# SuperNeuroMAT

***Release 2.0.0***

**Prasanna Date, Chathika Gunaratne, Shruti Kulkarni, Robert Patto**

**Apr 19, 2025**

# CONTENTS:

SuperNeuroMAT is a matrix-based simulator for simulating spiking neural networks, which are used in neuromorphic computing. It is one of the fastest, if not the fastest, simlators for simulating spiking neural networks.

Some salient features of SuperNeuroMAT are:

1. Support for leaky integrate and fire neuron model with the following parameters: neuron threshold, neuron leak, and neuron refractory period
2. Support for Spiking-Time-Dependent Plasticity (STDP) synapses with weights and delays
3. No restrictions on connectivity of the neurons, all-to-all connections as well as self connections possible
4. Constant leak supported
5. STDP learning can be configured to turn on/off positive updates and negative updates
6. Excessive synaptic delay can slow down the execution of the simulation, so try to avoid as much as possible
7. Leak refers to the constant amount by which the internal state (membrane potential) of a neuron changes in each time step of the simulation; therefore, zero leak means the neuron fully retains the value in its internal state, and infinite leak means the neuron never retains the value in its internal state
8. STDP implementation is extremely fast
9. The model of neuromorphic computing supported in SuperNeuroMAT is Turing-complete
10. All underlying computational operations are matrix-based and currently supported on CPUs

CHAPTER

ONE

# INSTALLATION

1. Install using `pip install superneuromat`
2. Update/upgrade using `pip install superneuromat --upgrade`

CHAPTER

TWO

# USAGE

1. In a Python script or on a Python interpreter, do `import superneuromat as snm`
2. The main class can be accessed by `snn = snm.SNN()`
3. Refer to docstrings in the source code or on the readthedocs page for the API

CHAPTER

THREE

# DEVELOPMENT

1. Clone the `superneuromat` repo: `git clone https://github.com/ORNL/superneuromat.git`
2. Add the path to `superneuromat` to your `$PYTHONPATH`: `export PYTHONPATH=$PYTHONPATH:/path/to/superneuromat`.
3. You may want to update the `$PYTHONPATH` in your `.bash_profile` or `.bashrc`.

CHAPTER

# FOUR

# CITE SUPERNEUROMAT

Please cite SuperNeuroMAT using:

```
@inproceedings{date2023superneuro,
  title={SuperNeuro: A fast and scalable simulator for neuromorphic computing},
  author={Date, Prasanna and Gunaratne, Chathika and R. Kulkarni, Shruti and Patton,␣
↪Robert and Coletti, Mark and Potok, Thomas},
  booktitle={Proceedings of the 2023 International Conference on Neuromorphic Systems},
  pages={1--4},
  year={2023}
}
```

**References for SuperNeuroMAT:**

- SuperNeuro: A Fast and Scalable Simulator for Neuromorphic Computing
- Neuromorphic Computing is Turing-Complete
- Computational Complexity of Neuromorphic Algorithms

CHAPTER

# FIVE

# SUPERNEUROMAT

class superneuromat.**SNN**(*backend='cpu'*, *num_mpi_ranks=1*)

Bases: object

Defines a spiking neural network (SNN) with neurons and synapses

**num_neurons**

Number of neurons in the SNN

**Type**
int

**neuron_thresholds**

List of neuron thresholds

**Type**
list

**neuron_leaks**

List of neuron leaks, defined as the amount by which the internal states of the neurons are pushed towards the neurons' reset states

**Type**
list

**neuron_reset_states**

List of neuron reset states

**Type**
list

**neuron_refractory_periods**

List of neuron refractory periods

**Type**
list

**num_synapses**

Number of synapses in the SNN

**Type**
int

**pre_synaptic_neuron_ids**

List of pre-synaptic neuron IDs

**Type**
list

`post_synaptic_neuron_ids`
List of post-synaptic neuron IDs

**Type**
list

`synaptic_weights`
List of synaptic weights

**Type**
list

`synaptic_delays`
List of synaptic delays

**Type**
list

`enable_stdp`
List of Boolean values denoting whether STDP learning is enabled on each synapse

**Type**
list

`input_spikes`
Dictionary of input spikes indexed by time

**Type**
dict

`spike_train`
List of spike trains for each time step

**Type**
list

`stdp`
Boolean parameter that denotes whether STDP learning has been enabled in the SNN

**Type**
bool

`stdp_time_steps`
Number of time steps over which STDP updates are made

**Type**
int

`stdp_Apos`
List of STDP parameters per time step for excitatory update of weights

**Type**
list

`stdp_Aneg`
List of STDP parameters per time step for inhibitory update of weights

**Type**
list

**create_neuron()**

Creates a neuron in the SNN

**create_synapse()**

Creates a synapse in the SNN

**add_spike()**

Add an external spike at a particular time step for a given neuron with a given value

**stdp_setup()**

Setup the STDP parameters

**setup()**

Setup the SNN and prepare for simulation

**simulate()**

Simulate the SNN for a given number of time steps

**print_spike_train()**

Print the spike train

**Caution**

1. Delay is implemented by adding a chain of proxy neurons. A delay of 10 between neuron A and neuron B would add 9 proxy neurons between A and B.
2. Leak brings the internal state of the neuron back to the reset state. The leak value is the amount by which the internal state of the neuron is pushed towards its reset state.
3. Deletion of neurons is not permitted
4. Multiple synapses from one neuron to any another neuron are not permitted
5. Input spikes can have a value
6. All neurons are monitored by default

**add_spike**(*time: int*, *neuron_id: int*, *value: float = 1.0*) → None

Adds an external spike in the SNN

**Parameters**

- **time** (*int*) – The time step at which the external spike is added
- **neuron_id** (*int*) – The neuron for which the external spike is added
- **value** (*float*) – The value of the external spike (default: 1.0)

**Raises**

**TypeError if** –

1. time is not an int 2. neuron_id is not an int 3. value is not an int or float

**count_spikes()**

Returns the spike count

**create_neuron**(*threshold: float = 0.0*, *leak: float = inf*, *reset_state: float = 0.0*, *refractory_period: int = 0*) → int

Create a neuron

**Parameters**

- **threshold** (*float*) – Neuron threshold; the neuron spikes if its internal state is strictly greater than the neuron threshold (default: 0.0)
- **leak** (*float*) – Neuron leak; the amount by which by which the internal state of the neuron is pushed towards its reset state (default: np.inf)
- **reset_state** (*float*) – Reset state of the neuron; the value assigned to the internal state of the neuron after spiking (default: 0.0)
- **refractory_period** (*int*) – Refractory period of the neuron; the number of time steps for which the neuron remains in a dormant state after spiking

**Returns**

Returns the neuron ID

**Raises**

- **TypeError if** –
  1. threshold is not an int or a float 2. leak is not an int or a float 3. reset_state is not an int or a float 4. refractory_period is not an int
- **ValueError if** –
  1. leak is less than 0.0 2. refractory_period is less than 0

**create_synapse**(*pre_id: int*, *post_id: int*, *weight: float = 1.0*, *delay: int = 1*, *stdp_enabled: bool = False*) → None

Creates a synapse in the SNN from a pre-synaptic neuron to a post-synaptic neuron with a given set of synaptic parameters (weight, delay and enable_stdp)

**Parameters**

- **pre_id** (*int*) – ID of the pre-synaptic neuron
- **post_id** (*int*) – ID of the post-synaptic neuron
- **weight** (*float*) – Synaptic weight; weight is multiplied to the incoming spike (default: 1.0)
- **delay** (*int*) – Synaptic delay; number of time steps by which the outgoing signal of the syanpse is delayed by (default: 1)
- **enable_stdp** (*bool*) – Boolean value that denotes whether or not STDP learning is enabled on the synapse (default: False)

**Raises**

- **TypeError if** –
  1. pre_id is not an int 2. post_id is not an int 3. weight is not a float 4. delay is not an int 5. enable_stdp is not a bool
- **ValueError if** –
  1. pre_id is less than 0 2. post_id is less than 0 3. delay is less than or equal to 0
- **RuntimeError if** –
  1. A synapse with the same (pre_id, post_id) already exists

**print_spike_train**()

Prints the spike train

**reset**(*internal_states=True*, *refractory_periods=True*, *internal_spikes=True*, *stdp_enabled_weights=True*, *spike_train=True*, *spike_count=True*, *input_spikes=True*) → None

Resets the state of the simulator

**Parameters**

- **internal_states** (*bool*) – Set to True if internal states of neurons should be reset, otherwise False (default: True)
- **refractory_periods** (*bool*) – Set to True if refractory periods of neurons should be reset, otherwise False (default: True)
- **internal_spikes** (*bool*) – Set to True if internal spikes of neurons should be reset, otherwise False (default: True)
- **stdp_enabled_weights** (*bool*) – Set to True if STDP enabled synaptic weights should be reset, otherwise False (default: True)
- **spike_train** (*bool*) – Set to True if spike train should be reset, otherwise False (default: True)
- **spike_count** (*bool*) – Set to True if spike count should be reset, otherwise False (default: True)
- **input_spikes** (*bool*) – Set to True if input spikes should be reset, otherwise False (default: True)

**Raises**

- **TypeError if** –

  1. internal_states is not bool 2. refractory_periods is not bool 3. internal_spikes is not bool 4. stdp_enabled_weights is not bool 5. spike_train is not bool 6. spike_count is not bool 7. input_spikes is not bool

- **Value error if** –

  1. internal_states is neither True nor False 2. refractory_periods is neither True nor False 3. internal_spikes is neither True nor False 4. stdp_enabled_weights is neither True nor False 5. spike_train is neither True nor False 6. spike_count is neither True nor False 7. input_spikes is neither True nor False

**setup**(*sparse='auto'*, *dtype=64*)

Choose the appropriate setup function based on backend

**Parameters**

- **sparse** (*bool*) – If True, forces simulation to use sparse computations (default: “auto”)
- **dtype** (*int*) – 32 or 64 for single or double precision operation (default: 64)

**Raises**

- **TypeError if** –

  1. sparse is not a bool 2. dtype is not an int

- **ValueError if** –

  1. dtype is not 32 or 64

**simulate**(*time_steps: int = 1000*) → None

Simulate the spiking neural network

**Parameters**

- **time_steps** (*int*) – Number of time steps for which the SNN is to be simulated
- **backend** (*string*) – Backend is either cpu or frontier

**Raises**

- **TypeError if** –
  1. time_steps is not an int 2. backend is not a string
- **ValueError if** –
  1. time_steps is less than or equal to zero 2. backend is not one of the following values: cpu, frontier

**stdp_setup**(*time_steps: int = 3*, *Apos: list = [1.0, 0.5, 0.25]*, *Aneg: list = [1.0, 0.5, 0.25]*, *positive_update: bool = True*, *negative_update: bool = True*) → None

Choose the appropriate STDP setup function based on backend

**Parameters**

- **time_steps** (*int*) – Number of time steps over which STDP learning occurs (default: 3)
- **Apos** (*list*) – List of parameters for excitatory STDP updates (default: [1.0, 0.5, 0.25]); number of elements in the list must be equal to time_steps
- **Aneg** (*list*) – List of parameters for inhibitory STDP updates (default: [1.0, 0.5, 0.25]); number of elements in the list must be equal to time_steps
- **positive_update** (*bool*) – Boolean parameter indicating whether excitatory STDP update should be enabled
- **negative_update** (*bool*) – Boolean parameter indicating whether inhibitory STDP update should be enabled

**Raises**

- **TypeError if** –
  1. time_steps is not an int 2. Apos is not a list 3. Aneg is not a list 4. positive_update is not a bool 5. negative_update is not a bool
- **ValueError if** –
  1. time_steps is less than or equal to zero 2. Number of elements in Apos is not equal to the time_steps 3. Number of elements in Aneg is not equal to the time_steps 4. The elements of Apos are not int or float 5. The elements of Aneg are not int or float 6. The elements of Apos are not greater than or equal to 0.0 7. The elements of Apos are not greater than or equal to 0.0
- **RuntimeError if** –
  1. enable_stdp is not set to True on any of the synapses

# PYTHON MODULE INDEX

## S

# INDEX