurons via MPI and is the simulator of choice for the Human Brain Project. Brian2 (Goodman and Brette, 2008) emphasizes equation-driven model specification, allowing researchers to define neuron models by their differential equations directly in Python. Both are targeted at computational neuroscience and offer minimal direct support for engineering applications such as robotics control loops, real-time sensor input, or safety-critical decision making. Where Jneopallium provides a harm-discriminator module, a loop-prevention subsystem, and transparency-logging integration, NEST and Brian2 provide simulation kernels and leave the higher-level engineering concerns to the user. The direction of trade-off is appropriate for each tool's audience but creates a real gap that Jneopallium addresses.

10.4 Nengo and the spiking-neural-network frameworks

Nengo (Eliasmith and Anderson, 2003), built on the Neural Engineering Framework, demonstrates that SNNs can implement nontrivial cognitive functions within biological energy and spike-rate constraints. Spaun (Eliasmith et al., 2012), built in Nengo, is the most ambitious cognitive-SNN model to date, performing a battery of cognitive tasks with 2.5 million spiking neurons. BindsNET, snnTorch, SpikingJelly, and Intel's Lava are more recent frameworks that bring SNN modeling into the PyTorch / TensorFlow ecosystem. The technical differences from Jneopallium are: SNN frameworks commit to spike-based communication throughout, while Jneopallium permits any typed signal; SNN frameworks tend to use uniform integrate-and-fire neurons throughout, while Jneopallium permits arbitrary neuron heterogeneity; and SNN frameworks generally lack the multi-timescale fast/slow loop construct that gives Jneopallium its natural fit to multi-rate biological signaling. The philosophical similarity is large, however: both Nengo and Jneopallium aim to use biological structure as a source of engineering robustness rather than as a research artifact.

10.5 Reinforcement-learning frameworks

Modern reinforcement-learning frameworks — Stable-Baselines3, Ray RLlib, Tianshou, CleanRL — are designed for training policies via gradient methods on Markov decision processes. They have become the dominant platforms for agent-based learning research and increasingly for robotics applications. Their relationship to Jneopallium is complementary rather than competitive. An RL agent can be embedded as one neuron-like component within a Jneopallium network, providing the policy-learning capability that the framework's neuron-level Hebbian and STDP learning is not designed to deliver at scale. Conversely, Jneopallium provides the safety, transparency, and biological scaffolding that pure RL frameworks lack: a planner whose outputs are gated by a harm discriminator, a forward model that can be queried during planning, an interoceptive module that can flag agent fatigue or dysregulation, and a transparency log that records every decision.

10.6 Comparative summary

Framework	Biological detail	Engineering practicality	Multi-detail support	Safety architecture	Multi-timescale	Modular composition
Jneopallium	User-selected	High	Yes	Built-in (harm discriminator)	Yes (5 timescales)	Yes (INeuronNetInput)
TensorFlow / PyTorch	Minimal	Very high	No	External tooling	No	Limited
NEURON / CoreNeuron	Very high (HH)	Research-oriented	No	None	No	Limited
NEST	High (point neurons)	Research-oriented	Limited	None	Limited	Limited
Brian2	High (equation-driven)	Research-oriented	Limited	None	Limited	Limited
Nengo	Medium (SNN)	Medium-high	No	Limited	Limited	Limited
RL frameworks (RLlib etc.)	Minimal	Very high	No	External tooling	No	Yes

The summary table reflects design priorities rather than absolute superiority. Each framework excels at its target domain. Jneopallium's claim is to a specific niche — biologically structured,

safety-architectured, multi-timescale, modularly composable autonomous systems — that no other framework currently fills directly.

11. Economic Impact

11.1 The framework's monetization paths

The original IJSR paper identified three monetization paths for the framework itself: building application-specific models for client products; providing hosted Jneopallium-as-a-Service the way cloud providers offer Apache Spark today; and FPGA-based acceleration for performance-critical deployments. The BSD 3-Clause license permits all three paths and additionally permits downstream proprietary forks. The framework's value to its operators thus decomposes into license-free use, integrator-services revenue, and hosting or acceleration revenue.

11.2 Application-domain impact estimates

More important than the framework's own monetization is the economic impact of the application-domain modules deployed in their respective sectors. We summarize each.

11.2.1 Brain-computer interfaces and neural prosthetics

The global neural-prosthetics market is currently estimated in the low-billions of US dollars annually and is projected by industry analysts to grow at compound annual rates exceeding 10 percent through the next decade as intracortical and electrocorticographic recording matures. The BCI module's specific contribution is the integration of decoder, effector driver, sensory feedback, and safety-gating in one auditable network. Per-patient cost savings come from reduced calibration time (the DriftTrackerNeuron and DecoderWeightNeuron support online recalibration), reduced surgical revisions (the StimulationSafetyGateNeuron's charge-density gating reduces electrode damage), and reduced training burden (the PersonalMotorLexiconNeuron lets patients invent gestures rather than relearning vendor-prescribed sets). The module's open-source posture also reduces vendor lock-in for clinical centers, which has historically been a barrier to clinical adoption of BCI technology.

11.2.2 Clinical decision support

Healthcare informatics is among the largest software markets globally, and clinical decision support is one of the highest-leverage informatics subdomains. Adverse drug events alone are estimated to cost healthcare systems several hundred billion dollars annually worldwide; the clinical module's drug-interaction memory and contraindication veto target precisely this loss class. The advisory-only posture (recommendation mode hard-coded, physician confirmation

required) is appropriate for FDA 510(k) Class II deployment and avoids the regulatory and liability burden of Class III autonomous treatment systems. The module's value lies less in replacing physician judgment than in providing physicians with structured access to guideline-derived recommendations under a verified pharmacokinetic and pharmacodynamic model — a capability that conventional electronic health record systems do not provide.

11.2.3 Cybersecurity

The endpoint detection-and-response and network detection-and-response markets are large and growing. The cybersecurity module's biological-immune system architecture has two specific operational advantages over conventional EDR/NDR. First, the principled separation of innate (signature) and adaptive (anomaly) detection with shared memory addresses the longstanding problem of alert fatigue: the AlertFatigueMonitorNeuron multiplier rises monotonically with false-positive rate, automatically tightening anomaly thresholds without operator intervention. Second, the never-permanent quarantine pattern addresses a real operational pain point: traditional security automation tools that quarantine indefinitely create operational risk through accumulated blocked legitimate traffic, while the IncidentTimelineNeuron and the automatic lift signal ensure that quarantines are explicitly justified and time-bounded.

11.2.4 Industrial process control

Industrial process control is a multi-hundred-billion-dollar market dominated by established DCS and SCADA vendors. The industrial module is unlikely to displace these in their core regulatory-control function, but it addresses a real and unsolved operational problem: the oscillation of cascaded control loops, which costs continuous-process plants substantial efficiency losses annually. The OscillationMonitorNeuron's automated, reversible, ACF-based intervention is a direct economic contribution. The per-loop deployment mode (SHADOW, ADVISORY, AUTONOMOUS) lowers the adoption barrier: a plant can run the framework in shadow mode on 99 percent of loops while transitioning a few high-confidence loops to autonomous, generating measured efficiency wins without committing the entire plant to a new control philosophy.

11.2.5 Swarm robotics

Swarm robotics is an emerging market with substantial growth in warehouse logistics, agricultural inspection, infrastructure monitoring, and search-and-rescue. The swarm module's specific contribution is the principled treatment of communication imperfection (every peer signal carries link quality), the k-witness Byzantine-tolerant isolation protocol, and the explicit no-LAWS hard constraint. The first two enable graceful degradation as agents drop out or become adversarial; the third positions the module for civilian deployment without legal or

ethical complications associated with lethal autonomous weapons. Open-source provenance and clean safety posture lower the barrier for civilian operators (logistics companies, agricultural cooperatives, public safety agencies) to deploy swarms without licensing specialized military-derived code.

11.2.6 Adaptive tutoring

EdTech is among the largest and most fragmented software markets globally. The tutoring module's specific contribution is the principled multi-timescale modeling of student state — moment-to-moment flow versus session-scale engagement versus week-scale forgetting — that conventional adaptive learning platforms typically reduce to a single Bayesian Knowledge Tracing posterior. The wellbeing guard (escalating from encouragement through break to redirect to escalate-to-human) addresses a real ethical concern in always-available digital tutoring: a system that pushes a frustrated learner indefinitely is harmful, regardless of its pedagogical intent. The module's instance-per-learner privacy posture is appropriate for K-12 and higher education deployments under FERPA, GDPR, and equivalent regimes.

11.3 Aggregate impact

Estimating aggregate economic impact across these six domains requires assumptions about adoption rate, substitution versus addition, and the fraction of value the framework captures versus the fraction captured by integrators and operators. We do not attempt a formal estimate. We do observe, however, that each application domain has on the order of tens of billions of dollars in addressable annual spend, and that the framework's open-source posture and modular composition lower adoption barriers in each. The cumulative addressable market is therefore substantial, with the dominant uncertainty lying in adoption rate rather than in addressable-market size.

12. Limitations and Future Work

The framework has limitations that should be honestly acknowledged. First, the Java implementation imposes a JVM-startup latency and a baseline memory overhead that make it less suitable than C++ or Rust for the smallest embedded deployments. The cluster gRPC mode targeting FPGAs is the planned response, but the FPGA back-end is not yet production-mature. Second, the framework's learning rules — Hebbian, STDP, metaplasticity — are local and biologically plausible but are not as efficient as backpropagation for training deep supervised classifiers. Where deep classification is the bottleneck, Jneopallium networks can incorporate a PyTorch model as a black-box neuron, but this is a workaround rather than a first-class solution. The longer-term research direction is biologically-plausible deep learning rules — feedback

alignment, predictive-coding-based learning, and equilibrium propagation — that operate within Jneopallium's local-update discipline.

Third, the harm-discriminator module's consequence model depends on the quality of the world model maintained by ConsequenceModelNeuron. A flawed world model produces flawed consequence projections, regardless of how structurally sound the harm-evaluation pipeline is. The framework provides no guarantee of world-model accuracy; this remains an empirical and deployment-specific concern. Fourth, the framework's per-tick performance overhead grows with the number of enabled modules. The current target — no more than 60 percent throughput degradation with all modules enabled — is documented but is a real cost compared to a stripped-down configuration. Fifth, the LLM verification mechanism relies on the system's internal model quality: if the internal model is itself flawed, verification may incorrectly reject correct LLM information. A multi-LLM consensus mechanism is on the future-work list.

Future work spans several directions. On the framework side: a JIT-compiled fast path for hot neurons; a GPU back-end for parallelizable layer computations; richer support for differentiable learning sub-networks embedded within otherwise local networks; and continued maturation of the FPGA back-end. On the modules side: an emotion-recognition module to extend the affect submodule with external observation rather than only interoceptive sources; a language-grounding module that grounds linguistic concepts in the embodiment module's body schema; and an explanation module that consumes TransparencyLogSignals and produces human-readable summaries. On the application-domain side: a financial-trading module with the same advisory-only and consequence-model discipline as the clinical module; an agriculture module integrating the swarm and embodiment modules for field robotics; and a creative-arts module exercising the curiosity and sleep modules in compositional and generative settings.

13. Conclusion

Jneopallium has matured from a small Java framework introduced in 2024 into a comprehensive open-source platform for biologically-grounded autonomous AI. Its core abstractions — typed signals with declared frequencies, neurons with multiple receptors, stateless processors parameterized on signal and neuron interfaces, dual fast/slow processing loops, structurally inviolable hard constraints — have proved sufficient to express not only an autonomous-AI architecture with a consequence-model harm discriminator and a loop-prevention subsystem, but also five biological extension modules (affect, embodiment, curiosity, glia, sleep), an optional LLM advisory subsystem, and six application-domain implementations spanning brain-computer interfaces, clinical decision support, cybersecurity, industrial process control, swarm robotics, and adaptive tutoring.

Three contributions are worth highlighting in summary. First, the framework demonstrates that a single set of core abstractions can support the full range of biological detail from perceptron-equivalent simplicity to limbic-system-and-glia complexity, with the level of detail chosen per neuron type rather than imposed globally. Second, the framework demonstrates that safety-architectured AI — harm discriminator, hard constraints, transparency logging, never-permanent intervention — can be implemented as an integral part of the cognitive substrate rather than as an external filter, with concrete benefits in auditability, learnability, and graceful degradation. Third, the framework demonstrates that biology and engineering are not necessarily in tension: the multi-timescale, multi-receptor, modularly-composable structure that biology has evolved is precisely the structure that engineering autonomous systems require.

The historical lineage from Hebb to the present has produced enormous successes — in deep learning, in computational neuroscience, in spiking neural networks, in reinforcement learning — but it has also produced a gap where engineering practicality, biological structure, and safety architecture should meet. Jneopallium is one attempt to fill that gap. The framework is open-source, biologically-cited, modular, optional, and audit-friendly. We hope that the community will use, extend, criticize, and improve it.

References

Araque, A., Carmignoto, G., Haydon, P.G., Oliet, S.H.R., Robitaille, R., Volterra, A. (2014). Gliotransmitters travel in time and space. Neuron 81(4), 728-739.

Buzsaki, G. (2015). Hippocampal sharp wave-ripple: A cognitive biomarker for episodic memory and planning. Hippocampus 25(10), 1073-1188.

Cogan, S.F. (2008). Neural stimulation and recording electrodes. Annual Review of Biomedical Engineering 10, 275-309.

Corbett, A.T., Anderson, J.R. (1995). Knowledge tracing: Modeling the acquisition of procedural knowledge. User Modeling and User-Adapted Interaction 4, 253-278.

Craig, A.D. (2002). How do you feel? Interoception and the anterior insula. Nature Reviews Neuroscience 3, 655-666.

Csikszentmihalyi, M. (1990). Flow: The Psychology of Optimal Experience. Harper and Row.

Damasio, A. (1994). Descartes' Error: Emotion, Reason, and the Human Brain. Putnam.

De Castro, L.N., Timmis, J. (2002). Artificial Immune Systems: A New Computational Intelligence Approach. Springer.

Diekelmann, S., Born, J. (2010). The memory function of sleep. Nature Reviews Neuroscience 11, 114-126.

Eliasmith, C., Anderson, C.H. (2003). Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems. MIT Press.

Eliasmith, C., Stewart, T.C., Choo, X., Bekolay, T., DeWolf, T., Tang, Y., Rasmussen, D. (2012). A large-scale model of the functioning brain. Science 338(6111), 1202-1205.

Farley, B., Clark, W.A. (1954). Simulation of self-organizing systems by digital computer. IRE Transactions on Information Theory 4(4), 76-84.

Fields, R.D. (2015). A new mechanism of nervous system plasticity: activity-dependent myelination. Nature Reviews Neuroscience 16(12), 756-767.

Flesher, S.N., Downey, J.E., Weiss, J.M., et al. (2021). A brain-computer interface that evokes tactile sensations improves robotic arm control. Science 372, 831-836.

Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R. (1994). Self-nonself discrimination in a computer. IEEE Symposium on Security and Privacy.

Foster, D.J., Wilson, M.A. (2006). Reverse replay of behavioural sequences in hippocampal place cells during the awake state. Nature 440, 680-683.

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

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics 36(4), 193-202.

Furber, S.B., Galluppi, F., Temple, S., Plana, L.A. (2014). The SpiNNaker Project. Proceedings of the IEEE 102(5), 652-665.

Georgopoulos, A.P., Schwartz, A.B., Kettner, R.E. (1986). Neuronal population coding of movement direction. Science 233, 1416-1419.

Gewaltig, M.-O., Diesmann, M. (2007). NEST (Neural Simulation Tool). Scholarpedia 2(4), 1430.

Gibson, E.M., Purger, D., Mount, C.W., et al. (2014). Neuronal activity promotes oligodendrogenesis and adaptive myelination in the mammalian brain. Science 344(6183), 1252304.

Goodman, D.F.M., Brette, R. (2008). Brian: a simulator for spiking neural networks in Python. Frontiers in Neuroinformatics 2, 5.

Hebb, D.O. (1949). The Organization of Behavior: A Neuropsychological Theory. Wiley.

Hines, M.L., Carnevale, N.T. (1997). The NEURON simulation environment. Neural Computation 9(6), 1179-1209.

Hochberg, L.R., Bacher, D., Jarosiewicz, B., et al. (2012). Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. Nature 485, 372-375.

Hubel, D.H., Wiesel, T.N. (1968). Receptive fields and functional architecture of monkey striate cortex. Journal of Physiology 195(1), 215-243.

Klyubin, A.S., Polani, D., Nehaniv, C.L. (2005). Empowerment: a universal agent-centric measure of control. IEEE Congress on Evolutionary Computation.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. Biological Cybernetics 43(1), 59-69.

LeDoux, J.E. (1998). The Emotional Brain: The Mysterious Underpinnings of Emotional Life. Simon and Schuster.

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. NeurIPS 33.

Lisman, J.E., Grace, A.A. (2005). The hippocampal-VTA loop: controlling the entry of information into long-term memory. Neuron 46(5), 703-713.

Maravita, A., Iriki, A. (2004). Tools for the body (schema). Trends in Cognitive Sciences 8(2), 79-86.

Miller, G.A. (1956). The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychological Review 63(2), 81-97.

Moses, D.A., Metzger, S.L., Liu, J.R., et al. (2021). Neuroprosthesis for decoding speech in a paralyzed person with anarthria. New England Journal of Medicine 385, 217-227.

Nygard, M. (2018). Release It! Design and Deploy Production-Ready Software, 2nd ed. Pragmatic Bookshelf.

Oudeyer, P.-Y., Kaplan, F. (2007). What is intrinsic motivation? A typology of computational approaches. Frontiers in Neurorobotics 1, 6.

Pandarinath, C., O'Shea, D.J., Collins, J., et al. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. Nature Methods 15, 805-815.

Pathak, D., Agrawal, P., Efros, A.A., Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. International Conference on Machine Learning.

Rakovskyi, D. (2024). Framework for natural neuron network modeling: the Jneopallium approach. International Journal of Science and Research 13(7), 284-286. DOI: 10.21275/SR24703042047.

Rakovskyi, D. (2024). Biologically-inspired autonomous AI architecture: neuron types, signal processors, loop prevention, and human-harm discriminator. Jneopallium framework documentation.

Rakovskyi, D. (2025). Integrating large language models as an optional external knowledge base into the Jneopallium natural neuron network framework. Jneopallium framework documentation.

Reynolds, C.W. (1987). Flocks, herds, and schools: a distributed behavioral model. ACM SIGGRAPH Computer Graphics 21(4), 25-34.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review 65(6), 386-408.

Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). Learning representations by back-propagating errors. Nature 323(6088), 533-536.

Russell, J.A. (1980). A circumplex model of affect. Journal of Personality and Social Psychology 39(6), 1161-1178.

Saper, C.B., Scammell, T.E., Lu, J. (2005). Hypothalamic regulation of sleep and circadian rhythms. Nature 437, 1257-1263.

Schafer, D.P., Lehrman, E.K., Kautzman, A.G., et al. (2012). Microglia sculpt postnatal neural circuits in an activity and complement-dependent manner. Neuron 74(4), 691-705.

Schultz, W., Dayan, P., Montague, P.R. (1997). A neural substrate of prediction and reward. Science 275(5306), 1593-1599.

Shannon, R.V. (1992). A model of safe levels for electrical stimulation. IEEE Transactions on Biomedical Engineering 39, 424-426.

Song, S., Miller, K.D., Abbott, L.F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. Nature Neuroscience 3(9), 919-926.

Sutton, R.T., Pincock, D., Baumgart, D.C., Sadowski, D.C., Fedorak, R.N., Kroeker, K.I. (2020). An overview of clinical decision support systems. npj Digital Medicine 3, 17.

Velliste, M., Perel, S., Spalding, M.C., Whitford, A.S., Schwartz, A.B. (2008). Cortical control of a prosthetic arm for self-feeding. Nature 453, 1098-1101.

Volterra, A., Meldolesi, J. (2005). Astrocytes, from brain glue to communication elements: the revolution continues. Nature Reviews Neuroscience 6, 626-640.

Von der Malsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. Kybernetik 14(2), 85-100.

Vygotsky, L.S. (1978). Mind in Society: The Development of Higher Psychological Processes. Harvard University Press.

Wamsley, E.J., Stickgold, R. (2011). Memory, sleep, and dreaming: experiencing consolidation. Sleep Medicine Clinics 6(1), 97-108.

Willett, F.R., Kunz, E.M., Fan, C., et al. (2023). A high-performance speech neuroprosthesis. Nature 620, 1031-1036.

Wilson, M.A., McNaughton, B.L. (1994). Reactivation of hippocampal ensemble memories during sleep. Science 265(5172), 676-679.

Wolpert, D.M., Miall, R.C., Kawato, M. (1998). Internal models in the cerebellum. Trends in Cognitive Sciences 2(9), 338-347.

Wu, W., Gao, Y., Bienenstock, E., Donoghue, J.P., Black, M.J. (2006). Bayesian population decoding of motor cortical activity using a Kalman filter. Neural Computation 18(1), 80-118.