# **SuperNeuroMAT**

Release 2.0.0

Prasanna Date, Chathika Gunaratne, Shruti Kulkarni, Mark Coletti

# **CONTENTS:**

		euromat package  Module contents	. 3
Ру	thon	odule Index	9
In	dex		11

SuperNeuroMAT is a fast and scalable simulator for spiking neural networks (SNNs). It is meant to be run on multi-core systems. SuperNeuroMAT models SNN computations using matrix and vector operations.

Key features of SuperNeuroMAT include: 1. Support for Leaky Integrate and Fire (LIF) neuron model 2. Support for Spike-Timing-Dependent Plasticity (STDP) learning mechanism

CONTENTS: 1

2 CONTENTS:

# SUPERNEUROMAT PACKAGE

# 1.1 Module contents

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)  $\rightarrow$  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

1.1. Module contents 5

#### count\_spikes()

Returns the spike count

**create\_neuron**(threshold: float = 0.0, leak: float = inf,  $reset\_state$ : float = 0.0,  $refractory\_period$ : int = 0)  $\rightarrow$  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) \rightarrow 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** (*boo1*) 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)  $\rightarrow$  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** (*boo1*) 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** (*boo1*) 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 (boo1) 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 -
  - internal\_states is neither True nor False 2. refractory\_periods is neither True nor False
     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 -

1.1. Module contents 7

1. dtype is not 32 or 64

**simulate**(*time steps: int* = 1000)  $\rightarrow$  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

 $stdp\_setup(time\_steps: int = 3, Apos: list = [1.0, 0.5, 0.25], Aneg: list = [1.0, 0.5, 0.25], positive\_update: bool = True, negative\_update: bool = True) <math>\rightarrow$  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** (*1ist*) 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** (*1ist*) 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 (boo1) Boolean parameter indicating whether excitatory STDP update should be enabled
- negative\_update (boo1) 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

 ${\tt superneuromat}, 3$ 

10 Python Module Index

# **INDEX**

```
Α
add_spike() (superneuromat.SNN method), 5
C
count_spikes() (superneuromat.SNN method), 5
create_neuron() (superneuromat.SNN method), 5, 6
create_synapse() (superneuromat.SNN method), 5, 6
Ε
enable_stdp (superneuromat.SNN attribute), 4
input_spikes (superneuromat.SNN attribute), 4
M
module
    superneuromat, 3
Ν
neuron_leaks (superneuromat.SNN attribute), 3
neuron_refractory_periods
                              (superneuromat.SNN
        attribute), 3
neuron_reset_states (superneuromat.SNN attribute),
neuron_thresholds (superneuromat.SNN attribute), 3
num_neurons (superneuromat.SNN attribute), 3
num_synapses (superneuromat.SNN attribute), 3
post_synaptic_neuron_ids (superneuromat.SNN at-
        tribute), 4
pre_synaptic_neuron_ids (superneuromat.SNN at-
        tribute), 3
print_spike_train() (superneuromat.SNN method),
R
reset() (superneuromat.SNN method), 7
S
setup() (superneuromat.SNN method), 5, 7
```

```
simulate() (superneuromat.SNN method), 5, 8
SNN (class in superneuromat), 3
spike_train (superneuromat.SNN attribute), 4
stdp (superneuromat.SNN attribute), 4
stdp_Aneg (superneuromat.SNN attribute), 4
stdp_Apos (superneuromat.SNN attribute), 4
stdp_setup() (superneuromat.SNN method), 5, 8
stdp_time_steps (superneuromat.SNN attribute), 4
superneuromat
    module, 3
synaptic_delays (superneuromat.SNN attribute), 4
synaptic_weights (superneuromat.SNN attribute), 4
```