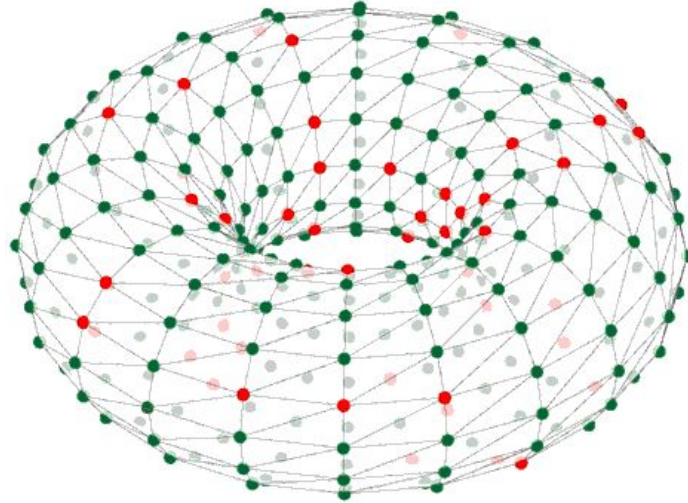


SpiNNaker Tutorial



Accessing SpiNNaker

Accessing SpiNNaker via Jupyter - Create a Collaboratory

<https://wiki.ebrains.eu/>

Log in to Wiki

The screenshot shows a web browser window with the URL `wiki.ebrains.eu/bin/view/Main/`. The page header includes the EBRAINS Collaboratory logo and navigation links for Collabs, Documentation, Support, Chat, and Community. The **Log-in** link is circled in red. The main content area features a green button labeled "Getting started" and an illustration of people collaborating around a large screen displaying a graph. Below the illustration, the text "Highlighted collabs" is visible, along with a "View all" link. The browser's address bar at the bottom shows `https://wiki.ebrains.eu/bin/view/Collabs/`.

Create a Collaboratory

The screenshot shows a web browser window with the URL `wiki.ebrains.eu/bin/view/Main/`. The page header includes the EBRAINS Collaboratory logo and a navigation menu with the following items: **Collabs** (circled in red), Documentation, Support, Chat, Community, a search icon, a notification bell, a user profile icon, and a **Log-out** button.

The main content area features the following text:

Collaborate.
Create reproducible science.
Discover EBRAINS services at work.
From anywhere.

Below this text is a green button labeled **Getting started**.

To the right of the text is an illustration of five stylized figures interacting with a large digital interface. One figure is sitting on top of the interface, another is pointing at a chart, and others are holding up documents or devices. The interface shows a line graph and a bar chart.

At the bottom of the page, there is a search bar with the placeholder text "Search word in all wiki pages" and a green search button.

Create a Collaboratory

The screenshot shows a web browser window with two tabs: 'JupyterHub' and 'Collabs - HBP Wiki'. The address bar shows the URL 'wiki.ebrains.eu/bin/view/Collabs/'. The page header includes the 'EBRAINS Collaboratory' logo and navigation links for 'Collabs', 'Documentation', 'Support', 'Chat', 'Community', and 'Log-out'. The main content area is titled 'Collab Search' and includes a search bar with the placeholder text 'Search word in collab titles & descriptions. You can use AND, OR and - operators.' and a green 'Search' button. Below the search bar, there are filter sections for 'FILTER COLLABS' (with 'Your favourites' selected) and 'YOUR ROLE' (with 'Admin' and 'Editor' selected). A central section titled 'Showing highlighted collabs' asks 'Do you want to promote your collab?' and features a workflow diagram. The diagram shows an 'INPUT' section with 'NII QuickNII and Virtualize' (Registration of images to reference brain atlas space), 'Customized atlas merge and coordinate files', and 'Nifty Quantifier' (Quantification of segmental voxels per brain atlas region). The 'OUTPUT' section shows 'Quantification reports' and 'Brain atlas coordinates'. A 'Bivette' step is also shown for 'Image representation to extract the labelling'. To the right, there is a section for 'EBRAINS Curation Services' with the text 'How to share your data'. A red circle highlights the 'Create a collab' button in the top right corner of the page.

Create a Collaboratory

JupyterHub Collabs - HBP Wiki

wiki.ebrains.eu/bin/view/Collabs?dbaction=create

Description

Describe what are the objectives of this collaboration workspace and the general context. This field is searched by the collab search tool. Use keywords in the description that will help find this collab.

Visibility

Private
Private collabs are only accessible to people added to the collab's Team. The collab contents are not viewable by other users.

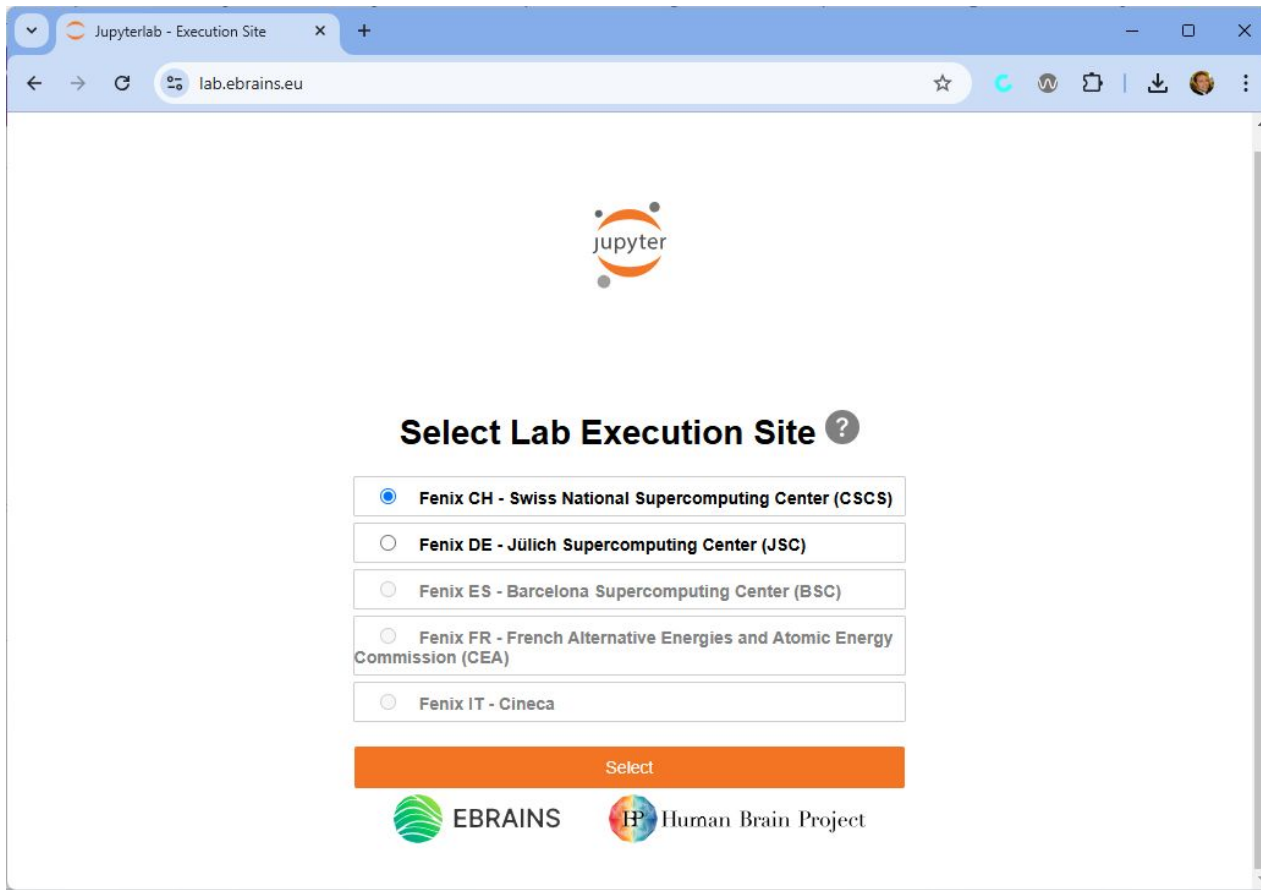
Public
Public collabs are accessible to everybody, even without an EBRAINS account. However, only members of the collab's Team with editor or admin permissions can modify anything in the collab.

Create Collab

Accessing SpiNNaker via Jupyter - Go to EBRAINS Lab

<https://lab.ebrains.eu/>

EBRAINS Lab - Choose a Site



The screenshot shows a web browser window with the address bar displaying "lab.ebrains.eu". The page content includes the Jupyter logo at the top center, followed by the heading "Select Lab Execution Site" with a help icon. Below this is a list of five radio button options for different execution sites. The first option, "Fenix CH - Swiss National Supercomputing Center (CSCS)", is selected. At the bottom of the selection area is an orange "Select" button. The footer contains the logos for EBRAINS and the Human Brain Project.

Jupyterlab - Execution Site



lab.ebrains.eu

jupyter

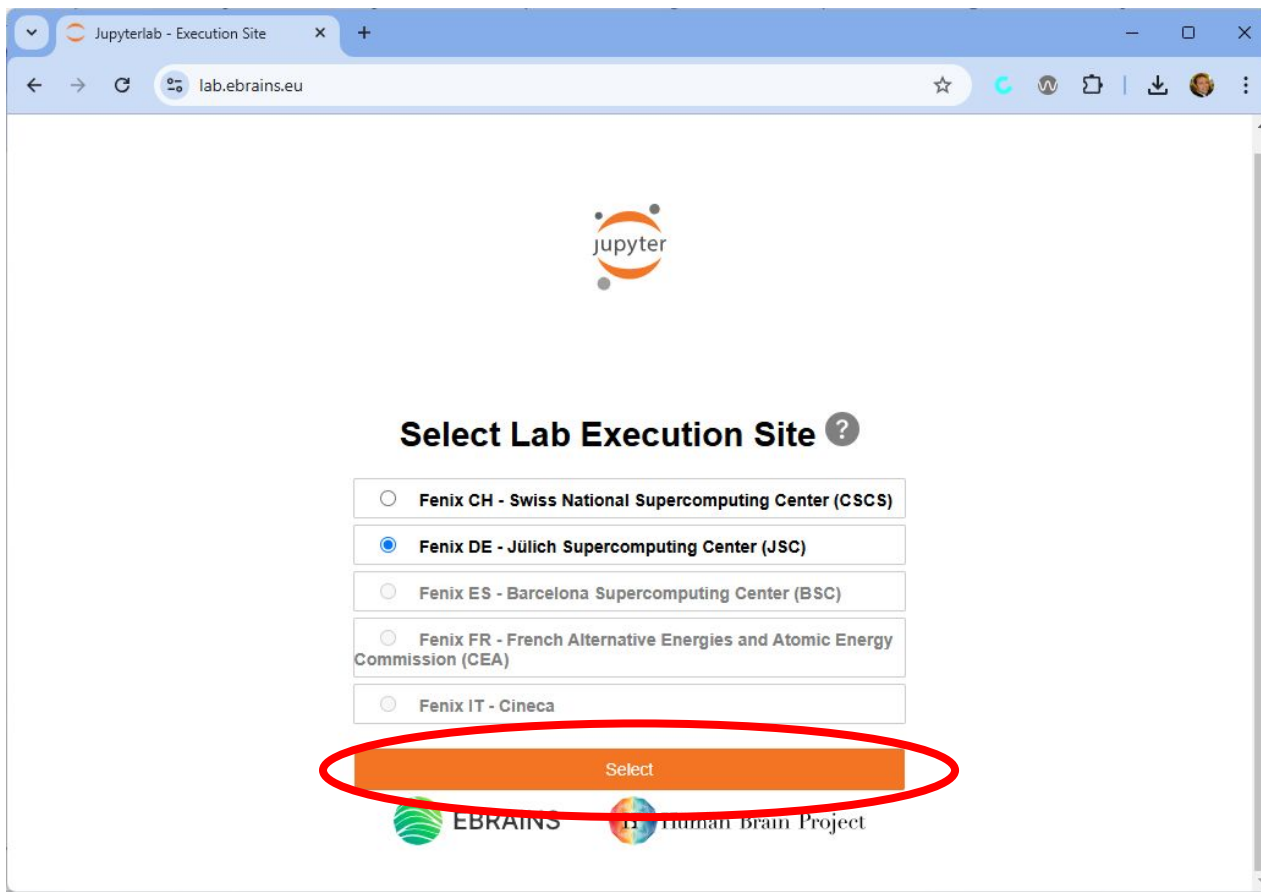
Select Lab Execution Site ?

- Fenix CH - Swiss National Supercomputing Center (CSCS)
- Fenix DE - Jülich Supercomputing Center (JSC)
- Fenix ES - Barcelona Supercomputing Center (BSC)
- Fenix FR - French Alternative Energies and Atomic Energy Commission (CEA)
- Fenix IT - Cineca

Select

 EBRAINS  Human Brain Project

EBRAINS Lab - Choose a Site



The screenshot shows a web browser window with the address bar displaying "lab.ebrains.eu". The page content includes the Jupyter logo at the top center. Below it, the heading "Select Lab Execution Site" is followed by a list of five radio button options. The second option, "Fenix DE - Jülich Supercomputing Center (JSC)", is selected. At the bottom of the selection area, an orange "Select" button is circled in red. The footer of the page features the EBRAINS logo and the Human Brain Project logo.

Jupyterlab - Execution Site

lab.ebrains.eu

jupyter

Select Lab Execution Site ?

- Fenix CH - Swiss National Supercomputing Center (CSCS)
- Fenix DE - Jülich Supercomputing Center (JSC)
- Fenix ES - Barcelona Supercomputing Center (BSC)
- Fenix FR - French Alternative Energies and Atomic Energy Commission (CEA)
- Fenix IT - Cineca

Select

EBRAINS Human Brain Project

EBRAINS Lab - Choose a Kernel

The screenshot shows the JupyterLab interface in a web browser. The browser address bar displays `lab.jsc.ebrains.eu/user/rowley/lab`. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. A memory usage indicator shows 256 / 2048 MB. On the left, there is a file browser sidebar with a search bar and a table of files:

Name	Last Modified
drive	7 months ago
shared	7 months ago

The main area is titled 'Launcher' and displays a grid of kernel options under the heading 'Notebook'. The kernels are arranged in a grid:

- Python 3 (ipykernel)
- EBRAINS_release_v0.1_202109
- EBRAINS-22.07
- EBRAINS-22.10
- EBRAINS-23.02
- EBRAINS-23.06
- EBRAINS-23.09
- EBRAINS-24.04** (highlighted with a red circle)
- EBRAINS-experimental
- R 3.6.3
- R-EBRAINS-23.02
- R-EBRAINS-23.06
- R-EBRAINS-23.09
- R-EBRAINS-24.04
- R-EBRAINS-

At the bottom of the interface, there is a status bar showing 'Simple' mode, a memory usage indicator of 187.44 / 2048.00 MB, and a 'Launcher' button.

EBRAINS Lab - Choose a Kernel

The screenshot displays the JupyterLab web interface. The browser address bar shows the URL: `lab.jsc.ebrains.eu/user/rowley/lab/tree/Untitled.ipynb`. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. A memory usage indicator shows 313 / 2048 MB. On the left, a file browser sidebar shows a search bar and a list of files and folders:

Name	Last Modified
/	
drive	7 months ago
shared	7 months ago
Untitled.ipynb	seconds ago

The main workspace contains a code editor for 'Untitled.ipynb' with the kernel 'EBRAINS-24.04' selected. The editor shows a single line of code: `[]:`. The bottom status bar indicates 'Simple' mode, 'EBRAINS-24.04 | Idle', and 'Mem: 234.07 / 2048.00 MB'.

Find your Collab: /shared/...

The screenshot displays the JupyterLab interface. The browser address bar shows the URL `lab.jsc.ebrains.eu/user/rowley/lab/tree/shared`. The left sidebar contains a file browser with a search bar and a list of folders. The folder `/ shared /` is circled in red. The right sidebar shows the 'Launcher' view, which displays a grid of notebooks under the 'shared' directory. The notebooks are organized by kernel type: Python 3 (ipykernel) and R. The Python notebooks include `EBRAINS_release_v0.1_202109`, `EBRAINS-22.07`, `EBRAINS-22.10`, `EBRAINS-23.02`, `EBRAINS-23.06`, `EBRAINS-23.09`, and `EBRAINS-24.04`. The R notebooks include `EBRAINS-experimental`, `R 3.6.3`, `R-EBRAINS-23.02`, and `R-EBRAINS-23.06`.

Name	Last Modified
/ shared /	
Andrew Public Test	a year ago
Andrew Rowley Default Quota	13 days ago
Andrew Rowley Default Quota 2	13 days ago
Andrew Test Collab	2 days ago
Andrew's Testing Collab	5 years ago
AngoraPy	2 years ago
API Catalogue	3 years ago
Async-Neuromorph	a year ago
Atlas pipeline workflow	a year ago
Atlas registration of neuronal mor...	a year ago
Atlas Viewer Development	3 years ago
Basal Ganglia Mean Field	2 years ago
BASSES Hands-on Session I - Han...	2 years ago
BASSES Hands-on Session II - Run...	2 years ago
BASSES Hands-on Session III - Sim...	2 years ago
BASSES Workshop Rome Event M...	2 years ago
Bayesian Virtual Epileptic Patient	2 years ago
BCCN-TVb-workshop-2022	2 years ago

Clone SpiNNaker Repository

The screenshot shows the JupyterLab interface. The top menu bar includes File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. The Git icon is circled in red. The file browser on the left shows a directory structure with files like SpiNNaker_01_test.ipynb and SpiNNaker_Collab_synfire.ipynb. The launcher on the right shows a grid of notebook environments, including Python 3 (ipykernel) and various EBRAINS and R environments.

Launcher

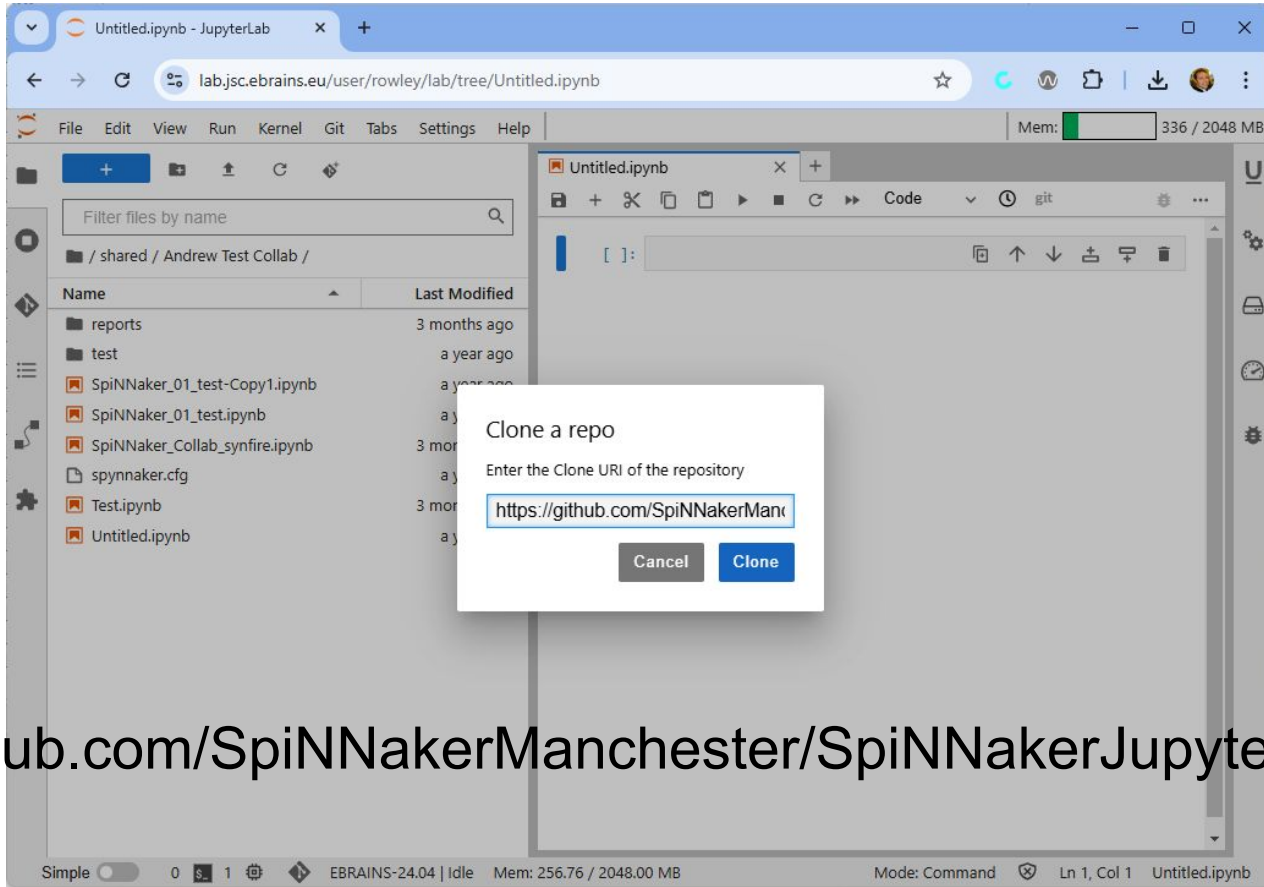
shared/Andrew Test Collab

Notebook

Environment	Python 3 (ipykernel)	EBRAINS_release_v0.1_202109	EBRAINS-22.07	EBRAINS-22.10
EBRAINS-23.02	EBRAINS-23.06	EBRAINS-23.09	EBRAINS-24.04	
EBRAINS-experimental	R 3.6.3	R-EBRAINS-23.02	R-EBRAINS-23.06	

Simple 1 0 Mem: 260.07 / 2048.00 MB Launcher

Clone SpiNNaker Repository



<https://github.com/SpiNNakerManchester/SpiNNakerJupyterExamples>

Clone SpiNNaker Repository

The screenshot displays the JupyterLab interface. The browser address bar shows the URL: `lab.jsc.ebrains.eu/user/rowley/lab/tree/shared/Andrew%20Test%20Collab/SpiNNakerJupyterE...`. The interface includes a menu bar (File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help) and a memory usage indicator (336 / 2048 MB).

On the left, the file browser shows the directory structure: `... / Andrew Test Collab / SpiNNakerJupyterExamples /`. The file list includes:

Name	Last Modified
00.Setup	seconds ago
01.RunningPyNNSimulations	seconds ago
02.LiveInputAndOutput	seconds ago
integration_tests	seconds ago
spinnaker_jupyter_examples	seconds ago
LICENSE	seconds ago
pyproject.toml	seconds ago
README.md	seconds ago
setup.cfg	seconds ago
setup.py	seconds ago

The '00.Setup' directory is circled in red. On the right, the 'Launcher' panel shows the path `shared/Andrew Test Collab/SpiNNakerJupyterExamples` and a 'Notebook' section with a grid of kernels:

- Python 3 (ipykernel)
- EBRAINS_release_v0.1_202109
- EBRAINS-22.07
- EBRAINS-22.10
- EBRAINS-23.02
- EBRAINS-23.06
- EBRAINS-23.09
- EBRAINS-24.04
- EBRAINS-experimental

The bottom status bar shows 'Simple' mode, a CPU icon, and memory usage: `Mem: 256.77 / 2048.00 MB`.

Open Setup Notebook

The screenshot displays the JupyterLab interface. The browser address bar shows the URL: `lab.jsc.ebrains.eu/user/rowley/lab/tree/shared/Andrew%20Test%20Collab/SpINNAkerJupyterE...`. The interface includes a menu bar (File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help) and a toolbar. On the left, a file browser shows the directory structure: `/ ... / SpiNNakerJupyterExamples / 00.Setup /`, with `Setup.ipynb` selected. The main area shows the notebook content, which includes a title `Jupyter Setup` and a code cell with the following Python code:

```

def check_should_create():
    """
    Detects if there is an existing cfg file and if so if it
    :return:
    """
    if not os.path.isfile(FULL_PATH):
        return True
    if OVERRIDE_EXISTING:
        return True

```

A 'Select Kernel' dialog box is overlaid on the notebook. It contains the text 'Select kernel for: "Setup.ipynb"' and a dropdown menu showing 'Python 3 (ipykernel)'. Below the dropdown are two buttons: 'No Kernel' and 'Select'.

The status bar at the bottom indicates: `Simple`, `0`, `1`, `No Kernel | Initializing`, `Mem: 256.77 / 2048.00 MB`, `Mode: Command`, `Ln 1, Col 1`, and `Setup.ipynb`.

Open Setup Notebook

The screenshot shows a JupyterLab environment with a notebook titled 'Setup.ipynb'. A modal dialog box titled 'Select Kernel' is centered on the screen, prompting the user to 'Select kernel for: "Setup.ipynb"'. The dropdown menu in the dialog shows 'EBRAINS-24.04' as the selected option. Below the dropdown are two buttons: 'No Kernel' and 'Select'. The background notebook content includes a section titled 'Jupyter Setup' with the following text:

The code below is needed to create a sPyNNaker configuration file, which is needed on the first run of the software. It will be necessary to start the Lab (it can also be added as the kernel if desired). A future image may already have the need to do this.

```

def check_should_create():
    """
    Detects if there is an existing cfg file and if so if it
    :return:
    """
    if not os.path.isfile(FULL_PATH):
        return True
    if OVERRIDE_EXISTING:
        return True
  
```

The status bar at the bottom of the JupyterLab interface indicates 'No Kernel | Initializing' and 'Mem: 257.83 / 2048.00 MB'.

Open Setup Notebook - Run

The screenshot shows a JupyterLab browser window. The browser address bar displays the URL: `lab.jsc.ebrains.eu/user/rowley/lab/tree/shared/Andrew%20Test%20Collab/SpiNNakerJupyterE...`. The interface includes a top menu bar with options: File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help. On the right side of the top bar, it shows memory usage: Mem: 411 / 2048 MB.

The left sidebar contains a file browser with a search bar labeled "Filter files by name". Below it, the file structure is shown as `/ SpiNNakerJupyterExamples / 00.Setup /`. A table lists files with columns "Name" and "Last Modified":

Name	Last Modified
Setup.ipynb	a minute ago

The main area displays a notebook titled "Setup.ipynb". The toolbar at the top of the notebook has a red circle around the "Run" button (represented by a double right-pointing arrow). The notebook content includes a section titled "Jupyter Setup" with the following text:

The code below is needed to create a sPyNNaker configuration file, which is needed on the first run of the software. It will be necessary to run this each time you start the Lab (it can also be added as the first cell in other notebooks if desired). A future image may already include this file and avoid the need to do this.

```
[ ]: import configparser
import os

OVERRIDE_EXISTING = False
FULL_PATH = os.path.expanduser('~/.spynaker.cfg')

def check_should_create():
    """
    Detects if there is an existing cfg file and if so if it
    :return:
    """
    if not os.path.isfile(FULL_PATH):
        return True
    if OVERRIDE_EXISTING:
        return True
```

The bottom status bar shows: Simple, 0, 2, EBRAINS-24.04 | Idle, Mem: 315.66 / 2048.00 MB, Mode: Command, Ln 1, Col 1, Setup.ipynb.

Open Setup Notebook - Run

The screenshot shows a JupyterLab browser window with the following elements:

- Browser Tab:** Setup.ipynb - JupyterLab
- Address Bar:** lab.jsc.ebrains.eu/user/rowley/lab/tree/shared/Andrew%20Test%20Collab/SpiNNakerJupyterE...
- Menu Bar:** File, Edit, View, Run, Kernel, Git, Tabs, Settings, Help
- Memory:** 411 / 2048 MB
- File Browser (Left):** Shows a directory structure with 'Setup.ipynb' selected, last modified 'a minute ago'.
- Launcher (Top):** Shows 'Setup.ipynb' as the active notebook.
- Notebook Content:**
 - Title:** Jupyter Setup
 - Text:** The code below is needed to create a sPyNNaker configuration file, which is needed on the first run of the software. It will be necessary to run this each time you start the Lab (it can also be added as the ...). A future image may already ... to do this.
 - Code:**

```

def check_should_create():
    """
    Detects if there is an existing cfg file and if so if it
    :return:
    """
    if not os.path.isfile(FULL_PATH):
        return True
    if OVERRIDE_EXISTING:
        return True
        
```
- Dialog Box (Center):**
 - Title:** Restart Kernel?
 - Text:** Do you want to restart the current kernel? All variables will be lost.
 - Buttons:** Cancel (grey), Restart (red)
- Status Bar (Bottom):** Simple, 0 s, 2, EBRAINS-24.04 | Idle, Mem: 315.66 / 2048.00 MB, Mode: Command, Ln 1, Col 1, Setup.ipynb

Open Setup Notebook - Run

The screenshot shows a JupyterLab environment with the following components:

- Browser Tab:** Setup.ipynb - JupyterLab
- Address Bar:** lab.jsc.ebrains.eu/user/rowley/lab/tree/shared/Andrew%20Test%20Collab/SpiNNakerJupyterE...
- File Browser:** Shows the directory structure: / SpiNNakerJupyterExamples / 00.Setup / Setup.ipynb (Last Modified: 2 minutes ago).
- Code Editor:** Contains Python code for creating a configuration file:


```
# Make sure there is a spynaker.cfg file
if check_should_create():
    with open(FULL_PATH, 'w') as cfg:
        cfg.write("[Machine]\n")
        cfg.write("spalloc_server = https://spinnaker.cs.man.\n")
        cfg.write("\n")
        cfg.write("[Java]\n")
        cfg.write("use_java=True\n")
        cfg.write("\n")
        cfg.write("[Reports]\n")
        cfg.write("read_provenance_data = False\n")
        print(f"New {FULL_PATH} created")
```
- Terminal Output:** A red circle highlights the message:


```
/opt/app-root/src/.spynaker.cfg already exists.
To replace it change OVERRIDE_EXISTING to True
```
- Contents Panel:** Lists two sections:
 1. Running PyNN Simulations on SpiNNaker
 2. Live Input and Output
- Status Bar:** Shows "Simple" mode, 0 files, 2 tabs, EBRAINS-24.04 kernel, and memory usage of 315.93 / 2048.00 MB.

Changing Kernel

The screenshot shows a JupyterLab environment. On the left is a file browser with a search bar and a list of files:

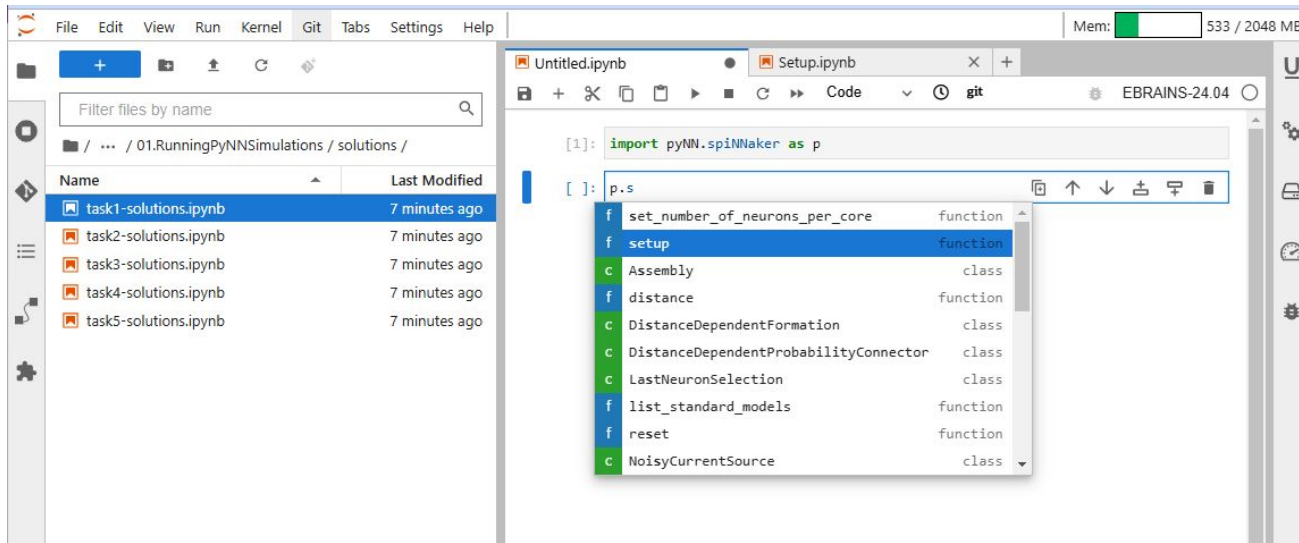
Name	Last Modified
task1-solutions.ipynb	14 minutes ago
task2-solutions.ipynb	14 minutes ago
task3-solutions.ipynb	14 minutes ago
task4-solutions.ipynb	14 minutes ago
task5-solutions.ipynb	14 minutes ago

The main area is a code editor with two tabs: 'Untitled.ipynb' and 'Setup.ipynb'. The code in the 'Setup.ipynb' tab is:

```
[1]: import pyNN.spiNNaker as p
[ ]: p.setup()
```

In the top right corner of the code editor, the kernel name 'EBRAINS-24.04' is displayed and circled in red. The status bar at the bottom shows 'Simple' mode, 'EBRAINS-24.04 | Idle', and 'Ln 1, Col 9'.

Jupyter Autocomplete



Jupyter Function Help

The screenshot shows a Jupyter Notebook window with two tabs: 'Untitled.ipynb' and 'Setup.ipynb'. The active cell in 'Setup.ipynb' contains the following code:

```
[1]: import pyNN.spiNNaker as p
[ ]: p.setup()
```

A help window is displayed over the code, showing the function signature and docstring for `p.setup()`:

```
Signature:
p.setup(
    timestep=0.1,
    min_delay='auto',
    max_delay=None,
    database_socket_addresses=None,
    time_scale_factor=None,
    n_chips_required=None,
    n_boards_required=None,
    **extra_params,
)
Docstring:
The main method needed to be called to make the PyNN 0.8 setup.
Needs to be called before any other function

:param timestep:
    the time step of the simulations in microseconds;
    if `None`, the configuration value is used
:type timestep: float or None
:param min_delay: the minimum delay of the simulation
```



Install Libraries

The screenshot shows a JupyterLab environment with a file browser on the left and a terminal window on the right. The file browser shows a directory structure with files like '00.Setup', '01.RunningPyNNSimulations', '02.LiveInputAndOutput', 'integration_tests', 'spinnaker_jupyter_examples', 'LICENSE', 'pyproject.toml', 'README.md', 'setup.cfg', and 'setup.py'. The terminal window shows the execution of the command `pip install gym` and its output, which includes the following text:

```
[1]: %pip install gym

Defaulting to user installation because normal site-packages is not writeable
Collecting gym
  Downloading gym-0.26.2.tar.gz (721 kB)
    721.7/721.7 kB 6.6 MB/s eta 0:0
0:00:00:0100:01
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: numpy>=1.18.0 in /srv/main-spack-instance-2402/spack/var/spack/environments/ebbrains-24-04/.spack-env/_view/dilrqvcful4yhyqdzwr424x6zawgsds/lib/python3.8/site-packages (from gym) (1.23.5)
Requirement already satisfied: cloudpickle>=1.2.0 in /srv/main-spack-instance-2402/spack/var/spack/environments/ebbrains-24-04/.spack-env/_view/dilrqvcful4yhyqdzwr424x6zawgsds/lib/python3.8/site-packages (from gym) (2.2.0)
Collecting gym-notices>=0.0.4 (from gym)
  Downloading gym_notices-0.0.8-py3-none-any.whl (3.0 kB)
Requirement already satisfied: importlib-metadata>=4.8.0 in /srv/main-spack-instance-2402/spack/var/spack/environments/ebbrains-24-04/.spack-env/_view/dilrqvcful4yhyqdzwr424x6zawgsds/lib/python3.8/site-packages (from gym) (6.6.0)
Requirement already satisfied: zipp>=0.5 in /srv/main-spack-instance-2402/spack/var/spack/environments/ebbrains-24-04/.spack-env/_view/dilrqvcful4yhyqdzwr424x6zawgsds/lib/python3.8/site-packages (from importlib-metadata>=4.8.0->gym) (3.17.0)
Building wheels for collected packages: gym
  Building wheel for gym (pyproject.toml) ... done
  Created wheel for gym: filename=gym-0.26.2-py3-none-any.whl size=820775 sh
```

The terminal window also shows the system memory usage as 523 / 2048 MB and the current mode as Command. The bottom status bar indicates the current file is 'Untitled.ipynb' at line 1, column 17.

Terminal Access

The screenshot displays the JupyterLab web interface. The browser address bar shows the URL: `lab.jsc.ebrains.eu/user/rowley/lab/tree/shared/Andrew%20Test%20Collab/SpiNNakerJupyterExamples`. The interface includes a top menu bar with options like File, Edit, View, Run, Kernel, Git, Tabs, Settings, and Help. A memory usage indicator shows 535 / 2048 MB. On the left, a file browser shows a directory structure with files like `00.Setup`, `01.RunningPyNNSimulations`, `02.LiveInputAndOutput`, `integration_tests`, `spinnaker_jupyter_examples`, `LICENSE`, `pyproject.toml`, `README.md`, `setup.cfg`, and `setup.py`. The main area is the Launcher, which contains icons for various environments: EBRAINS-experimental, R 3.6.3, R-EBRAINS-23.02, R-EBRAINS-23.06, R-EBRAINS-23.09, R-EBRAINS-24.04, and R-EBRAINS-experimental. Below these is an 'Other' section with icons for Terminal, Text File, Xircuits File, Markdown File, Python File, R File, and Show Contextual Help. The Terminal icon, which is a black square with a white '\$_' symbol, is circled in red. The bottom status bar shows 'Simple' mode and a memory usage of 509.67 / 2048.00 MB.

Terminal Access

The screenshot shows a JupyterLab environment with a file browser on the left and a terminal window on the right. The file browser displays a directory structure with the following files and folders:

Name	Last Modified
00.Setup	16 minutes ago
01.RunningPyNNSimulations	16 minutes ago
02.LiveInputAndOutput	16 minutes ago
integration_tests	16 minutes ago
spinnaker_jupyter_examples	16 minutes ago
LICENSE	16 minutes ago
pyproject.toml	16 minutes ago
README.md	16 minutes ago
setup.cfg	16 minutes ago
setup.py	16 minutes ago

The terminal window shows the current directory path:

```
jovyan@jupyter-rowley:/mnt/user/shared/Andrew Test Collab/SpiNakerJupyterExamples$
```

Edit SpiNNaker Configuration

The screenshot displays a JupyterLab environment. On the left, the file explorer shows the following directory listing:

Name	Last Modified
reports	3 months ago
SpiNNakerJupyterExamples	an hour ago
test	a year ago
SpiNNaker_01_test-Copy1.ipynb	a year ago
SpiNNaker_01_test.ipynb	a year ago
SpiNNaker_Collab_synfire.ipynb	3 months ago
spynnaker.cfg	a year ago
Test.ipynb	3 months ago
Untitled.ipynb	a year ago

The terminal window on the right shows the following command and output:

```

jovyan@jupyter-rowley:/mnt/user/shared/Andrew Test Collab/SpiNNakerJupyterExamples$ nano ~/
.spynnaker.cfg
  
```

The terminal title bar indicates the file being edited is 'spynnaker.cfg'. The system tray at the bottom shows 'Simple' mode, 2 tabs, and memory usage of 337.80 / 2048.00 MB.

Set Reports Options

The screenshot shows a JupyterLab interface with a terminal window open. The terminal is displaying the contents of a file named `.spynaker.cfg` located at `/opt/app-root/src/.spynaker.cfg`. The configuration is as follows:

```

[Machine]
spalloc_server = https://spinnaker.cs.man.ac.uk/spalloc/

[Java]
use_java = True

[Reports]
default_report_file_path = /opt/app-root/src/
read_provenance_data = False
  
```

The terminal window also shows a list of keyboard shortcuts at the bottom:

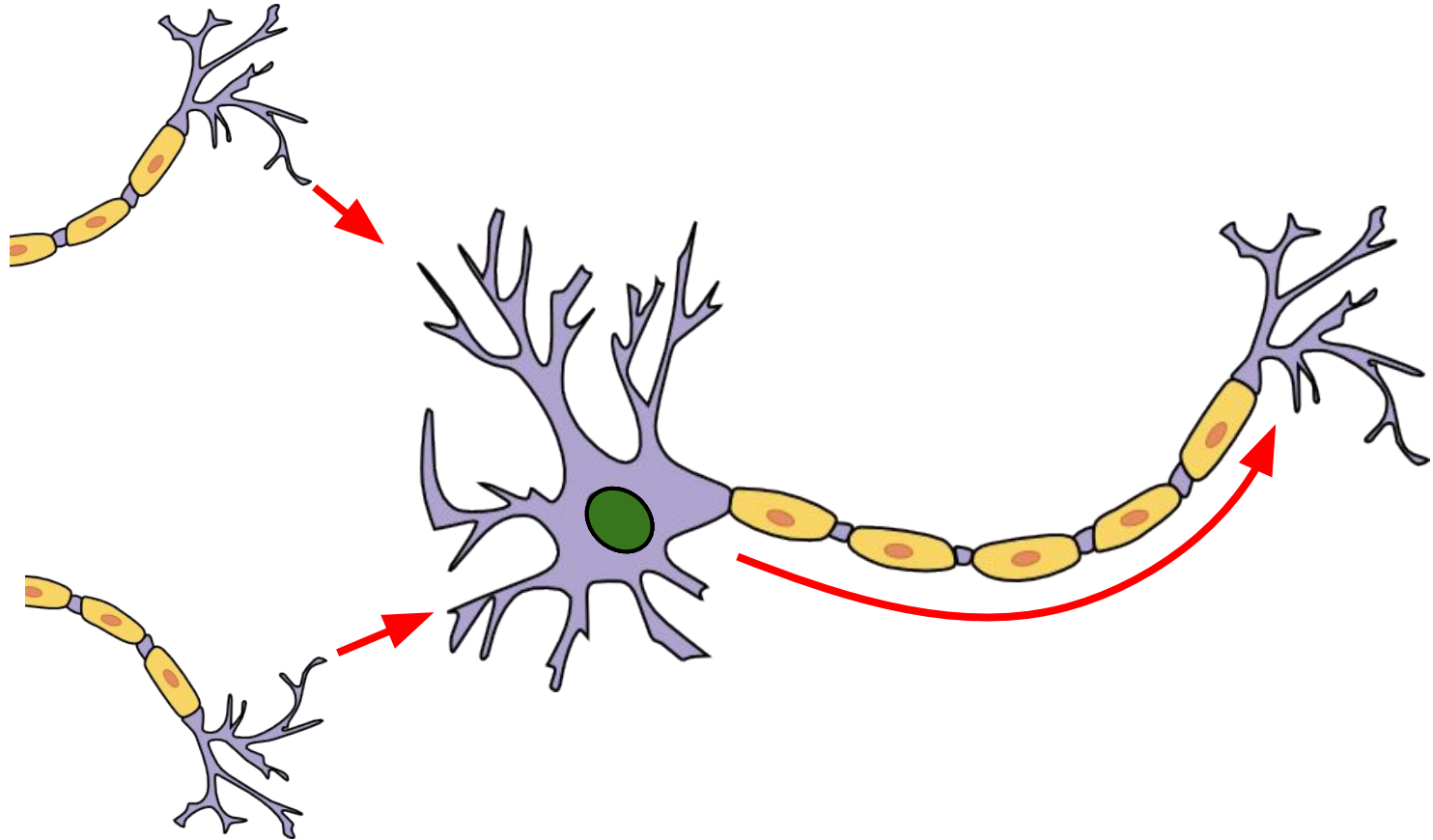
```

^G Get Help      ^O Write Out    ^W Where Is     ^K Cut Text     ^C Cur Pos     ^_ Previous
^X Exit          ^R Read File    ^\ Replace      ^U Paste Text   ^G Go To Line  ^N Next
  
```

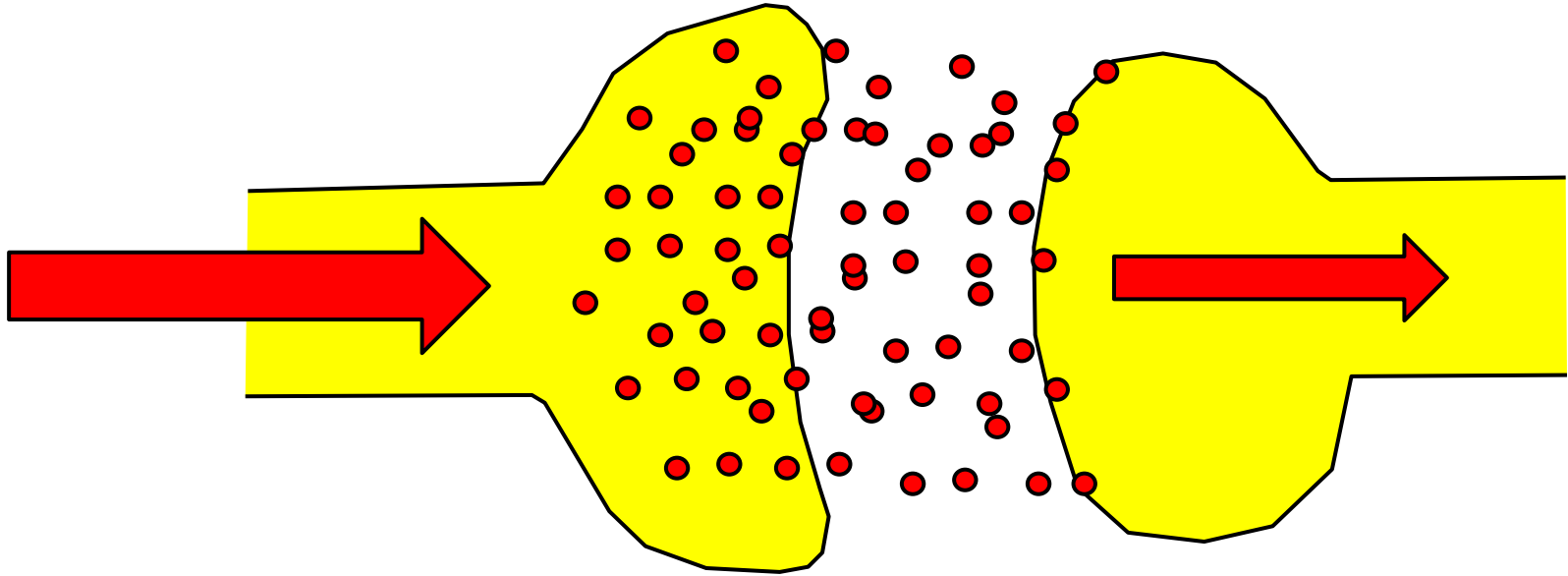
The JupyterLab interface includes a file browser on the left showing a directory structure with 'drive' and 'shared' folders. The top menu bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Git', 'Tabs', 'Settings', and 'Help'. The status bar at the bottom indicates memory usage: 'Mem: 551.05 / 2048.00 MB'.

Spiking Neural Networks

Spiking Neural Networks

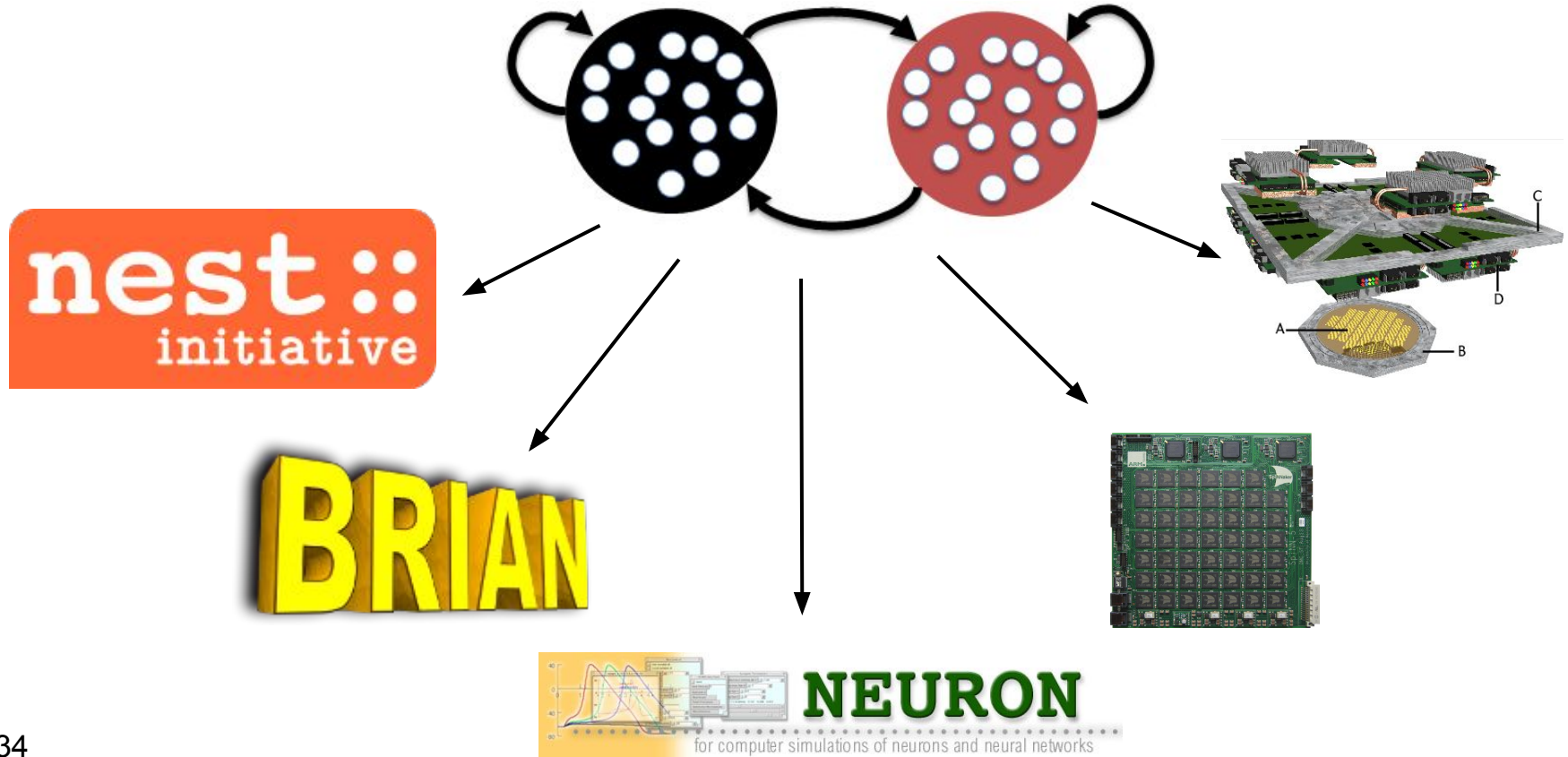


Spiking Neural Networks



PyNN

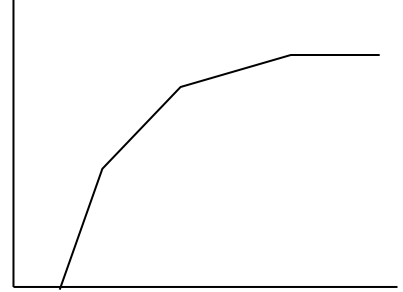
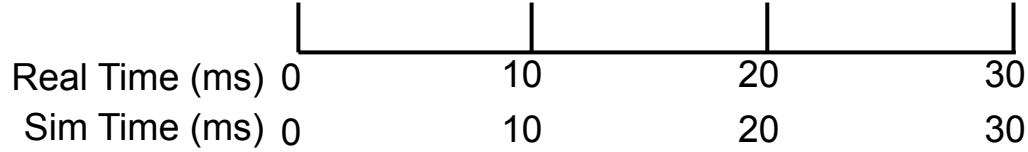
What is PyNN?



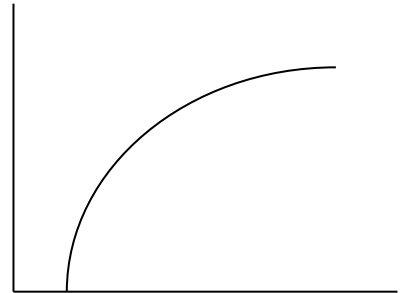
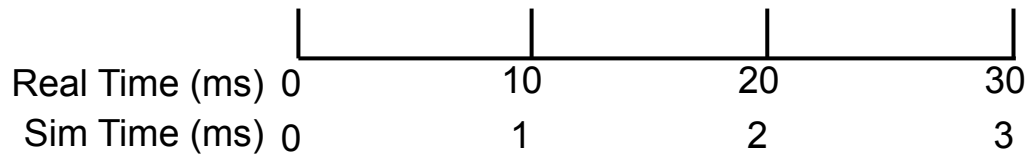
PyNN - Setup

```
import pyNN.spiNNaker as p
```

```
p.setup(timestep=1.0)
```

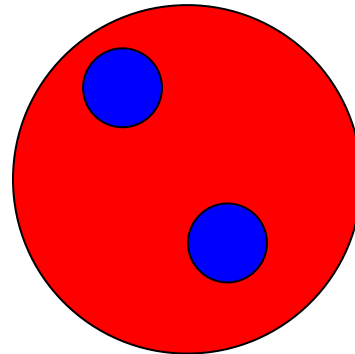
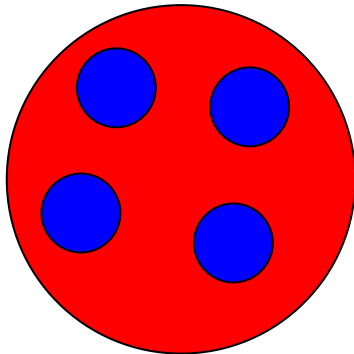


```
p.setup(timestep=0.1)
```



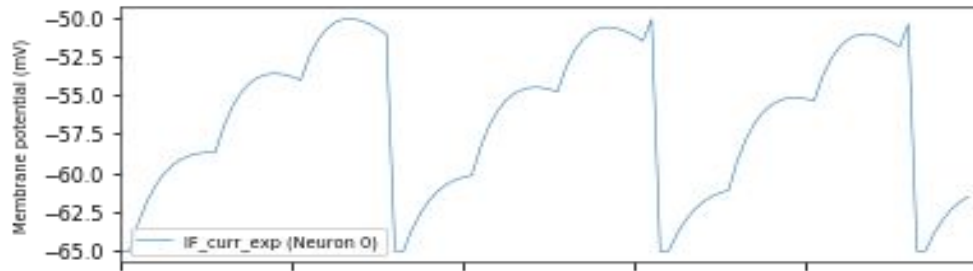
PyNN - Populations

```
pop_1 = p.Population(  
    4, p.IF_curr_exp(), label="Fred")  
pop_2 = p.Population(  
    2, p.IF_curr_exp(), label="Bob")
```

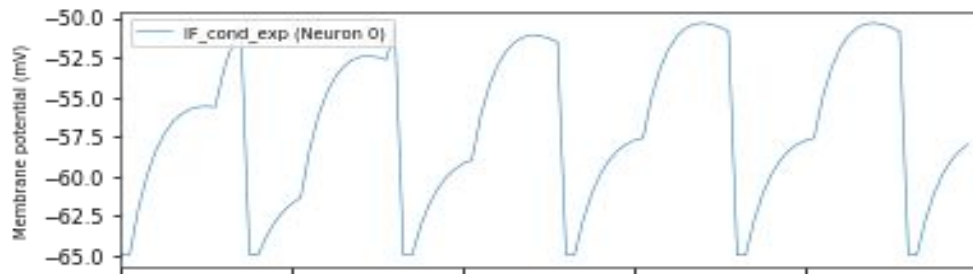


PyNN Populations - Models

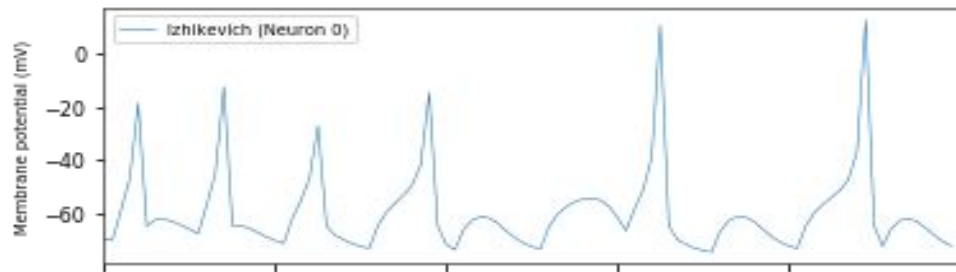
`p.IF_curr_exp(...)`



`p.IF_cond_exp(...)`

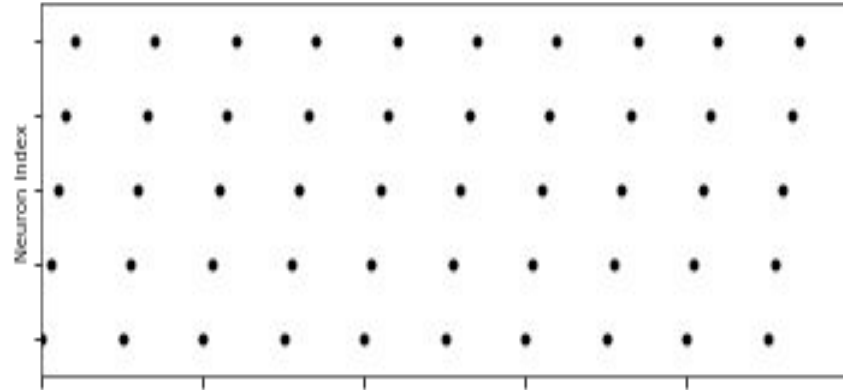


`p.Izhikevich(...)`

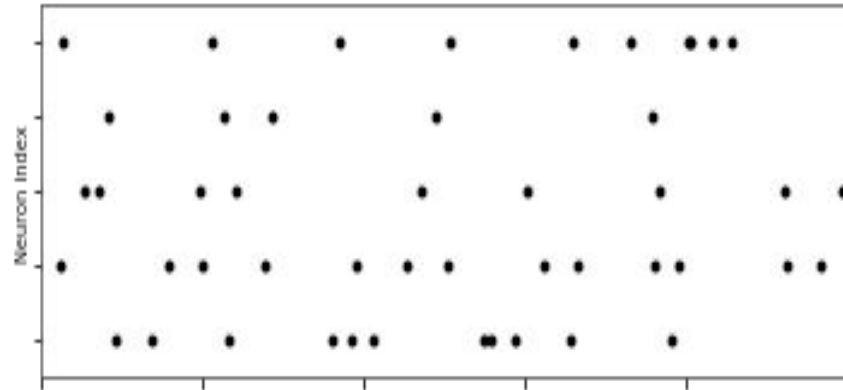


PyNN Populations - Models

```
p.SpikeSourceArray(
    spike_times=[...])
```

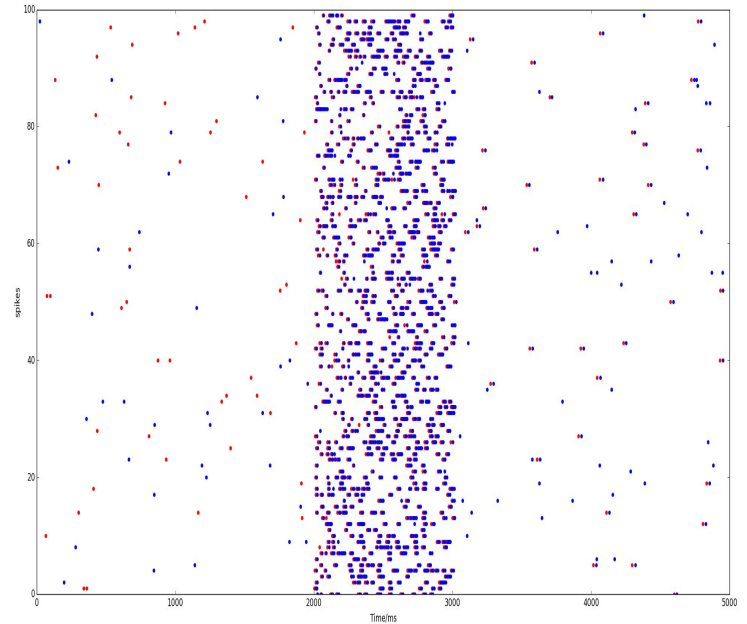
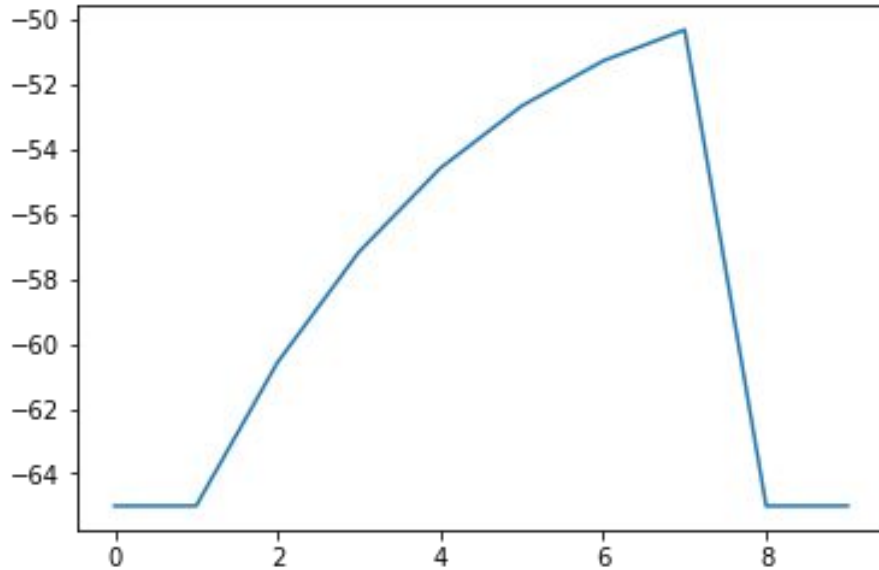


```
p.SpikeSourcePoisson(
    rate=...)
```



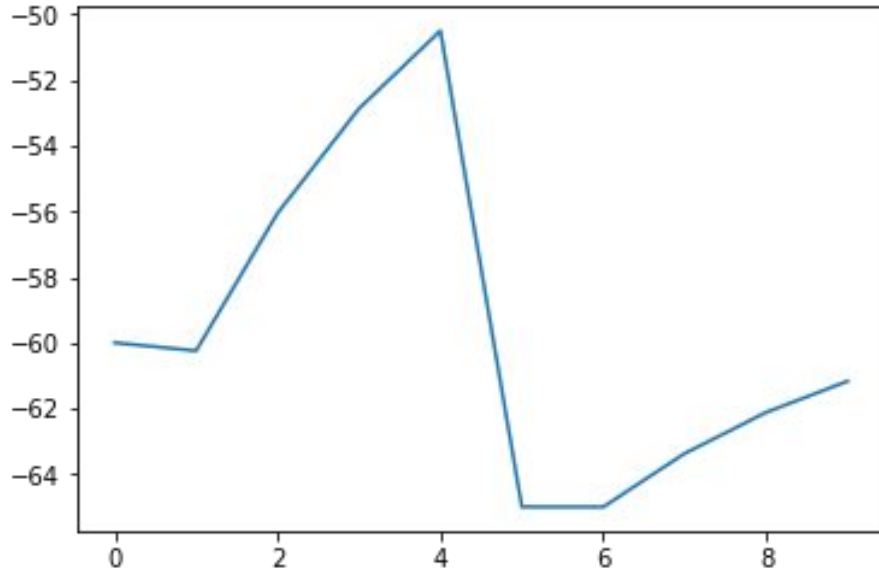
PyNN Populations - Recording

```
pop_1.record(["v", "spikes"])
```



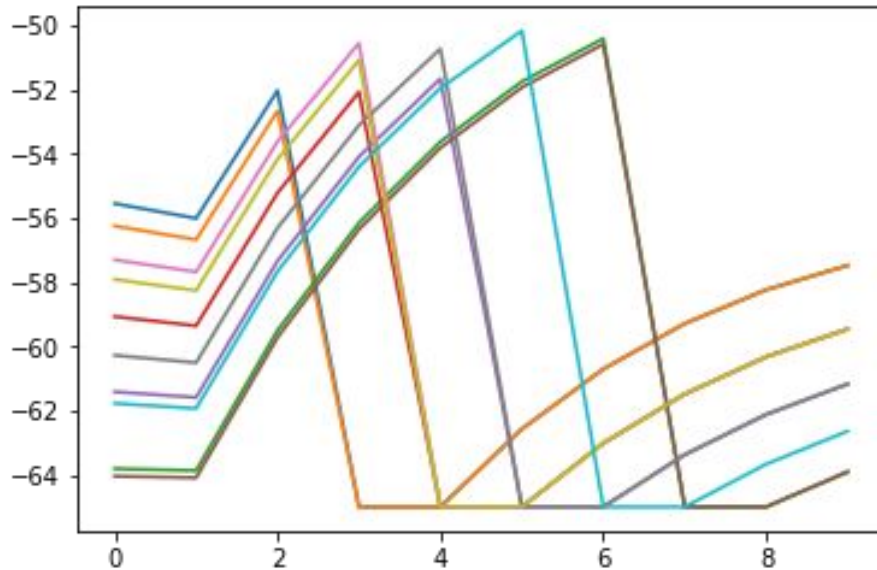
PyNN Populations - Initialize

```
pop_1.initialize(v=-60)
```



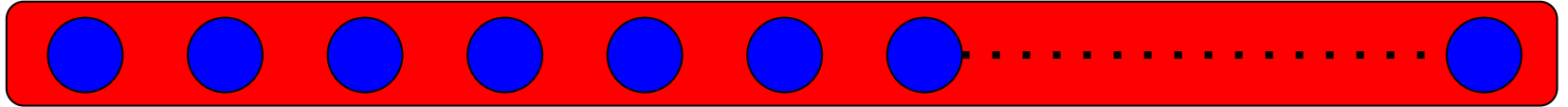
PyNN - Random Parameters

```
pop_1.initialize(v=p.RandomDistribution(  
    "uniform", low=-65.0, high=-55.0))
```



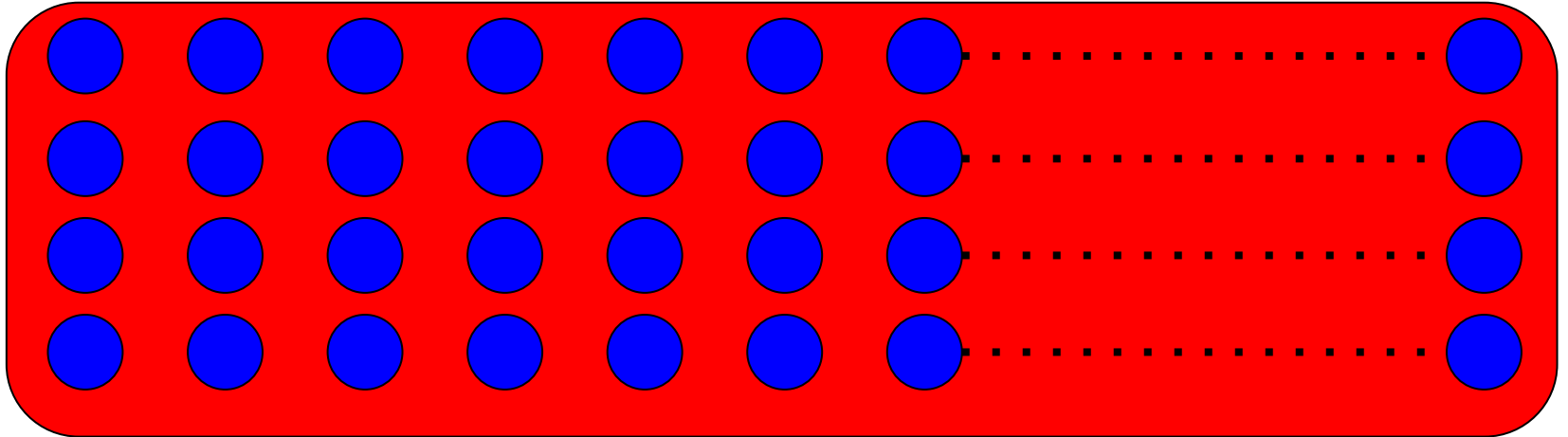
PyNN Populations - Structure

```
pop_1 = p.Population(  
    100, p.IF_curr_exp())
```



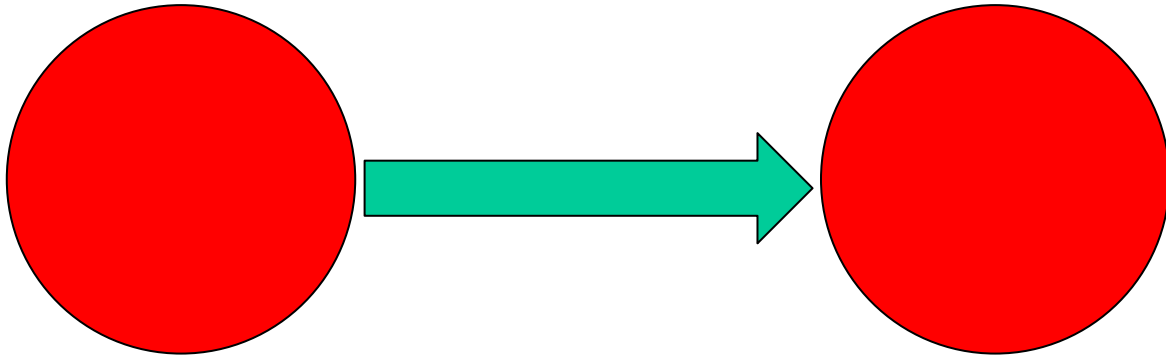
PyNN Populations - Structure

```
pop_1 = p.Population(  
    100, p.IF_curr_exp(),  
    structure=Grid2D(4 / 25))
```



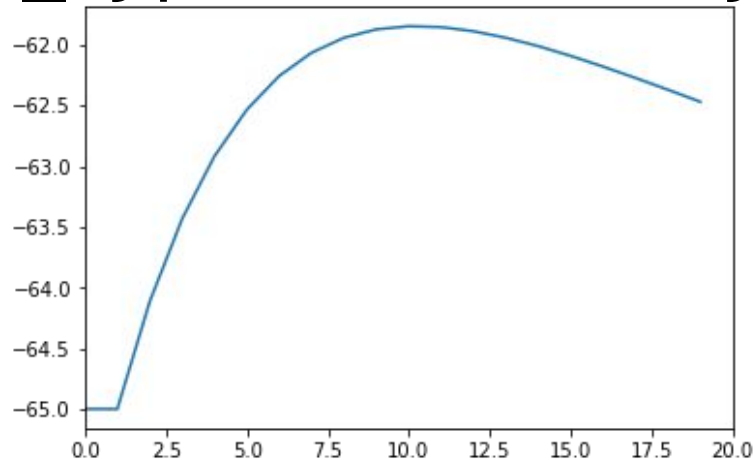
PyNN - Projections

```
proj = p.Projection(  
    pop_1, pop_2,  
    p.OneToOneConnector(),  
    p.StaticSynapse(  
        weight=1.0, delay=2.0))
```



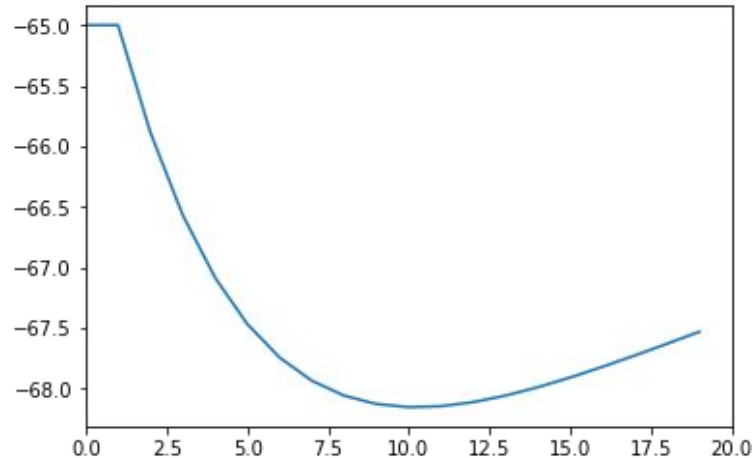
PyNN - Projections

```
proj = p.Projection(  
    pop_1, pop_2,  
    p.OneToOneConnector(),  
    p.StaticSynapse(weight=1.0),  
    receptor_type="excitatory")
```

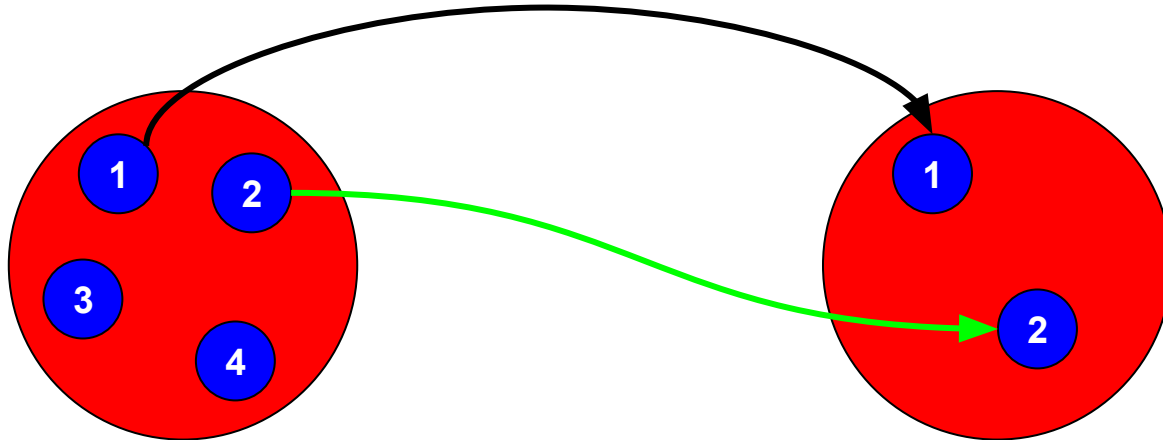


PyNN - Projections

```
proj = p.Projection(  
    pop_1, pop_2,  
    p.OneToOneConnector(),  
    p.StaticSynapse(weight=1.0),  
    receptor_type="inhibitory")
```

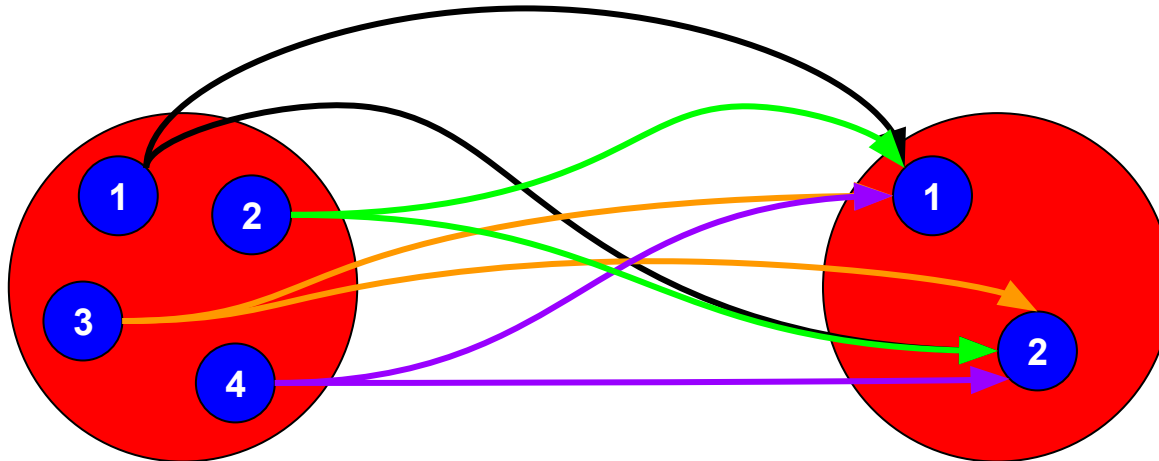


```
p.OneToOneConnector()
```



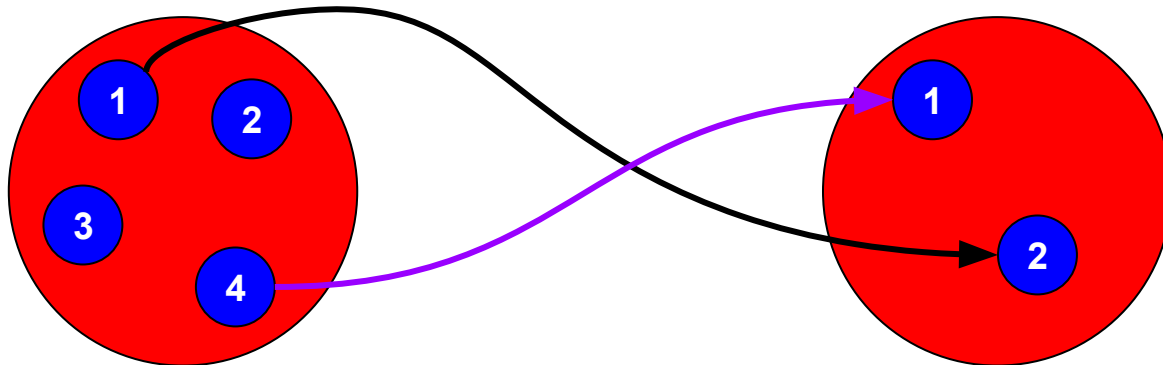
PyNN Projections - Connectors

```
p.AllToAllConnector()
```



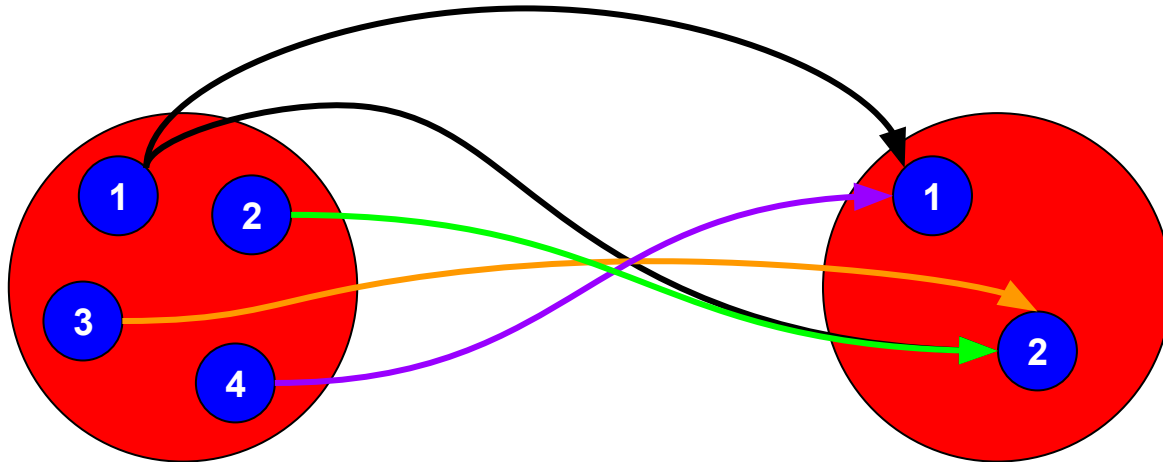
PyNN Projections - Connectors

`p.FixedProbabilityConnector(p=0.1)`



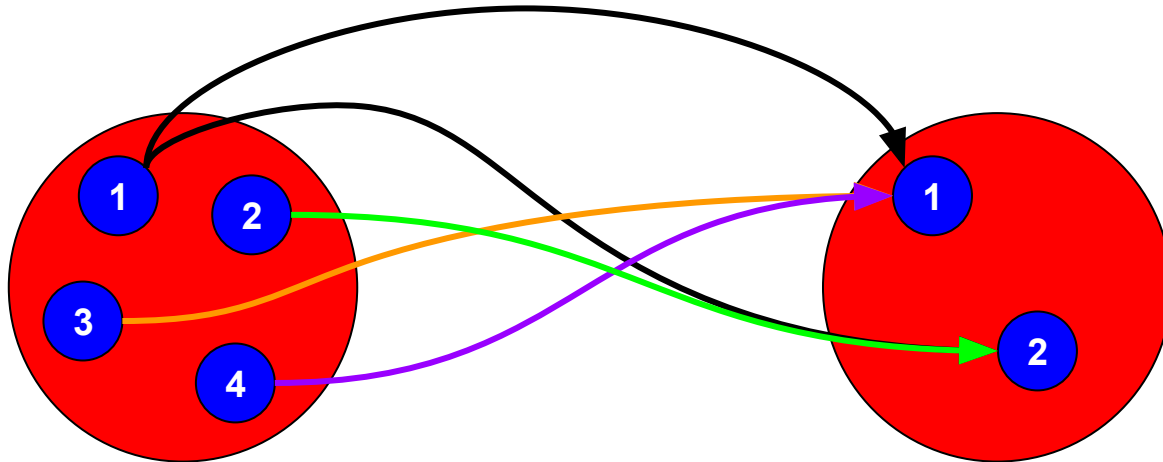
PyNN Projections - Connectors

`p.FixedProbabilityConnector(p=0.5)`



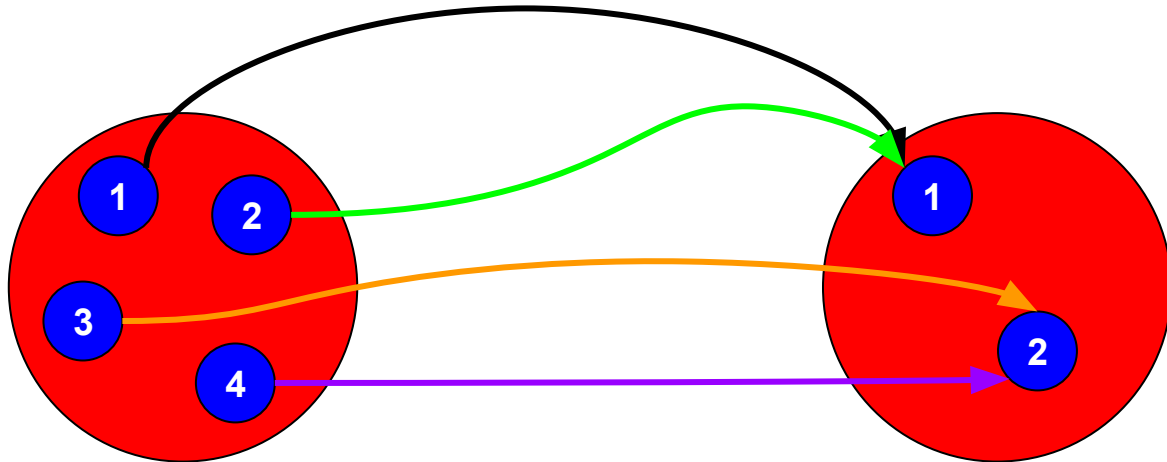
PyNN Projections - Connectors

`p.FixedTotalNumberConnector(5)`



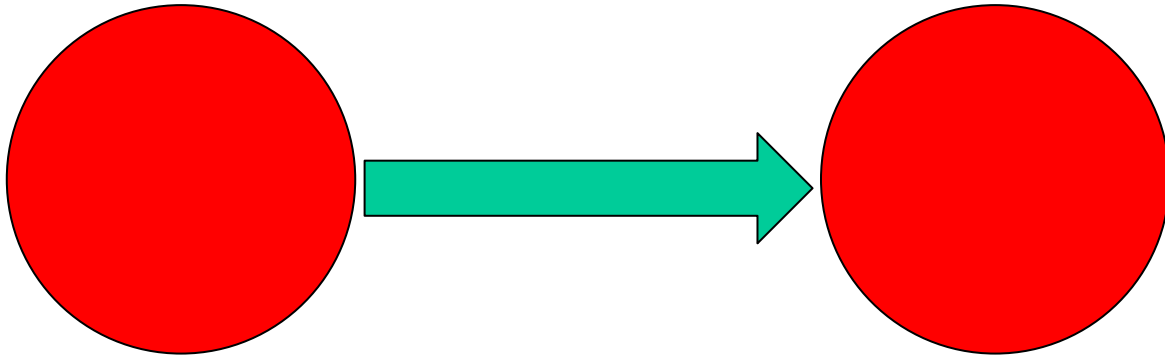
PyNN Projections - Connectors

```
p.FromListConnector(  
    [(1, 1), (2, 1), (3, 2), (4, 2)])
```



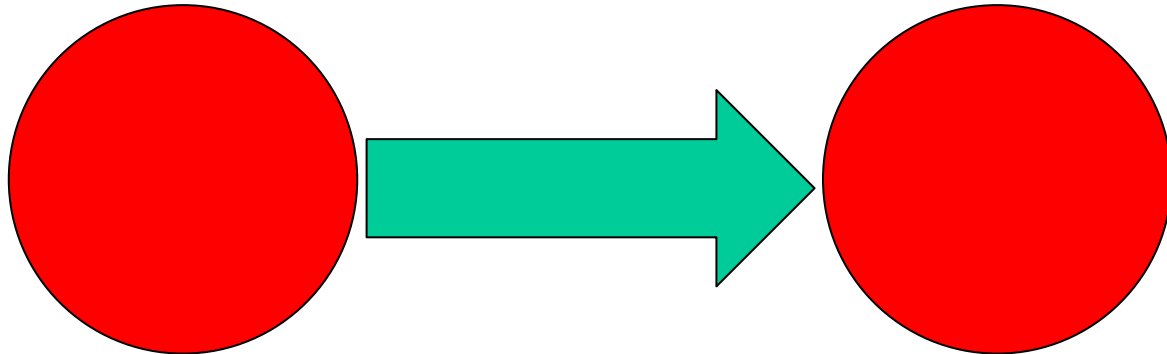
PyNN - Static Synapse Types

```
p.StaticSynapse(weight=1.0, delay=2.0)
```



PyNN - Static Synapse Types

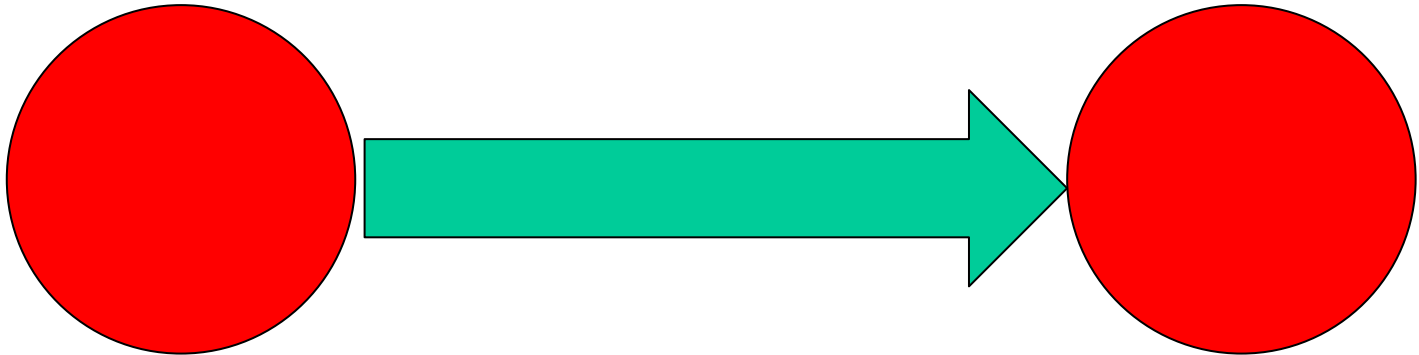
```
p.StaticSynapse(weight=5.0, delay=2.0)
```



PyNN - Static Synapse Types

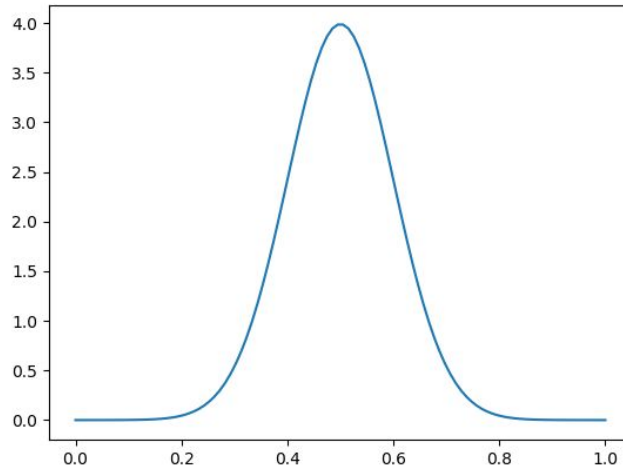
```
p.StaticSynapse(weight=5.0, delay=3.0)
```

(timestep \leq delay \leq 144 * timestep)



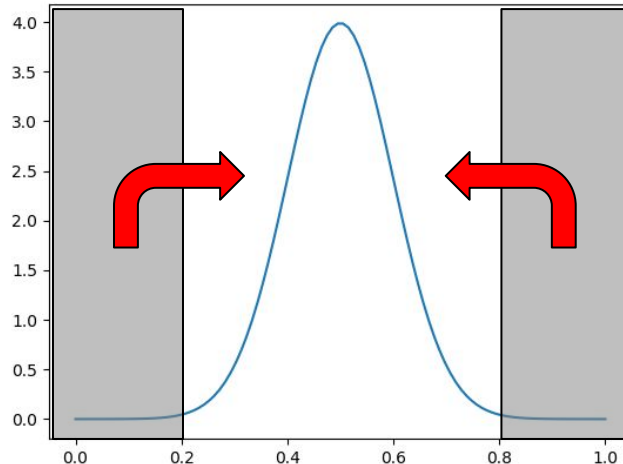
PyNN - Random Weights and Delays

```
weight_dist = p.RandomDistribution(  
    "normal", mu=0.5, sigma=0.1)  
p.StaticSynapse(  
    weight=weight_dist, delay=3.0)
```



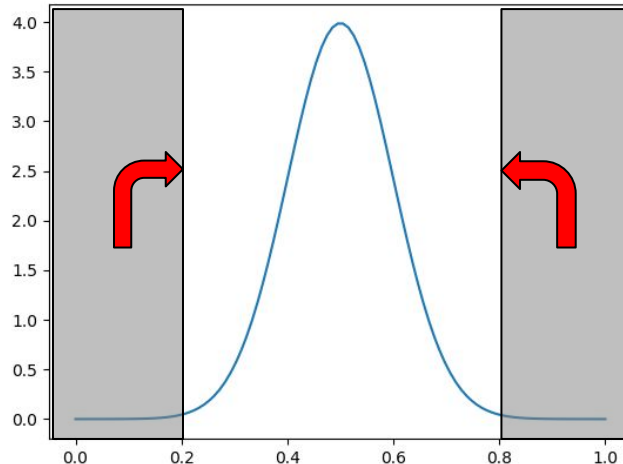
PyNN - Random Weights and Delays

```
weight_dist = p.RandomDistribution(  
    "normal_clipped",  
    mu=0.5, sigma=0.1, low=0.2 high=0.8)
```

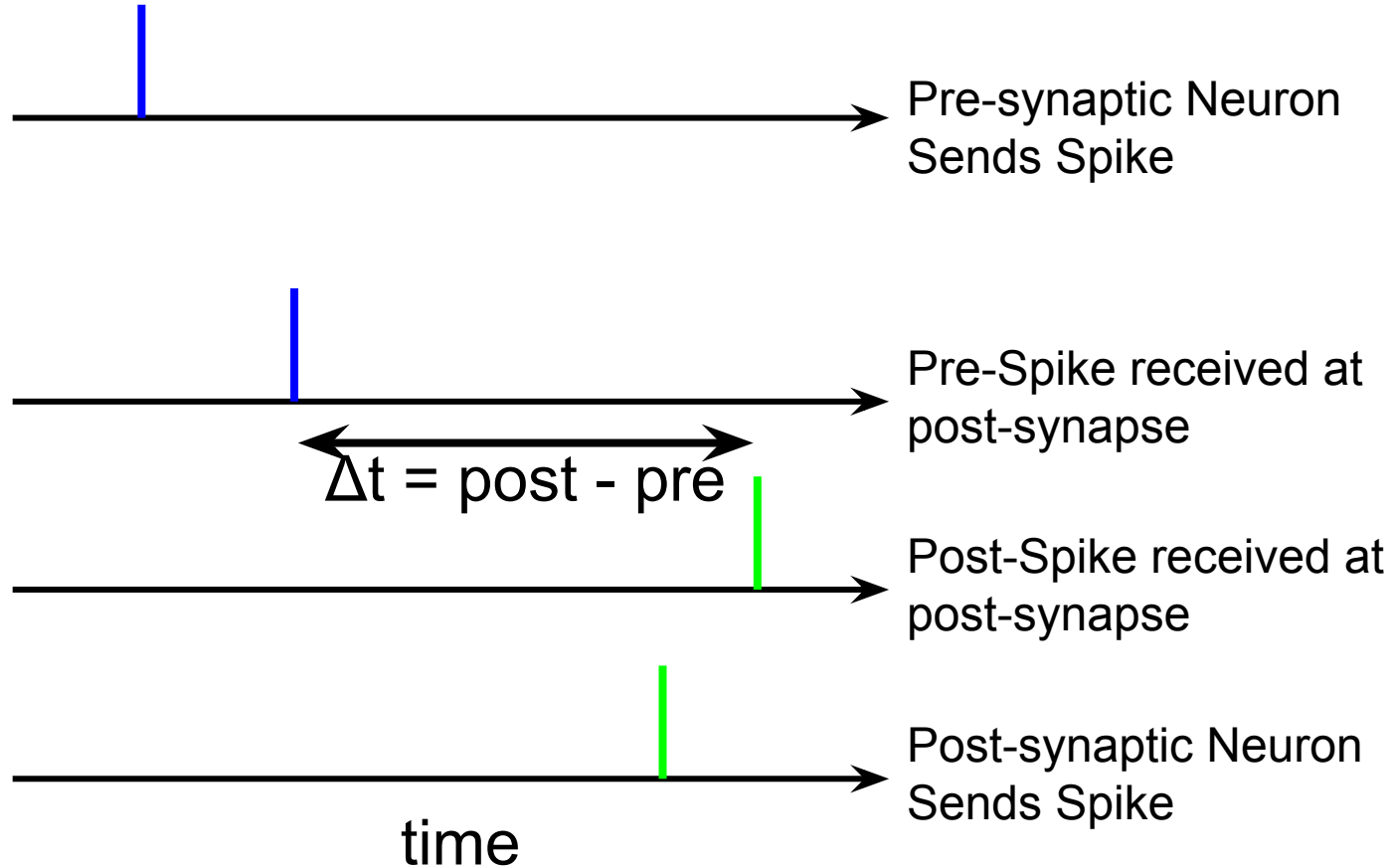
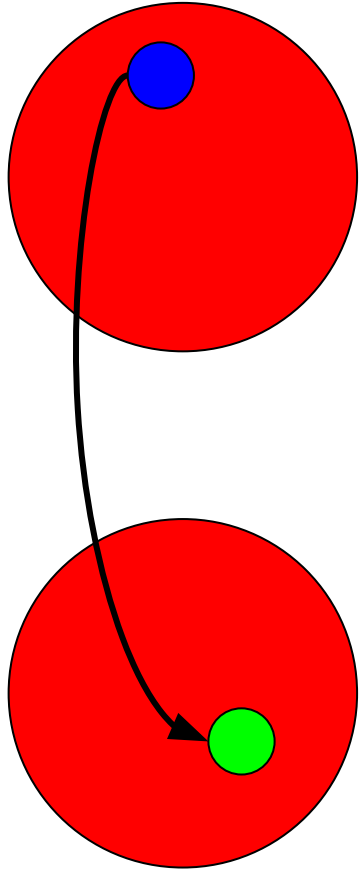


PyNN - Random Weights and Delays

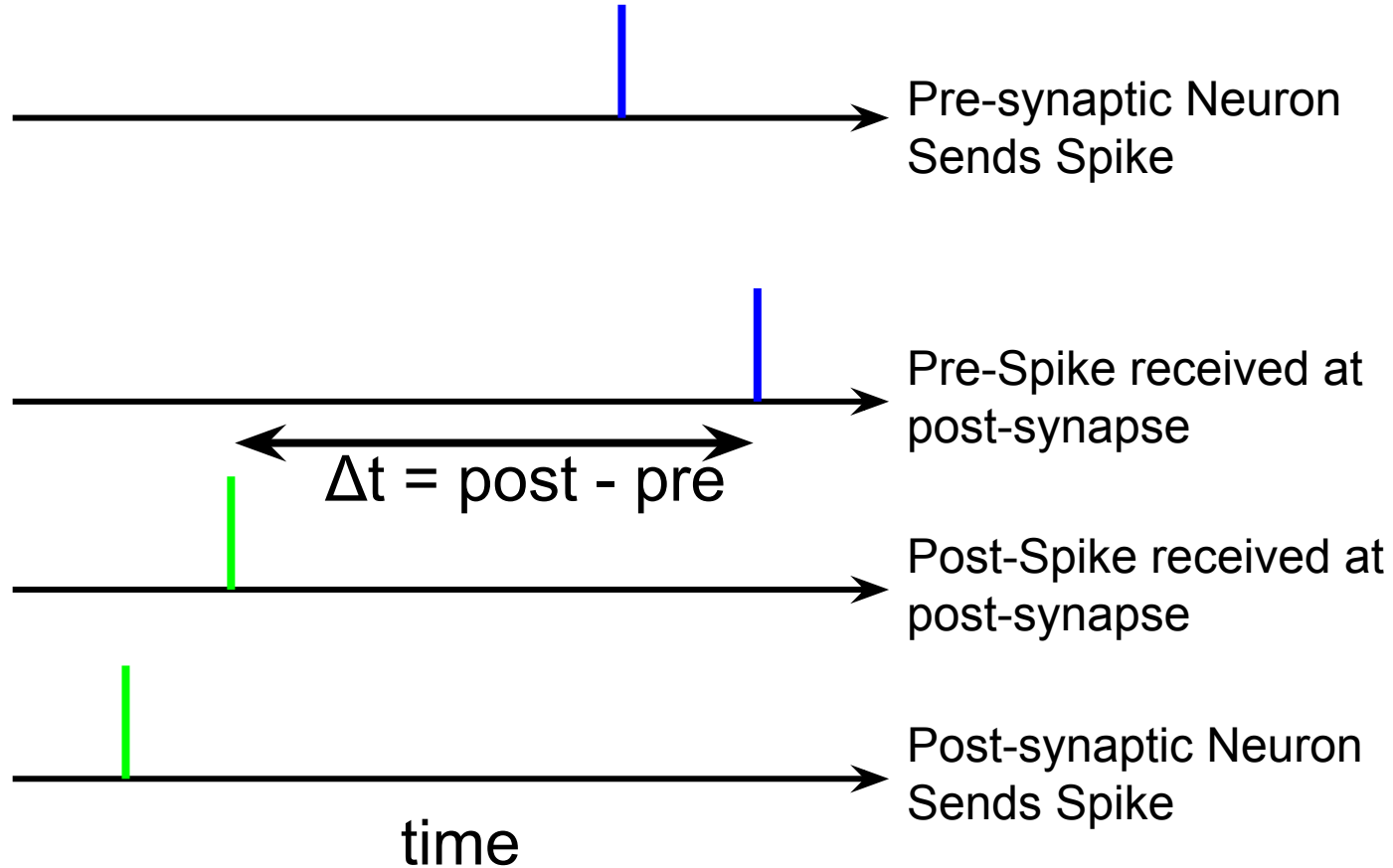
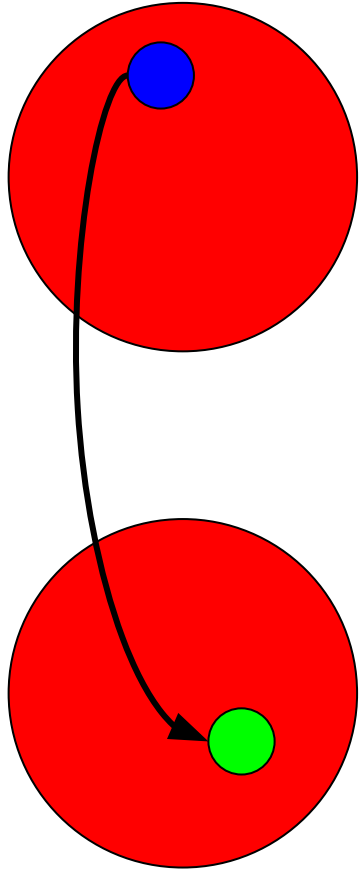
```
weight_dist = p.RandomDistribution(  
    "normal_clipped_to_boundary",  
    mu=0.5, sigma=0.1, low=0.2 high=0.8)
```



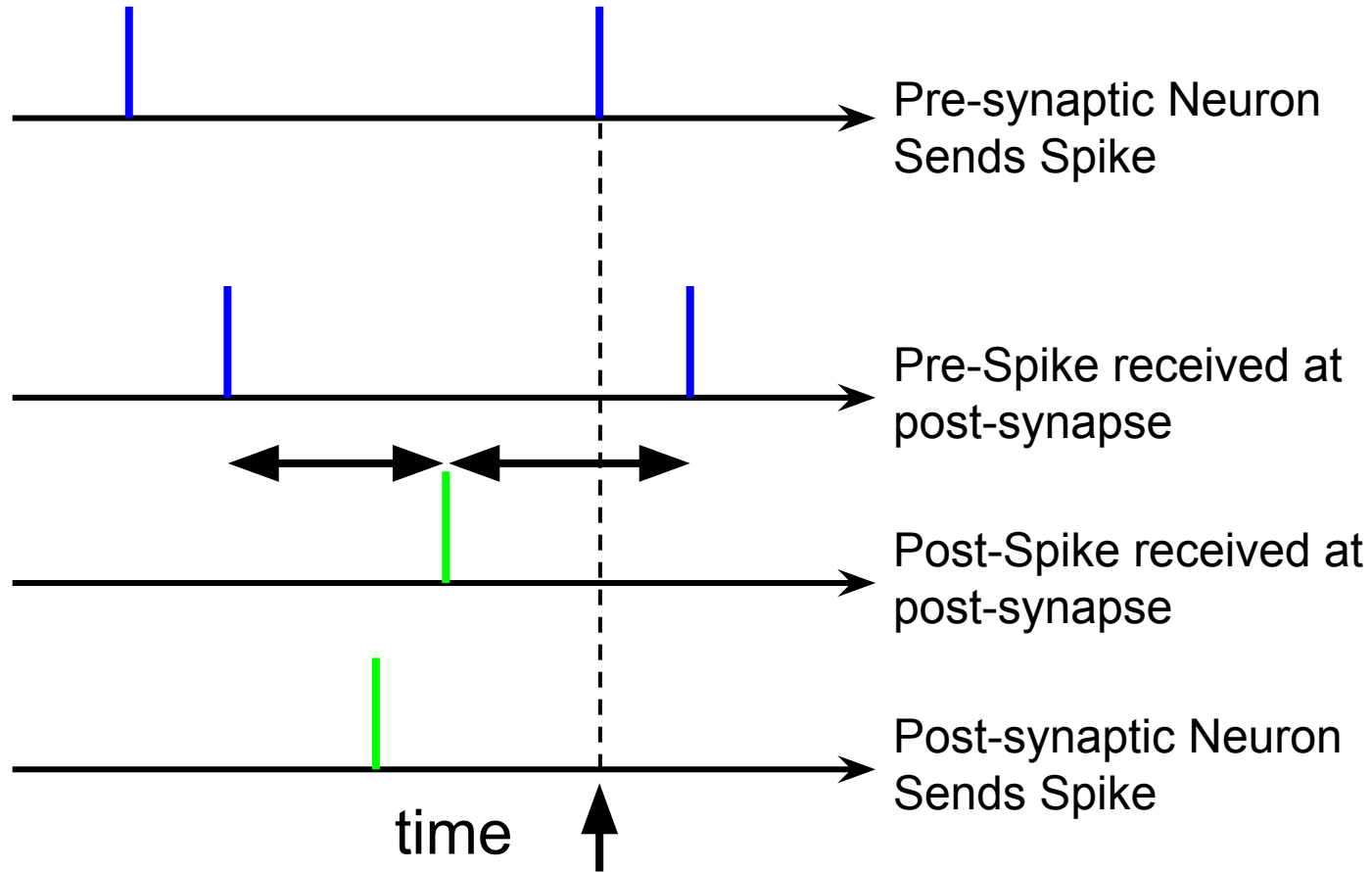
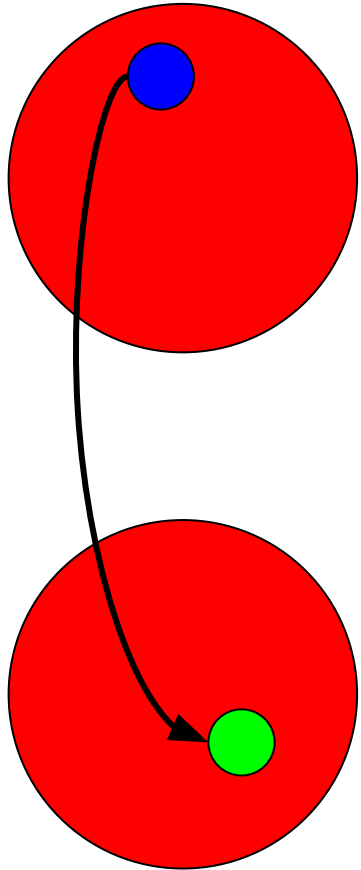
Spike Timing Dependent Plasticity



Spike Timing Dependent Plasticity

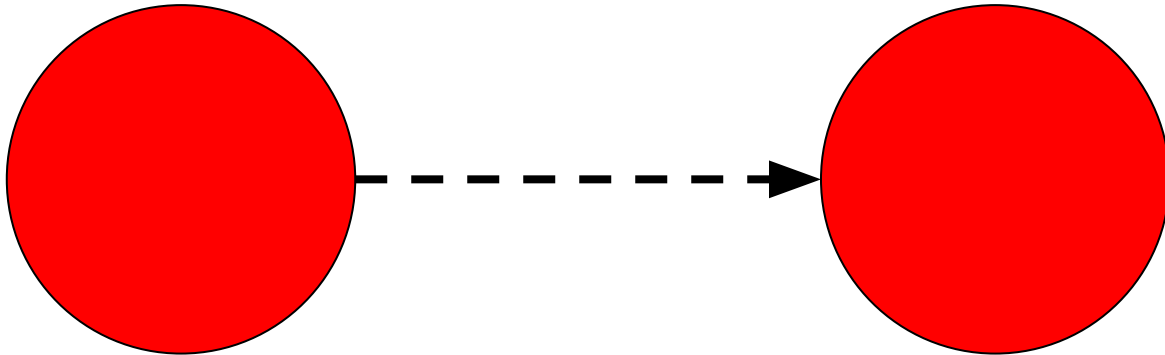


STDP - Deferred Execution



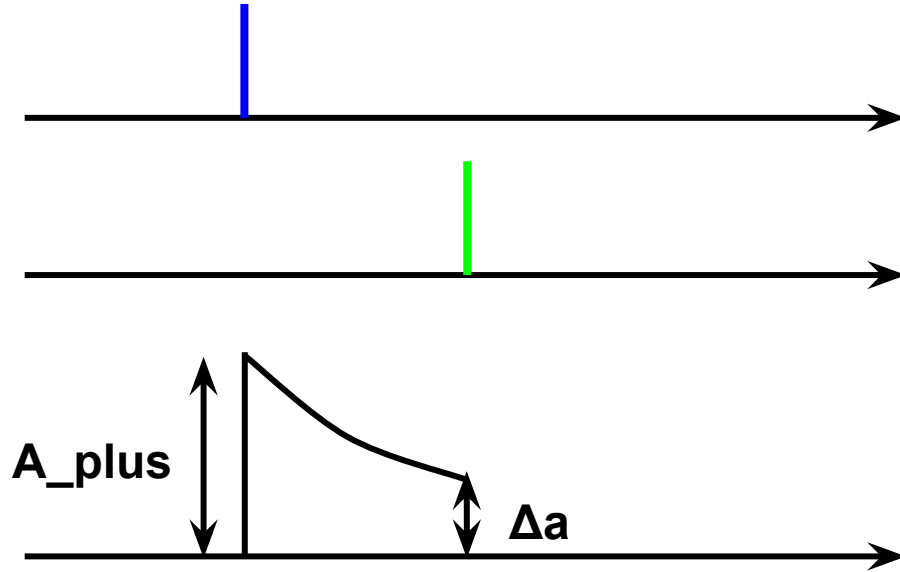
PyNN - STDP Synapse Types

```
p.STDPMechanism(  
    timing_dependence=?,  
    weight_dependence=?,  
    weight=0.0, delay=2.0)
```



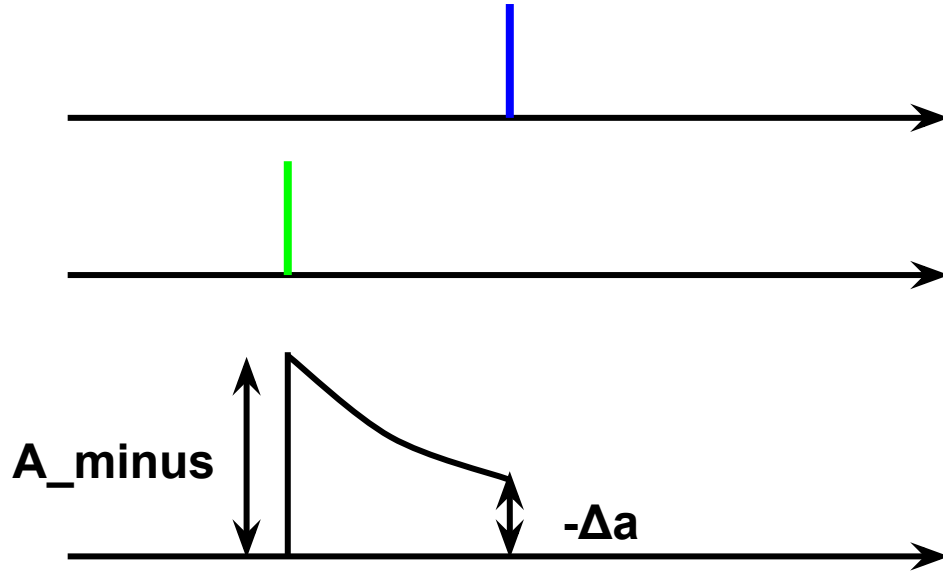
PyNN - Timing Dependence

```
sim.SpikePairRule(tau_plus=20.0, tau_minus=20.0,  
                  A_plus=0.5, A_minus=0.5)
```



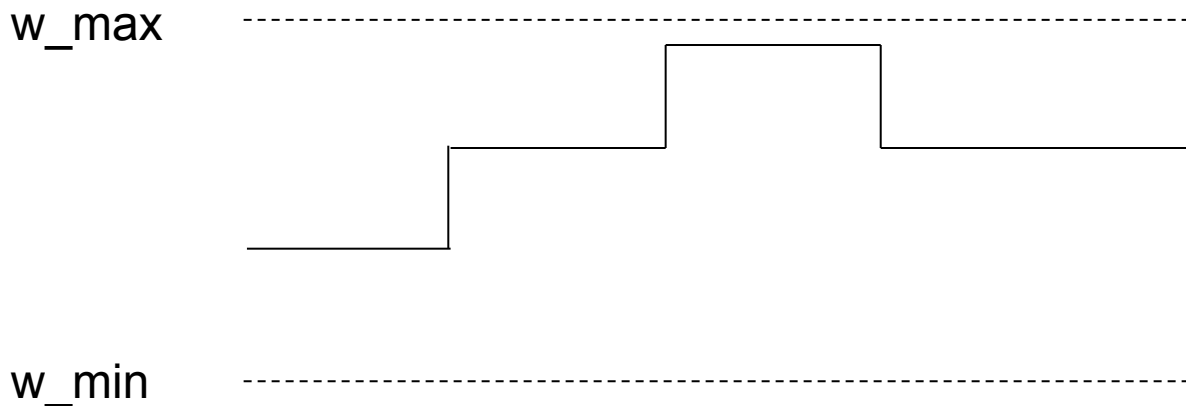
PyNN - Timing Dependence

```
sim.SpikePairRule(tau_plus=20.0, tau_minus=20.0,  
                  A_plus=0.5, A_minus=0.5)
```



PyNN - Weight Dependence

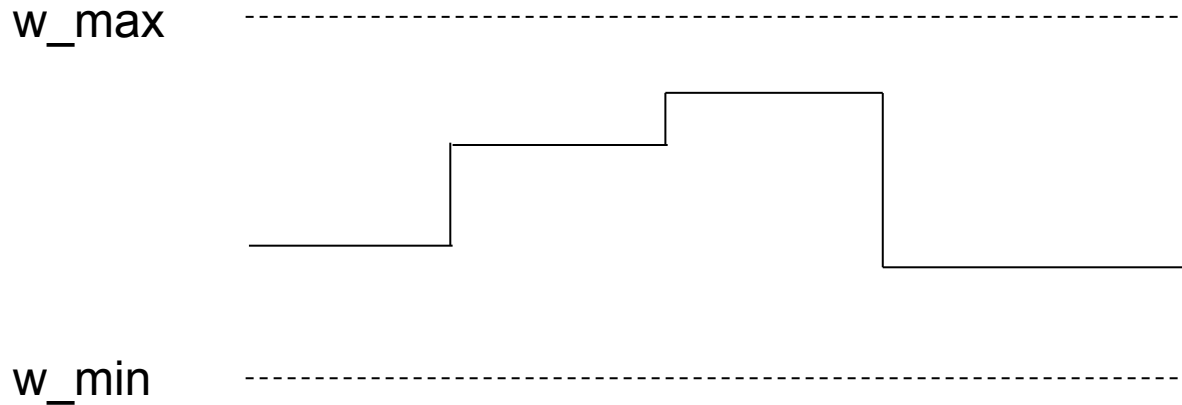
```
sim.AdditiveWeightDependence(w_max=5.0, w_min=0.0)
```



$$\Delta w = \Delta a (w_{\max} - w_{\min})$$

PyNN - Weight Dependence

```
sim.MultiplicativeWeightDependence(w_max=5.0,w_min=0.0)
```



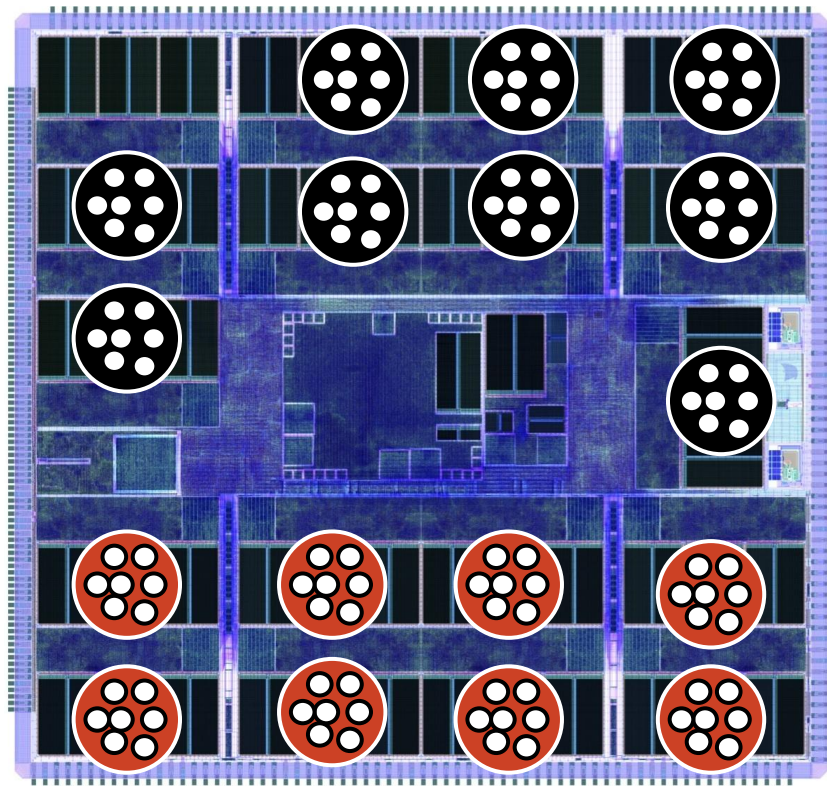
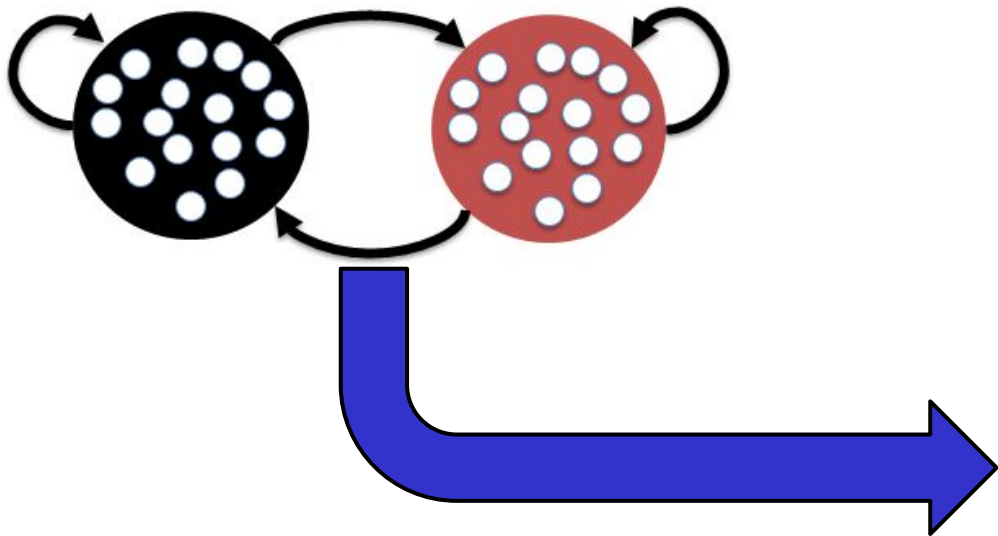
$$\Delta w = \Delta a (w - w_{\min}) \text{ if } \Delta a < 0 \text{ (Depression)}$$

$$\Delta w = \Delta a (w_{\max} - w) \text{ if } \Delta a > 0 \text{ (Potentiation)}$$

Running on SpiNNaker

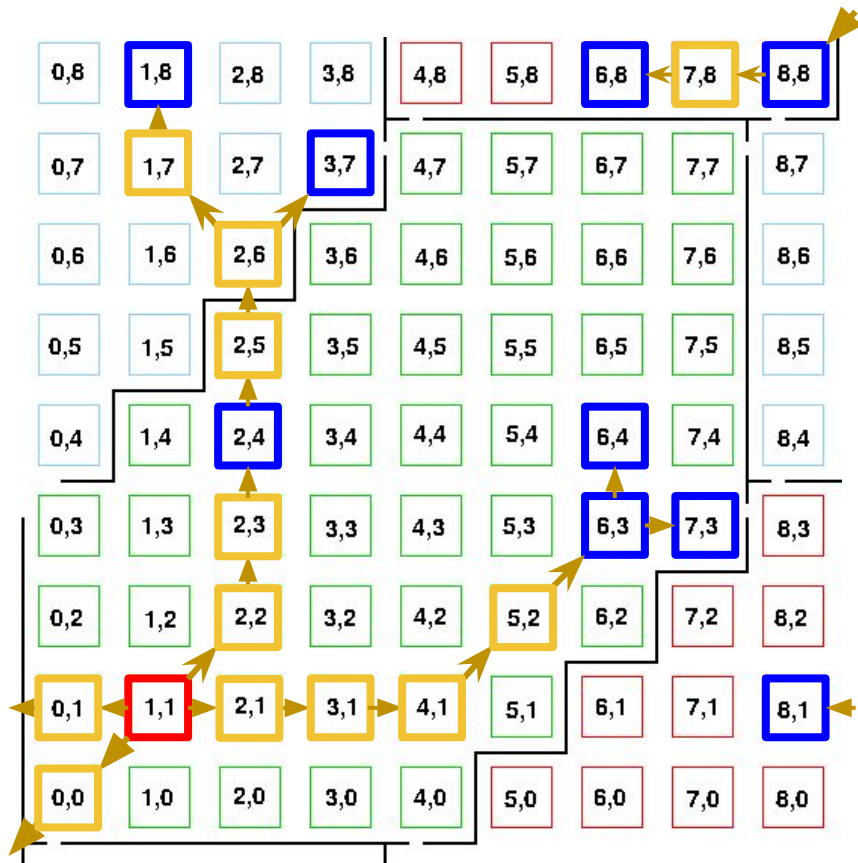
PyNN - Run

`p.run(100)`



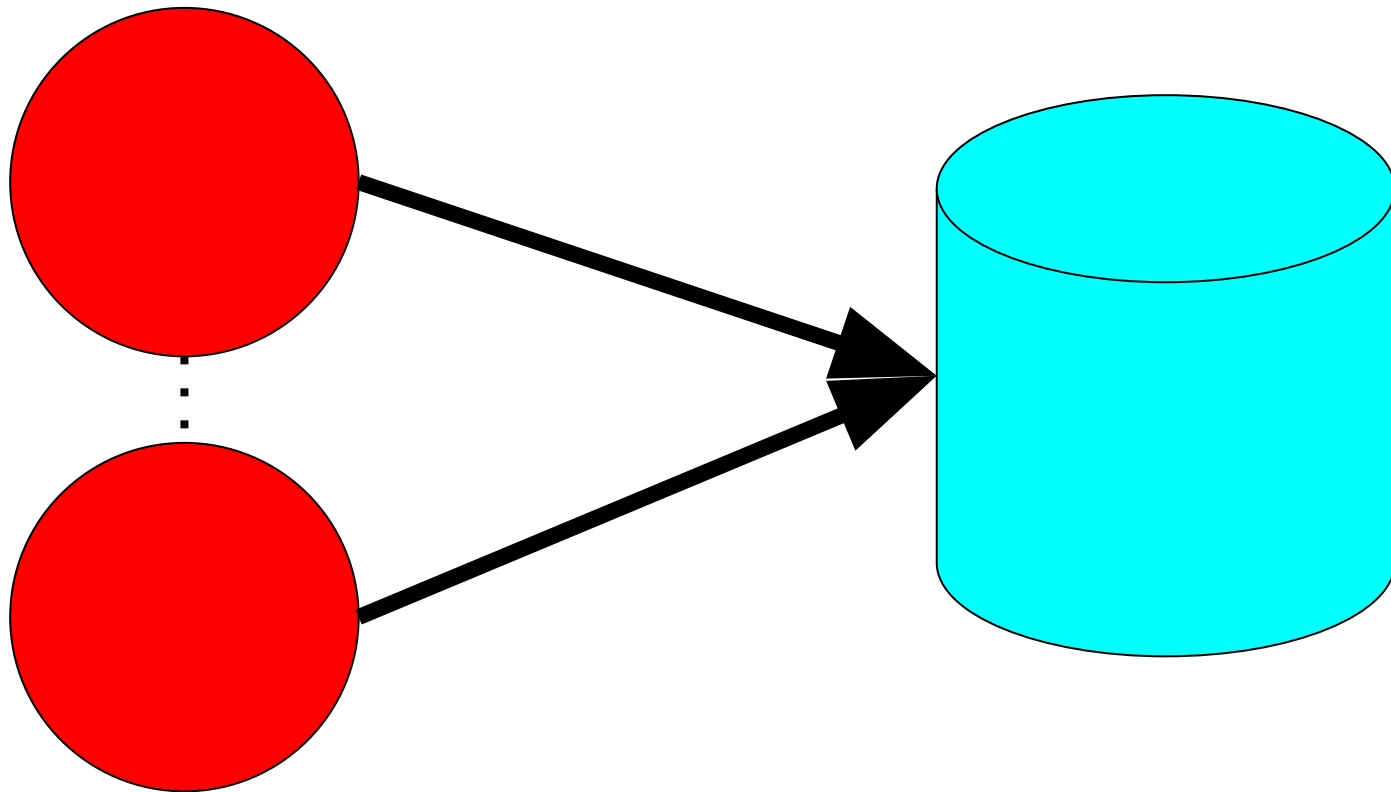
PyNN - Run

p.run(100)



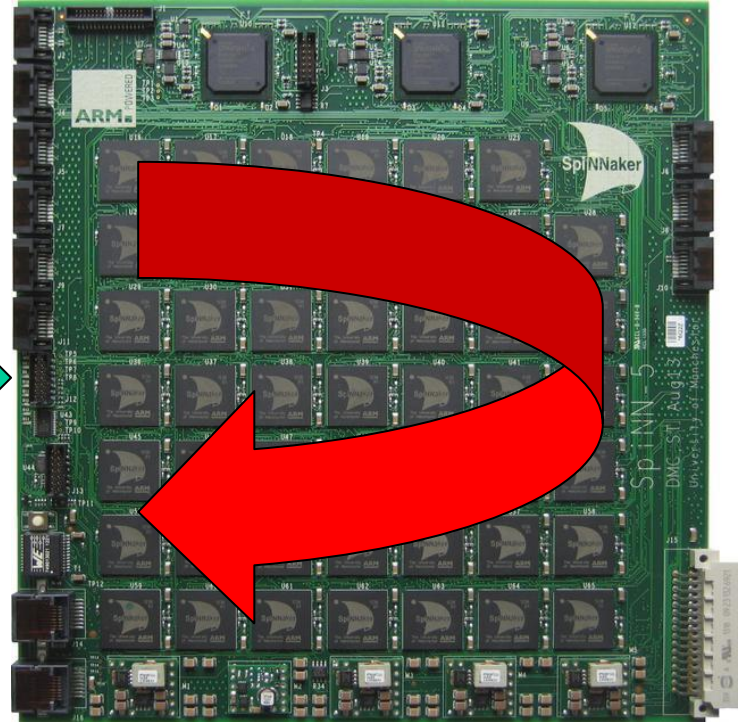
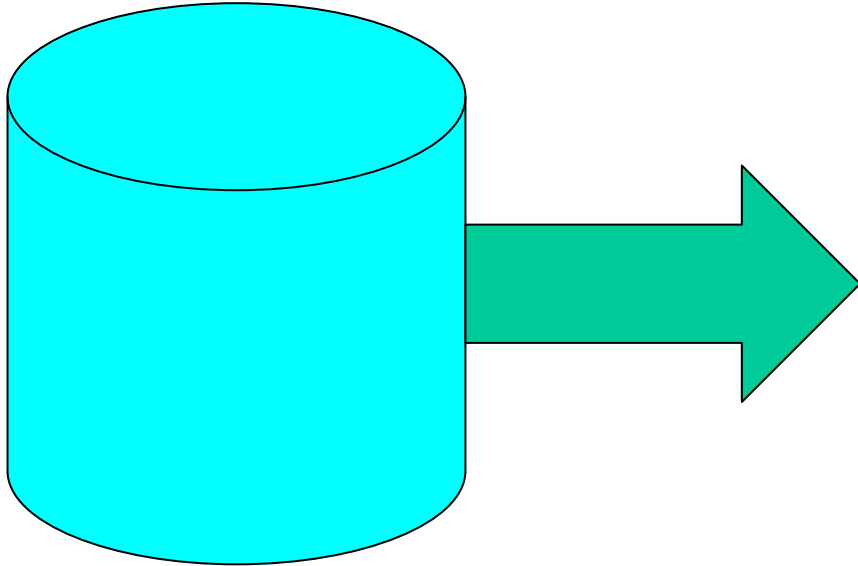
PyNN - Run

```
p.run(100)
```



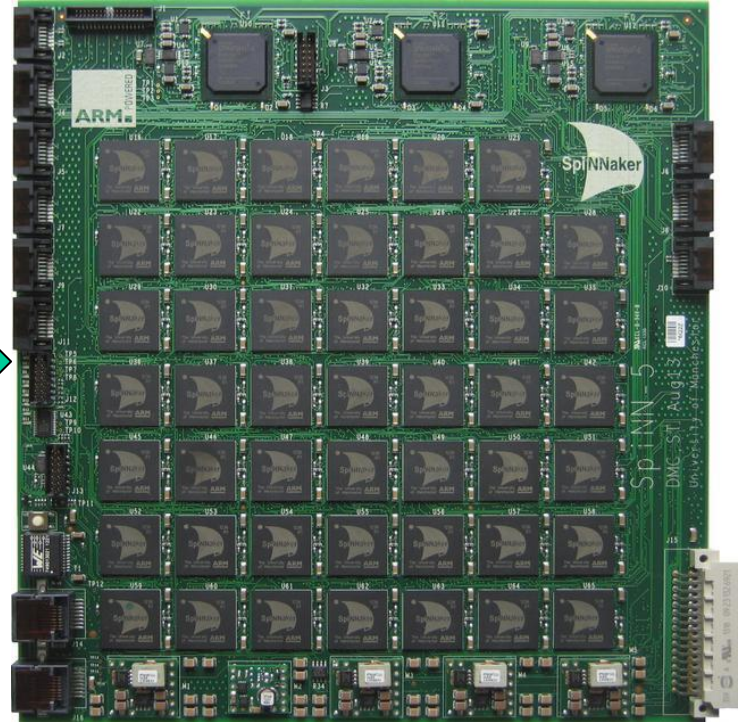
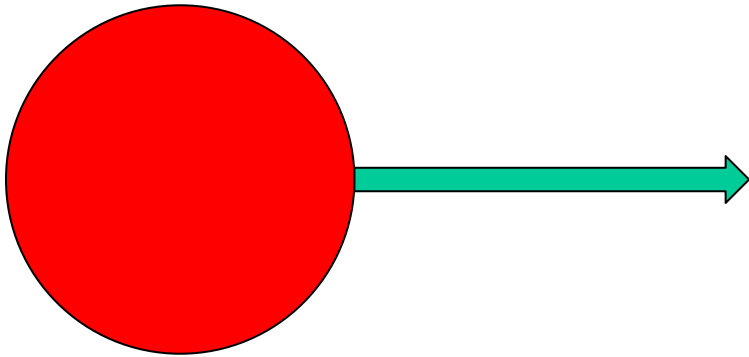
PyNN - Run

```
p.run(100)
```



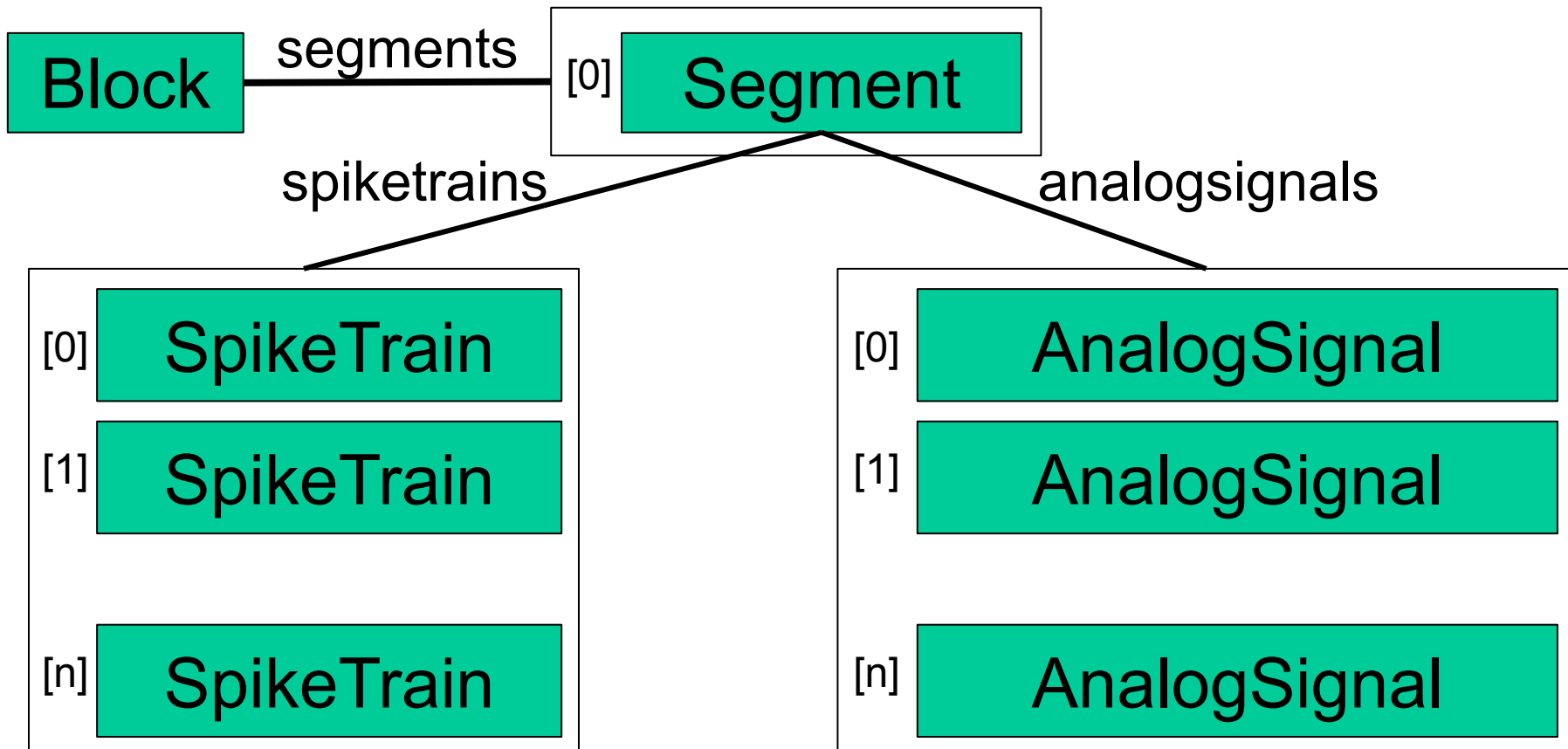
PyNN - Change and Run Again

```
pop_1.set(i_offset=5.0)  
p.run(50)
```

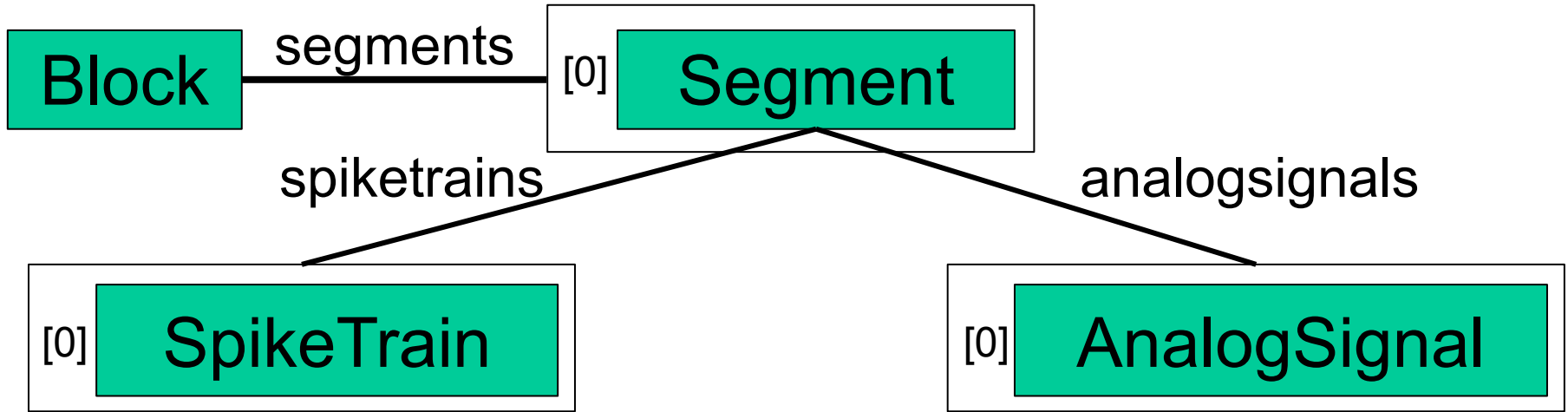


Reading Results

PyNN - Neo Data



PyNN - Neo Data



```

data = pop_1.get_data(["v", "spikes"])
v = data.segments[0].analogsignals
V = data.segments[0].filter(name="v")[0]
spikes = data.segments[0].spiketrains
  
```

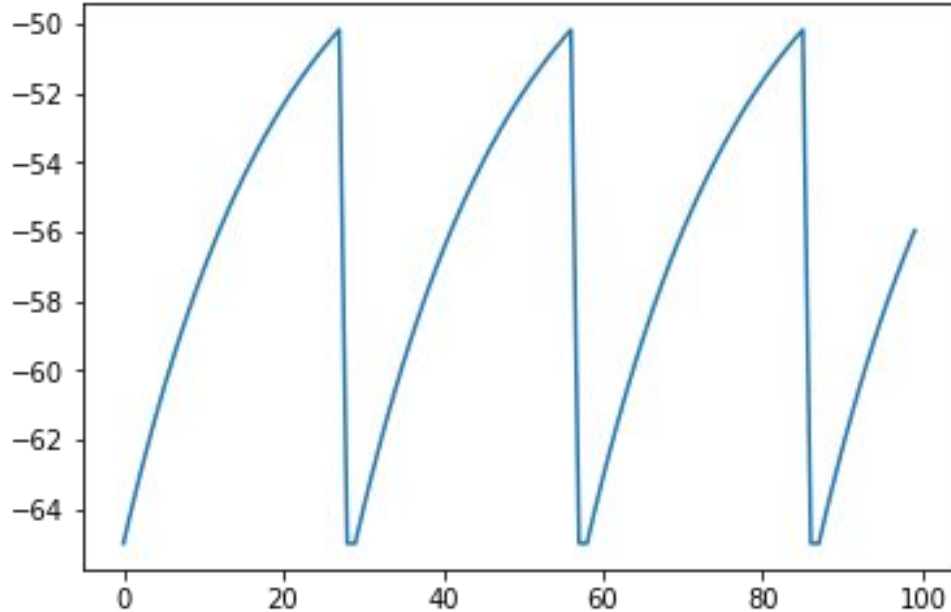
PyNN - Plotting

```
import matplotlib.pyplot as plt
```

```
plt.figure()
```

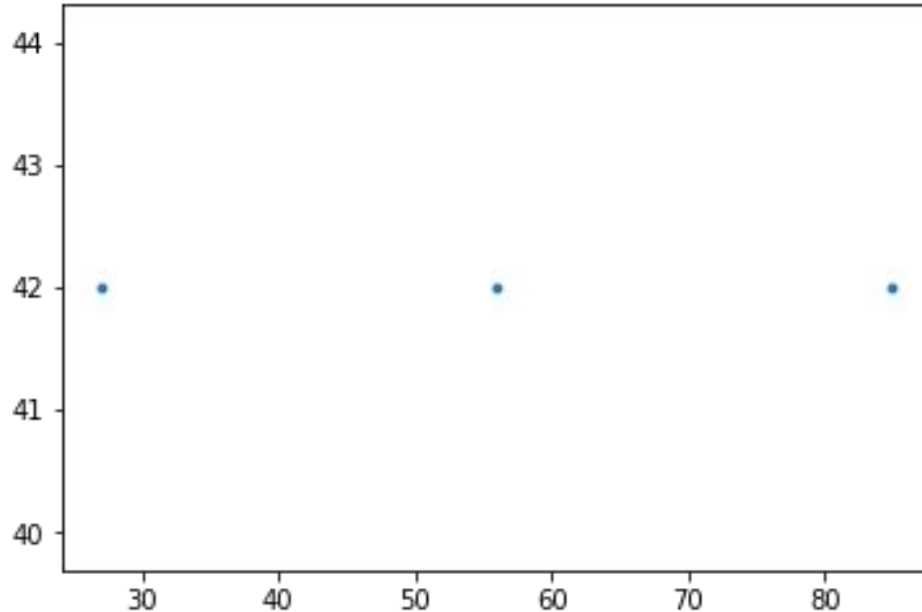
```
plt.plot(v[0].times, v[0])
```

```
plt.show()
```



PyNN - Plotting

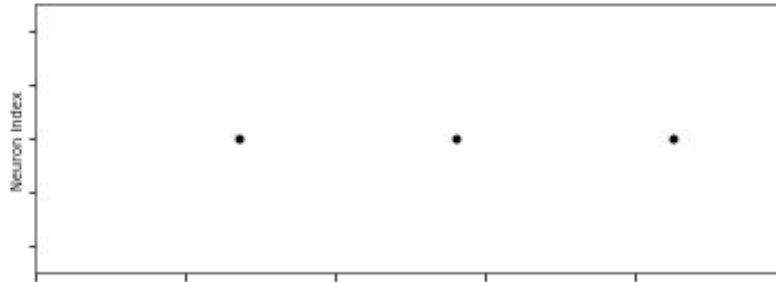
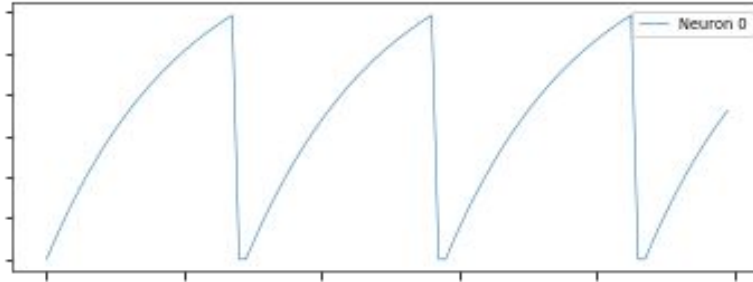
```
plt.figure()  
y = [1 for i in range(len(spikes[0]))]  
plt.plot(spikes[0], y, ".")  
plt.show()
```



PyNN - Plotting Neo Directly

```
from pyNN.utility.plotting import Figure, Panel
Figure(
```

```
    Panel(*data.segments[0].analogsignals),
    Panel(data.segments[0].spiketrains)
)
```



PyNN - Get Weights and Delays

```
synapses = proj.get(  
    ["weight", "delay"], "list")
```

```
array([(0, 5, 0.756, 1.), (0, 6, 0.316, 1.), (0, 7, 0.885, 2.),  
      (0, 8, 0.421, 1.), (1, 4, 0.618, 1.), (1, 7, 0.438, 1.),  
      (1, 9, 1.607, 1.), (2, 0, 0.129, 1.), (2, 2, 1.055, 1.),  
      (2, 3, 1.319, 1.), (2, 9, 0.422, 1.), (3, 1, 0.328, 1.),  
      (3, 3, 0.456, 1.), (3, 6, 0.566, 1.), (4, 0, 1.046, 1.),  
      (4, 1, 1.199, 1.), (4, 2, 0.831, 1.), (5, 0, 1.643, 1.),  
      (5, 2, 1.165, 1.), (5, 3, 0.902, 1.), (5, 5, 1.627, 1.),  
      (6, 0, 2.143, 1.), (6, 5, 0.635, 1.), (6, 7, 0.704, 1.),  
      (7, 0, 1.914, 1.), (7, 4, 0.289, 1.), (7, 5, 2.058, 1.),  
      (7, 6, 0.428, 2.), (7, 7, 0.639, 1.), (7, 9, 0.616, 2.),  
      (8, 0, 1.039, 1.), (8, 1, 0.576, 1.), (8, 4, 1.563, 2.),  
      (8, 8, 0.995, 1.), (9, 0, 1.686, 1.), (9, 9, 0.631, 2.)])
```


PyNN - Get Weights and Delays

```
synapses = proj.get(  
    "weight", "array")
```

```
array([[ nan,  nan,  nan,  nan,  nan,  0.756,  0.316,  0.885,  0.421,  nan],  
       [ nan,  nan,  nan,  nan,  0.618,  nan,  nan,  0.438,  nan,  1.607],  
       [0.129,  nan,  1.055,  1.319,  nan,  nan,  nan,  nan,  nan,  0.422],  
       [ nan,  0.328,  nan,  0.456,  nan,  nan,  0.566,  nan,  nan,  nan],  
       [1.046,  1.199,  0.831,  nan,  nan,  nan,  nan,  nan,  nan,  nan],  
       [1.643,  nan,  1.165,  0.902,  nan,  1.627,  nan,  nan,  nan,  nan],  
       [2.143,  nan,  nan,  nan,  nan,  0.635,  nan,  0.704,  nan,  nan],  
       [1.914,  nan,  nan,  nan,  0.289,  2.058,  0.428,  0.639,  nan,  0.616],  
       [1.039,  0.576,  nan,  nan,  1.563,  nan,  nan,  nan,  0.995,  nan],  
       [1.686,  nan,  nan,  nan,  nan,  nan,  nan,  nan,  nan,  0.631]])
```

<http://neuralensemble.org/docs/PyNN/>

PyNN 0.9.4 documentation »

[next](#) | [modules](#) | [index](#)



Table of Contents

- PyNN: documentation
 - Developers' Guide
 - API reference
 - Old documents
 - Indices and tables

Next topic

[Introduction](#)

This Page

[Show Source](#)

Quick search

Go

PyNN: documentation

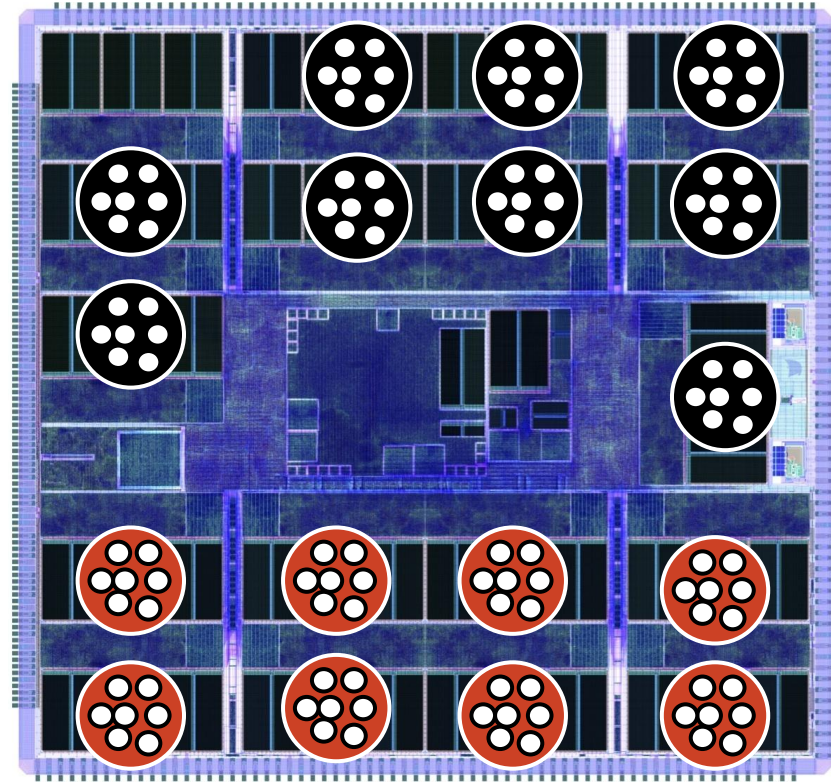
- [Introduction](#)
- [Installation](#)
- [Quickstart](#)
- [Building networks](#)
- [Injecting current](#)
- [Recording spikes and state variables](#)
- [Data handling](#)
- [Simulation control](#)
- [Model parameters and initial values](#)
- [Random numbers](#)
- [Backends](#)
- [Running parallel simulations](#)
- [Units](#)
- [Importing from and exporting to other formats](#)
- [Examples](#)
- [Publications about, relating to or using PyNN](#)
- [Contributors and licence](#)
- [Release notes](#)

Additional Non-PyNN Features

Neurons Per Core

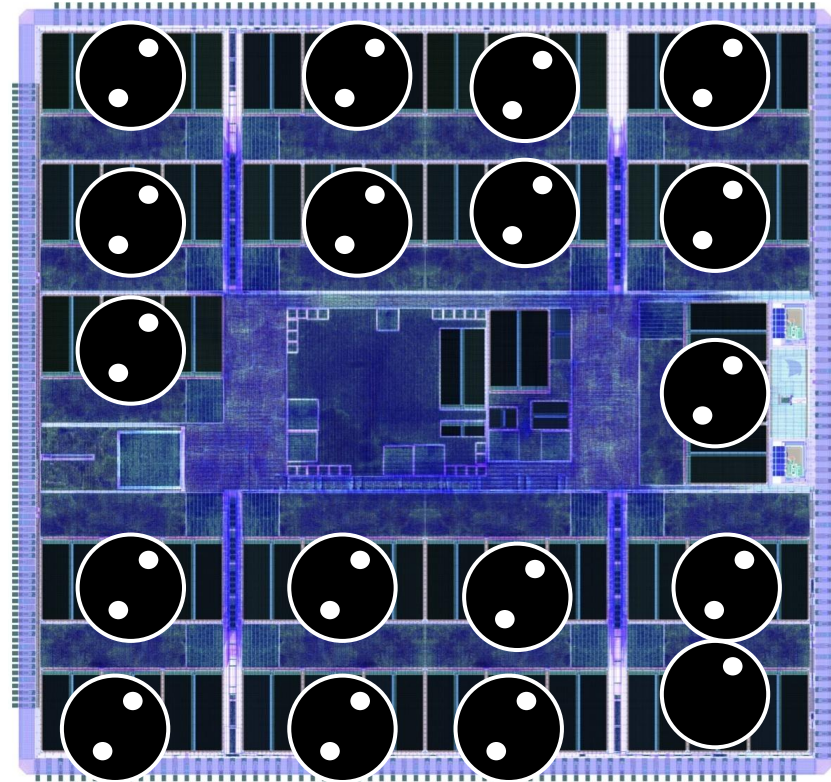
```
p.set_number_of_neurons_per_core(  
    p.IF_curr_exp, 7)
```

Default: 256



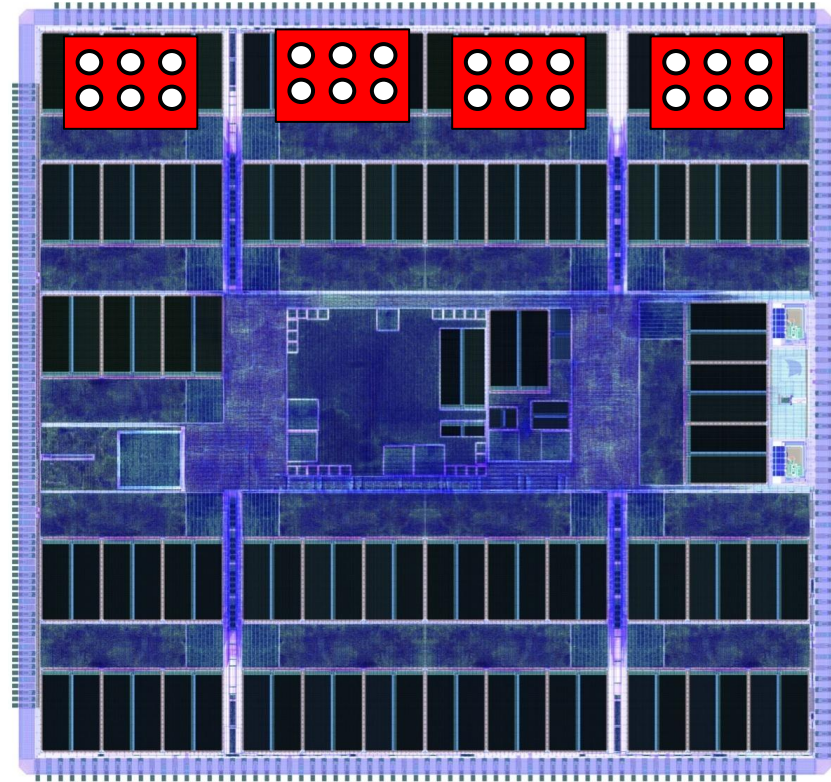
Neurons Per Core

```
p.set_number_of_neurons_per_core(  
    p.IF_curr_exp, 2)
```



Neurons Per Core

```
p.set_number_of_neurons_per_core(  
    p.IF_curr_exp,  
    (3, 2))
```



Convolution Connections

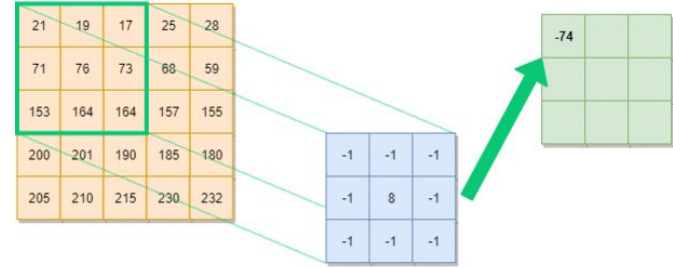
```
con = p.ConvolutionConnector(kernel=[...])
```

```
in_sz = (640, 480)
```

```
inp = p.Population(  
    in_sz[0] * in_sz[1], ...  
    structure=Grid2D(in_sz[0] / in_sz[1])
```

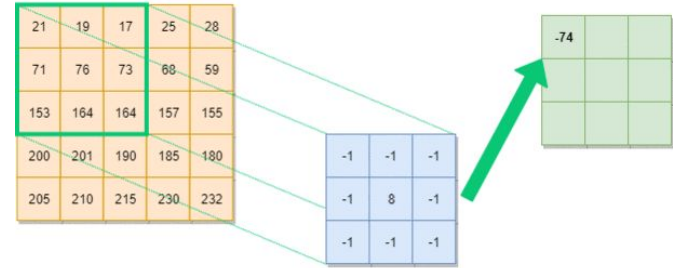
```
out_sz = conn.get_post_shape(in_sz)
```

```
outp = p.Population(  
    out_sz[0] * out_sz[1], ...  
    structure=Grid2D(out_sz[0] / out_sz[1])
```



Convolution Connections

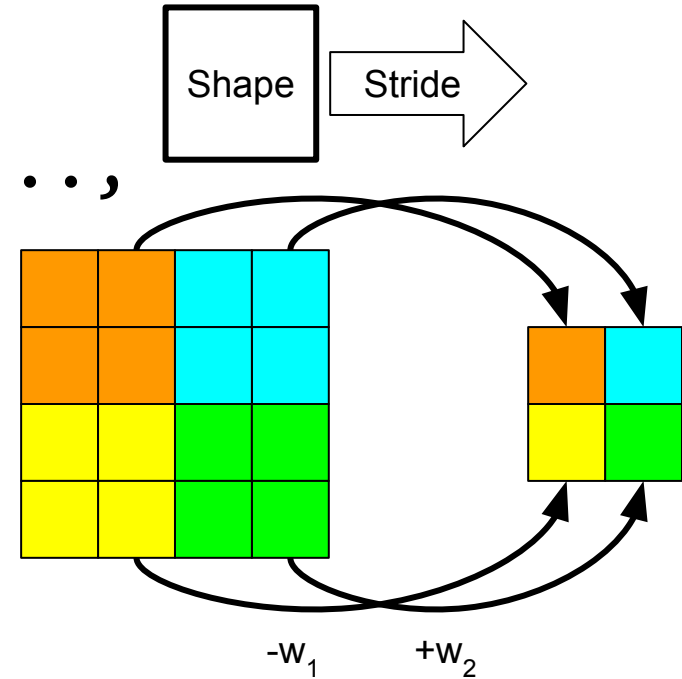
```
con = p.ConvolutionConnector(kernel=[...])
```



```
p.Projection(inp, outp, con, p.Convolution())
```

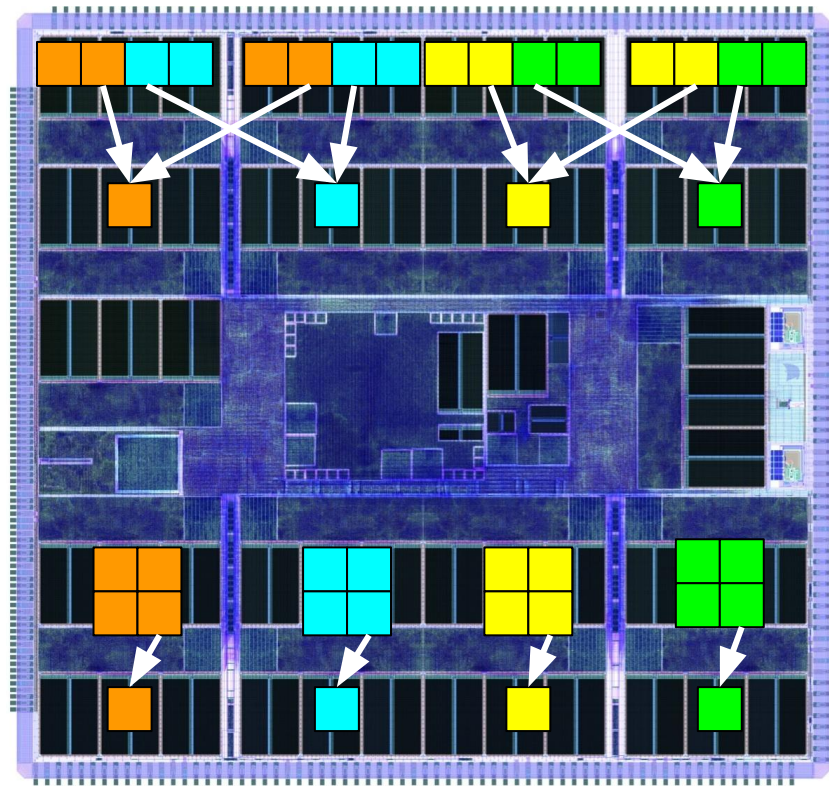
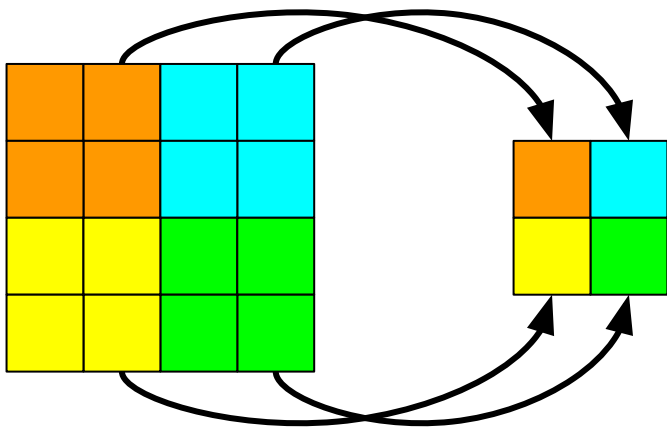

Pool Dense Connections

```
con = p.PoolDenseConnector(
    kernel=[...], pool_shape=...,
    pool_stride=...,
    positive_synapse_type=...,
    negative_synapse_type=...)
```



```
p.Projection(inp, outp, con, p.PoolDense())
```

Use of Population Structure

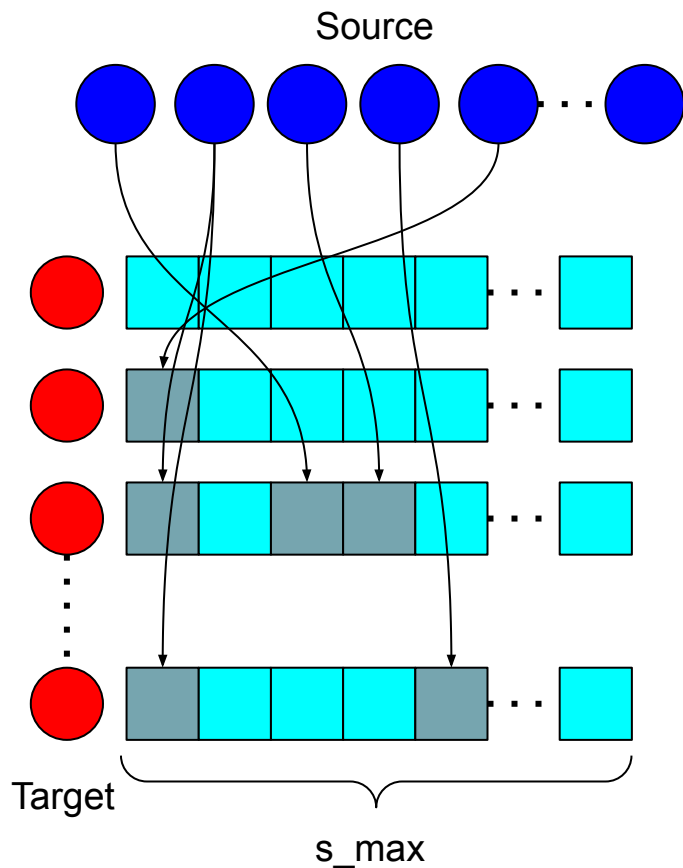


Structural Plasticity

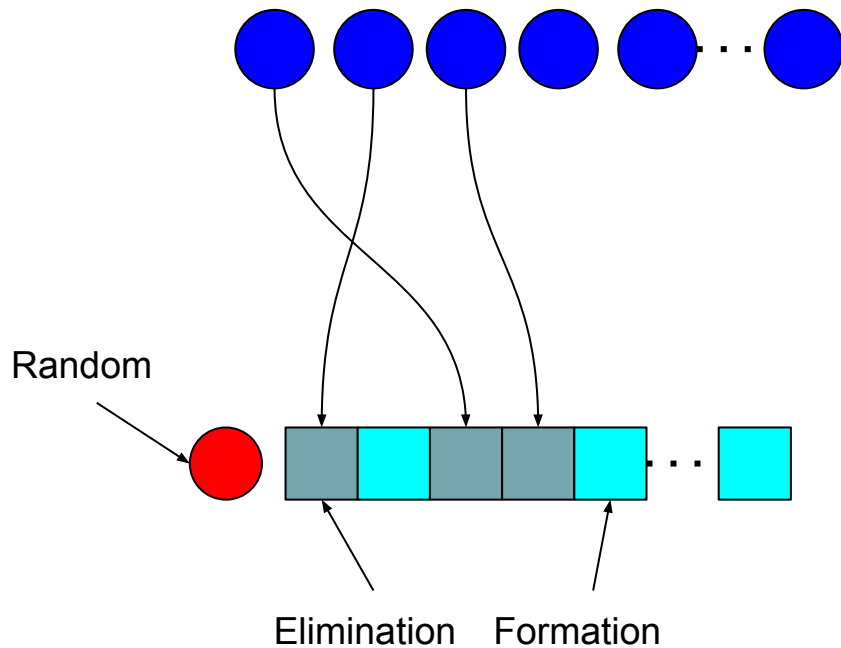
```
p.StructuralMechanism(
  f_rew=..., s_max=...,
  weight=..., delay=...,
  ...)
```



f_rew

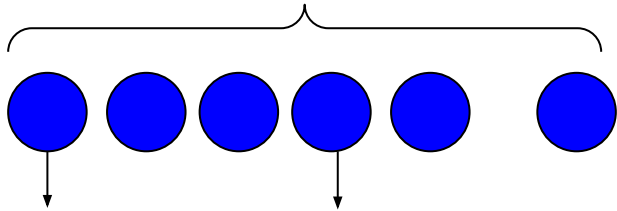


Structural Plasticity

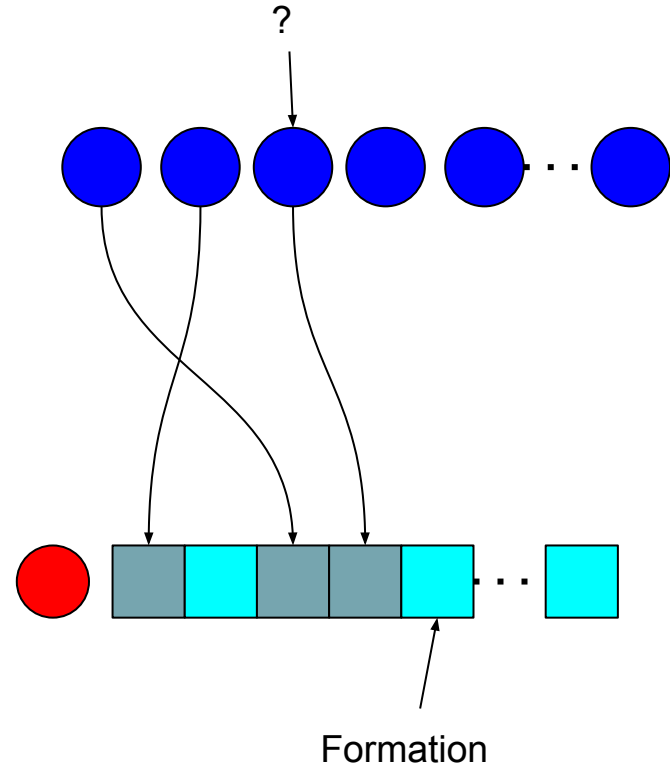


Structural Plasticity

```
p.StructuralMechanism(
  ...
  partner_selection=,
  with_replacement=, ...)
p.RandomSelection()
```



```
p.LastNeuronSelection()
```

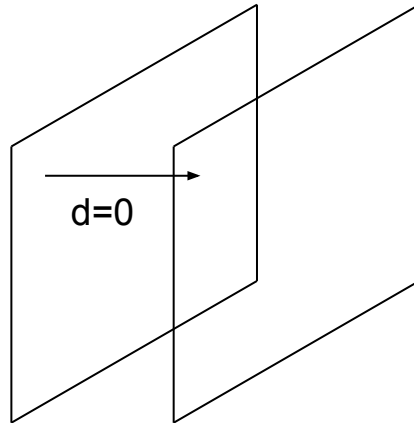


Structural Plasticity

```
p.StructuralMechanism(  
  ...
```

```
  ...
```

```
  formation=p.DistanceDependentFormation(  
    grid=[...], p_form_forward=...,  
    sigma_form_forward=...),  
  ...)
```



Structural Plasticity

```
p.StructuralMechanism(  
  ...
```

```
  ...
```

```
  elimination=p.RandomByWeightElimination(  
    threshold=...,  
    prob_elim_depressed=...,  
    prob_elim_potentiated=...),  
  ...)
```

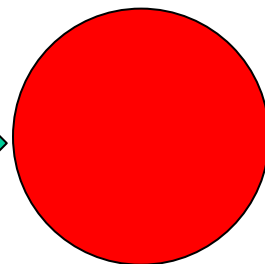


Live Input and Output

Input from Environment: Spikes

send_spikes

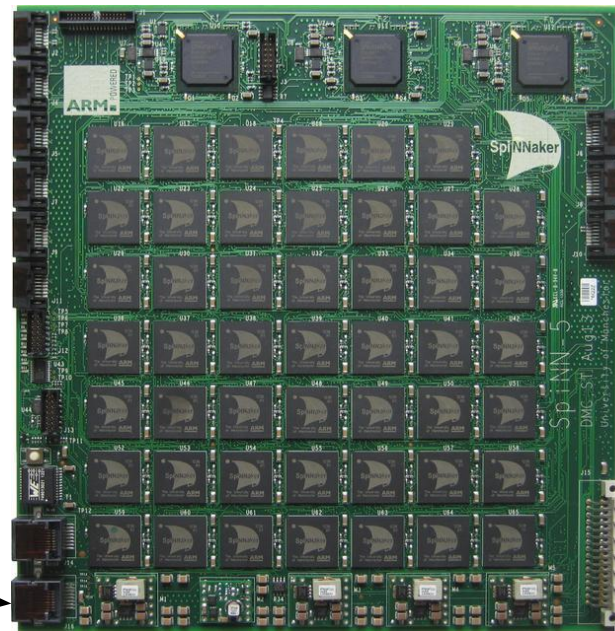
SpikeInjector
label="injector"



send_spikes("injector", [0])

```
SpynnakerLiveSpikesConnection(  
    send_labels=["injector"])
```

Multicast Key(s)
(Spike)



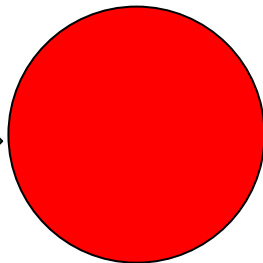
Input from Environment: Rates

add_poisson_live_rate_control



set_rates

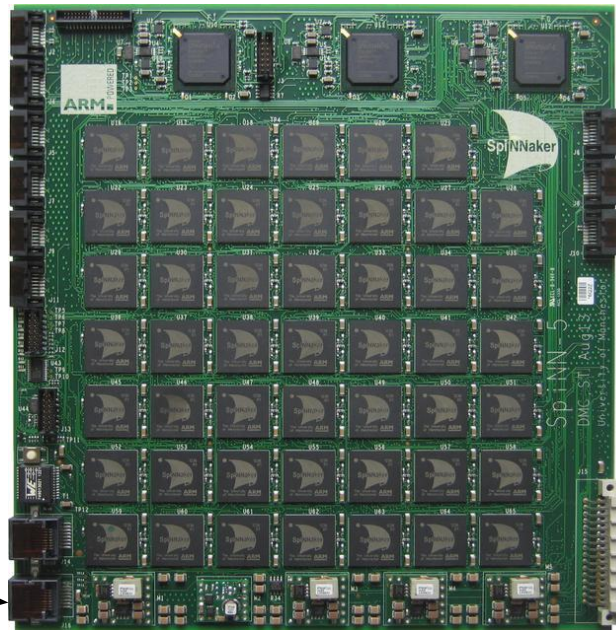
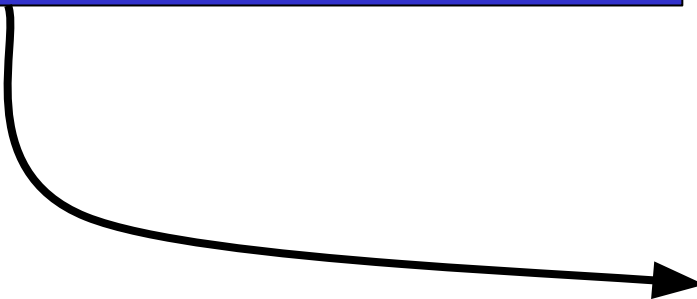
SpikeSourcePoisson
label="input"



set_rates("input", [(0, 10)])

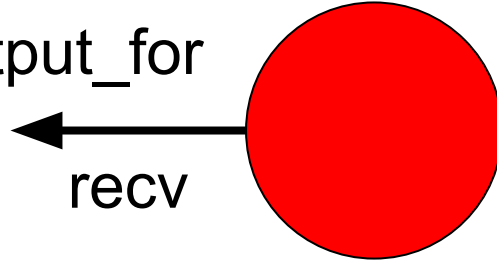
```
SpynakerPoissonControlConnection(  
    poisson_labels=["input"])
```

Multicast Key(s)
and Payload(s)



Output to Environment: Spikes

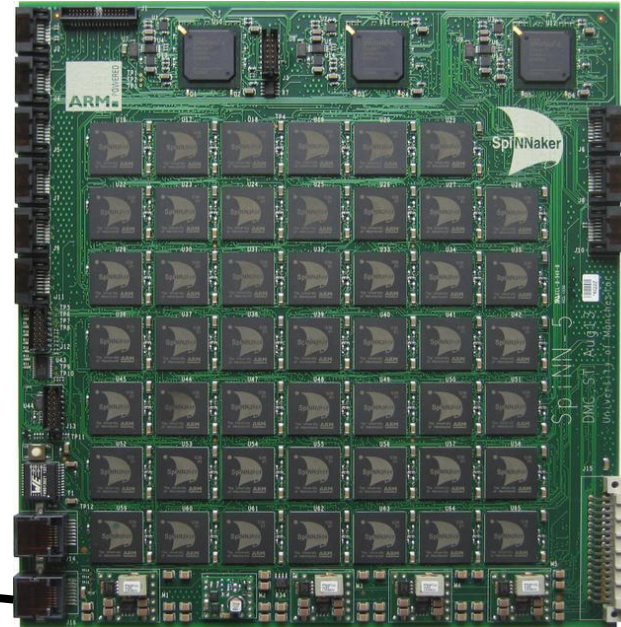
activate_live_output_for



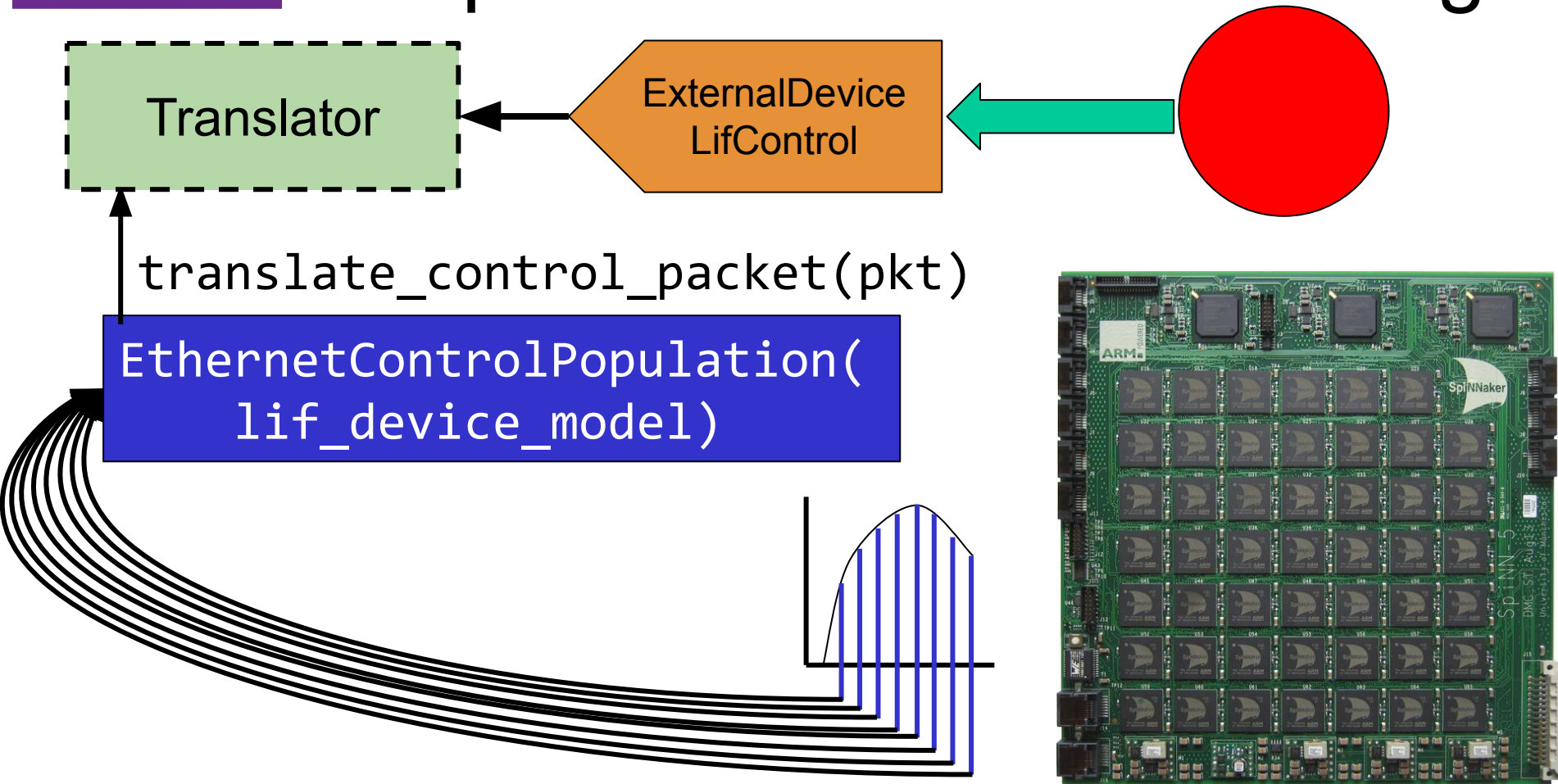
recv("pop", time, neuron_ids)

```
SpynakerLiveSpikesConnection(  
    receive_labels=["pop"])
```

Multicast Key(s)
(Spike)



Output to Environment: Voltage



SpiNNaker Tutorials

The screenshot displays the JupyterLab interface. The top navigation bar includes 'File', 'Edit', 'View', 'Run', 'Kernel', 'Git', 'Tabs', 'Settings', and 'Help'. The address bar shows the URL: `lab.jsc.ebrains.eu/user/rowley/lab/tree/shared/Andrew%20Test%20Collab/SpiNNakerJupyterE...`. The left sidebar contains a file browser with a search bar and a list of files and folders. The folder `01.RunningPyNNSimulations` is circled in red. The right sidebar shows a 'Launcher' window with the title `shared/Andrew Test Collab/SpiNNakerJupyterExamples`. It displays a grid of notebook environments, including 'Python 3 (ipykernel)', 'EBRAINS_release_v0.1_202109', 'EBRAINS-22.07', 'EBRAINS-22.10', 'EBRAINS-23.02', 'EBRAINS-23.06', 'EBRAINS-23.09', 'EBRAINS-24.04', and 'EBRAINS-experimental'. The bottom status bar shows 'Simple' mode, '0' files, '1' kernel, and 'Mem: 256.77 / 2048.00 MB'.

Name	Last Modified
00.Setup	seconds ago
01.RunningPyNNSimulations	seconds ago
02.LiveInputAndOutput	seconds ago
Integration tests	seconds ago
spinnaker_jupyter_examples	seconds ago
LICENSE	seconds ago
pyproject.toml	seconds ago
README.md	seconds ago
setup.cfg	seconds ago
setup.py	seconds ago

SpiNNaker Tutorials on Jupyter

<https://lab.ebrains.eu/>

<https://github.com/SpiNNakerManchester/>

SpiNNakerJupyterExamples

```
import matplotlib.pyplot as plt
```

```
...
```

```
plt.savefig("fig.png")
```

01. Running PyNN Simulations

02. Live Input And Output

<https://shorturl.at/oJWFf>

PyNN Documentation:

<http://neuralensemble.org/docs/PyNN/>