

## APPENDIX

### A Biological Neurodynamic Models

**Model for neuron** Neurodynamic models require neuron models to describe the dynamics of neurons, *i.e.*, signal firing. Multiple models for neurons have been proposed; among them, the Leaky Integrate-and-Fire (LIF) neuron Gerstner et al. [2014] has been proved to be a biologically realistic and computationally feasible one. The state of LIF neurons can be expressed as:

$$C_m \frac{dv(t)}{dt} = -\frac{C_m}{\tau_m} [v(t) - v(rest)] + i_e(t) + i_i(t) + I_{offset} + I_{injection}(t), \quad (1)$$

$$\text{if } V(t) > V_{th}, V_t \leftarrow V_{reset}, \quad (2)$$

where  $C_m$  is the neuron capacity,  $v(t)$  is the membrane voltage,  $\tau_m$  is the membrane time constant,  $v(rest)$  is the resistant voltage,  $i_e(t)$  and  $i_i(t)$  are the excitatory and inhibitory input currents, respectively, and  $I_{offset}$  and  $I_{injection}(t)$  are the constant input currents. If the membrane voltage  $v(t)$  reaches the threshold  $V_{th}$ , the neuron will issue a signal, and  $v(t)$  will be held at  $v(rest)$  for a refractory period  $\tau_{ref}$ . Once this refractory period ends, the neuron follows this expression until it issues the next signal again.

**Model for synapse** In addition to neurons, the connections between them, *i.e.*, synapses, are crucial in biological neural networks. Gerstner et al. [2014] Synapses transmit electrical signals from presynaptic to postsynaptic neurons, it defines how signals fired by neurons affect the membrane voltage of a postsynaptic neuron. In the model for synapses, it has the impulse signal in presynaptic neurons as the input, and the postsynaptic current as the output. The postsynaptic current (PSC) can be expressed as:

$$I_{syn} = g(t)(V_{post} - E_{syn}), \quad (3)$$

where  $I_{syn}$  is the postsynaptic current,  $V_{post}$  is the postsynaptic voltage,  $E_{syn}$  is the reversal potential of the synapse; its value determines whether a synapse is either excitatory or inhibitory, and  $g$  is the conductance on synapses.

In phenomenological models, *i.e.*, synapse is modelled based on the postsynaptic current, rather than the dynamical features of the ion channels, the time-dependence of  $g(t)$  can be expressed as:

$$g_t = \bar{g} \sum_{t^{(f)}} s(t - t^{(f)}), \quad (4)$$

where  $\bar{g}$  is a constant describing the synapse weight,  $t^{(f)}$  denotes the moment that the signal arrives at the synapse,  $s(t - t^{(f)})$  is a time-dependent function that describes the effect of signal firing on  $g_t$ . Using an exponential decay to model the synapses, we obtain:

$$s(t) = e^{-t/\tau} H(t), \quad (5)$$

where  $\tau$  is a temporal constant, and  $H(t)$  is the heaviside function. the  $g(t)$  differential equation of  $g_t$ , with respect to the current time is therefore:

$$\frac{dg}{dt} = -\frac{g}{\tau} + \bar{g} \sum_{t^{(f)}} \delta(t - t^{(f)}). \quad (6)$$

Where  $\delta$  function is zero anywhere but  $t = 0$ , and  $t$  is the current time.

The parameters for neuron and synapse models employed in the work described in this paper are summarised in table 1.

Table 1: Parameters for the Biological Neurodynamic Model.

Parameter	Value	Unit
Resting potential ( $V_{\text{rest}}$ )	-60	mV
Reset potential ( $V_{\text{reset}}$ )	-60	mV
Initial membrane potential ( $V_0$ )	-55	mV
Threshold potential ( $V_{\text{th}}$ )	-50	mV
Membrane capacitance ( $C_m$ )	1	pF
Membrane time constant ( $\tau_m$ )	20	ms
Synaptic time constant ( $\tau_{\text{syn}}$ )	5	ms

## B Biological Neurodynamic Networks

**Excitatory–Inhibitory Balance Models** Excitatory–inhibitory (E–I) balance Vreeswijk and Sompolinsky [1996] is a hallmark of cortical microcircuitry, whereby excitatory and inhibitory synaptic inputs dynamically counterbalance each other to maintain stable yet flexible neural activity. This principle, rooted in electrophysiological recordings from the neocortex and hippocampus, has inspired a class of computational models that preserve this balance at both the single-neuron and population level. These E–I balanced networks typically consist of spiking neurons or rate-based models partitioned into excitatory and inhibitory subpopulations, with constrained synaptic weights (e.g., Dale’s law Strata and Harvey [1999]) and recurrent connectivity that gives rise to asynchronous irregular activity, criticality, and efficient coding. The E–I architecture has been shown to support robust computations, including working memory, pattern decorrelation, and gain control, by leveraging biologically plausible dynamics rather than task-specific training. Notably, such models often operate without supervised gradient-based optimization, instead relying on biologically motivated plasticity rules or structured connectivity to enable function.

The parameters for the biological neurodynamic network (BNN) employed in the work described in this paper are summarised in table 2.

Table 2: Parameters for the Biological Neurodynamic Model.

Parameter	Value	Unit
Simulation duration	100	timestep
Simulation step size	0.1	ms
Number of neurons	10E + 10I + number of input&output	-
Learnable Params	synaptic weights and delays, noisy input	-

## C Spiking Neural Networks

Spiking neural networks (SNNs) Maass [1997], Maass and Markram [2004] emulate the dynamics of real neurons by encoding and transmitting information via discrete spikes. Neurons in SNNs integrate synaptic inputs over time and emit spikes when membrane potentials exceed threshold, introducing an inherent temporal dimension and nonlinearity that makes them both biologically plausible and computationally distinct. Unlike MLPs, where activations are continuously differentiable, SNNs rely on non-differentiable events, posing challenges for standard optimization methods and prompting the development of surrogate gradients, spike-based learning rules, and biologically inspired plasticity mechanisms. Furthermore, the asynchronous and sparse firing nature of SNNs facilitates event-driven computation, offering potential gains in energy efficiency when deployed on neuromorphic hardware. Thus, while MLPs excel in data-rich, high-throughput regimes with dense numerical representations, SNNs offer a promising alternative for real-time, low-power, and biologically grounded computation, particularly in scenarios demanding temporal precision and structural interpretability.

## D Artificial Neural Networks

Alongside our proposed biological neurodynamic networks (BNNs), the following artificial neural networks LeCun et al. [2015] are considered in the work described in this paper.

**Multilayer perceptrons** Multilayer perceptrons (MLPs) Hornik et al. [1989] are feedforward neural networks that compute deterministic, continuous-valued transformations through stacked layers of weighted summations and pointwise nonlinearities. As universal function approximators, MLPs form the backbone of modern deep learning, enabling supervised and unsupervised learning across vision, language, and control domains. Their success is rooted in architectural simplicity, differentiability, and compatibility with efficient gradient-based training methods such as backpropagation. However, MLPs operate in discrete time with synchronous updates and dense, analog activations—features that stand in sharp contrast to the event-driven, temporally sparse computations observed in biological neural circuits.

**Recurrent Neural Networks** Recurrent neural networks (RNNs) Rumelhart et al. [1986] are a foundational class of artificial neural models developed to handle sequential and temporally structured data. Unlike feedforward architectures, RNNs maintain internal state through recurrent connections, allowing them to capture dynamic dependencies across time. Variants such as Long Short-Term Memory (LSTM) Hochreiter and Schmidhuber [1997] networks and Gated Recurrent Units (GRUs) were introduced to overcome vanishing gradient issues and have since become standard tools in natural language processing, time series forecasting, and reinforcement learning. While RNNs exhibit impressive performance in engineering tasks, they lack constraints from biological connectivity and synaptic dynamics. Their recurrent activity arises from parameterized weight matrices and nonlinear activations, rather than the structured E–I coupling observed in cortical networks. As such, RNNs prioritize trainability and function approximation over mechanistic interpretability. Efforts to bridge this gap have emerged through hybrid models, such as balanced RNNs or spiking RNNs with biologically inspired constraints, but a fundamental divergence remains: E–I balance models are grounded in the physical dynamics of neural tissue, while RNNs abstract away biological detail to maximize learning flexibility.

The key differences between MLP, SNN and our BNN are summarised in table 3.

Table 3: Comparison of different network architectures.

Network Architecture	Artificial Neural Network	Spiking Neural Network	Biological Neurodynamic Network
Basic Unit	Weighted sum + nonlinear activation	integrate-and-fire	LIF + Synaptic dynamics
Neuronal Dynamics	Static	Temporal integration	Membrane potential + conductance dynamics
Information Encoding	Continuous values	Spike trains	Continuous dynamical state
Computational Efficiency	High (GPU friendly)	Medium (requires event-driven framework)	Low to Medium (complex simulation)
Application Domain	Image recognition, NLP	Brain-inspired computation	Brain modeling, biophysical simulations

## E Simulation Framework

The parallel neurodynamics simulation framework ENLARGE Qu et al. [2023] is employed. We mainly use the fine-grained network representation and hierarchical communication architecture in ENLARGE.

Fine-grained impulse neural network represents and stores neuron and synapse parameters, and network topology separately. The neuron and synapse parameters are firstly separated and grouped according to the characteristics of the computational units and distributed memory. The communication paths are then optimised by the segmentation algorithm to reduce the redundant communication between clusters. Compressed sparse row (CSR) analogy is used for network topology information to describe the connection information of each individual neurons and synapses. For the fine-grained pulsed neural network representation, the delay information is integrated into the CSR representation to form a compact network representation, which dramatically saves the storage requirement.

The hierarchical communication architecture divides cluster communication into three levels: process level, intra-node level and inter-node level. It provides two main modules: the converter module and the communication module. The former locates in the process, which queries the firing neuron ID from the firing list, and converts it into the corresponding shadow neuron ID; it also rearranges the shadow neuron IDs according to their destinations. The latter handles most of the communication and synchronisation. All core computations are performed in the process, the firing list is stored and accessed at each computation phases for data sharing. The synchronisation occurs only at the inter-node level. As the global time must be synchronised within each time step, the hierarchical communication architecture transmits the data in a blocking manner in order to reduce the cost. This

blocking communication process can also be used as a synchronisation signal, instead of the actual synchronisation process of the global time.

## F Evolutionary Algorithm

We considered two implementations of the evolutionary algorithms, CMA-ES Hansen et al. [2003] and NES Wierstra et al. [2014]. The pseudocode is presented in **Algorithm 1** and **Algorithm 2**.

---

### Algorithm 1: Parallel CMA-ES Algorithm

---

**Input:** Initial network parameters  $\theta_0$ , noise standard deviation  $\sigma$ .  
**Output:** Optimal network weights  $W$ .  
**Initialise:**  $m$  nodes,  $n$  tasks.  
**while**  $generation < generation\ limit$  **do**  
    **for** each nodes  $i=1$  to  $m$  **do**  
        **for** each tasks  $j=1$  to  $n$  **do**  
            Generate a polpulation of network parameters using CMA-ES sampler;  
            Parallel compute fitness score  $F_j$  and return to CMA-ES;  
            Update CMA-ES sampler's distribution;  
        **end**  
        Record best parameter  $\sigma_i$ , best score  $F_i$  of all tasks;  
        **if**  $generation = migrate\ generation$  **then**  
            Exchange best parameter  $\theta_i$ , best score  $F_i$  to every other nodes;  
        **end**  
    **end**  
**end**  
Save network weights with highest score  $W_{best}$ .

---



---

### Algorithm 2: Parallel NES Algorithm

---

**Input:** Initial network parameters  $\theta_0$ , noise standard deviation  $\sigma$ .  
**Output:** Optimal network weights  $W$ .  
**Initialise:**  $m$  nodes,  $n$  tasks.  
**while**  $generation < generation\ limit$  **do**  
    **for** each nodes  $i=1$  to  $m$  **do**  
        **for** each tasks  $j=1$  to  $n$  **do**  
            Generate a polpulation of network parameters using NES sampler;  
            Parallel compute fitness score  $F_j$  and return to NES;  
            Update NES sampler's distribution;  
        **end**  
        Record best parameter  $\sigma_i$ , best score  $F_i$  of all tasks;  
        **if**  $generation = migrate\ generation$  **then**  
            Exchange best parameter  $\theta_i$ , best score  $F_i$  to every other nodes;  
        **end**  
    **end**  
**end**  
Save network weights with highest score  $W_{best}$ .

---

The parameters for evolutionary algorithms employed in the work presented in this paper are listed in table 4.

## G Reinforcement Learning Training Algorithms

We consider two RL training techniques, PPO Schulman et al. [2017] and TRPO Wierstra et al. [2014] in the current work. The parameters during network training presented in the work described in this paper are listed in table 5.

Table 4: Parameters of Evolutionary Algorithm.

Parameter	Value
Population size ( $\lambda$ )	10
Parent size ( $\mu$ )	$\lambda/2$
Initial step size ( $\sigma$ )	0.5
Initial mean	Uniform( $(0, 1)^n$ )
Max generation	500

Table 5: PPO and TRPO parameters used for Mujoco tasks.

Parameter	Value	Description
Policy Net	MLP	2 layers with 64 units each
Learning rate	$3 \times 10^{-4}$	Adam optimizer
Discount factor ( $\gamma$ )	0.99	Reward discounting
GAE lambda ( $\lambda$ )	0.95	For Generalized Advantage Estimation
Batch size	64	Mini-batch size for updates
Rollout length	2048	Timesteps per batch
Epochs per update	10	Number of gradient steps per batch

## H Experiments

**CartPole** CartPoleBarto et al. [1983] is a 2-dimensional control task in which the intelligent agent needs to keep a pole balanced upright by applying discrete left and right thrusts. The task, which is widely regarded as a standard test for assessing the responsiveness and stability of a controller. The state space is four-dimensional, comprising the cart’s position and velocity, as well as the pole’s angle and angular velocity. The 2-dimensional action space consists of two discrete actions representing forces applied to the left or right. The reward of the intelligent agent is determined by the positive feedback gained from successfully maintaining the equilibrium state at each time step, and the task continues until the pole tilt angle or position exceeds a threshold value.

**MuJoCo Tasks** The MuJoCo tasksTodorov et al. [2012] we chose are mainly robot motion control problems with a high-dimensional state space (8-27 dimensions) and a continuous action space (1-8 dimensions), where the state contains information about the position, velocity, angle, and angular velocity of the body parts, and the action represents the control signals applied to the actuated joints. The reward functions primarily based on the the distance travelled along a specific axis, complemented by an energy penalty term and balance constraints. Each task is set as a finite time trajectory of 1000 steps.

## References

- A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, Sept. 1983. ISSN 2168-2909. doi: 10.1109/tsmc.1983.6313077. URL <http://dx.doi.org/10.1109/TSMC.1983.6313077>.
- W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski. Neuronal Dynamics, jul 24 2014. URL <http://dx.doi.org/10.1017/CB09781107447615>.
- N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1): 1–18, 2003.
- S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov. 1997. ISSN 1530-888X. doi: 10.1162/neco.1997.9.8.1735. URL <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.
- K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, Jan. 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL [http://dx.doi.org/10.1016/0893-6080\(89\)90020-8](http://dx.doi.org/10.1016/0893-6080(89)90020-8).
- Y. LeCun, G. Hinton, and Y. Bengio. Deep learning (2015), y. lecun, y. bengio and g. hinton. *Nature*, 521, 2015.
- W. Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 10, 1997. ISSN 08936080. doi: 10.1016/S0893-6080(97)00011-7.
- W. Maass and H. Markram. On the computational power of circuits of spiking neurons. *Journal of Computer and System Sciences*, 69, 2004. ISSN 00220000. doi: 10.1016/j.jcss.2004.04.001.
- P. Qu, H. Lin, M. Pang, X. Liu, W. Zheng, and Y. Zhang. Enlarge: An efficient snn simulation framework on gpu clusters. *IEEE Transactions on Parallel and Distributed Systems*, 34, 2023. ISSN 15582183. doi: 10.1109/TPDS.2023.3291825.
- D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct. 1986. ISSN 1476-4687. doi: 10.1038/323533a0. URL <http://dx.doi.org/10.1038/323533a0>.
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.
- P. Strata and R. Harvey. Dale’s principle. *Brain Research Bulletin*, 50(5–6):349–350, Nov. 1999. ISSN 0361-9230. doi: 10.1016/S0361-9230(99)00100-8. URL [http://dx.doi.org/10.1016/S0361-9230\(99\)00100-8](http://dx.doi.org/10.1016/S0361-9230(99)00100-8).
- E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *IEEE International Conference on Intelligent Robots and Systems*, 2012. doi: 10.1109/IROS.2012.6386109.
- C. V. Vreeswijk and H. Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274, 1996. ISSN 00368075. doi: 10.1126/science.274.5293.1724.
- D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber. Natural evolution strategies. *Journal of Machine Learning Research*, 15, 2014. ISSN 15337928.