

Purpose: a quick, print-friendly sheet capturing only the core API surface and the default knobs. This complements the v2 design spec and the original Theory of Change document.

CORE CONCEPTS • Every neuron exposes two methods: `onInput(value, ...)` -> `fired (bool)`, and `onOutput(amplitude)`. • Two-phase tick in Region: Phase A (inject + local routing), Phase B (inter-layer flush), then finalize outputs. • Buses per layer: `inhibition_factor` (≤ 1) and `modulation_factor` (default 1), both decay each tick. • Slots (weights) live inside neurons; thresholds adapt via `T0` imprint + `T2` homeostasis.

DEFAULTS (recommended starting values) `beta` (EMA horizon) = 0.01 `eta` (threshold adapt) = 0.02 `r_star` (target spike rate) = 0.05 `epsilon_fire` (`T0` slack) = 0.01 `t0_slack` (same-stimulus) = 0.02 `slot_hit_saturation` = 10000 `output_smoothing` (EMA) = 0.20 `inhibition_factor` (example) = 0.80 (for 1-2 ticks) `modulation_factor` (example) = 1.20 (for 1-2 ticks)

REGION API (language-neutral names; see per-language bindings) `add_input_layer_2d(h, w, gain=1.0, epsilon_fire=0.01)` -> `layerIndex` `add_layer(excitatory_count, inhibitory_count, modulatory_count)` -> `layerIndex` `add_output_layer_2d(h, w, smoothing=0.2)` -> `layerIndex` `bind_input(port: string, layerIndexes: [int])` `connect_layers(srcLayer, dstLayer, probability, feedback=False)` `tick_image(port: string, image[h][w])` -> `metrics{ delivered_events, total_slots, total_synapses }` (`get_layers()`) -> list of layers (optional helper used by demos)

LAYER API `forward(value: float)` # scalar injection (Phase A)
`forward_image(image[h][w])` # shape-aware injection (Phase A, `InputLayer2D`)
`propagate_from(sourceIndex: int, value: f64)` # destination-side hook (`OutputLayer2D` overrides)
`bus.decay()` # called by Region each tick

NEURON API (unified contract) `onInput(value, [modulation, inhibition])` -> `fired: bool`
`onOutput(amplitude)` -> `None` `fire(amplitude)` # internal; outputs never call this
`fired_last: bool` `last_input_value: float` `slots: map[int, Weight]`

WEIGHT (SLOT) API fields: `strength_value`, `threshold_value`, `ema_rate`, `first_seen`, `hit_count`
`reinforce(modulation, inhibition)` `update_threshold(input_value)` -> `fired: bool` # includes `T0` + `T2` hybrid logic

OUTPUT LAYER/NEURON FINALIZATION `OutputNeuron.onInput(...)` # gate only, no fire
`OutputNeuron.onOutput(a)` # accumulate sum/count `OutputNeuron.end_tick()` # compute EMA:
`output_value = (1-s)*prev + s*(sum/count)` `OutputLayer2D.end_tick()` # copy each neuron's `output_value` into the frame

TWO-PHASE TICK (Region) Phase A: inject → per-layer: `onInput` → if fired then `onOutput` → local propagation Phase B: flush tracts once → finalize outputs (`end_tick`) → decay buses

INVARIANTS • Output neurons never propagate (never call fire). • Layers call `neuron.onOutput(value)` only if `fired` is true. • Input/Output neurons are single-slot by design; hidden neurons may allocate more slots over time.

METRICS (suggested per tick) `delivered_events`, `total_slots`, `total_synapses`, `output_mean`, `output_nonzero`

LANGUAGE BINDINGS (names) Python: `add_input_layer_2d`, `add_layer`, `add_output_layer_2d`, `tick_image`, `end_tick` Java: `addInputLayer2D`, `addLayer`, `addOutputLayer2D`, `tickImage`, `endTick` C++: `addInputLayer2D`, `addLayer`, `addOutputLayer2D`, `tickImage`, `endTick` Mojo: `add_input_layer_2d`, `add_layer`, `add_output_layer_2d`, `tick_image`, `end_tick`