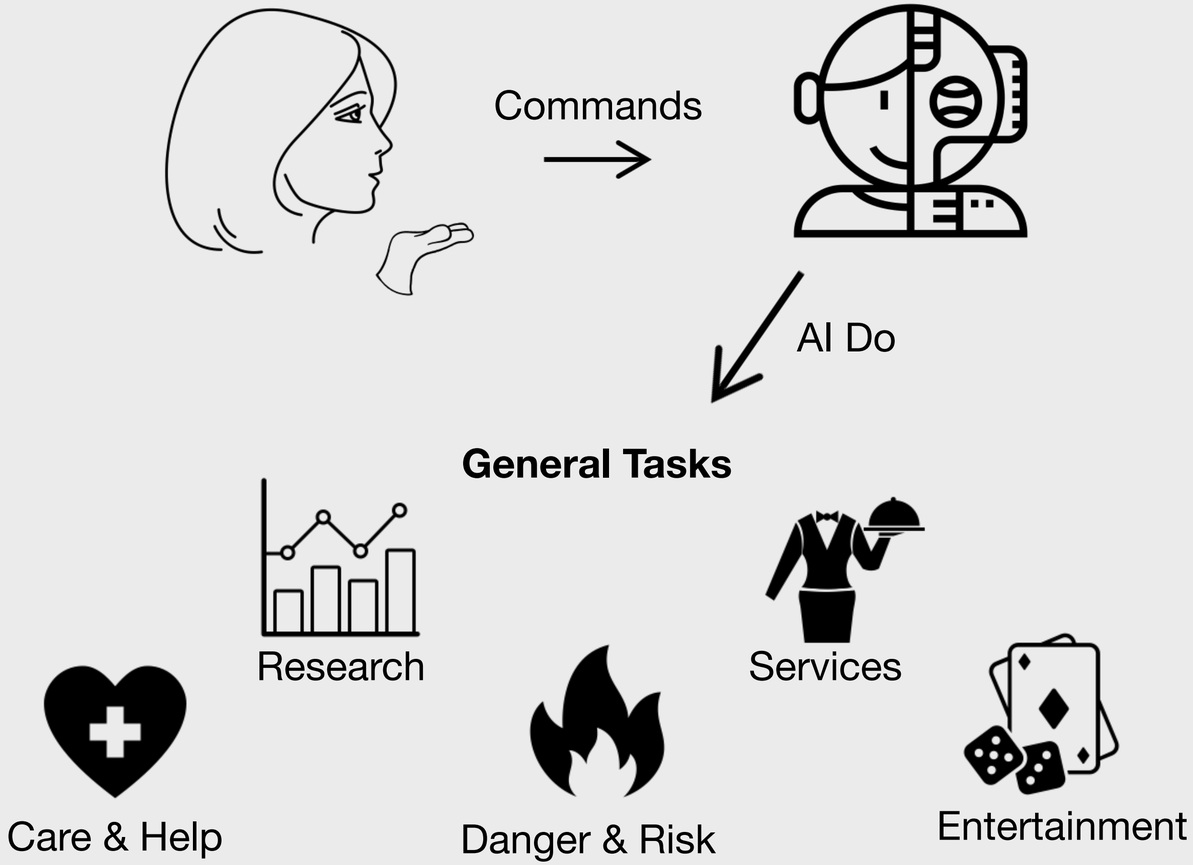
WheatNNLeek:
一個Rust麥蔥神經網路專案
A Spiking Neural Network Project by Rust

Bali Hsu 許世豪
Chief Research Officer
Libgirl Co., Ltd. 書卷姐股份有限公司

LIBGIRL

balisun@libgirl.com
https://libgirl.com

Libgirl 書卷姐股份有限公司



dong

Features

- a machine learning platform for production AI application
- free lunch: project framework to start quickly but highly-customizable
- hosting and managing trained models for you
- deploying hosted model as API with just one command
- team collaboration facilities
- for CLI lovers

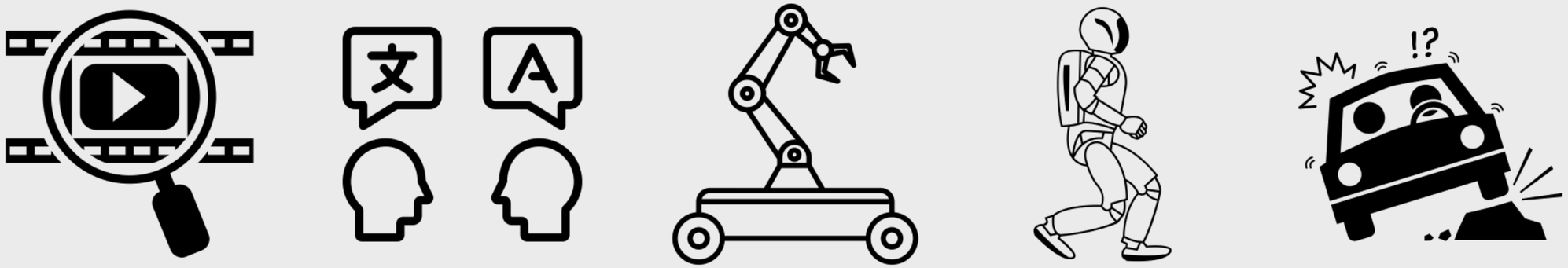
Human-Like AI: the long-term goal.

dong: MLOps platform

LIBGIRL

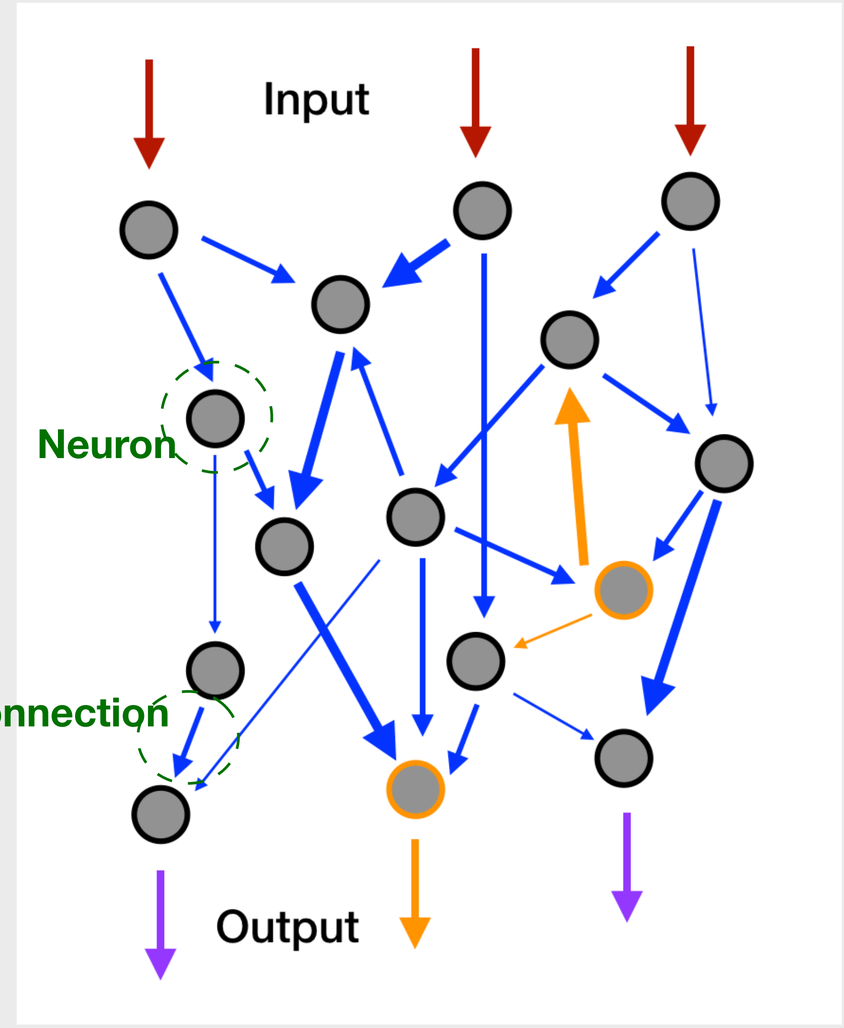
Spiking Neural Networks

- The next generation neural networks.
- The candidate for HLAI implementation, by Libgirl’s research.
- Promising for dynamic, interactive and contextual tasks.

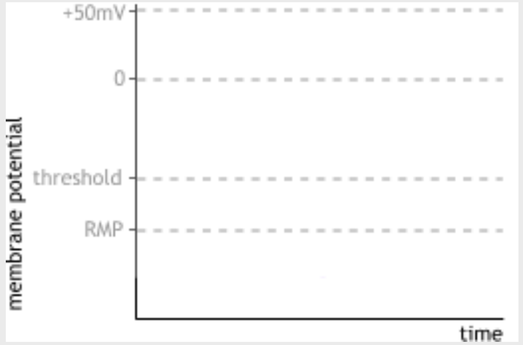


LIBGIRL

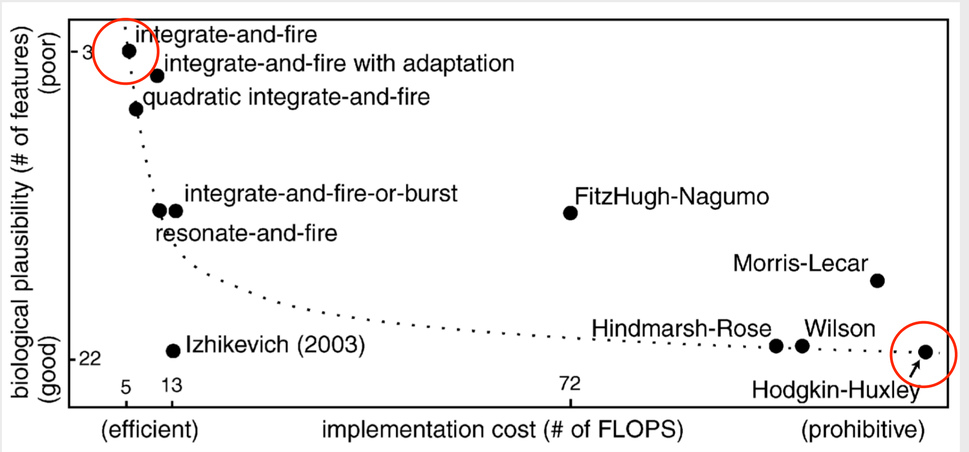
Spiking Neural Networks



Neuron Membrane Potential



Complexity ∝ Computing Power ∝ Cost



E. Izhikevich, IEEE Trans Neural Netw 15-5, 1063-1070 (2004)

LIF Model

$$\tau_m \frac{du}{dt} = -u(t) + RI(t)$$

$$\text{if } v \geq v_{\text{thresh}}, \text{ then } v \leftarrow c$$

Hodgkin–Huxley Model

$$I = C_m \frac{\mathrm{d}V_m}{\mathrm{d}t} + \bar{g}_{\mathrm{K}} n^4 (V_m - V_K) + \bar{g}_{\mathrm{Na}} m^3 h (V_m - V_{Na}) + \bar{g}_l (V_m - V_l)$$

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n$$

$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m$$

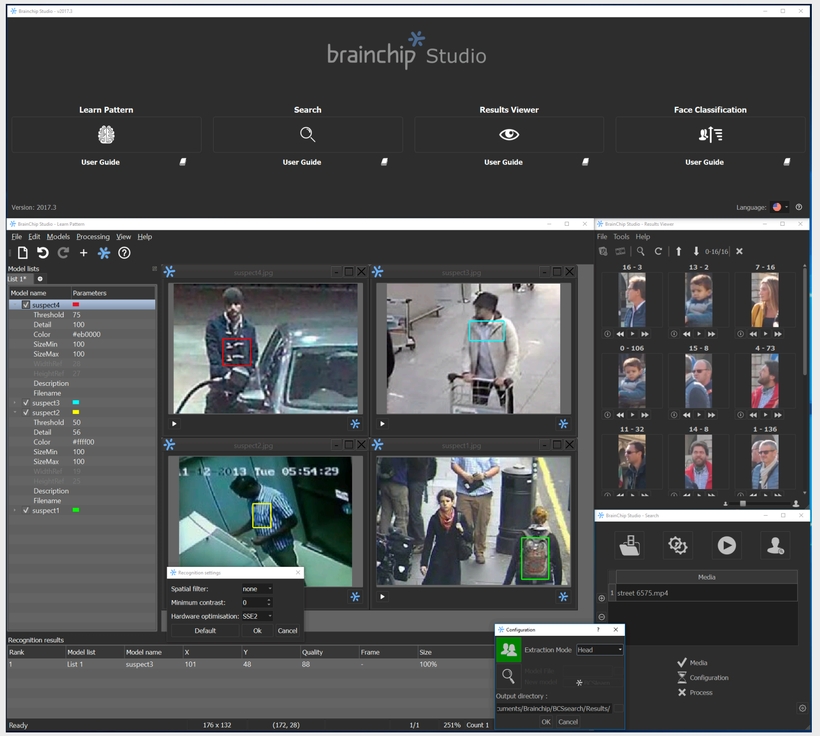
$$\frac{dh}{dt} = \alpha_h(V_m)(1-h) - \beta_h(V_m)h$$

LIBGIRL

Status of SNN Development

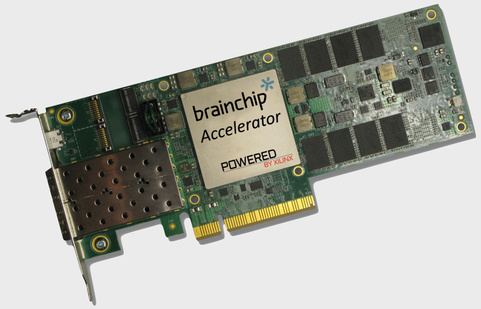
- General SNN learning algos are still missing, so Libgirl is making them.
- A commercial example: facial recognition in videos.

brainchip
The Neuromorphic Computing Company



SECURITY today
2017
NEW PRODUCT
OF THE YEAR

winner 1
2017 Innovation
AWARDS



https://www.brainchipinc.com/

LIBGIRL

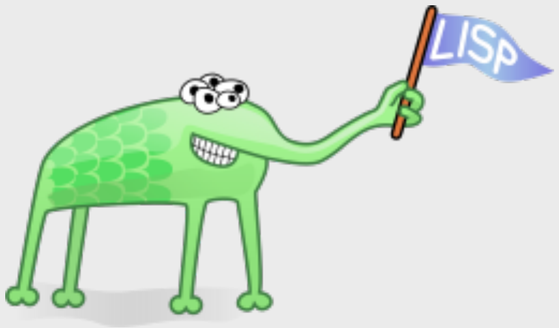
麥 神經網路 蔥

WheatNNLeek

https://github.com/libgirlenterprise/WheatNNLeek

- Existing SNN packages are mostly based on C++ / Python.
- We are rewriting everything, then why not make a new one?

Created by Mozilla



Created by John McCarthy

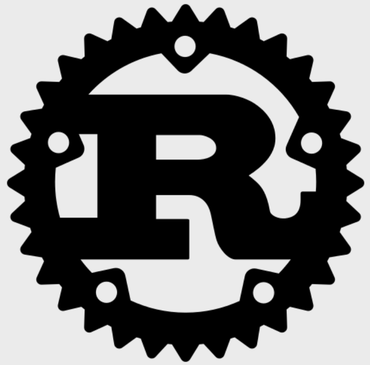
LIBGIRL

麥 神經網路 蔥

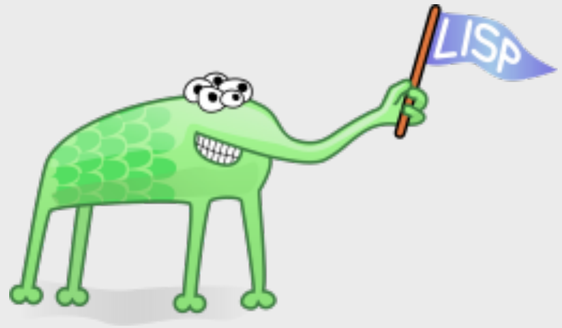
WheatNNLeek

https://github.com/libgirlenterprise/WheatNNLeek

- Why Rust?
  - Performance: system language without GC.
  - “Fearless concurrency”
  - Safety: ownership, lifetime, type inference.
- Why Common Lisp?
  - Interactive: REPL.
  - Programmable, easy to make a DSL: macro.
  - Fast prototyping.



Created by Mozilla



Created by John McCarthy

LIBGIRL

Rust - Common Lisp interface

```rust
#[no_mangle]
pub extern "C" fn Network_connect(id0: usize, id1: usize) -> *mut c_char {
    let network = NETWORK.clone();

    let mut network = network.lock().unwrap();
    let population1 = (*network).get_population_by_id(id0);
    let population2 = (*network).get_population_by_id(id1);
    let result = (*network).connect(
        &population1,
        &population2,
        &all_to_all::Connector::default(),
        &static_connection::Connection::default(),
    );
    let ret = CString::new(serde_json::to_string(&result).unwrap()).unwrap();
    ret.into_raw()
}
```

```lisp
(defun network-connect (pop-id1 pop-id2)
  (let ((p (%network-connect pop-id1 pop-id2)))
    (unwind-protect
         (let ((string (cffi:foreign-string-to-lisp p)))
           (and string
                (jonathan:parse string)))
      (%json_string_free p))))
```

extern Rust functions to C

Call C functions in Common Lisp

LIBGIRL

SNN training procedures in CL

Output Files

Open data

Create Neurons

Connect Neurons

```
(defun train (weight-save-filepath theta-save-filepath)
  (cl-wheatnnleek-cffi/ffi::network-clear)
  (with-open-file (weight-output-stream (uiop:ensure-pathname weight-save-filepath)
                                        :direction :output
                                        :if-exists :supersede)
    (with-open-file (theta-output-stream (uiop:ensure-pathname theta-save-filepath)
                                         :direction :output
                                         :if-exists :supersede)
      (setf *training-data* (get-mnist-training-data))
      (setf *training-labels* (get-mnist-training-label))
      (create-neurons)
      (let* ((input-population-id (getf *input-layer-population* :|id|))
             (excitatory-population-id (getf *excitatory-layer-population* :|id|))
             (inhibitory-population-id (getf *inhibitory-layer-population* :|id|))
             (stdp-connection-ids (network-stdp-connect input-population-id excitatory-population-id 10d0)))
        (network-static-connect excitatory-population-id
                                inhibitory-population-id
                                5d0
                                "linear"
                                "Excitatory")
        (network-static-connect inhibitory-population-id
                                excitatory-population-id
                                0d0
                                "all_to_all_except_diagonal"
                                "Inhibitory")
        (network-record-spikes excitatory-population-id)
```

⋮

LIBGIRL

SNN training procedures in CL

Training

Close data

Export training results

Configurations

Macro

Training process on every input

```
(loop-run-image-input excitatory-population-id image-label "training" nil
  (dotimes (i 28)
    (dotimes (j 28)
      (network-set-static-poisson-freq (+ (first (getf *input-layer-population*
                                                       :|neuron_ids|))
                                          (* i 28)
                                          j)
                                       0d0)))
  (network-set-property excitatory-population-id
                        "fix_theta"
                        1d0)
  (network-run 150d0))
(mnist-database:close-data *training-data*)
(mnist-database:close-data *training-labels*)
(loop for connection-id in stdp-connection-ids
      do (let ((conn-info (network-get-conn-info-by-id connection-id)))
           (format weight-output-stream
                   "~{~a~^ ~}~%"
                   (mapcar #'(lambda (keyword)
                               (getf conn-info keyword))
                           '(:|source| :|target| :|weight|)))))
(format theta-output-stream
        "~{~a~%~}"
        (network-get-property excitatory-population-id
                              "theta")))))
```

LIBGIRL

Customizable Usage by CL Macro

```
(defmacro loop-run-image-input (excitatory-population-id-symbol image-label-symbol prompt-message-prefix-string each-image-form &body body)
  `(loop for i from 0 below (mnist-database:number-of-images *training-data*)
         for k from 0 below *training-data-size-to-use*
         do (let ((image (mnist-database:image *training-data* i))
                  (,image-label-symbol (mnist-database:label *training-labels* i)))
              (format t "~a:~a/~a~a" ,prompt-message-prefix-string (1+ k) *training-data-size-to-use* #\Return)
              (force-output)
              (set-input-layer-firing-freq image)
              (let ((run-network t)
                    (intensity-plus 0))
                (loop while run-network
                      do (block continue
                           (progn
                             (network-clear-spike-records ,excitatory-population-id-symbol)
                             (network-set-property ,excitatory-population-id-symbol
                                                   "fix_theta"
                                                   0d0)
                             (network-run 350d0)
                             (let ((spike-records (network-get-spike-records)))
                               (if (> (apply '+
                                             (mapcar #'(lambda (neuron-spike-record)
                                                         (neuron-firing-count neuron-spike-record))
                                                     spike-records))
                                      5)
                                   (setf run-network nil)
                                   (progn
                                     (set-input-layer-firing-freq image
                                                                  :intensity-plus (incf intensity-plus))
                                     (return-from continue))))
                             ,@body))))
              ,each-image-form)))
```

Configurations

Training process on every input

LIBGIRL

Run a SNN in Rust

Make a network

```
fn main() {
    let mut sp0 = Supervisor::new(Time::new::<m_S>(0.5));
```

⋮

Make a neuron population

```
// build population for LIF.
let name_pp_lif = String::from("LIF Population");
let pp_lif = SimpleFiringPopulation::new();
sp0.add_firing(
    name_pp_lif.clone(),
    Arc::downgrade(&pp_lif)
);
```

Define neuron parameters

```
// build automatic-firing LIF.
let params_lif_auto = ParamsLIF {
    v_rest: Voltage::new::<m_V>(0.),
    v_reset: Voltage::new::<m_V>(13.5),
    r_m: Resistance::new::<M_Ohm>(1.),
    tau_m: Time::new::<m_S>(30.),
    tau_refrac: Time::new::<m_S>(3.),
    v: Voltage::new::<m_V>(0.),
    v_th: Voltage::new::<m_V>(15.),
    i_e: Current::new::<n_A>(20.),
    gen_dirac_v: Voltage::new::<m_V>(1.4),
    tau_m_imtt: Time::new::<m_S>(25.),
};
```

Make neurons

```
pp_lif.lock().unwrap().add(params_lif_auto.build());

// buil non-automatic-firing LIF.
let params_lif_non_auto = ParamsLIF {
    v_rest: Voltage::new::<m_V>(14.5),
    v_reset: Voltage::new::<m_V>(14.5),
    r_m: Resistance::new::<M_Ohm>(1.),
    tau_m: Time::new::<m_S>(30.),
    tau_refrac: Time::new::<m_S>(0.),
    v: Voltage::new::<m_V>(14.5),
    v_th: Voltage::new::<m_V>(15.),
    i_e: Current::new::<n_A>(0.),
    gen_dirac_v: Voltage::new::<m_V>(1.4),
    tau_m_imtt: Time::new::<m_S>(25.),
};

pp_lif.lock().unwrap().add(params_lif_non_auto.build());
```

⋮

Make a connection population

```
// build Synapse-Dirac-Delta-V
let name_pp_syn = String::from("SynapseDiracV Population");
let pp_syn = SimplePassivePopulation::new();
sp0.add_passive(
    name_pp_syn.clone(),
    Arc::downgrade(&pp_syn)
);
```

Define connection parameters

```
// build Synapse-Dirac-Delta-V
let params_syn = ParamsSynapseDiracV {
    w: Ratio::new::<ratio>(0.5),
    w_max: Ratio::new::<ratio>(1.),
    w_min: Ratio::new::<ratio>(0.),
    delay: Time::new::<m_S>(1.),
    stdp_pre_amount: Ratio::new::<ratio>(-0.005), // be careful for +/- !!
    tau_stdp_pre: Time::new::<m_S>(20.),
    stdp_post_amount: Ratio::new::<ratio>(0.005),
    tau_stdp_post: Time::new::<m_S>(15.),
    synapse_flag: SynapseFlag::STDP, // decide use STDP or Static.
};
```

Make connections

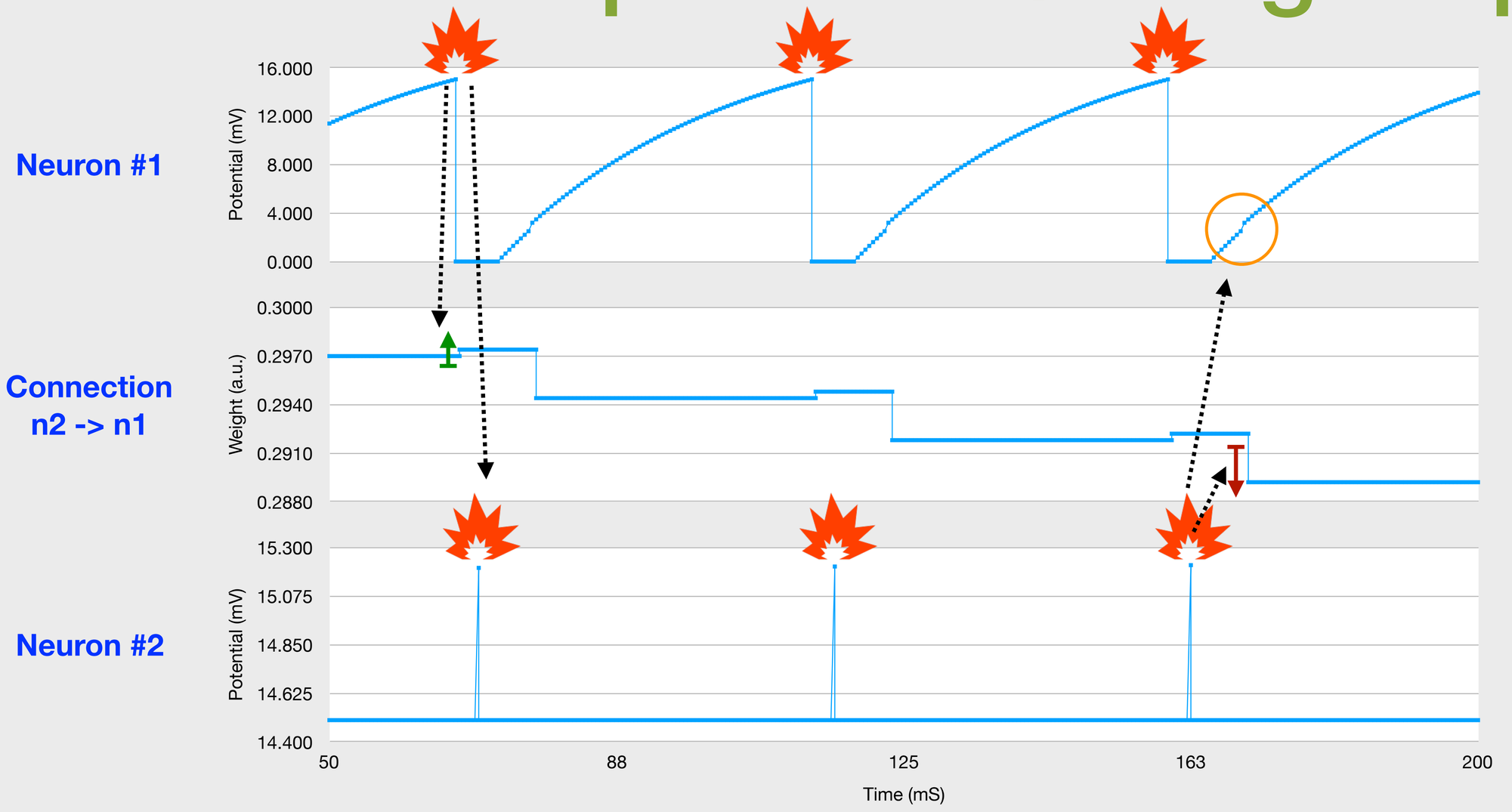
```
// n1 -> n2
let n1 = pp_lif.lock().unwrap().agent_by_id(0);
let n2 = pp_lif.lock().unwrap().agent_by_id(1);
pp_syn.lock().unwrap().add(params_syn.build_to_active(n1, n2));

// n2 -> n1
let n1 = pp_lif.lock().unwrap().agent_by_id(0);
let n2 = pp_lif.lock().unwrap().agent_by_id(1);
pp_syn.lock().unwrap().add(params_syn.build_to_active(n2, n1));
```

Run

```
println!("start run.");
sp0.run(Simulate::Concurrent, RunMode::ForwardStepping, Time::new::<m_S>(1000.0));
```

LIBGIRL

Membrane potential / weights plots



Aaaa

LIBGIRL

uom: Type-Safe Zero-Cost Dimensional Analysis

- Let the rust type system enforce correctness on your equations!

https://crates.io/crates/uom

https://github.com/iliekturtles/uom

LIBGIRL

uom: Type-Safe Zero-Cost Dimensional Analysis

Example: Compile error on mismatched dimensionality

```
fn main() {
    let some_currrent = Current::new::<n_A>(1.0);
    let some_voltage = Voltage::new::<m_V>(1.0);
    let mistake = some_currrent + some_voltage;
}
```

```
error[E0308]: mismatched types
  --> src/main.rs:13:35
   |
13 |     let mistake = some_currrent + some_voltage;
   |                                   ^^^^^^^^^^^^ expected struc\
t `typenum::int::Z0`, found struct `typenum::int::PInt`
```

Equation

$$\tau_m \frac{du}{dt} = -u(t) + RI(t)$$

Type Definition

```
v_rest: Voltage,
v_reset: Voltage,
r_m: Resistance,
tau_m: Time,
tau_refrac: Time,
v: Voltage,
v_th: Voltage,
i_e: Current,
gen_dirac_v: Voltage,
```

Implementation

```
fn continuous_evolve(&mut self, dt: Time) {
    self.v += rk4(
        |y| (-(y - self.v_rest) + self.i_e * self.r_m) / self.tau_m,
        self.v,
        dt);
}
```

LIBGIRL

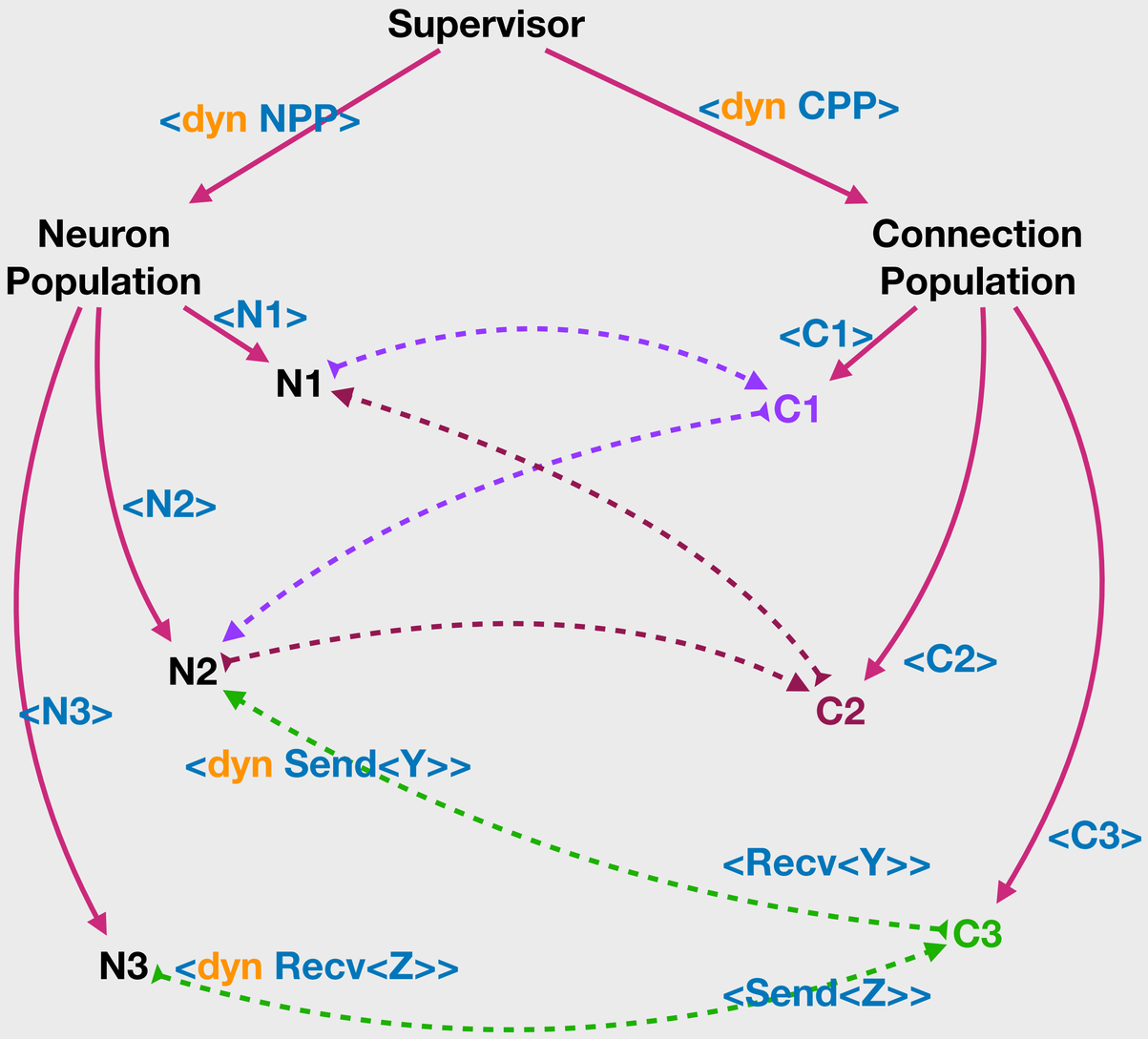
Impression on Rust programming

“If you find it difficult to compile, your might have to reflect on your logic.”
「如果你發現很難 compile 過，你可能要好好檢討你自己。」

– a senior engineer 一位資深工程師

LIBGIRL

A Scenario in WheatNNLeek

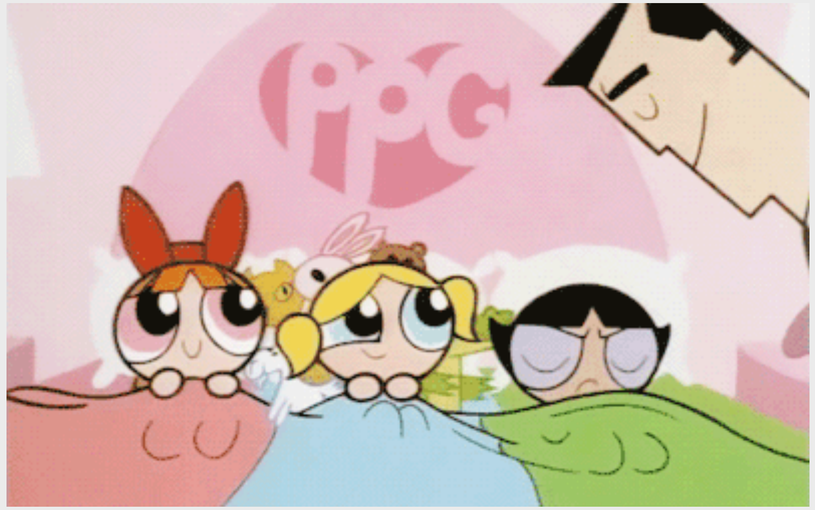


Object	Methods in scenarios			
	Config by Population	Check connection config	Step-by-step run	Response on neuron's event
Neuron	config() by population	mode() by connection	evolve() by population	
Connection	config() by population	recheck_config() by population		response() by neuron
population	config() by supervisor	recheck_config() by supervisor	evolve() by supervisor	

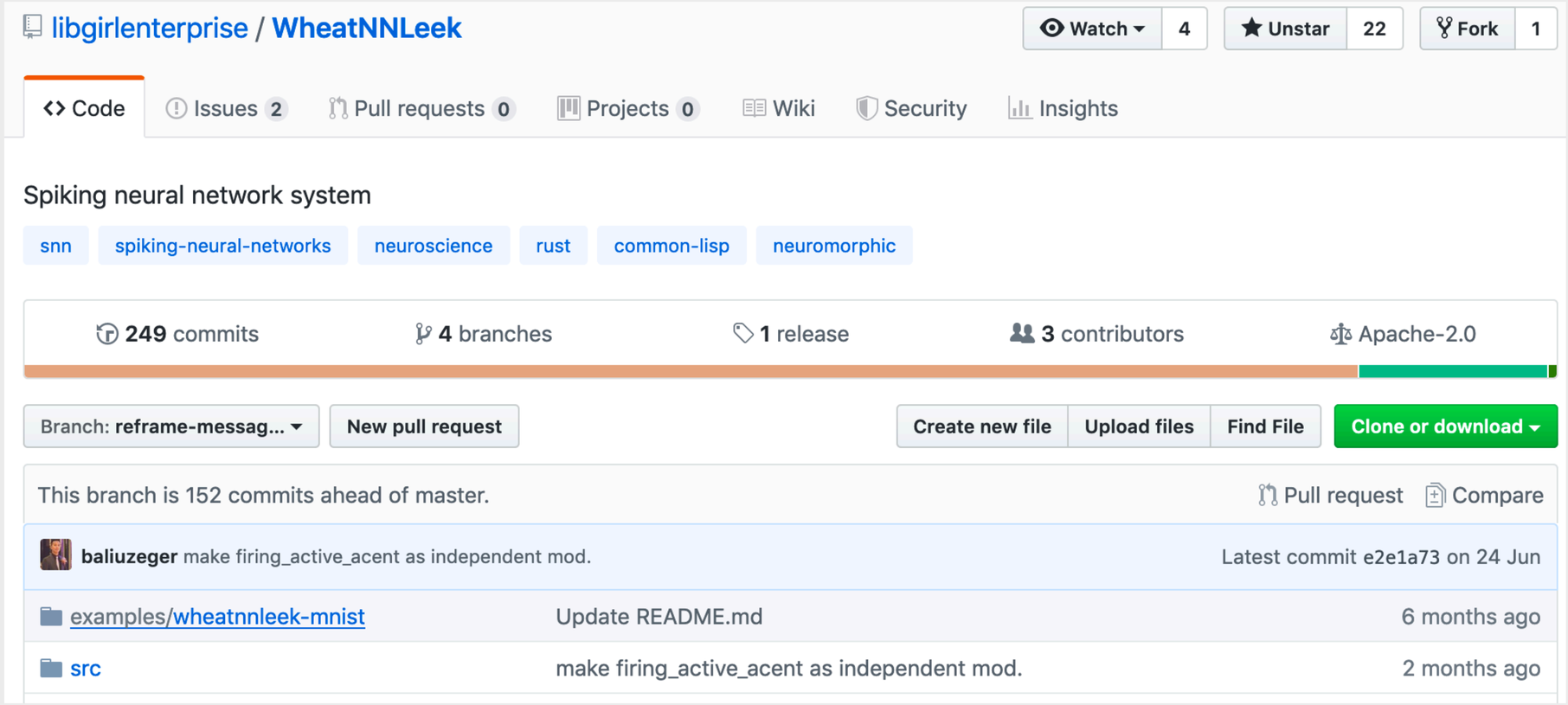
LIBGIRL

Impression on Rust programming

“Once it compiles, the world is saved. thanks to Rust!”
「在成功 compile Rust 專案以後，碼農的一天又恢復了和平。」



Let’s play with WheatNNLeek!



LIBGIRL

balisun@libgirl.com
https://libgirl.com