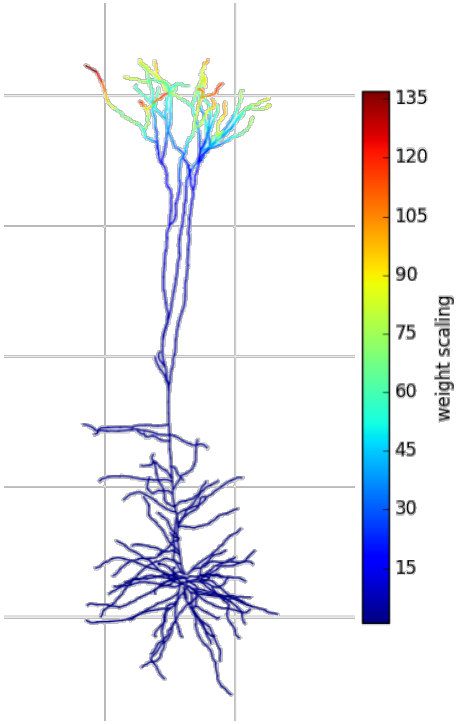
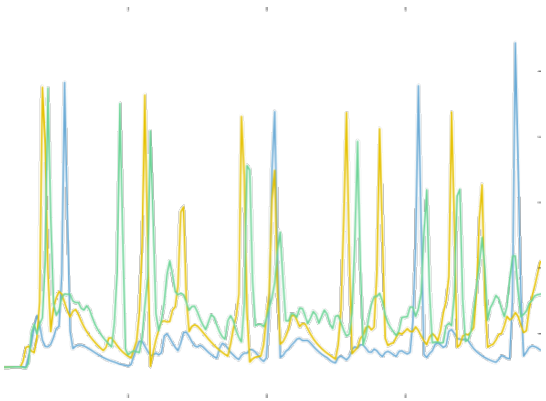
NetPyNE

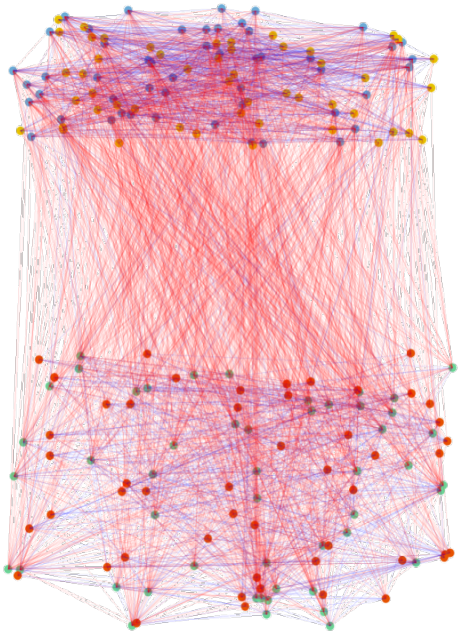
(**Net**works in **Py**thon and **NE**URON)

A Python package to facilitate the development, simulation and analysis of biological neuronal networks in NEURON



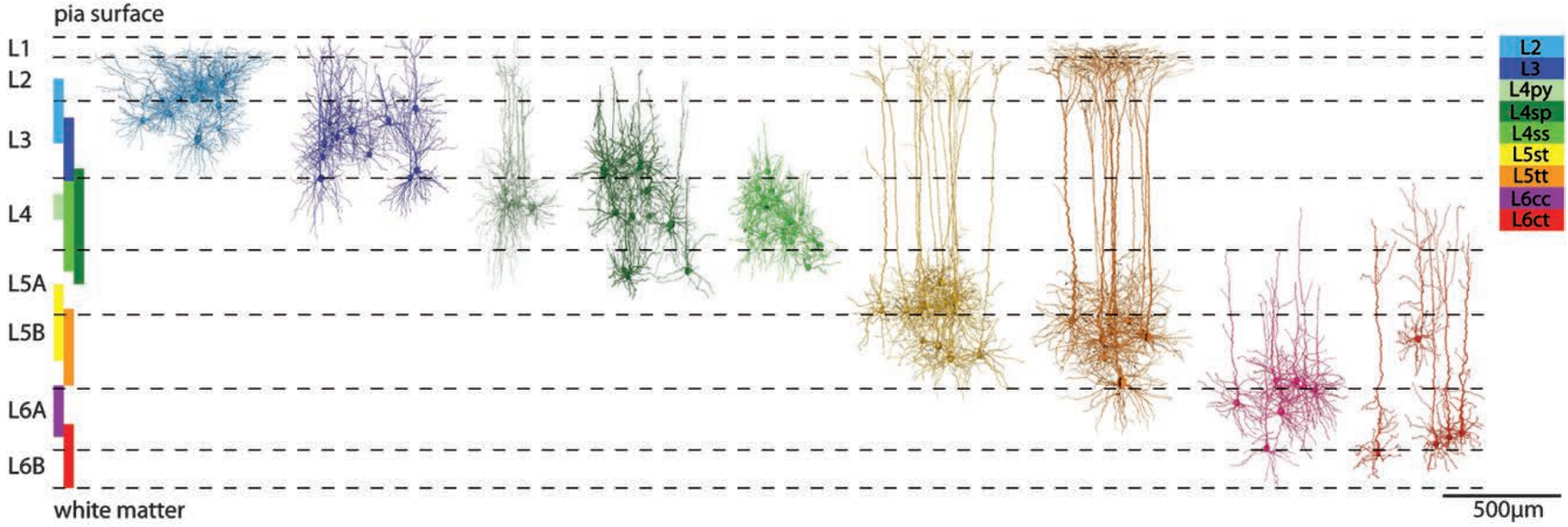
www.netpyne.org





Motivation

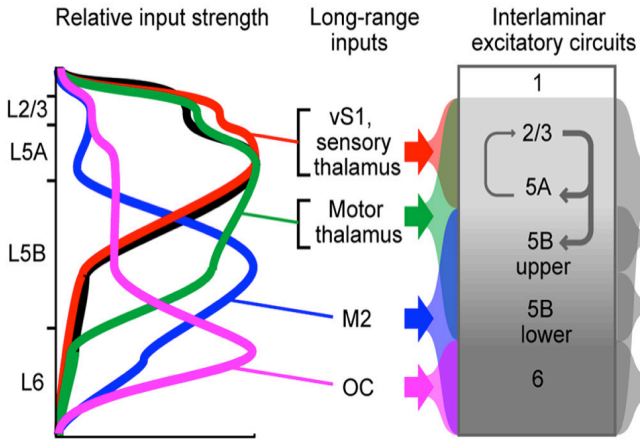
- Facilitate incorporation of experimental data at multiple scales



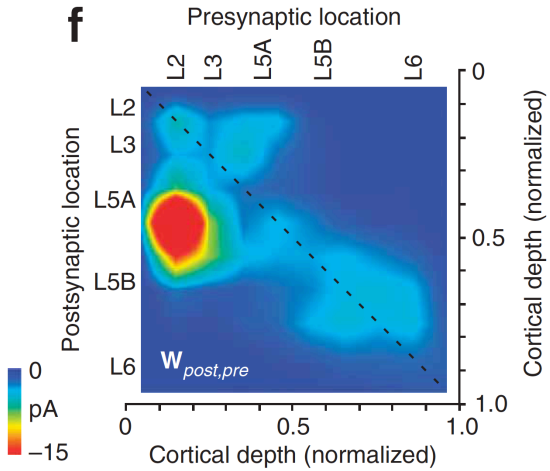
Motivation

- Facilitate incorporation of experimental data at multiple scales

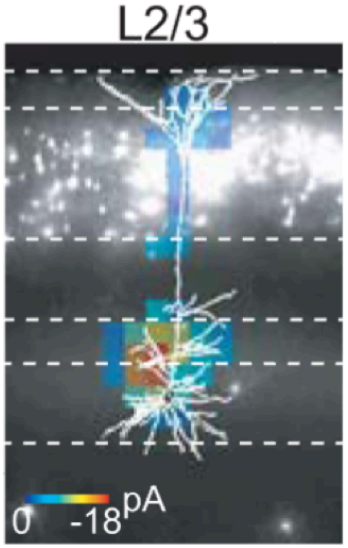
Long-range inputs



Local microcircuits



Dendritic inputs

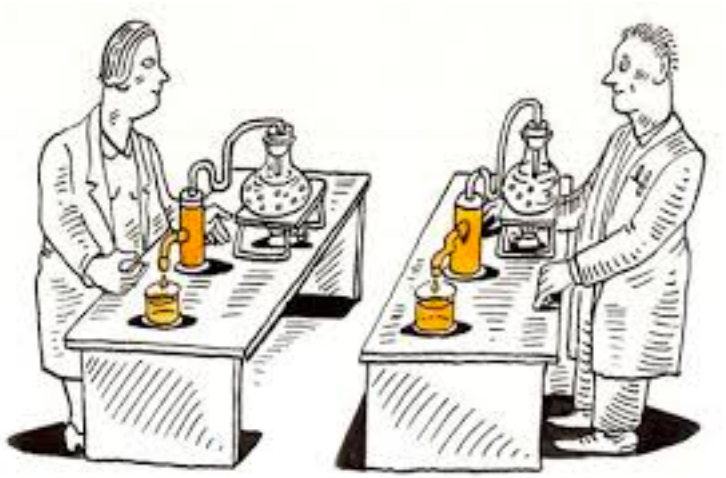


Motivation

- Separate model parameters from implementation
- Standardize format – easy to read, interpret, edit, share etc

```
popParams['EXC_L2'] = {
 'cellType': 'PYR',
 'yRange':    [100, 400],
 'numCells':  50}
```

```
for cellParams in range(pop['numCells']):
      cell = sim.Cell(cellParams)
      cell.tags['y'] = numpy.random(100,400)
      cell.tags['cellType'] = 'PYR'
```

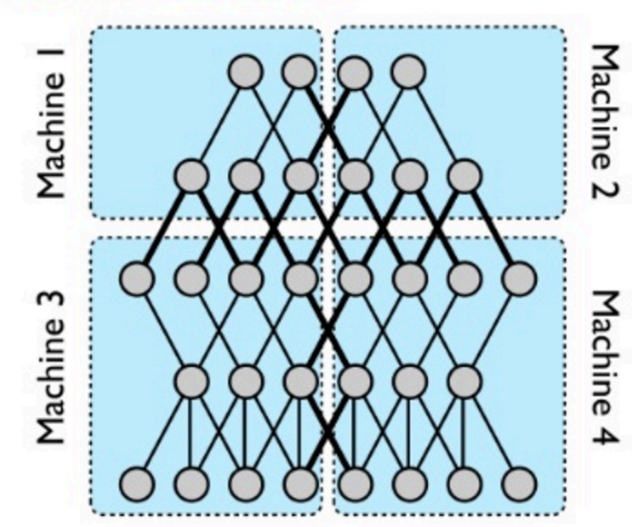


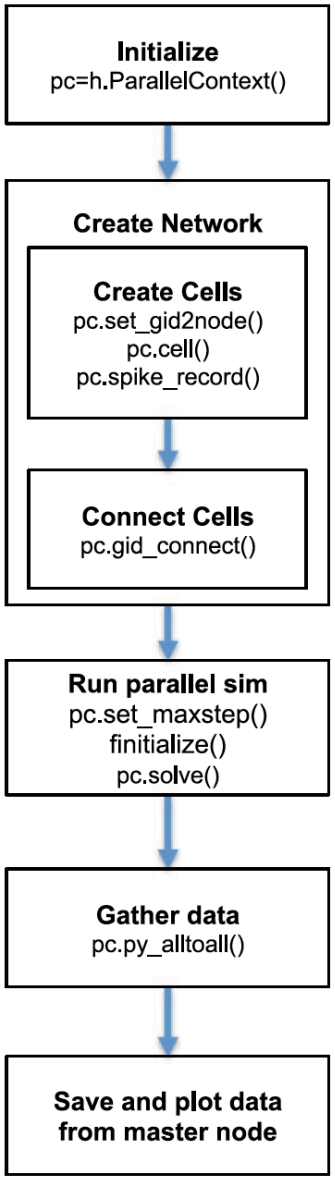
Replicate: get same thing to run again

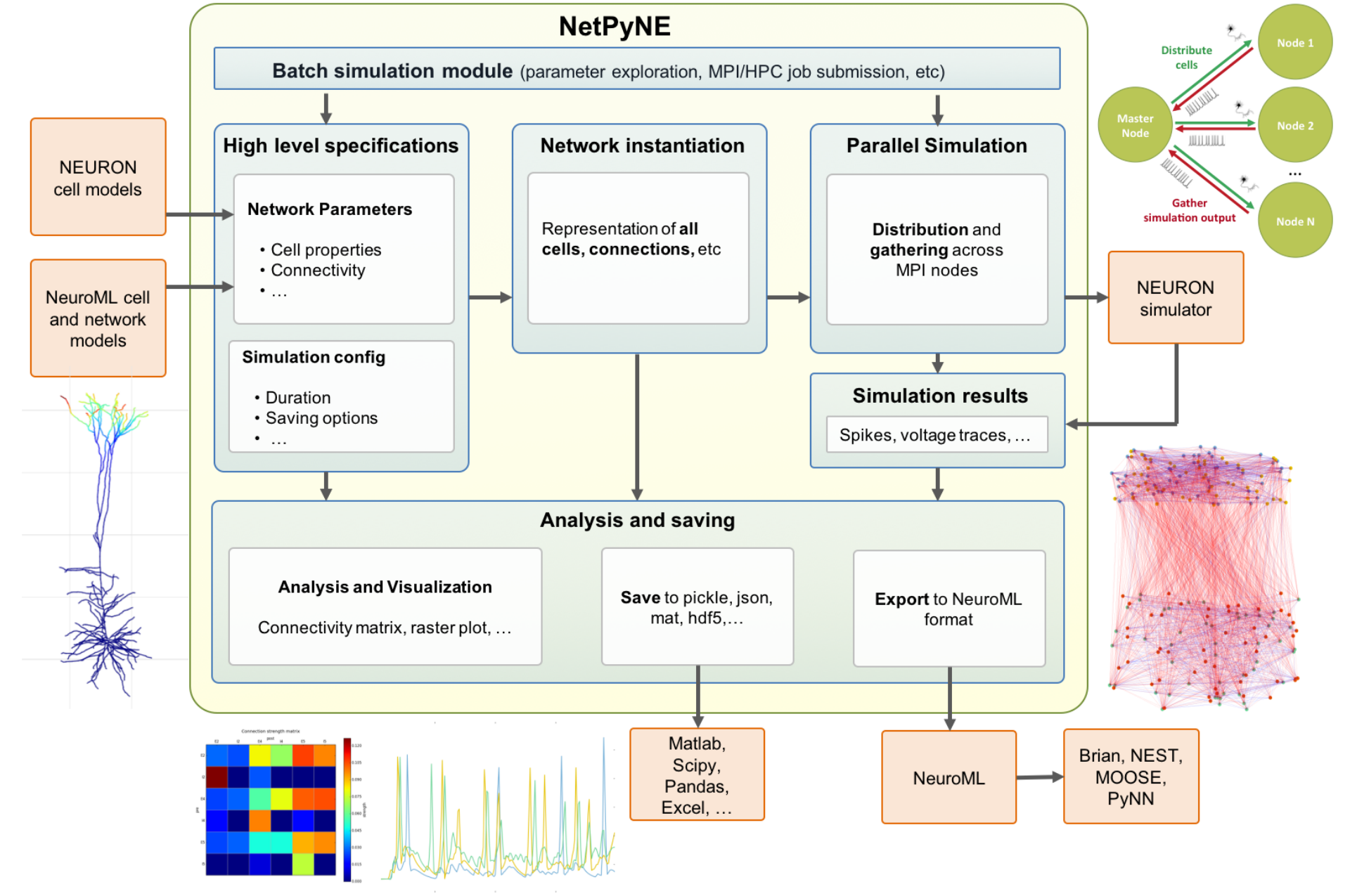
Reproduce: make it youself

Motivation

- Facilitate model parallelization (HPCs)
- Batch parameter exploration/optimization






NetPyNE
Batch simulation module (parameter exploration, MPI/HPC job submission, etc)
NEURON cell models
NeuroML cell and network models
High level specifications
Network Parameters
• Cell properties
• Connectivity
• …
Simulation config
• Duration
• Saving options
• …
Network instantiation
Representation of all cells, connections, etc
Parallel Simulation
Distribution and gathering across MPI nodes
Simulation results
Spikes, voltage traces, …
Distribute cells
Master Node
Node 1
Node 2
Node N
Gather simulation output
NEURON simulator
Analysis and saving
Analysis and Visualization
Connectivity matrix, raster plot, …
Save to pickle, json, mat, hdf5,…
Export to NeuroML format
Matlab, Scipy, Pandas, Excel, …
NeuroML
Brian, NEST, MOOSE, PyNN

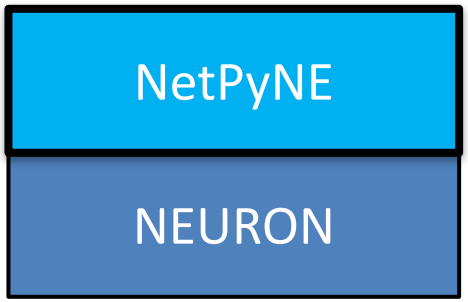
High level specifications

NEURON

```
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

pre1Con = h.NetCon(preCell1.soma(0.5)._ref_v,
                   postSyn1,
                   sec=preCell1.soma)
pre1Con.delay = 1
pre1Con.weight[0] = 0.001
pre1Con.threshold = 0
```

High level specifications

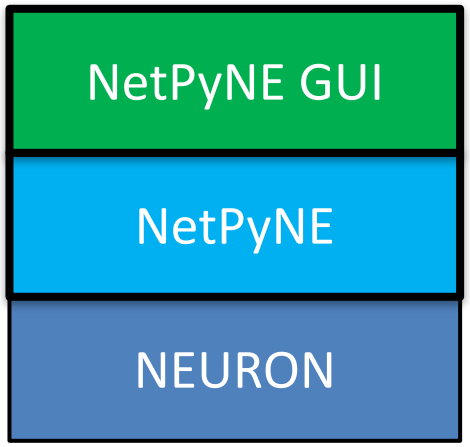


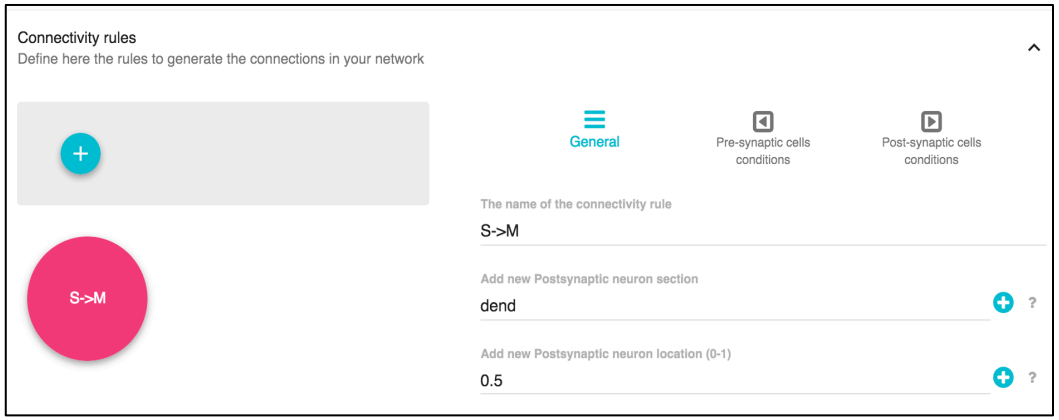

```
## Cell connectivity rules
netParams.connParams['S->M'] = {
        'preConds': {'pop': 'S'},
        'postConds': {'pop': 'M'},
        'probability': 0.5,
        'weight': 0.01,
        'delay': 5,
        'synMech': 'exc'}
```

```
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

pre1Con = h.NetCon(preCell1.soma(0.5)._ref_v,
                   postSyn1,
                   sec=preCell1.soma)
pre1Con.delay = 1
pre1Con.weight[0] = 0.001
pre1Con.threshold = 0
```

High level specifications






```
## Cell connectivity rules
netParams.connParams['S->M'] = {
        'preConds': {'pop': 'S'},
        'postConds': {'pop': 'M'},
        'probability': 0.5,
        'weight': 0.01,
        'delay': 5,
        'synMech': 'exc'}
```

```
# add exc connection
postSyn1 = h.ExpSyn(postCell.dend(0.5))
postSyn1.tau = 2
postSyn1.e = -90

pre1Con = h.NetCon(preCell1.soma(0.5)._ref_v,
                   postSyn1,
                   sec=preCell1.soma)
pre1Con.delay = 1
pre1Con.weight[0] = 0.001
pre1Con.threshold = 0
```

High level specifications

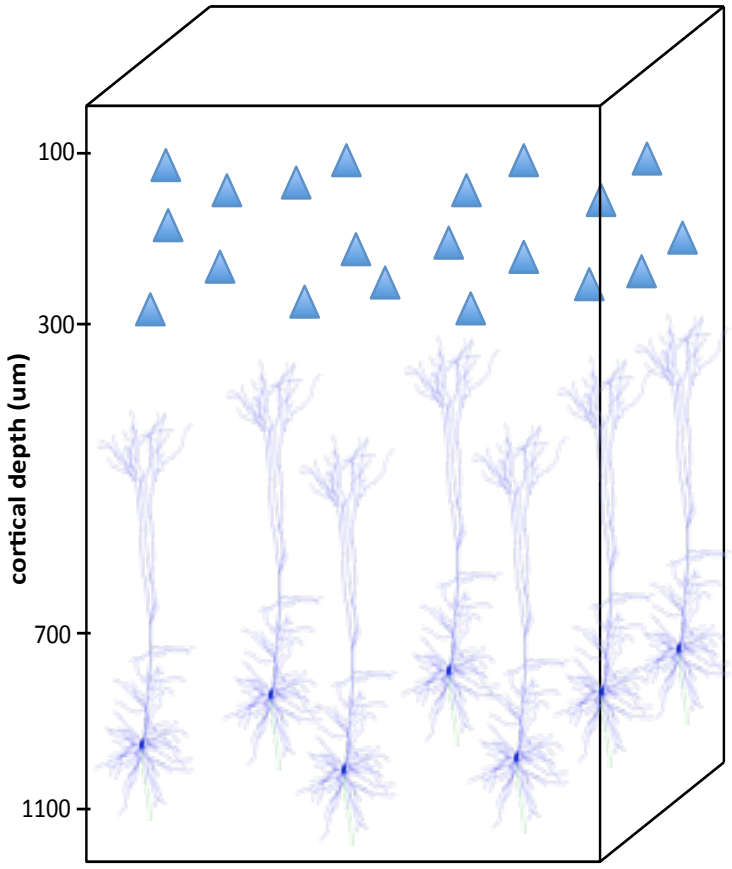
- A **standardized, declarative** Python format (JSON-like, lists and dicts) to define:
  - ***Populations****:* cell type, number of neurons or density, spatial extent, ...
  - ***Cell properties:*** Morphology, biophysics, implementation, …
  - ***Synaptic mechanisms:*** Time constants, reversal potential, implementation, …
  - ***Stimulation:*** Spike generators, current clamps, spatiotemporal properties, …
  - ***Connectivity rules:*** conditions of pre- and post-synaptic cells, different functions, …
  - ***Simulation configuration:*** duration, saving and analysis, graphical output, ...

High level specifications

- **Example** of populations and cell parameters

```
popParams['EXC_L2'] = {
 'cellType': 'PYR',
 'cellModel': 'simple',
 'yRange':   [100, 400],
 'numCells': 50}
```

```
popParams['EXC_L5'] = {
 'cellType': 'PYR',
 'cellModel': 'complex',
 'yRange':   [700, 1100],
 'density':  80e3}
```

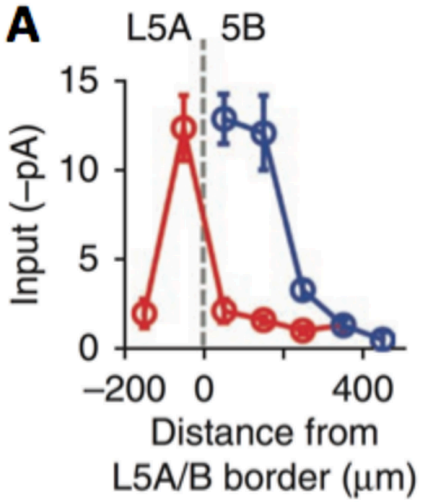



```
cellParams['PYR_simple'] = {
 'conds': {'cellType': 'PYR', 'cellModel': 'simple'},
 'secs': {'soma': {'geom': {'diam': 18.8, 'L': 18.8},
                   {'mechs': {'hh': {'gnabar': 0.12,
                                     'gkbar':0.036,
                                     'gl':0.003,
                                      'el': -70}}}
```

```
importCellParams(
 label = 'PYR_complex',
 conds = {'cellType': 'PYR', 'cellModel': 'complex'},
 fileName = 'L5_pyr_full.hoc',
 cellName = 'PYR_template')
```

High level specifications

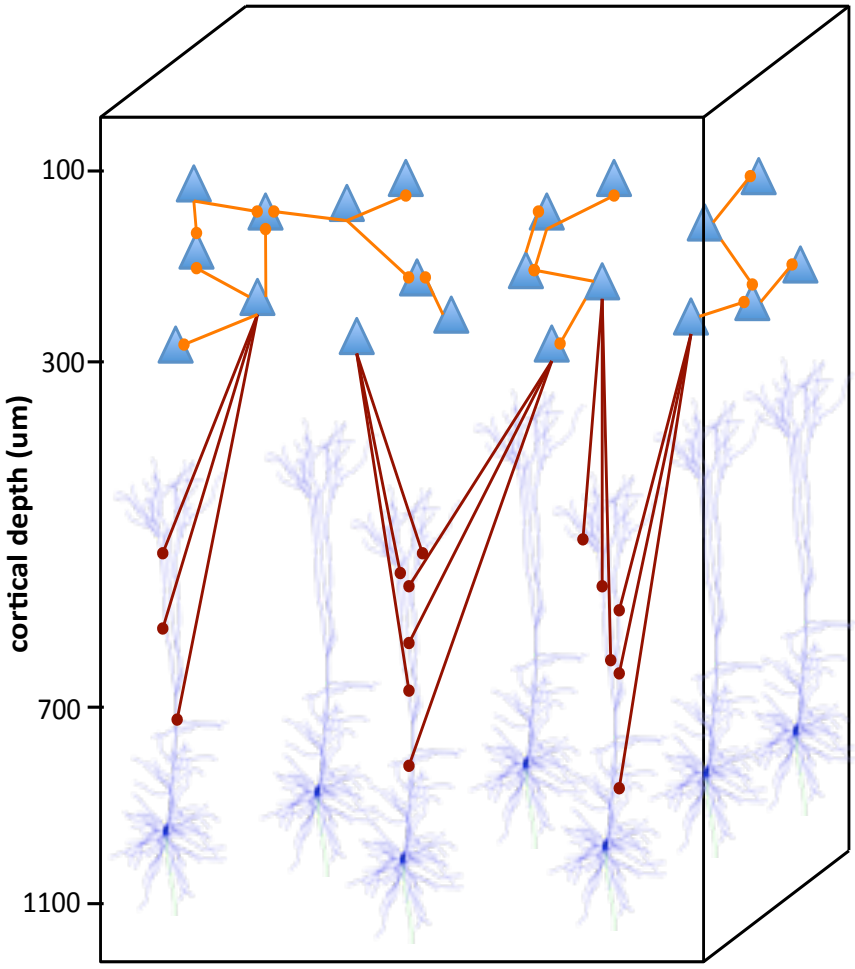
- ***Connectivity:***
  - **Flexible connectivity rules** based on pre- and post-synaptic cell properties (eg, type or location).
  - Connectivity **functions** available: all-to-all, probabilistic, convergent, divergent, and explicit list.
  - Parameters (eg, probability, weight, delay) as a **function of pre/post-synaptic spatial properties**, eg, delays or probability that depend on distance between cells or cortical depth.
  - Easily add synapses with **learning** mechanisms (STDP and RL) and **gap junctions.**



Incorporate connectivity data as a function of cortical depth (not layers)

High level specifications

- **Example** of connectivity rules



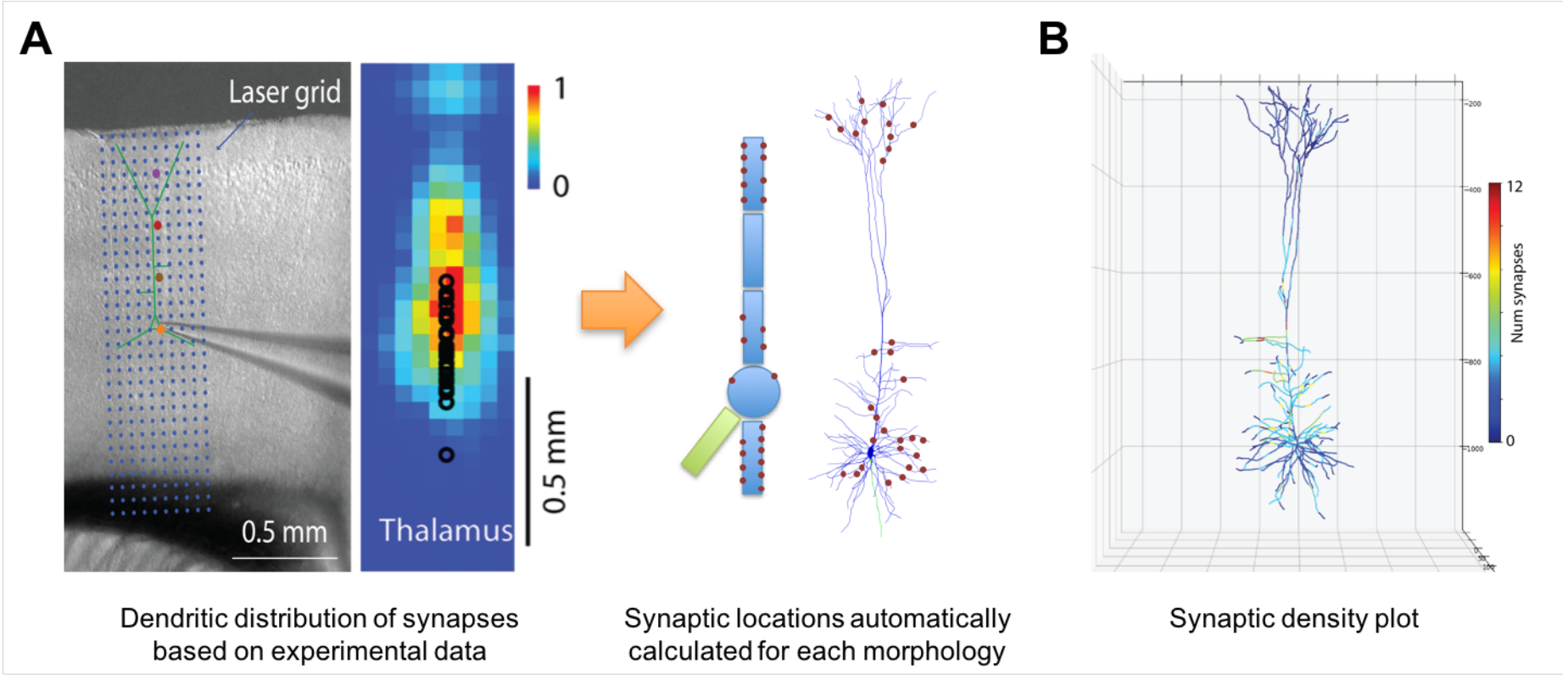

```
connParams['L2->E2'] = {
 'preConds':    {'y': [100, 400]},
 'postConds':   {'popLabel': 'EXC_L2'},
 'probability': '1*exp(-dist_3D/200)',
 'weight':      0.4,
 'delay':       5,
 'synMech':     'AMPA'}
```

```
connParams['E2->L5'] = {
 'preConds':    {'popLabel': 'EXC_L2'},
 'postConds':   {'y': [700,1100], 'cellModel': 'complex'},
 'convergence':  25,
 'weight':      '0.001 * post_ynorm',
 'delay':       'dist_3D/propVelocity',
 'sec':         'allDend',
 'synMech':     'AMPA',
 'synsPerConn':  3}
```

```
synMechParams['AMPA'] = {
 'mod':     'Exp2Syn',
 'tau1':    0.8,
 'tau2':    5.3,
 'e':        0}
```

High level specifications

- ***Connectivity:***
  - Specify dendritic **distribution of synapses** as 1D or 2D density map, or based on distance from section (eg, 100 µm around soma)
  - Synaptic distribution automatically **adapted to morphology** of each cell model



High-level specifications

Integration of Reaction-Diffusion (RxD)

1) Metabotropic glutamate receptors (mGluR) activate

2) increase IP3 in cytosol

3) ER IP3R channels open

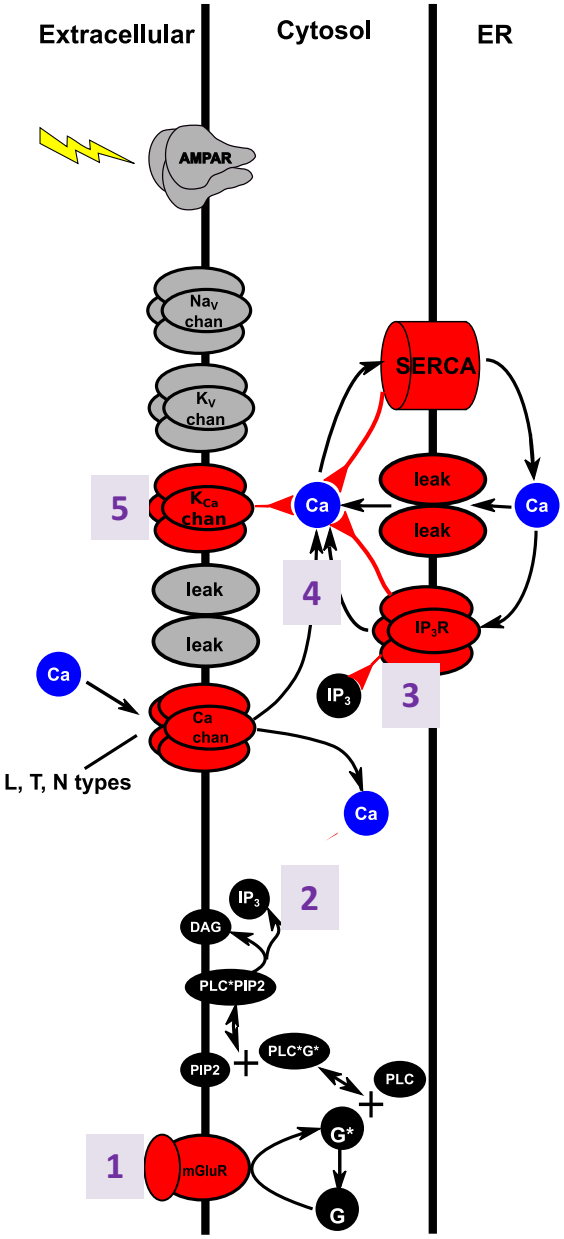
4) ER Ca released to cytosol

5) kBK / Kca channels (sensitive to Ca) open

6) K flows inside cell

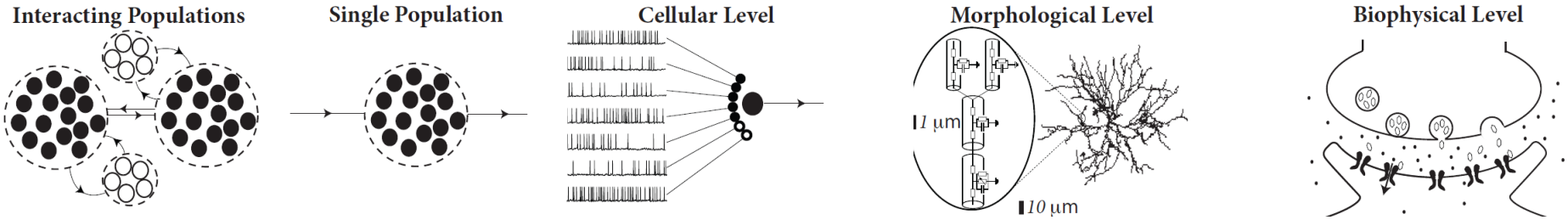
7) hyperpolarizing K current

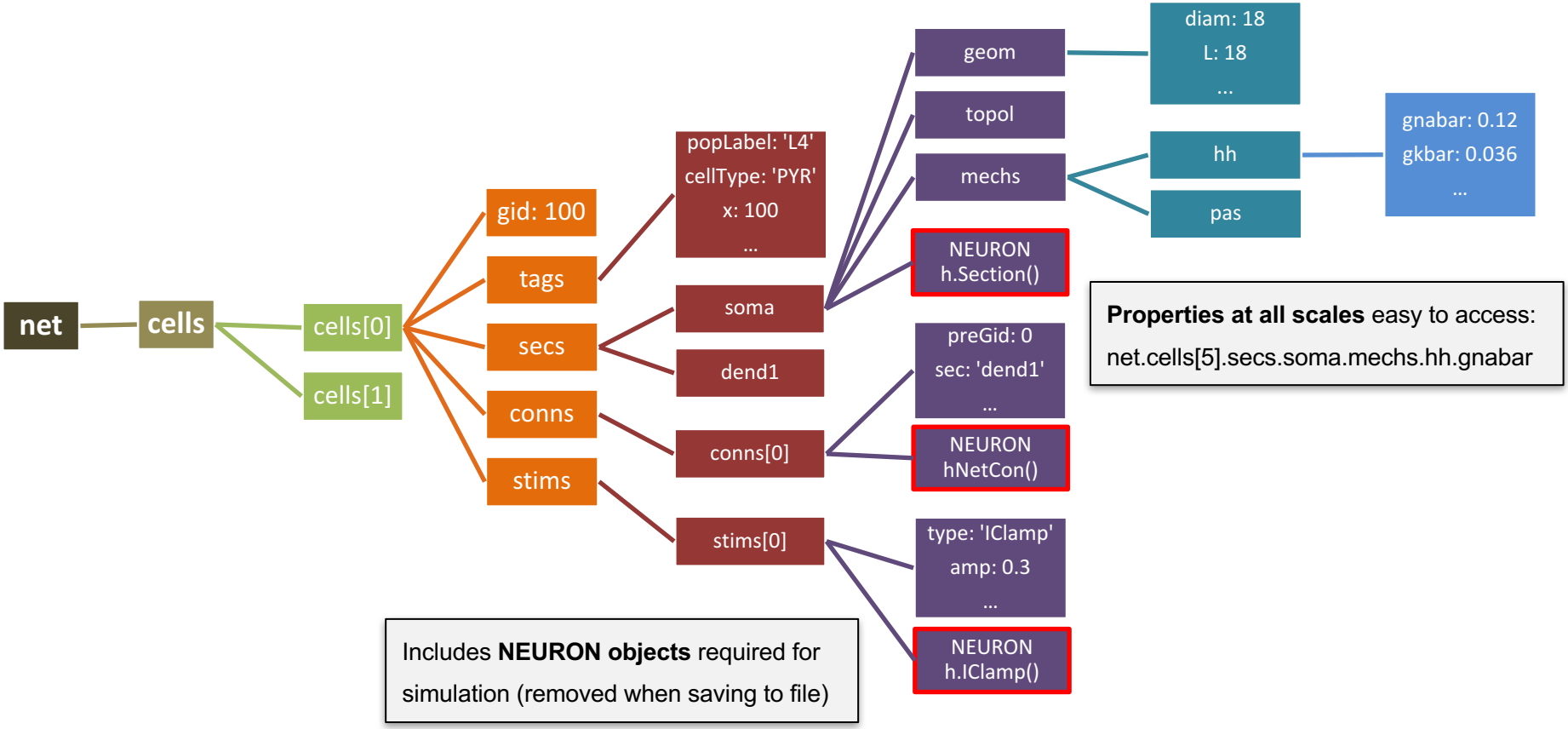
8) reduces cell firing



Network instantiation

- Network instance as **standardized hierarchical** Python strucutre (JSON-like, lists and dicts)

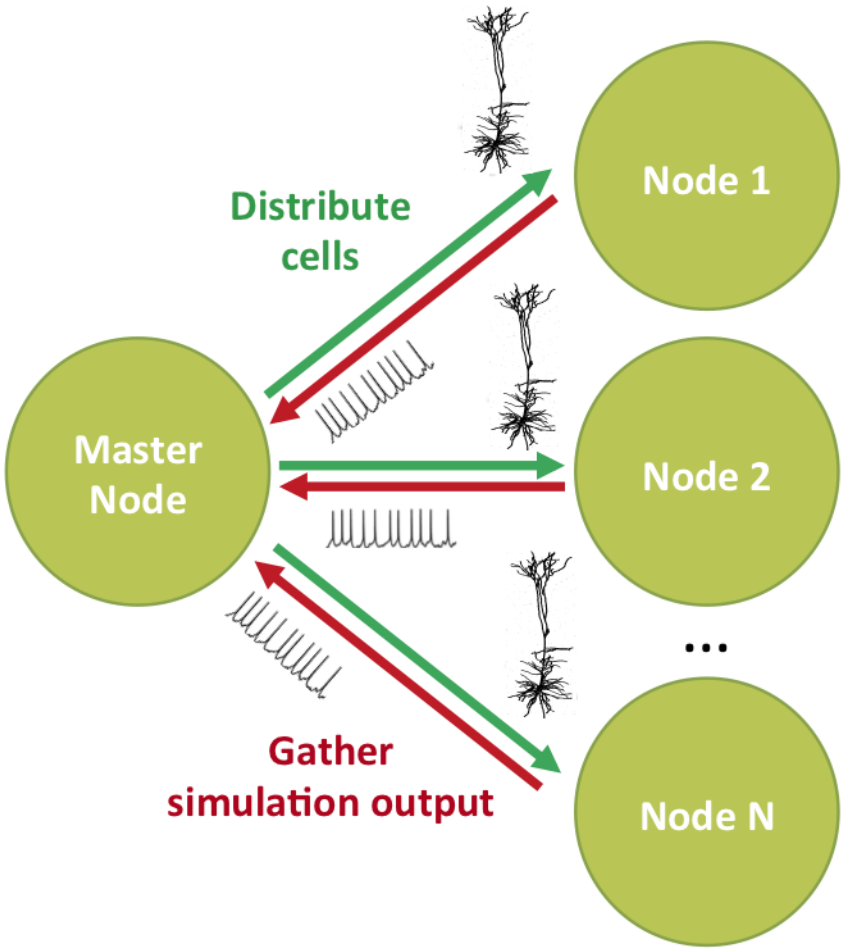




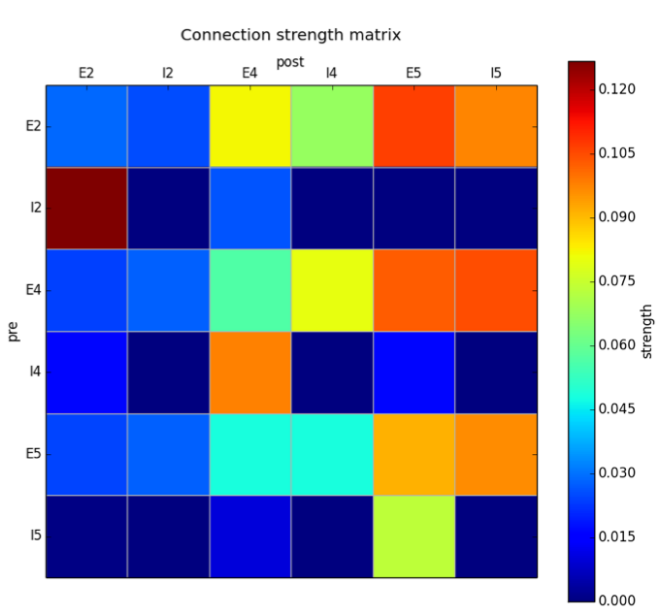
Parallel Simulation

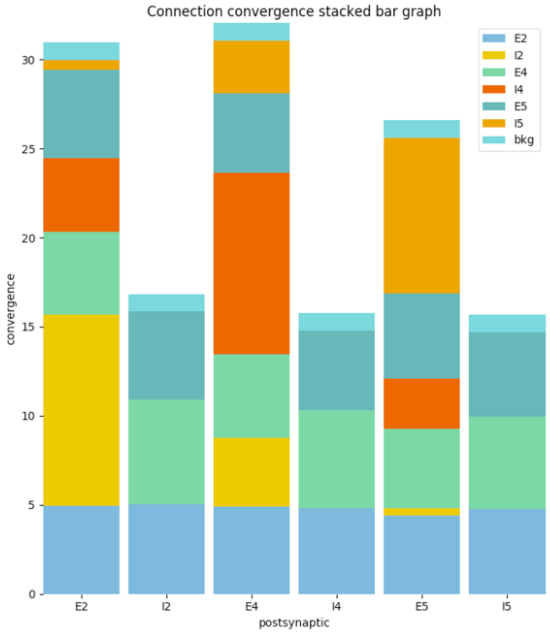
- Set up for MPI **parallel simulation** across multiple nodes (via NEURON simulator).

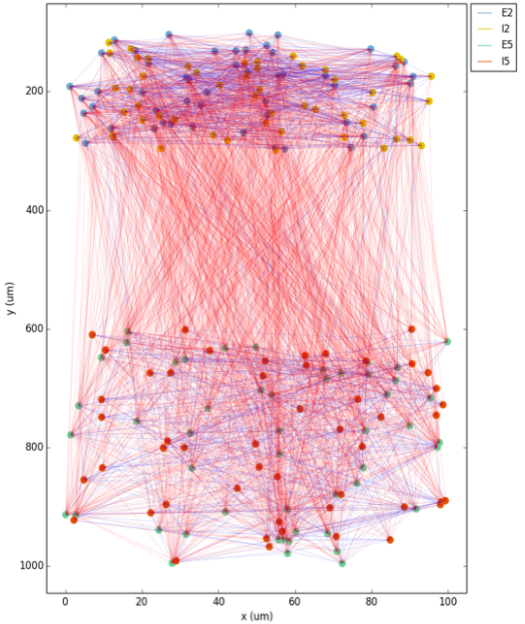
- Takes care of balanced **distribution** of cells and **gathering** of simulation output from nodes.

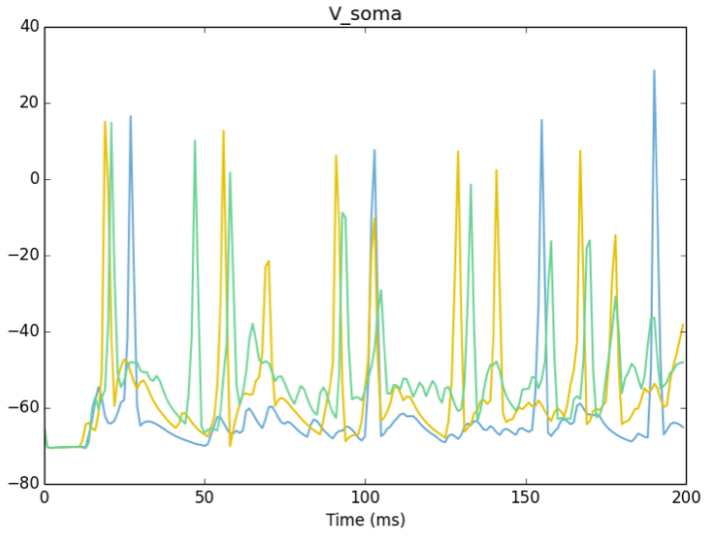


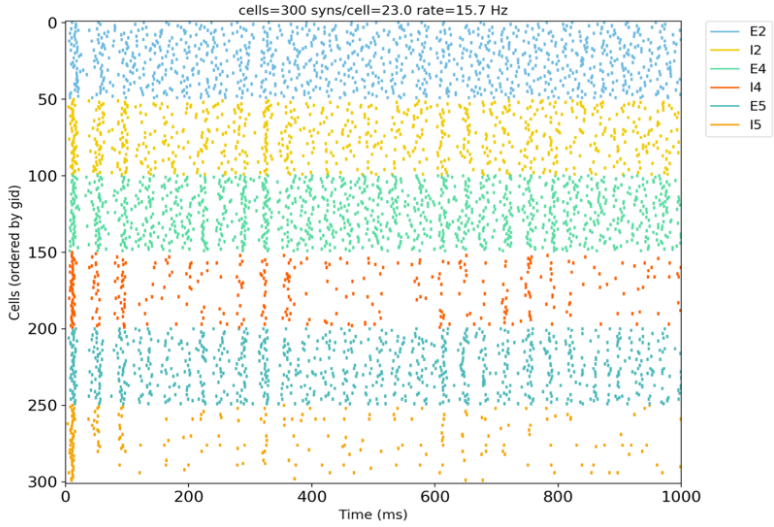
Analysis

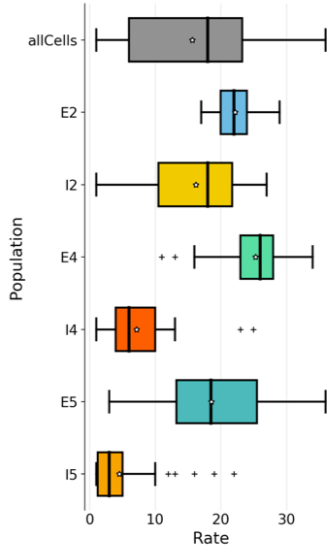

Connection strength matrix
post
E2
I2
E4
I4
E5
I5
pre
strength
0.120
0.105
0.090
0.075
0.060
0.045
0.030
0.015
0.000


Connection convergence stacked bar graph
E2
I2
E4
I4
E5
I5
bkg
convergence
postsynaptic

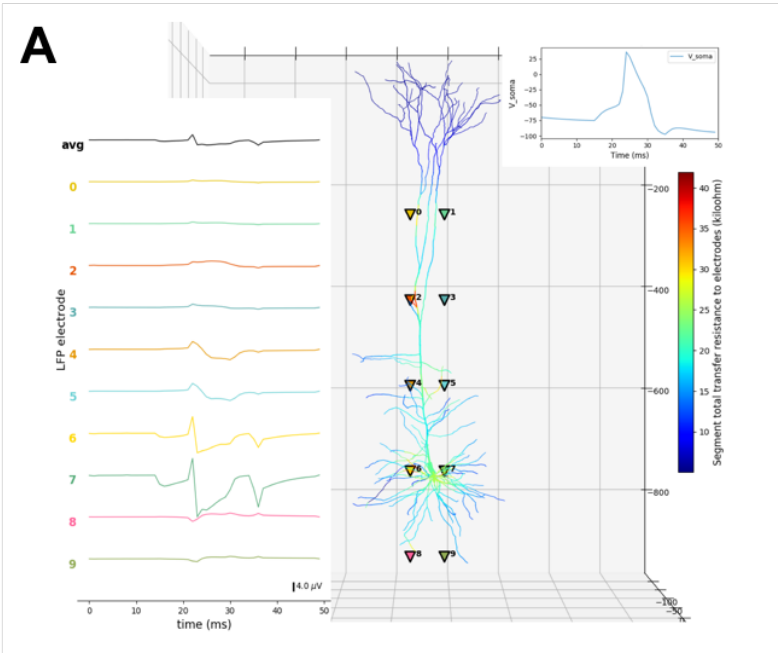

E2
I2
E5
I5
y (um)
x (um)


V_soma
Time (ms)


cells=300 syns/cell=23.0 rate=15.7 Hz
Cells (ordered by gid)
Time (ms)
E2
I2
E4
I4
E5
I5

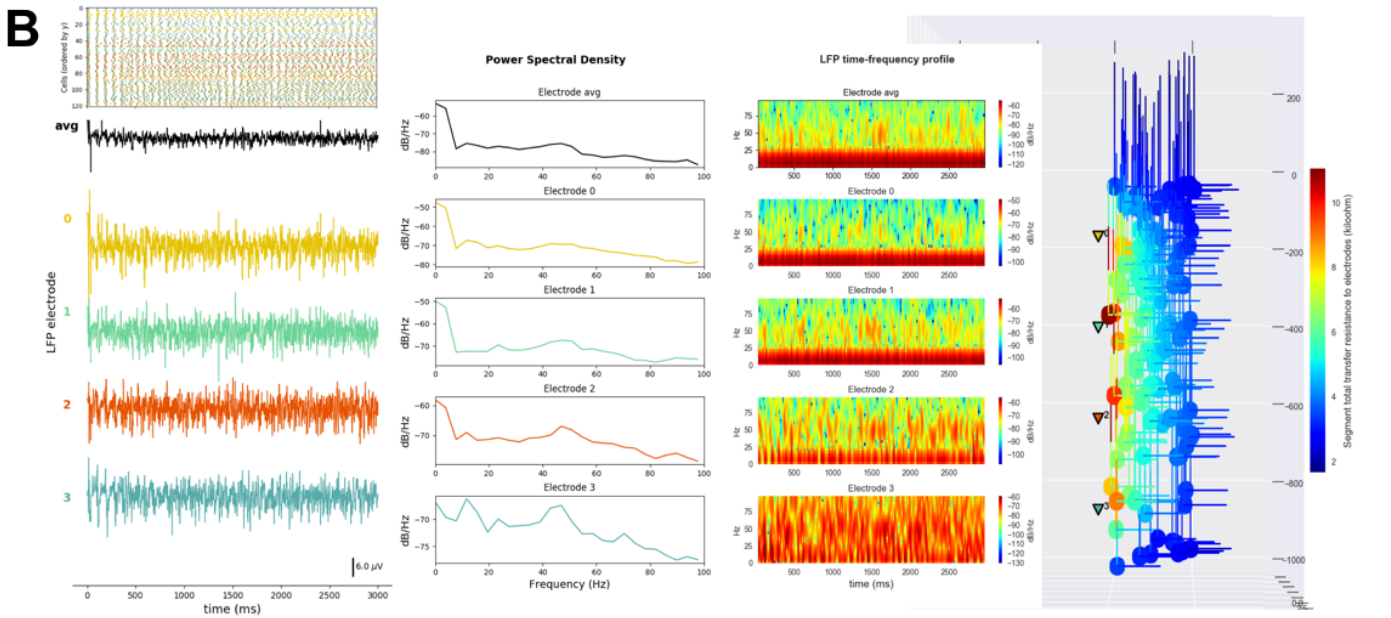

allCells
E2
I2
E4
I4
E5
I5
Population
Rate

LFP



Single cell

Network



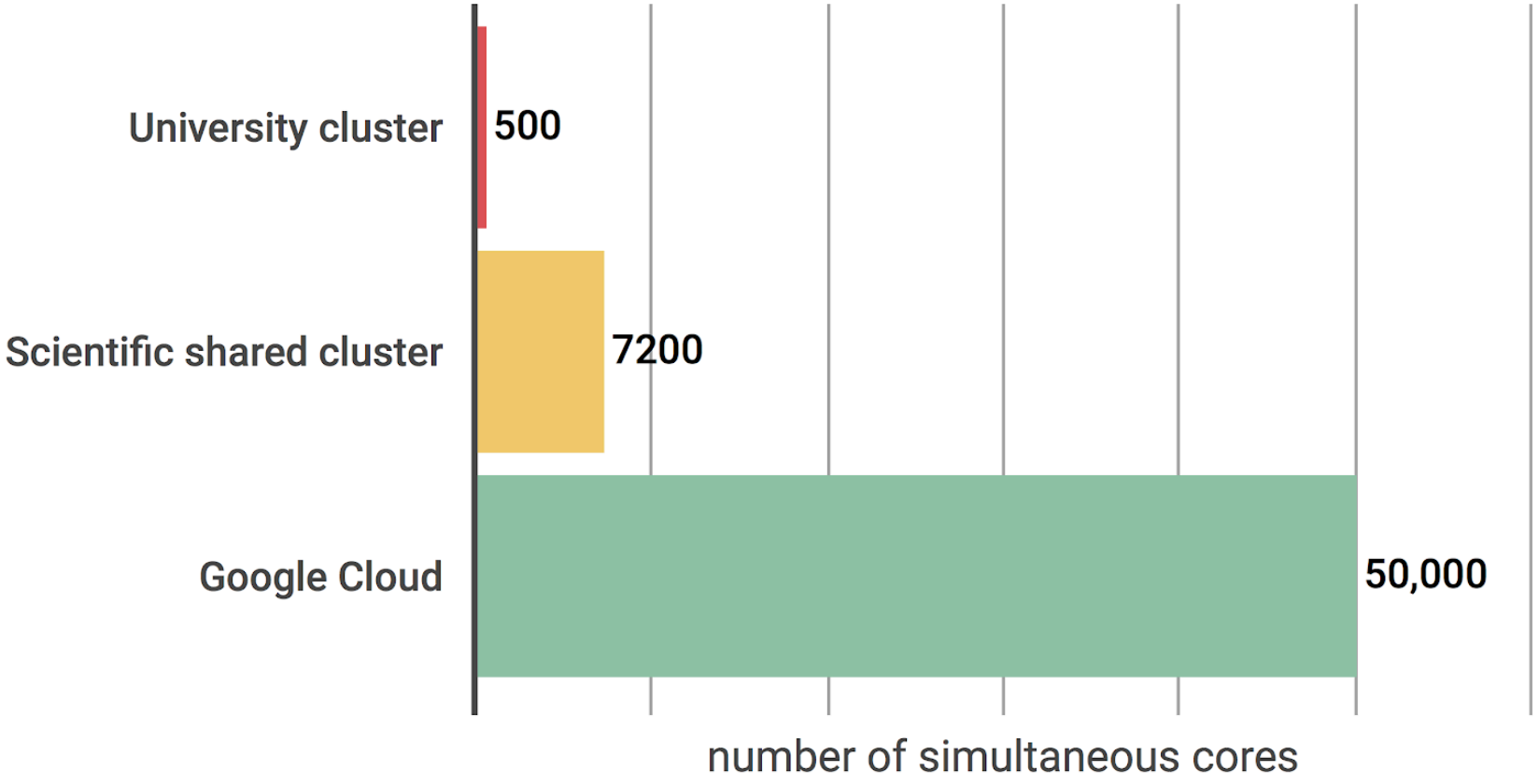
Batch Parallel Simulations

- **Easy specification** of parameters and range of values to explore in batch simulations.

- **Pre-defined, configurable** setups to automatically **submit jobs** in multicore machines (Bulletin board) or supercomputers (SLURM or PBS Torque)

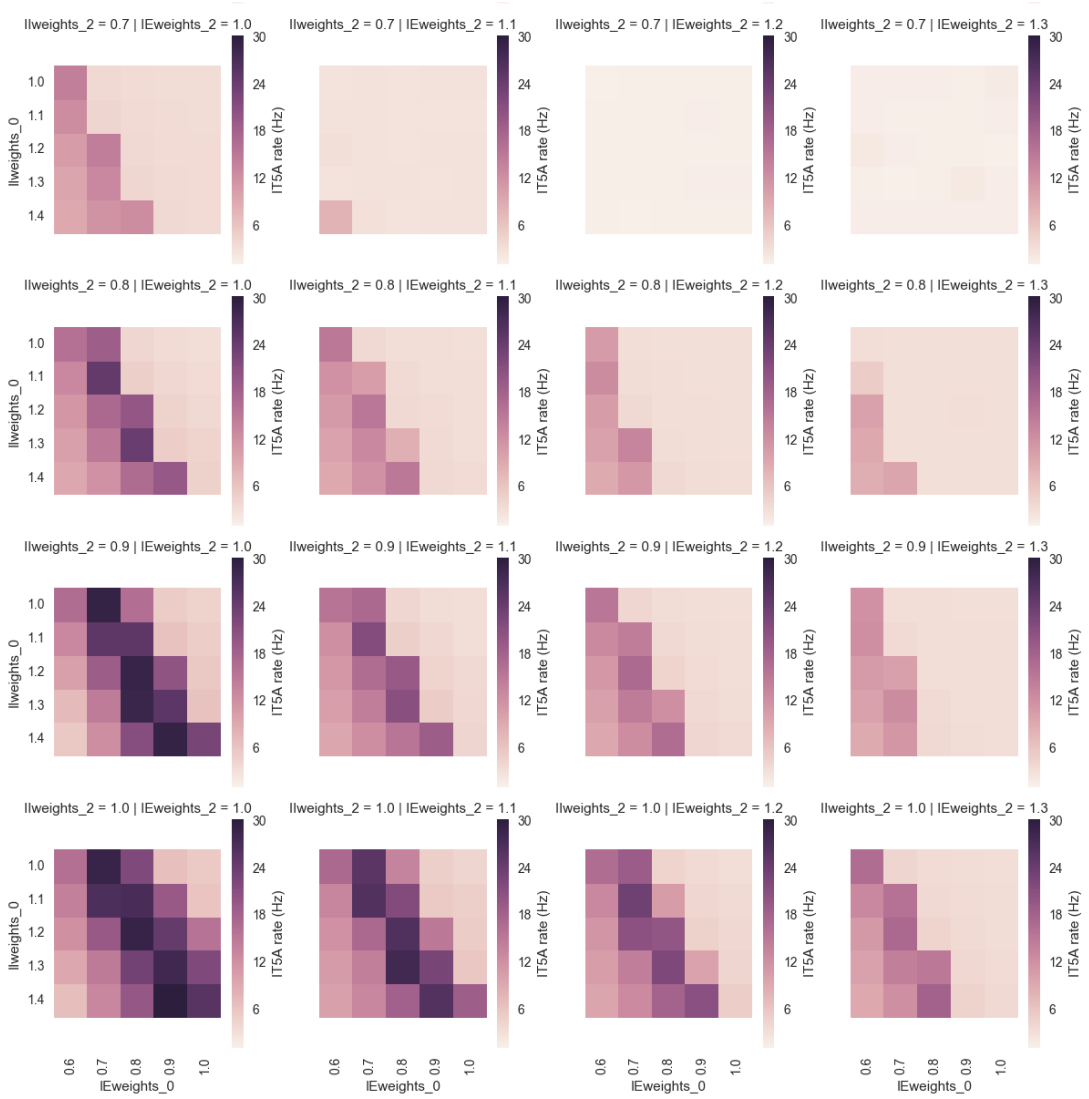
SDSC SAN DIEGO SUPERCOMPUTER CENTER

Batch Parallel Simulations



Batch Simulation Analysis

- **Analysis and visualization** of multidimensional batch simulation results.



Data saving and exporting

- **Save and load** high-level specifications, network instance, simulation config and/or simulation results.
- **Multiple formats** supported: pickle, JSON and Matlab (CSV and HDF5 in progress)
- **Export/import** network instance to/from **NeuroML.**
- Network instance **connections** included within each postsynaptic cell; 2 formats available:

long format (list of dicts)

```
"conns": [
  {
    "loc": 0.5,
    "weight": 0.002,
    "preGid": 137,
    "label": "PYR->PYR",
    "delay": 12.22717948106687,
    "sec": "soma",
    "synMech": "AMPA"
  },
  {
    "loc": 0.5,
    "weight": 0.002,
    "preGid": 193,
    "label": "PYR->PYR",
    "delay": 13.612656026095003,
    "sec": "soma",
    "synMech": "AMPA"
  },
```

short format (list of lists)

```
"compactConnFormat": [
  "preGid",
  "sec",
  "loc",
  "synMech",
  "weight",
  "delay"
],
```

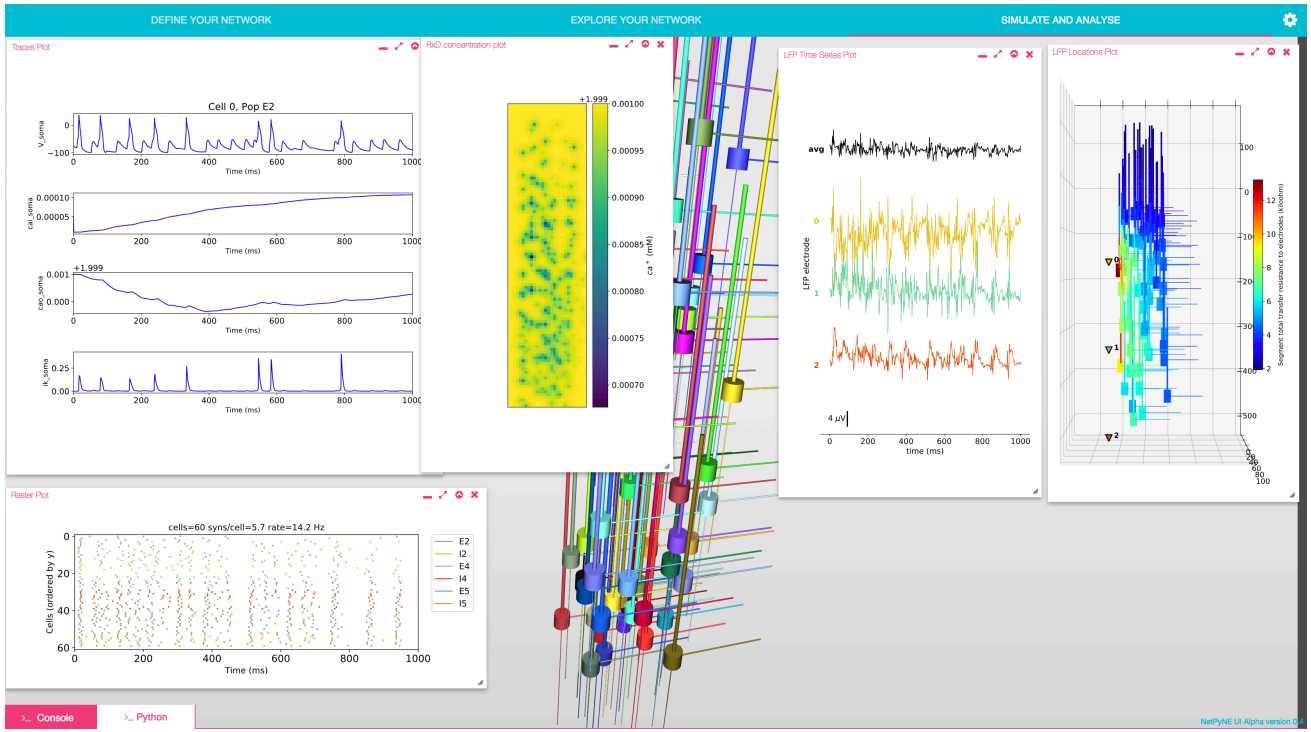
```
"conns": [
  [
    137,
    "soma",
    0.5,
    "AMPA",
    0.002,
    12.22717948106687
  ],
  [
    193,
    "soma",
    0.5,
    "AMPA",
    0.002,
    13.612656026095003
  ],
```

Data saving and exporting

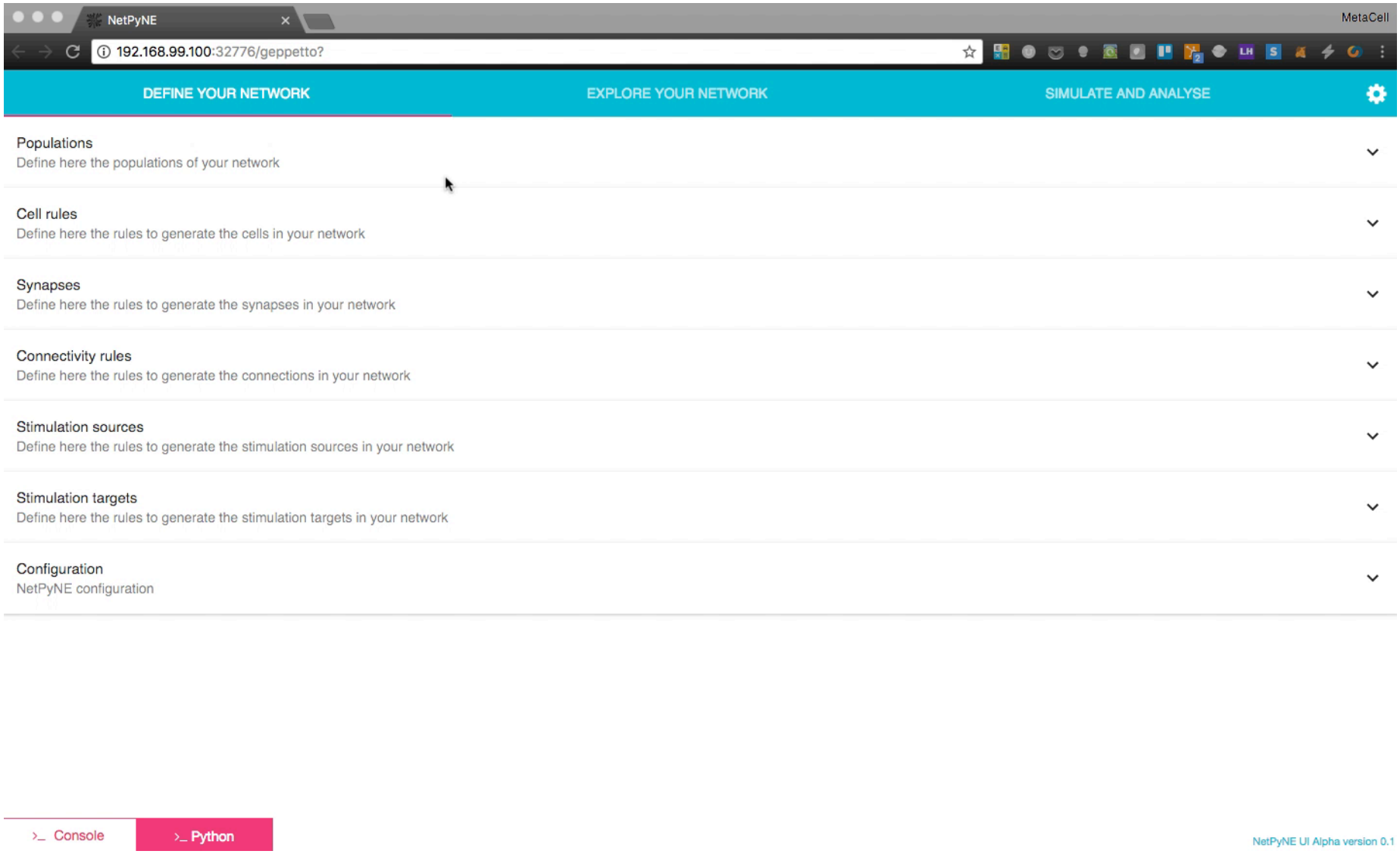
- **Features under development:**
  - Distributed saving to file on compute nodes (HDF5)
  - Saving to file at intervals/checkpoints
  - Import/Export to SONATA format

Development, simulation and analysis in GUI

- Useful for:
  1) Students/beginners
  2) Prototypng model (can export to script)
  3) Exploring/visualizing existing models

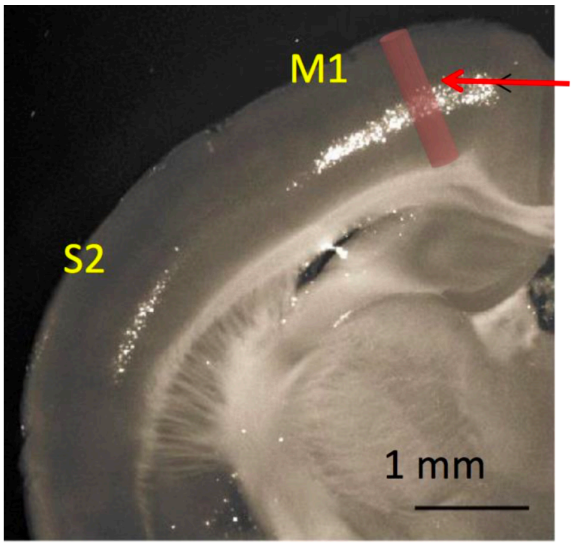


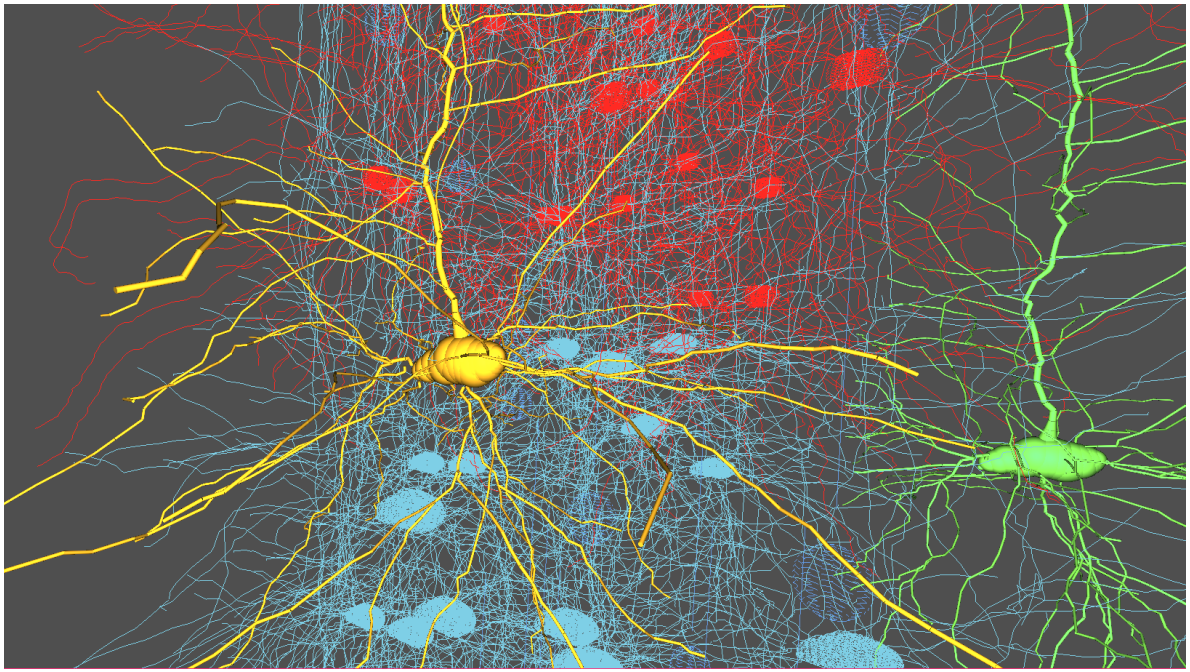
Development, simulation and analysis in GUI

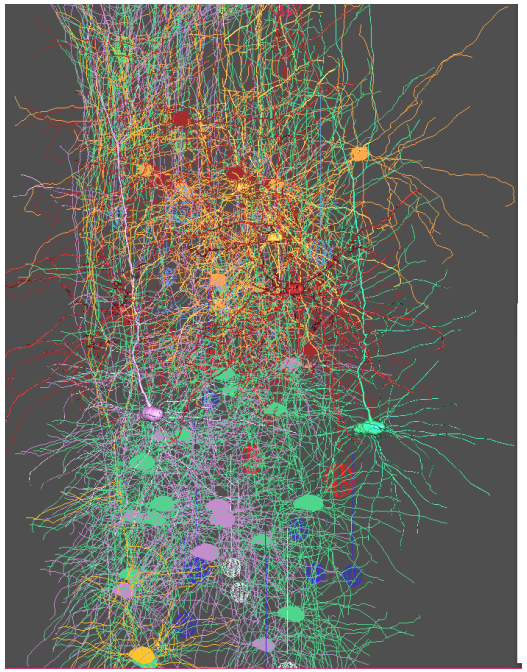


M1 microcircuits model

- Data-driven multiscale network model of **M1 microcircuits**
- Full scale cylindric volume of **300 µm** (diameter) x **1350 µm** (cortical depth)
- **~10,000 neurons** of 5 classes distributed in 15 populations
- ~**30M** synapses

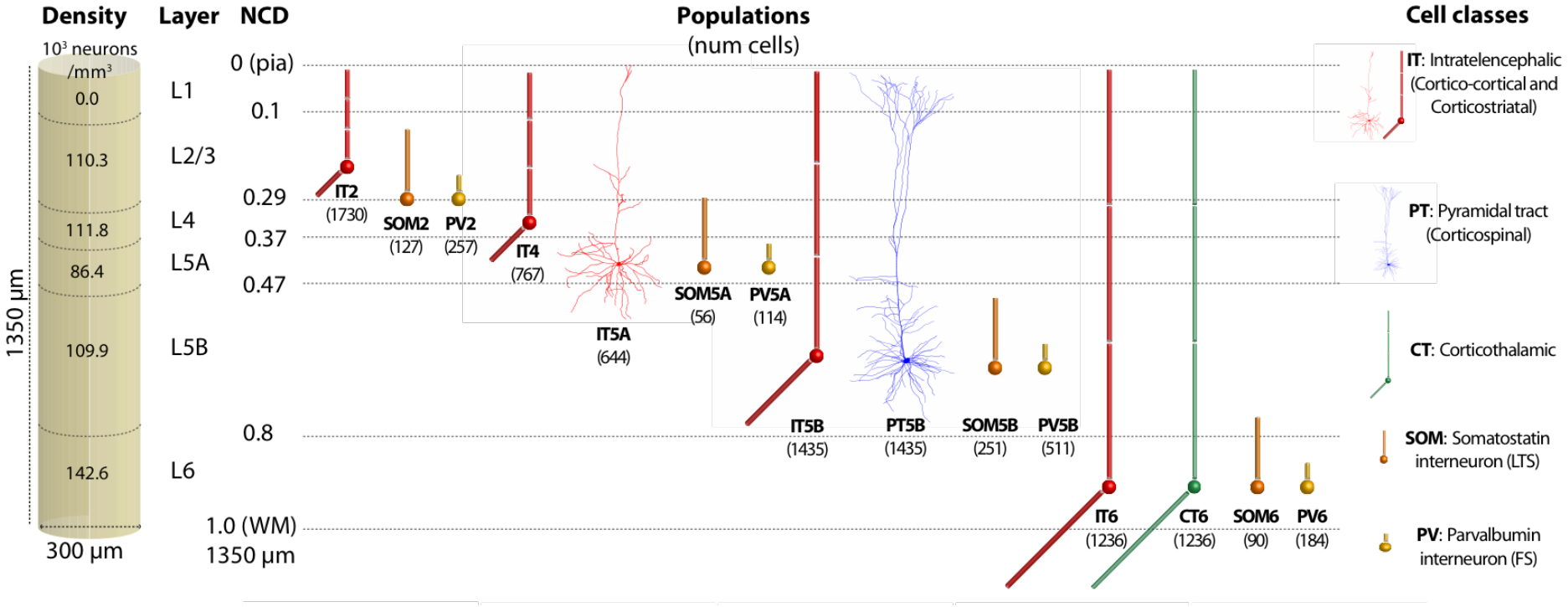




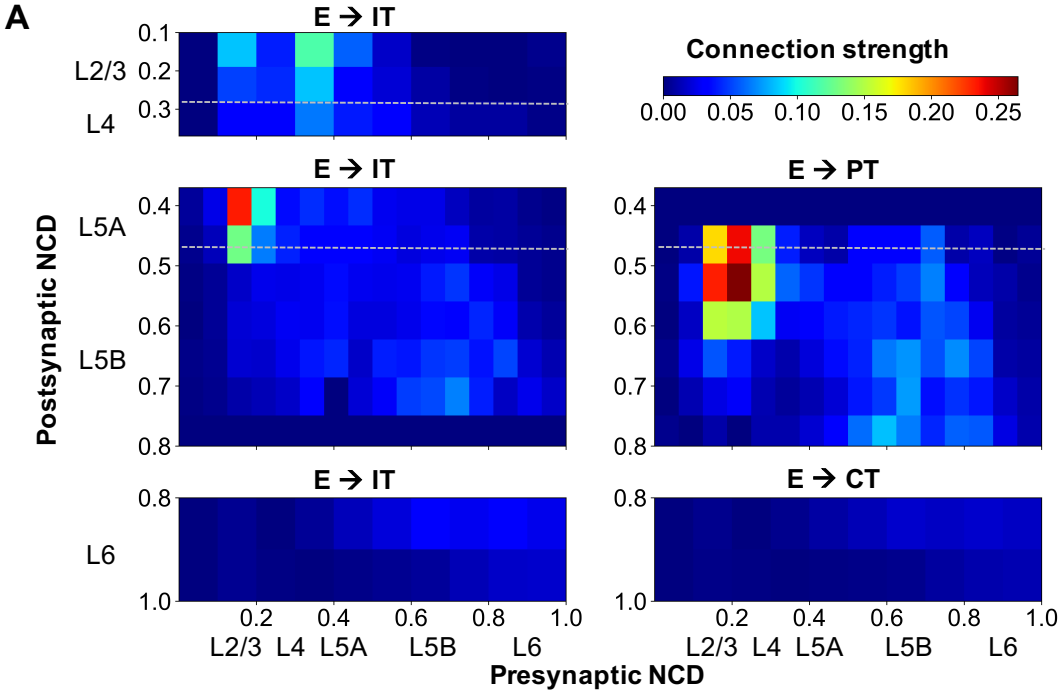


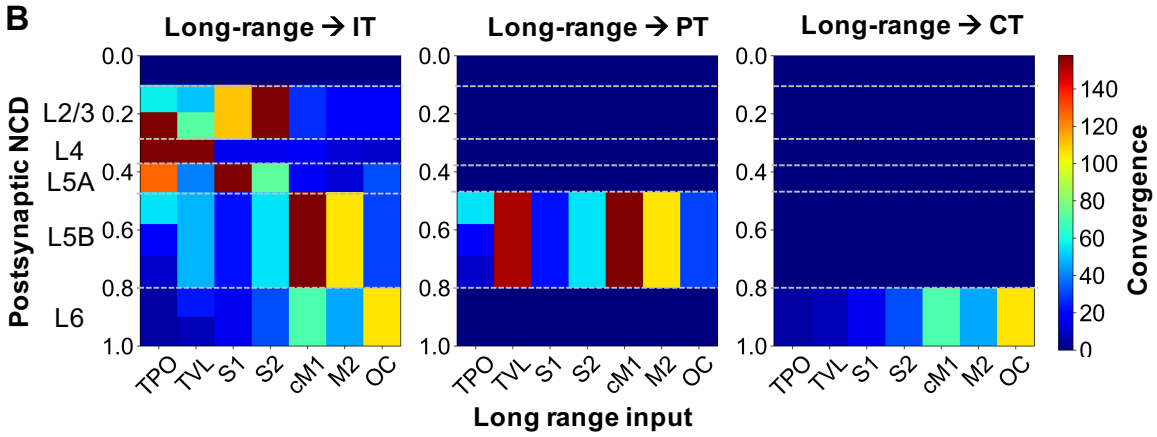
M1 microcircuits model

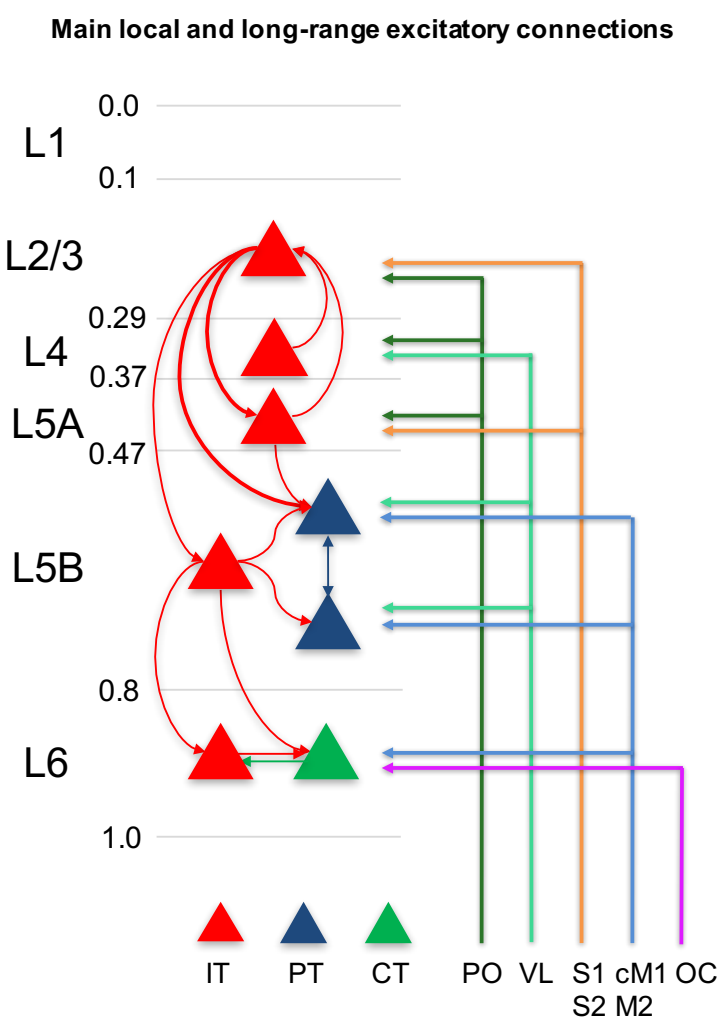
- Data-driven multiscale network model of **M1 microcircuits**



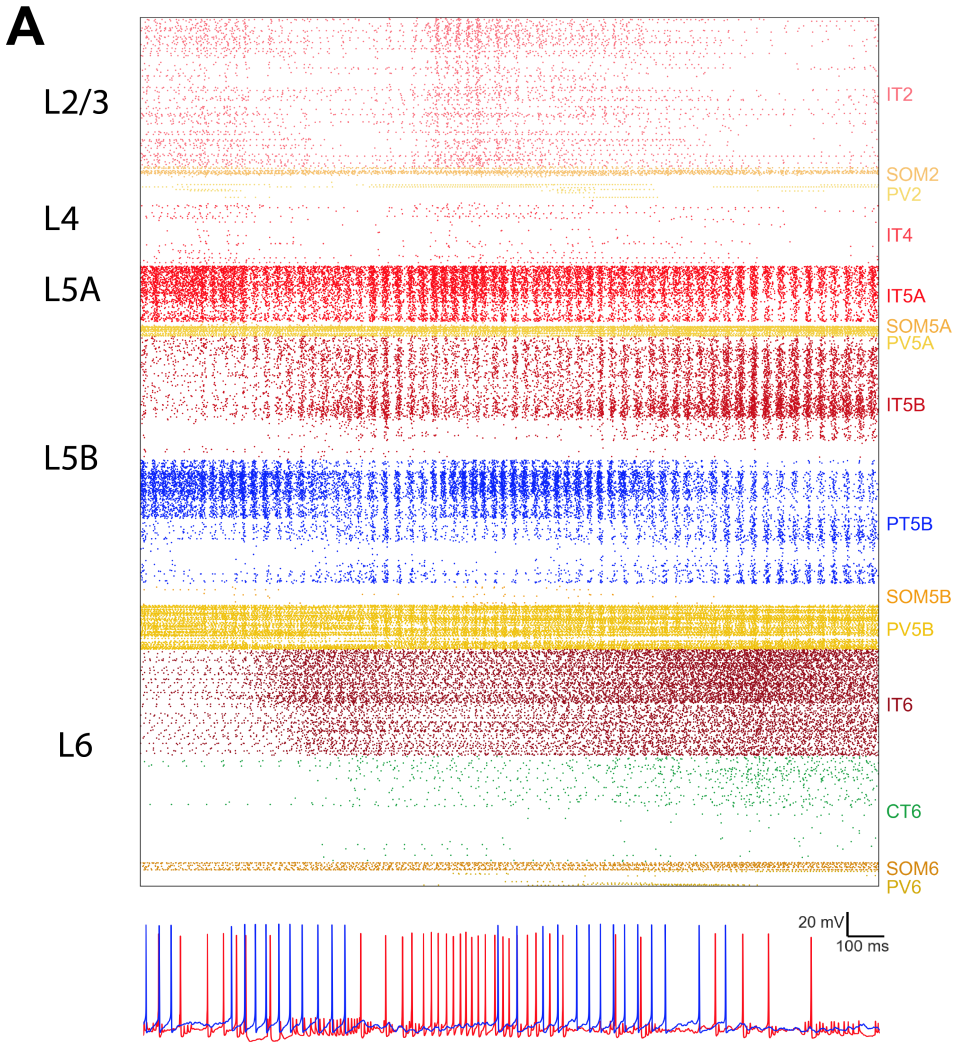
M1 microcircuits model

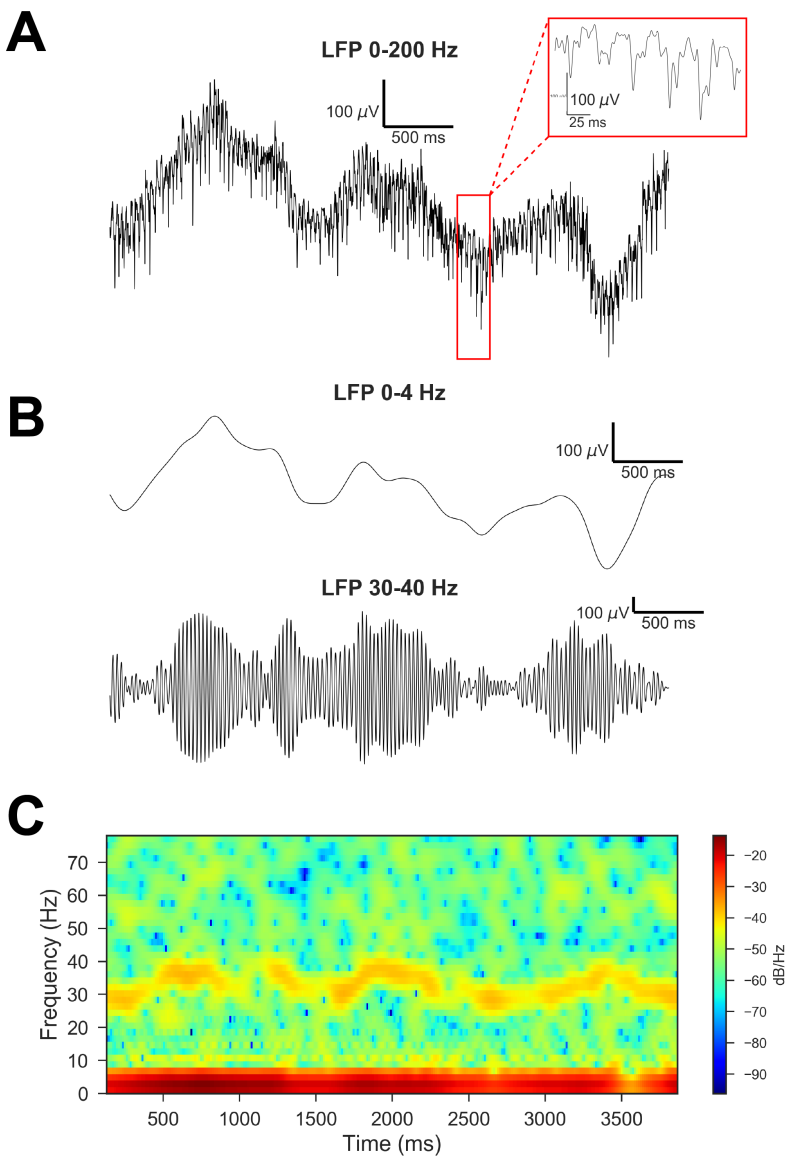






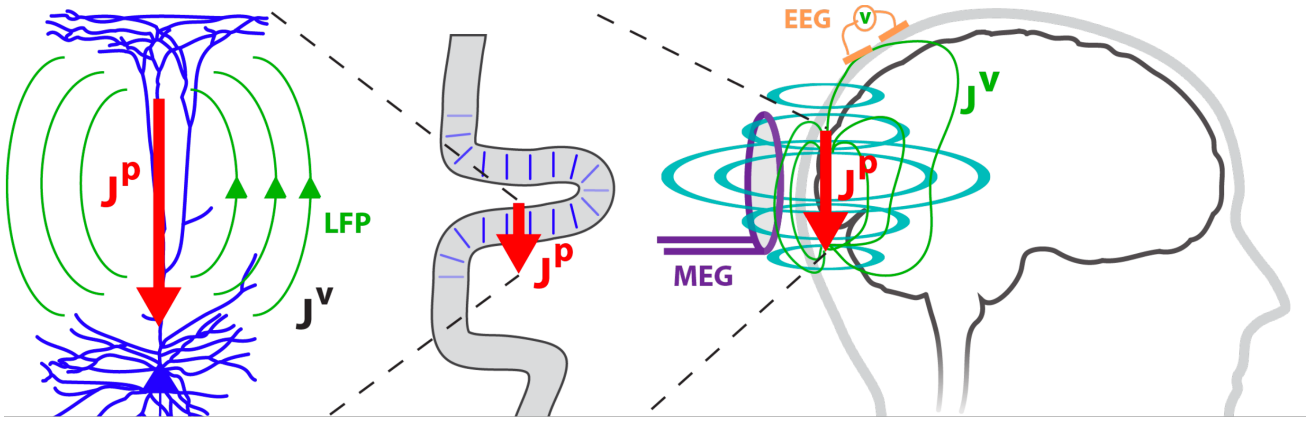
M1 microcircuits model





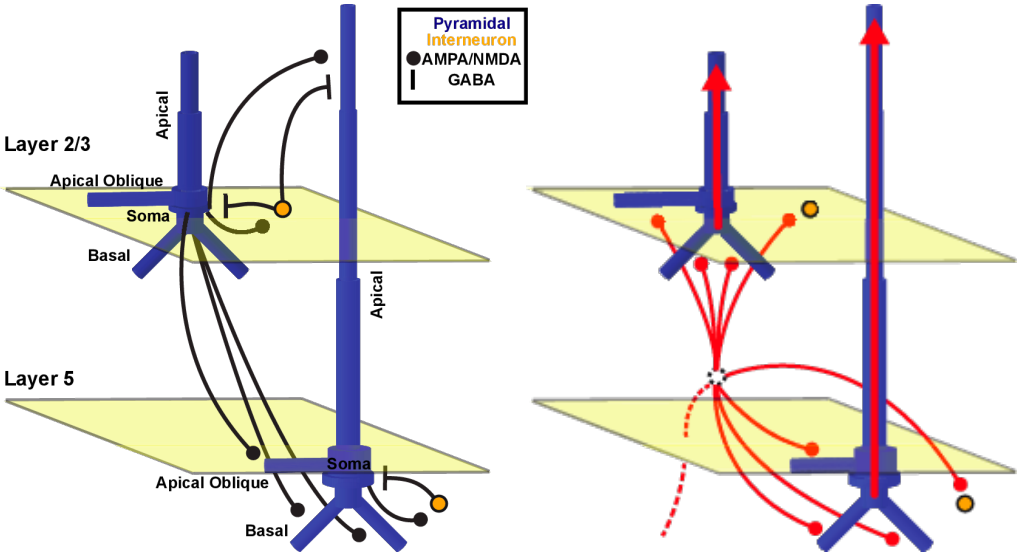
Human Neocortical Neurosolver

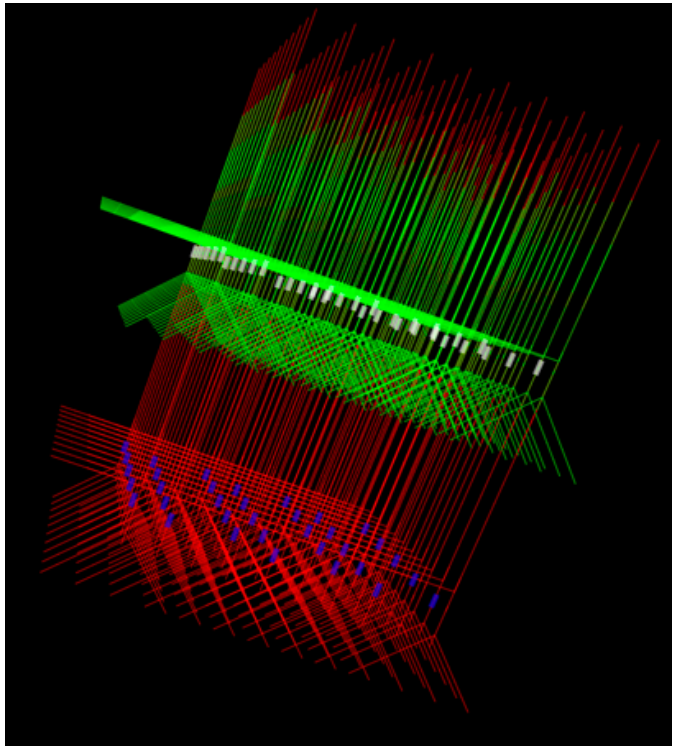
- Stephanie Jones (Brown University), PI of NIH BRAIN R01
- Tool to reproduce/understand EEG/MEG signals using biophysical circuit model



Human Neocortical Neurosolver

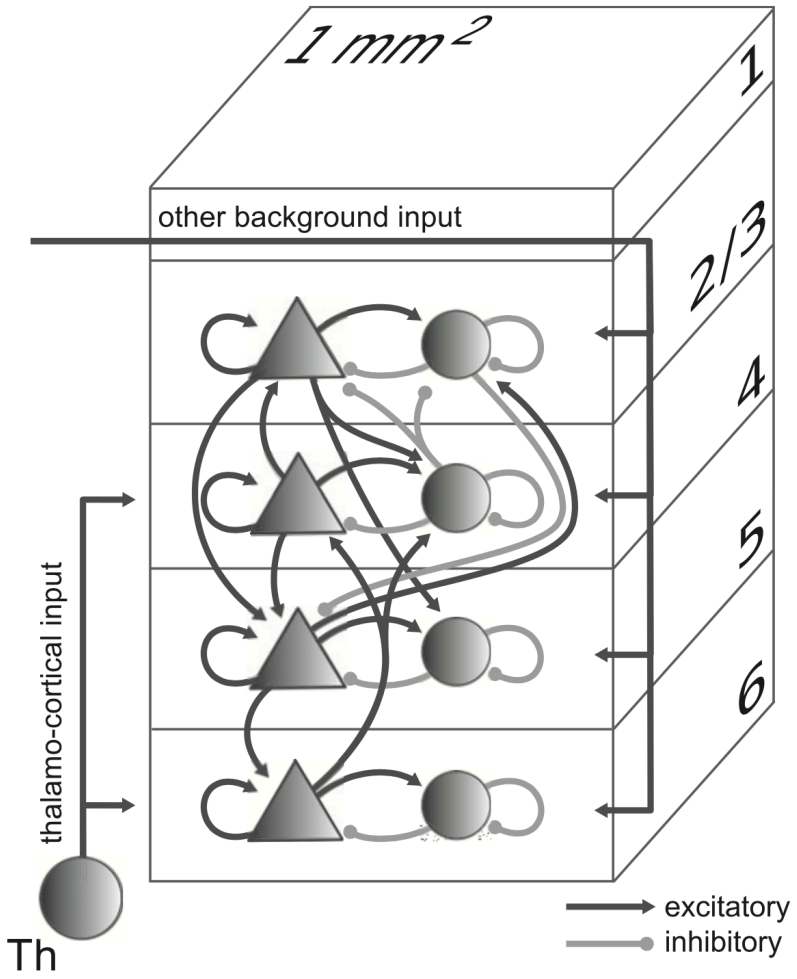
- ❑ Converted circuit model to NetPyNE
- ❑ Facilitate scaling, extension and customization





Potjan's & Diesmann model

- ~80k neurons (point model in NMODL)
- ~300M synapses
- Converted to NetPyNE
- Executed on Google Cloud



Potjan's & Diesmann model

NEST **NetPyNE/NEURON**

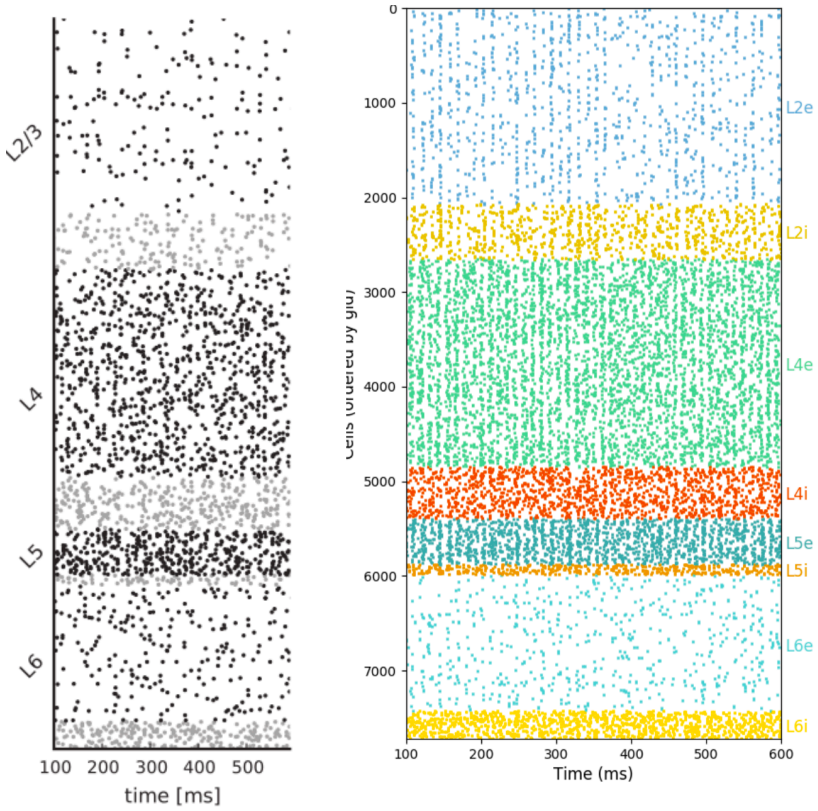


Figure 2: Raster plots network models scaled down to 100% of the original size (with around 80,000 neurons). NEST model on the left and NetPyNE model on the right.

NEST **NetPyNE/NEURON**

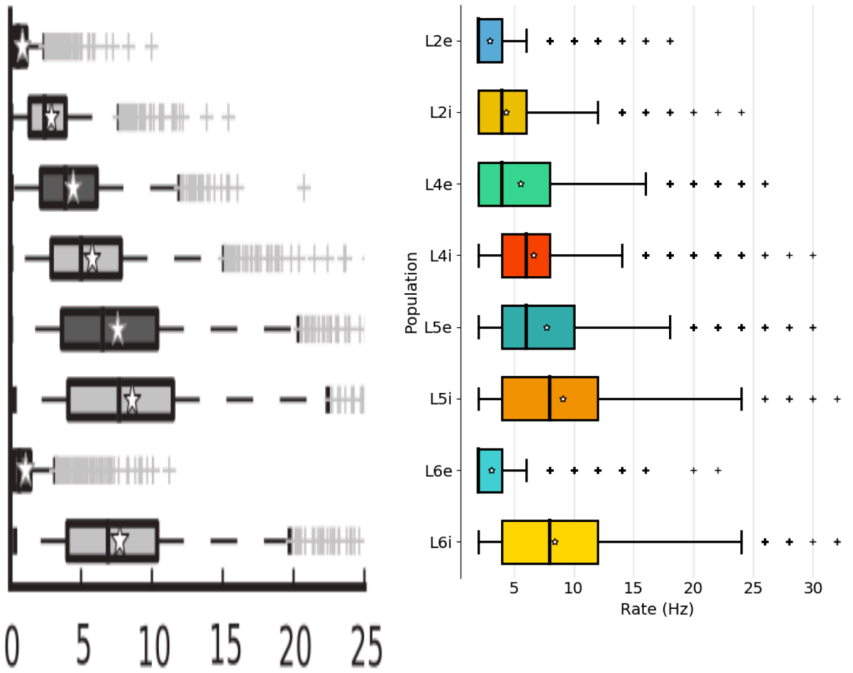
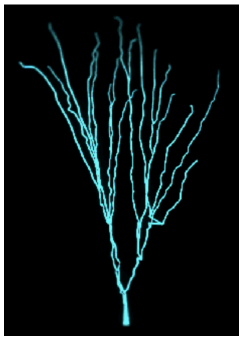
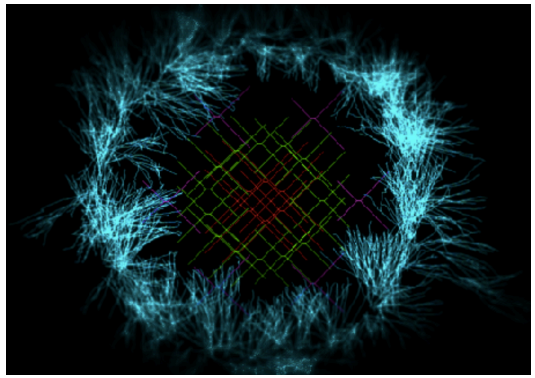


Figure 3: Statistics of spiking activity of all 8 neural populations for rescaling of the PD model to 100% of its original size (around 80,000 neurons). NEST model on the left and NetPyNe model on the right.

NetPyNE: Existing models



- Other models in progress:
  - Traub **thalamocortical** network (P. Gleeson, UCL / S. Crook, Arizona)
  - Hippocampus **CA3** (B. Tessler, SUNY DMC)
  - **Spinal cord** circuits (V. Caggiano, IBM Watson)
  - **Striatal** microcircuits (Hanbing/Christina Weaver, Franklin and Marshall College)
  - **V1** network (Vinicius/Antonio Roque, Sao Paulo University)
  - **Cerebellum** (Sergio Salines/Stefano Masoli, University of Pavia)
  - **Dentate Gyrus** (F. Rodriguez, SUNY DMC)
  - **Ischemia** in cortical network (Alex Seidenstein, SUNY DMC)
  - **TMS/tDCS** network (Aman Aberra, Duke University)
  - **LFP** oscillations (Christian Fink, Ohio Wesleyan)
  - **Dendritic** computations (Birgit Kriener, Oslo)
  - Thalamocortical **epilepsy** network (Andrew Knox, Cincinatti Hospital)
  - Full list of 43 models: https://drive.google.com/open?id=1bkWHakgZoEkYIkzrAS8sIKCvO5PSuUXLLRjNdN2pseY



NetPyNE: Acknowledgments

- **Contributors:**
  - Salvador Dura-Bernal (SUNY DMC)
  - Ben Suter (Northwestern)
  - Matteo Cantarelli (Metacell Ltd)
  - Adrian Quintana (EyeSeeTea Ltd)
  - Dario del Piano (Metacell Ltd)
  - Facundo Rodriguez (SUNY DMC)
  - Cliff Kerr (Sydney)
  - Padraig Gleeson (UCL)
  - Robert McDougal (Yale)
  - Michael Hines (Yale)
  - Gordon MG Shepherd (Northwestern)
  - William Lytton (SUNY DMC)

- **Lab website:** www.neurosimlab.org
- **NetPyNE Website:** www.netpyne.org
- **NetPyNE-UI Website:** www.github.com/MetaCell/NetPyNE-UI
- **Github:** www.github.com/Neurosim-lab/netpyne (open source development; contributions welcome)

- **Funding:**
  - NIH Grant U01EB017695
  - NIH Grant R01EB022903
  - NIH Grant R01MH086638
  - NYS Grant DOH01-C32250GG-3450000

SUNY Downstate Medical Center

Northwestern University

UCL