[Re] Connectivity reflects coding: a model of voltage-based STDP with homeostasis

René Larisch[1]

1 Professorship for Artificial Intelligence, Department of Computer Science, Chemnitz University of Technology, D-09107 Chemnitz, Germany

rene.larisch@informatik.tu-chemnitz.de

Editor
Name Surname

Reviewers
Name Surname
Name Surname

Received Feb, 1, 2019
Accepted Feb, 1, 2019
Published Feb, 1, 2019

A reference implementation of

→ Connectivity reflects coding: a model of voltage-based STDP with homeostasis, C. Clopath, L. Büsing, E. Vasilaki and W. Gerstner, In: Nature Neuroscience 13.3 (2010), pp. 344–352, doi= 10.1038/nn.2479

Introduction

Since the first description of spike timing-dependent plasticity (STDP) [2], different models of STDP have been published to reproduce different experimental findings. Early implementations such as pair-based STDP learning rules failed to reproduce some experimental observations, for example triplet or quadruplets experiments [8].

Clopath et al. [4] introduced a STDP model which is able to reproduce the experimental findings of triplet studies. They propose a biologically-motivated model with a voltage-based learning rule, where the occurrence of long term depression (LTD) or long term potentiation (LTP) depends on the postsynaptic membrane voltage. Clopath and colleagues could reproduce how the occurrence of LTD and LTP depends on the depolarizing of the postsynaptic membrane potential, as observed in voltage-clamp [6] and stationary-depolarization experiments [1]. Further, they could reproduce experimental findings such as spike pair repetition and triplet experiments [9], as well as spike bursting experiments [5]. They were able to show that their learning rule can develop stable weights, as needed for learning the receptive fields of simple cells in the primary visual cortex (V1). They implemented a homeostatic mechanism to control the level of generated LTD, based on the relationship between the average postsynaptic membrane potential and a reference value. Their model led to two different connectivity structures, depending on the spiking behavior of the neurons: if the neurons fire strongly at the same time, they build strong bidirectional connections (as in correlation-based Hebbian learning). If they fire in a specific temporal order, their connectivity structure follows that order (temporal coding).

Methods

Overview

The original model was implemented in Matlab (http://modeldb.yale.edu/144566) to demonstrate the stable learning of weights. This model reimplementation is written in Python (2.7, 3.4 or later) with the help of the neuro-simulator ANNarchy[1] [10] (version 4.6.6), numpy (version 1.11.0) and matplotlib (version 1.5.1). The reimplementation is mainly based on the description of neuron model and learning rule in the original publication [4]. Because of the lack of further description of the homeostatic mechanism and the neural behavior after a emitted spike, the Matlab code is used as the second reference for this reimplementation.

Model description

Neural model

Clopath et al. [4] used an adaptive exponential integrate-and-fire (AdEx) neuron in their model. The exact neural model is mainly derived from the description in the Matlab source code (Eq. 1):

$$C\frac{du}{dt} = -g_L\,(u - E_L) + g_L\,\Delta_T\,e^{\frac{u-V_T}{\Delta_T}} - w_{ad} + z + I \tag{1}$$

where u is the membrane potential, C the membrane capacitance, g_L the leak conductance and E_L the resting potential. The slope factor (Δ_T) and the spiking threshold (V_T) are describing the behavior of the exponential term.

If the membrane potential (u) is above V_T, the neuron spikes and the membrane potential increases exponentially. To simulate the spike upswing for 2 ms after a spike was emitted, they used the so-called *resolution trick*: they simulate the complete process in the membrane potential through a spike once with high precision and integrated over the complete process to calculate the entire change in the membrane potential. For simulations, they used the integrated value and fixed the membrane potential for 2 ms. This means that the membrane potential is set to 29.4 mV after a spike, one millisecond later to 29.4 mV +3.462 mV and another millisecond later to $E_L + 15$ mV +6.0984 mV.

The depolarizing spike afterpotential is z and it decays over time to zero (Eq. 2).

$$\tau_z\frac{dz}{dt} = -z \tag{2}$$

The hyperpolarization current is described by w_{ad} (Eq. 3). After a spike, w_{ad} is increased by b and decreases exponentially to the resting potential E_L otherwise.

$$\tau_{w_{ad}}\frac{dw_{ad}}{dt} = a(u - E_L) - w_{ad} \tag{3}$$

The adaptive spiking threshold (V_T) is set to $V_{T_{max}}$ after a spike and decays exponentially to $V_{T_{rest}}$ (Eq. 4).

$$\tau_{V_T}\frac{dV_T}{dt} = -(V_T - V_{T_{rest}}) \tag{4}$$

The values for the parameters of the model are taken from the original paper.

Synaptic model

The proposed learning rule consists of two terms: long term potentiation (LTP) and long term depression (LTD). The LTP term (Eq. 5) controls the increase in synaptic efficiency:

$$LTP = A_{LTP}\,\bar{x}_i\,(u - \theta_+)_+\,(\bar{u}_+ - \theta_-)_+ \tag{5}$$

[1] https://bitbucket.org/annarchy/annarchy

A_{LTP} is the learning rate for LTP. The parameters θ_+ and θ_- are plasticity thresholds for the membrane potential (u) and its temporal average ($\bar{u}_+$, Eq. 6), respectively. The original paper does not mention the meaning of these thresholds. The chosen value $\theta_+ = -45.3$ mV is above the spiking threshold. This suggests that this threshold prevents the occurrence of LTP if the postsynaptic neuron has not spiked already. With $\theta_- = E_L$, LTP only occurs if the membrane potential is above the resting potential.

$$\tau_+ \frac{d\bar{u}_+}{dt} = -\bar{u}_+ + u \tag{6}$$

After each presynaptic spike, the spike train $\bar{x}_i$ is increased by 1 and decays exponentially with a time constant τ_x (Eq. 7). The spike counter X_i is 1 when the presynaptic neuron spikes at time t, otherwise 0.

$$\tau_x \frac{d\bar{x}_i}{dt} = -\bar{x}_i + X_i \tag{7}$$

With this term, LTP occurs when the presynaptic spike trace ($\bar{x}_i$) is above zero, the postsynaptic membrane potential u is over the threshold θ_+ and the membrane potential trace $\bar{u}_-$ is above θ_-. This happens whenever the postsynaptic neuron spikes shortly after the presynaptic neuron or if the membrane potential is high long enough, i.e. when $\bar{u}_-$ exceeds θ_-.

The LTD term (Eq. 8) governs the decrease of the synaptic efficiency:

$$LTD = A_{LTD} \, (\frac{\bar{\bar{u}}}{u_{ref}^2}) \, X_i \, (\bar{u}_- - \theta_-)_+ \tag{8}$$

The presynaptic spike counter (X_i) is set to one after a spike and zero otherwise. $\bar{u}_-$ is a second trace of the postsynaptic membrane potential similar to $\bar{u}_+$, but with $\tau_- > \tau_+$. If it exceeds the threshold θ_- and a presynaptic spike is emitted, LTD occurs. This happens when the presynaptic neurons spikes after the postsynaptic one.

The amplitude of the LTD term, and with that the balance between LTP and LTD, is adjusted with respect to the ratio between $\bar{\bar{u}}$ and a reference value (u_{ref}^2), hence implementing a homeostatic mechanism (Eq. 9).

$$\tau_{\bar{\bar{u}}} \frac{d\bar{\bar{u}}}{dt} = [(u - E_L)^2] - \bar{\bar{u}} \tag{9}$$

The homeostatic variable $\bar{\bar{u}}$ is computed over the quadratic difference between the postsynaptic membrane potential and the resting potential (E_L). When the postsynaptic neuron fires frequently, $\bar{\bar{u}}$ increases, leading to a higher level of LTD and weight decreases. In contrast, a lower postsynaptic activity decreases the level of LTD and the weights can increase. Through the ratio of $\bar{\bar{u}}$ with u_{ref}^2, this mechanism can enforce the connections to decrease down to the minimum weight bound or increase to the maximum weight bound. This requires a hard upper and lower bound for the weights and leads to a binomial distribution of the weights. The weight change over time depends on both the positive LTP term and the negative LTD term (Eq. 10):

$$\frac{dw}{dt} = LTP - LTD \tag{10}$$

All parameters of the neuron model and the basis set of parameters for the learning rule are taken from the original publication. Some parameters of the learning rule differ from experiment to experiment, in particular the reference value of the homeostatic mechanism (u_{ref}^2), the learning rates for the LTP and LTD terms (A_{LTP} and A_{LTD}), θ_- and the maximum weight value (w_{max}). A table with the different parameters for each task is presented in Table 1 and Table 2.

Table 1: Changed parameters for connectivity experiments.

Task	Parameter	Value
Rate based connectivity	w_{max}	$0.25nA$
Temporal based connectivity	w_{max}	$0.30nA$
Stable weight by Gaussian input	w_{max}	$3.0nA$
Stable weight by Gaussian input	A_{LTD}	$1.4 * 10^{-3}$
Stable weight by Gaussian input	A_{LTP}	$0.8 * 10^{-3}$
receptive fields	A_{LTP}	$9.6 * 10^{-5}$
receptive fields	A_{LTD}	$8.4 * 10^{-5}$

Table 2: Changed parameters for weight change experiments.

Task	Parameter	Value
STDP learning window	$\bar{\bar{u}}$	$80mV^2$
STDP learning window	θ_-	$-60.0mV$
triplet experiment	$\bar{\bar{u}}$	$120mV^2$

Reproduction of experiments

Parameter setup

The experimental protocols are based on the description on the publication of Clopath et al. [4]. The learning rule was mainly implemented according to the available Matlab source code. The development of the synapses depends on the behavior of the postsynaptic membrane potential, especially for the first two milliseconds after a postsynaptic spike. Although ANNarchy makes it easy to write down the model equations to build up networks, the processing order of the equations is strictly defined by ANNarchy [10].

At each simulation step, neural variables are first updated using spikes emitted at the previous time steps. Spikes are then emitted using the defined conditions. Synaptic variables are then updated (including weight changes if presynaptic or postsynaptic event have been detected). Finally, the value of all desired variables is recorded for that step. Because of this design choice, the execution order for differential equations and non-differential equations of the synapses and neurons is different from the order mentioned in the published Matlab source code. This forced us to change the value of some parameters or some of the simulations. These changed values can be found in the associated Python files and in the tables below.

The chosen integration time step can have an influence of the computation result, as well. In the original publication, no integration time step is mentioned. In the published Matlab source code, a time step of $dt = 1$ ms is chosen, which we also use.

Experiment descriptions

In the original publication, the authors reproduce spike timing triplet experiments in the visual cortex of rats [9]. Furthermore, they investigate the resulting connectivity structure depending on the spiking behavior.

To validate the reimplementation, we reproduce the voltage clamp experiment (Fig. 1h in [4]), the classical spike timing-dependent learning window (Fig. 2a in [4]), the frequency repetition task to reproduce a triplet experiment (Fig. 2b in [4]), the burst timing-dependent plasticity experiment (Fig. 3 in [4]), the influence of spiking order to connectivity (Fig. 4a, down and Fig. 4b, down in [4]) and the emergence of receptive fields by presenting natural scenes (Fig. 7d in [4]).

To reproduce the voltage clamp experiment, the presynaptic neuron spikes with a constant firing rate of $25Hz$ for $50ms$. The postsynaptic membrane potential is changed from a fixed value of $-80mV$ to $0mV$. We recorded for different values of the postsynaptic membrane potential the weight change. Resulting changes in the learning rule are implemented as mentioned in the original publication.

To reproduce the STDP learning window, we create a list of discrete time points where the pre- or postsynaptic neurons should emit spikes. The presynaptic neuron spikes every 50 ms. The postsynaptic neuron spikes in a range from 1 ms to 15 ms before or after the presynaptic neuron. Both neurons are AdEx neurons and connected to one input neuron each to control the spiking behavior. As mentioned in the original publication the variable of the homeostatic mechanism is set to ($\bar{\bar{u}} = u_{ref}$).

For the repetition frequency experiment and the triplet experiment, the number of pre- and postsynaptic spike pairs increases from a pair frequency of 0.1 Hz to 50 Hz. The time between a pre- and postsynaptic spike of a pair is 10 ms. As in the previous experiment, we implement a network with two AdEx neurons connected to each other in order to observe the weight change. To reproduce this experiment, it was necessary to set $\bar{\bar{u}}$ to a fixed value, as mentioned in the original publication [4]. The parameter changes are shown in Table 2.

Clopath and colleagues used three burst timing-dependent plasticity experiments. In the first task, they changed the number of postsynaptic spikes from one up to three, with either $+10$ ms or -10 ms between the presynaptic and the first postsynaptic spike. The postsynaptic neuron fires at 50 Hz. In the second experiment, they recorded the weight change when one presynaptic spike is followed by three postsynaptic spikes with varying postsynaptic firing rates (from 20 Hz to 100 Hz). As in the first experiment, they observe the weight change for the case when the first postsynaptic spikes appears 10 ms after or 10 ms before the presynaptic spike. The variation of the time lag between one presynaptic spike and three postsynaptic spikes is the focus of the third experiment. For that, the postsynaptic neuron fires with constant 50 Hz and the time lag varies from -100 to $+60$ ms. To implement these experiments, we implement a network with two of the AdEx neurons. Each of these neurons is connected to one input neuron to control the discrete time points of the spiking events. For all three burst spiking experiments, the normal parameter set is used, and $\bar{\bar{u}}$ is set to a fixed value, as mentioned in the original publication.

To analyze the connectivity dependence on the number of spikes, a small network with ten interconnected AdEx neurons is built. Each neuron receives an input from one additional neuron, with Poisson-distributed spike patterns. The firing rate of each Poisson neuron is increased from 2 Hz to 20 Hz, influencing the firing rate of the 10 corresponding neurons in the network. To analyze how the temporal order of spiking influences the connectivity, the recurrent connected neurons receives input from neurons, they spiking $20ms$ one after another to realize a temporal spiking order of the ten recurrent neurons. To establish stable weights, the normal homeostatic mechanism is used.

For the emergence of V1 simple-cell-like receptive fields, a network with one postsynaptic AdEx neuron and $16 \times 16 \times 2$ presynaptic neurons is used. As mentioned in the original publication [4], the activity of the presynaptic population depends on the pixel values of a 16×16 pixel sized patch, cut out of pre-whitened natural scenes [7]. The maximum input firing rates in the original publication are set to 37.5 Hz. In the here presented reimplementation, the maximum firing rate is set to 50 Hz. Further, the learning rates for the LTP term (A_{LTP}) and the LTD term (A_{LTD}) are reduced (see Table 1). The pixel values of the patch are normalized with the maximum pixel value of the current image and divided into an ON (only positive pixel values) and an OFF (only negative pixel values) image. The presynaptic population spikes are generated by a Poisson process. To establish stable weights, the normal homeostatic mechanism is used.

Besides the experiments from the original publication, we reimplemented the experiment for the emergent of stable weights out of the Matlab source code. The emergence of stable weights was achieved by presenting a Gaussian input over 500 presynaptic neurons and one postsynaptic AdEx neuron. For every trial (125 ms) ten Gaussian patterns are created to determine the activity of the 500 input neurons. As in the Matlab source code, the learning rates (A_{LTP} and A_{LTD}) are increased by a factor of ten to speed up the learning. The experiments for stable weight learning, to show a specific connectivity pattern, require the original homeostatic mechanism. Changes to the default parameters for this tasks are shown in Table 1.

Non-reproduced experiments

One experiment that is not reproduced is the experiment with ten excitatory and three inhibitory neurons, using stochastic Poisson input (Fig. 5 in [4]). In the original publication, they presented the emergence of a stable receptive fields and showed that the strength of synapses depends on the input firing rate. Here, we reproduce the emergence of receptive fields by presenting natural scenes and reproduce how the strength of the neuron activity influences the connectivity order.

The second experiment not reproduced is the experiment using the same network structure but with moving input patterns (Fig. 6 in [4]). Clopath et al. [4] demonstrated with this experiment that the strength of synapses can depend on the temporal order of emergent spikes and that the receptive field moves over the time, if the input is moving. We reproduce the synapse weight development, depending on the temporal order of spikes. The moving receptive fields are not reproduced here, but by reproducing receptive fields generally, we assume that moving receptive fields would emerge with the here proposed reimplementation.

Reimplementation

The reimplementation was done with Python 2.7 (Python $>$ 3.4 also works) and the neurosimulator ANNarchy [10] (version 4.6.6 or later). With ANNarchy, it is possible to implement neuronal and synaptic behavior by defining the corresponding mathematical equations in a text format, which are solved by ANNarchy using the desired numerical method. ANNarchy supports rate-based and spiking networks and provides a way to combine both kinds of neuronal networks. The network description is done in Python and code generation is used to produce optimized C++ code allowing a good parallel performance.

The implementation of the network for the voltage-clamp experiment, the pairing repetition task, the STDP learning window and the burst spiking experiments can be found in `net_fix.py`, where $\bar{\bar{u}} = u_{ref}$ as mentioned previously to switch off the homeostatic mechanism. For the connectivity experiments, the emergence of V1 simple-cell-like receptive fields and for the emergence of stable weights, the homeostatic mechanism dynamic as described in the original publication [4] is used. The network with a dynamic homeostatic mechanism can be found in `net_homeostatic.py`. The following explanation of the network is from the implementation in `net_homeostatic.py`.

Network implementation

To achieve a correct behavior of the learning rule, a correct implementation of the membrane potential behavior, specially after a spike, is necessary. The proposed reimplementation relies on the original source code written in Matlab. ANNarchy allows to define easily the equations governing neurons and synapses, but the internal execution order of the equations is fixed and different from the original implementation. In the original Matlab source code, there is a counter variable to implement the right behavior of the membrane potential as shown in the code passage below. The presented code passage is from the `aEIF.m` file, which is contained in the source code published on modeldb. After the neuron spikes, the counter is set to one.

In the next calculation step, the changes in the membrane voltage is set to 32.863 mV. One step later, the membrane potential is set to -49.5 mV.

```
if counter ==2
    u = E_L+15+6.0984;
    w = w+b;
    w_tail = w_jump;
    counter = 0;
    V_T = VT_jump+VT_rest;
end

% Updates of the variables for the aEIF
udot = 1/C*(-g_L*(u-E_L) + g_L*Delta_T*exp((u-V_T)/Delta_T)
       - w +w_tail+ I);
wdot = 1/tau_w*(a*(u-E_L) - w);
u= u + udot;
w = w + wdot;
w_tail = w_tail-w_tail/tau_wtail;
V_T = VT_rest/tau_VT+(1-1/tau_VT)*V_T;

if counter == 1
    counter = 2;
    u = 29.4+3.462;
    w = w-wdot;
end

if (u>th && counter ==0)
    u = 29.4;
    counter = 1;
end
```

The code below shows the definition of the neural equations in ANNarchy. As in the Matlab source code, we use a counter variable to control the behavior of the membrane potential for time steps after a spike, together with the ANNarchy own 'if' statement. With this variable, we can add the necessary 3.462 mV on the membrane potential one step after the spike, and set it to -49.5 mV after the second time.

```
neuron_eqs = """
dvm/dt  = if state>=2:+3.462 else:
          if state==1: -(vm + 49.5)+
          1/C*(Isp - (wad+b))+ g_Exc else:
          1/C * ( -gL * (vm - EL) + gL * DeltaT *
          exp((vm - VT) / DeltaT) - wad + z )+
          g_Exc: init = -70.6
dvmean/dt = (pos(vm - EL)**2 - vmean)/taumean    :init = 0.0
dumeanLTD/dt = (vm - umeanLTD)/tauLTD : init=-70.0
dumeanLTP/dt = (vm - umeanLTP)/tauLTP : init =-70.0
dxtrace /dt = (- xtrace )/taux
dwad/dt = if state ==2:0 else:
          if state==1:+b/tauw else:
          (a * (vm - EL) - wad)/tauw : init = 0.0
```

```
dz/dt = if state==1: -z+Isp-10 else:
        -z/tauz  : init = 0.0
dVT/dt = if state==1: +(VTMax - VT)-0.4 else:
         (VTrest - VT)/tauVT  : init=-50.4
dg_Exc/dt = -g_Exc/tau_gExc
state = if state > 0: state-1 else:0
Spike = 0.0"""
```

To implement the necessary equations, they are typed in one multi-string variable (`neuron_eqs`). The variable `vm` describes the membrane potential u, `vmean` the homeostatic variable $\bar{\bar{u}}$, `umeanLTD` and `umeanLTP` represent u_- and u_+, respectively. The variable `xtrace` describes $\bar{x}$, `wad` is w_{ad}, `z` is z, `g_Exc` is the input current and `Spike` is the spike counter (X). To implement the resolution trick, we use a extra discrete variable `state`. With that, we control the behavior of the different variables after a spike to reproduce the behavior of the variables as in the Matlab source file. The neuron spikes only if the membrane potential exceeds the threshold and if the `state` variable is equal to zero.

```
spkNeurV1 = Neuron( parameters = params,
                    equations=neuron_eqs,
                    spike="""(vm>VT) and (state==0)""",
                    reset="""vm = 29.0
                          state = 2.0
                          VT = VTMax
                          Spike = 1.0
                          xtrace+= 1/taux""")
```

To define a neuron model, ANNArchy provides the `Neuron` object, which expects a string object for the parameters (`parameters`), the equations that describes the neuronal behavior (`equations`), a string that define the conditions to release a spike (`spike`) and a string that defines the changes in the variables after a spike (`reset`).

```
equatSTDP = """
ltdTerm = if w>wMin : (aLTD*(post.vmean/urefsquare)*
          pre.Spike * pos(post.umeanLTD - thetaLTD)) else : 0.0
ltpTerm = if w<wMax : (aLTP * pos(post.vm - thetaLTP)*
        (pre.xtrace)* pos(post.umeanLTP - thetaLTD)) else : 0.0
deltaW = ( -ltdTerm + ltpTerm)
          dw/dt = deltaW :min=0.0"""
```

As for the neuron model, the equations for the STDP learning are defined by strings of the differential equations. `ltdTerm` describes the LTD term and `ltpTerm` the LTP term. Variables of the pre- or postsynaptic neuron, defined in the neuron model, can be addressed with the prefixes `pre.` and `post.`, respectively. With the `if w>wMin` statement in `ltdTerm`, the weight only decreases if the weight is above the lower bound. In `ltpTerm` a analogous term is implemented to avoid that weights exceed the upper bound. The parameters are defined in a string, analogous to the parameters of the neuron model. The parameter `urefsquare` is the homeostatic reference parameter u^2_{ref}. The learning rates A_{LTD} and A_{LTP} are defined by `aLTD` and `aLTP`, respectively. The threshold θ_- is defined by `thetaLTD` and θ_+ by `thetaLTP`. The parameter `transmit` is either zero or one, depending on whether the synaptic current for the experiment should transmit or not.

```
ffSyn = Synapse( parameters = parameterFF,
    equations= equatSTDP,
    pre_spike='''g_target += w*transmit''')
```

ANNarchy provides a `Synapse` object, that expects a `parameters` argument, the `equations` and a description of what happens when the pre-synaptic neuron spikes (`pre_spike`). After a pre-synaptic spike, the input current of the postsynaptic neurons increases by the value of the synaptic weight. `g_target` is an alias for the postsynaptic conductance that should be increased (`g_Exc` in the AdEx neurons).

Implementation of the Experiments

The implementation of the different experiments are in the corresponding python files. To perform an experiment, the network with the neuron populations and the weights between them must be initialized.

To create a population, ANNarchy provides the `Population` object. The `geometry` argument expects a tuple or a integer and defines the spatial geometry, respectively the number of neurons in the population. The `neuron` arguments expects a `Neuron` object. It defines the used model for population neurons. There is also a set of predefined `Population` objects in ANNarchy, for example the `PoissonPopulation`. This object provides a population of spiking neurons, whose spiking behavior follows a Poisson distribution with a given rate.

```
poisPop = PoissonPopulation(geometry=10, rates=100.0)
pop_Ten = Population(geometry=10, neuron=spkNeurV1, name="pop_Ten")
```

To connect two neuron populations and define the weight matrix between them, ANNarchy provides the `Projection` object. The `pre` argument defines the pre-synaptic population and the `post` argument the postsynaptic population. The `target` argument defines the target variable of the postsynaptic neuron, which is increased by the weight value after a pre-synaptic spike.

```
projInp_Ten = Projection(
    pre = poisPop,
    post= pop_Ten,
    target='Exc'
).connect_one_to_one(weights = 30.0)

projTen_Ten = Projection(
    pre= pop_Ten,
    post= pop_Ten,
    target= 'Exc',
```

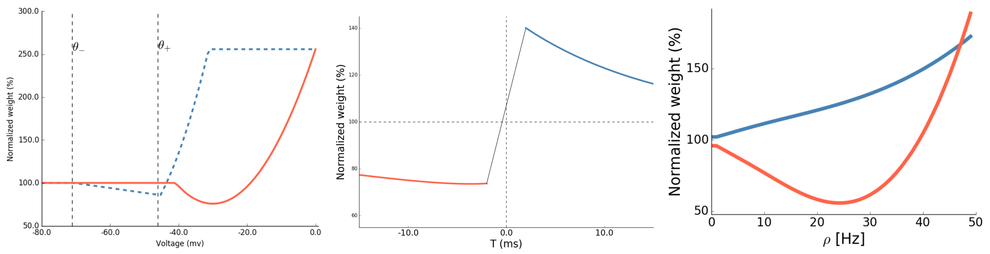

Figure 1: Replication of experimental findings. Left, weight change in the Voltage clamp experiment. The blue line presents the weight change with the parameter set for the visual cortex. The red line presents the weight change with the parameter set for the hippocampus. **Middle**, the classic STDP learning window. On the x-axis is the time of a postsynaptic spike in relation to the presynaptic spike presented. **Right**, weight changes as a function of pair frequency repetition. Pre-post pairs are the blue line and post-pre pairs the red line.

```
    synapse= ffSyn
).connect_all_to_all(weights = 0.1,allow_self_connections=True)
```

Results

To prove the correctness of the proposed reimplementation, different experiments of the original paper have been reproduced. Although most results are reproduced successfully, some experiments could not be absolutely reproduced or exhibit small differences. All weight changes are shown relatively to the initial weight values, which are not shown in the original publication. Because of that, initial values are mainly found experimentally and are shown in the corresponding tables of parameters.

Voltage-Clamp experiment

The results are shown in Fig. 1-left. The blue line represents the weight change with the standard parameter set for the visual cortex and the red line represents the weight change with the parameter set for the hippocampus, as mentioned in the original publication [4]. The both dotted lines mark the θ_- and θ_+ from the learning rule, with the standard data set. With the visual cortex data set (blue line), what is equal to the standard data set, the weight decreases slightly if the membrane potential exceeds the θ_- threshold and increases after it exceeds θ_+. With the hippocampus data set (red line) ($theta_- = -41.0mV$, $theta_+ = -38.0mV$, $A_{LTD} = 3.8 * 10^{-4}$,$A_{LTP} = 0.2 * 10^{-4}$), the weight decreases at a postsynaptic membrane voltage value of $-41.0mV$ and increases around $-20mV$. This matches with the results in the Clopath et al. [4] publication.

Pair-based and triplet STDP experiments

The classic pair-based spike timing learning window is presented in Fig. 1-middle. If the postsynaptic neuron spikes before the presynaptic one, LTD occurs (red line). If the postsynaptic neuron spikes after the presynaptic one, LTP occurs (blue line). The x-axis represents the time difference between pre- and postsynaptic spikes, relative to the postsynaptic spike. The resulting graph is similar to the one presented in the original publication. A slight difference can be seen in the higher positive and negative changes: in the original publication the normalized weight at the time gap -10 ms

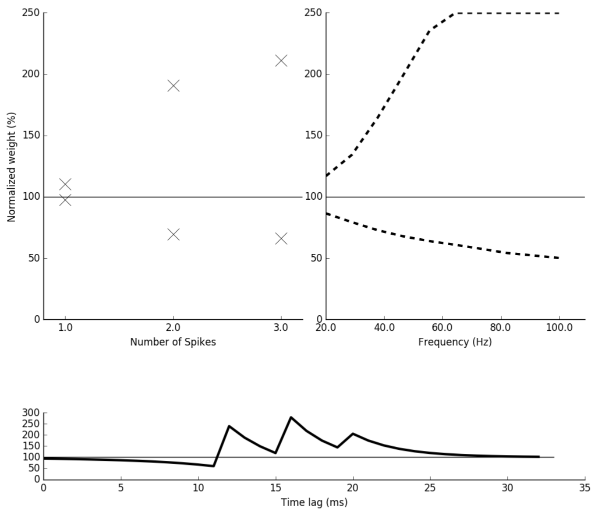

Figure 2: Burst experiments Upper left, weight change as a function of the numbers of postsynaptic spikes. **Upper right**, weight change as a function of the frequency between three postsynaptic spikes. **Down**, weight change as a function of the time between the first of three postsynaptic spikes and one presynaptic spike.

is around 70%, while it is around 80% in our simulation. This could be caused by a different internal processing of ANNarchy, but we coud not isolate the reason.

The analysis of the pairing repetition frequency task is shown in Fig. 1-right. With lower repetition frequencies, post-pre pairs (red line) lead to LTD. At a repetition frequency around 30 Hz, the post-pre pairs are under the influence of the next post-pre pair and the post-pre-post triplets lead to LTP. If the repetition frequency of post-pre pairs is around 50 Hz, the same amount of LTP emerges as in pre-post pairs. These results are similar to the original paper.

Spike bursts

In the original publication, Clopath et al. [4] studied the nonlinearity of STDP by using protocols of spike bursts. The result from the reimplementation of the first spiking burst experiment is shown in Fig. 2-upper-left. The upper marks represent the weight change with +10 ms, the lower marks the weight change with −10 ms between the presynaptic and the first postsynaptic spikes. As in the original publication from Clopath et al. [4] and the experimental paper of Nevian and Sakmann [5], one postsynaptic spike, independently of the spiking order, leads only to a small change in the weight. A second spike leads to bigger change, especially when the postsynaptic neurons spikes after the presynaptic one.

The second task is the weight change, depending on the frequency between three postsynaptic spikes (Fig. 2-upper-right). As in the previous experiment, the upper line represents the weight changes when the first postsynaptic spike appears 10 ms after the presynaptic spike; the lower line when the first postsynaptic spike appears 10 ms before the presynaptic one. As shown in Clopath et al. [4], a higher frequency leads to a higher change of the synaptic weight.

The third task, the weight change as a function of the time between the one presynaptic spike and the first of three postsynaptic spikes, is presented in Fig. 2-down. The curve is also very similar to the one presented in Clopath et al. [4].

The label on the y-axes, as described in the original publication [4], suggested the weight value in percentage. But, as mentioned in the original experimental paper by Nevian and Sakmann [5], the graphs should actually show the weight changes in percentage, relative to the initial weight value.

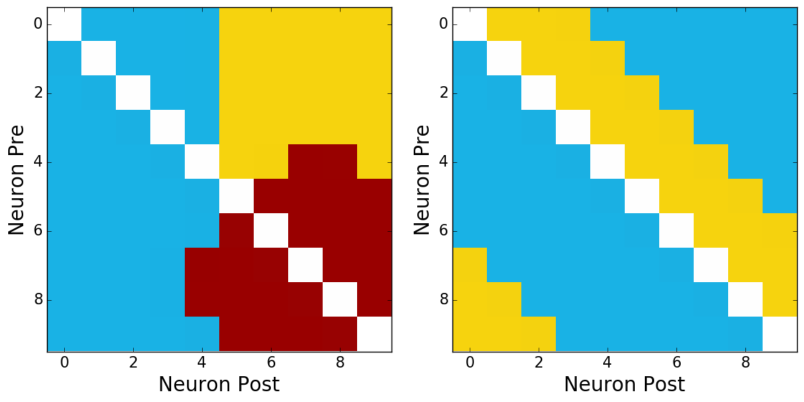

Figure 3: Different connectivity patterns. Depending on the spiking activity, different connectivity patterns emerge between the neurons. The color scheme is similar to that in the original publication. Weak connections are blue, strong unidirectional connections are yellow, and strong bidirectional connections are red. **Left**, Neurons with similarly high firing rates develop strong bidirectional connections. **Right**, Connection patterns follows the temporal order of the occurred spikes. The y-axis shows the index of the presynaptic neuron, the x-axis the index of the postsynaptic neurons

Connectivity analysis

In addition to the replication of experimental findings of pair-based and triplet STDP experiments, Clopath et al. [4] presented how the synaptic connectivity (emerging from the proposed learning rule) is influenced by the spiking behavior of the neurons. Fig. 3-left shows the obtained connectivity structure if neurons fire at different frequencies. The color scheme is similar to the original publication: weak connections (above $\frac{3}{4}$ of the maximum activity) are blue, strongly unidirectional connections are yellow while strong bidirectional connections are red. Neurons with similarly high firing rates develop strong bidirectional connections, because they are often active at the same time. This suggests that learning is based on the correlation between the neuronal activities in a Hebbian manner. Weak connections are assigned to neurons with a low firing rate below 5 Hz. If the postsynaptic neuron is firing with a rate above 5 Hz, strong unidirectional weights emerge. This is in line with the connection pattern presented in the original paper (Fig. 4 in [4]).

If the neurons fire in a specific temporal order, this sequence is reflected in the connection pattern (Fig. 3-left). As in the original paper, the connections from neurons which are firing a long time after or before the postsynaptic neuron are weak, while they are strong to neurons which fired a short time after the neurons.

Receptive fields

In the previously published Matlab source code, the emergence of stable weights was demonstrated using a one dimensional input with 500 input values. At every time step, a subset of close neurons are active. The emergent stable weights are shown in Fig. 4-right. After 500 epochs, a spatially related subset of weights increased to the maximum and the other weight values decrease down to zero. This leads to a specific selectivity for the postsynaptic neuron.

The emergence of a cluster of strong synaptic weights can be interpreted as the formation of a receptive field, which defines the selectivity of the neuron, similar to the receptive fields in the primary visual cortex (V1). To reproduce that, one postsynaptic neuron receives input from 512 presynaptic neurons, as described in the original publication. Every presynaptic neuron corresponds to one pixel of the 16 $\times$ 16 pixel input, divided in an ON-part for the positive and an OFF-part for the negative values

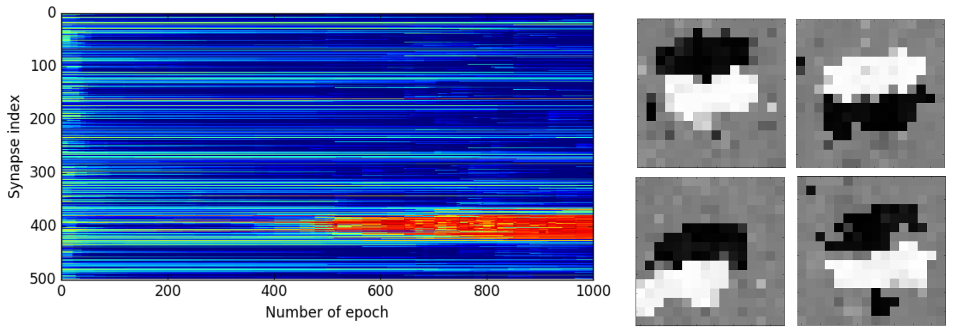

Figure 4: Stable weights on Poisson distributed inputs. Left Colors show the weight value at the end of the current epoch from the presynaptic neuron to a single postsynaptic neuron. Weight values around zero are blue and weight values around the maximum weight value of 3 are red. The y-axis denotes the presynaptic index indicated and the x-axis shows the number of epochs. **Right** Four different V1 simple cells like receptive fields, generated by four simulation runs.

($16 \times 16 \times 2 = 512$). The 200000 presented patches are randomly cut out of ten natural scenes [7], and each patch is presented for 200 ms. The emergent receptive fields are shown in Fig. 4-left.

Conclusion

Our reimplementation of voltage based STDP learning rule from Clopath et al. [4] is able to reproduce the experimental data and the emergence connectivity structures proposed in the original paper as well as the emergent of orientation selective receptive fields, like those in the primary visual cortex (V1). In comparison to the graphs in the original publication, our reimplementation shows little differences in the curve shapes in the STDP window (Fig. 1-middle), the weight change as a function of the pair-repetition frequency (Fig. 1-left) and for the weight change as a function of the burst frequency of the postsynaptic neuron (Fig. 2-upper-right). However, the curves show the same tendency as in the original publication.

The description of the learning rule in the original publication comprises enough details to understand the different components and their interaction. However, two main components of the learning rule have not been described adequately to allow a direct reimplementation: the 'resolution trick' of the membrane potential after spike emission, and the equation for the homeostatic mechanism ($\bar{\bar{u}}$). The emergent of stable weights with high and low values depends on a functional homeostatic mechanism, as mentioned in the original publication [4]. But the formula to calculate the homeostatic variable $\bar{\bar{u}}$ is not described in the publication. The dependency of $\bar{\bar{u}}$ on the homeostatic mechanism is necessary to implement the right behavior of the membrane potential. Because of this, the reimplementation has greatly benefited from the release of the source code on modelDB, where the correct behavior of the neuron and the homeostatic mechanism is written. Furthermore, initial weight values are not given in the original publication, this complicates the reproduction of the experiments. Initial weights for the experiments can be found in the corresponding python files. Please note that the weights can change from task to task to achieve the proportional values for the figures.

ANNarchy as a framework makes it easy to define a neuron model and learning rule to set up a network. Despite of this advance, the calculation order of the equations is controlled by ANNarchy and can lead to a slightly different behavior and little differences in the results. Furthermore, the change of some parameters for most tasks was necessary. But this seems to be a regular problem in reimplementing models [3].

Acknowledgment

This work was supported by the European Social Fund (ESF) and the Freistaat Sachsen.

References

[1] Alain Artola, Susanne Bröcher, and Wolf Singer. "Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex". In: *Nature* 347 (1990), pp. 69–72. DOI: 10.1038/347069a0. URL: http://dx.doi.org/10.1038/347069a0.

[2] Guo-Qiang Bi and Mu-Ming Poo. "Synaptic Modifications in Cultured Hippocampal Neurons: Dependence on Spike Timing, Synaptic Strength, and Postsynaptic Cell Type". In: *Journal of Neuroscience* 18.24 (1998), pp. 10464–10472. ISSN: 0270-6474. eprint: http://www.jneurosci.org/content/18/24/10464.full.pdf. URL: http://www.jneurosci.org/content/18/24/10464.

[3] Romain Brette et al. "Simulation of networks of spiking neurons: A review of tools and strategies". In: *Journal of Computational Neuroscience* 23.3 (Dec. 2007), pp. 349–398. ISSN: 1573-6873. DOI: 10.1007/s10827-007-0038-6. URL: https://doi.org/10.1007/s10827-007-0038-6.

[4] Claudia Clopath et al. "Connectivity reflects coding: a model of voltage-based STDP with homeostasis". In: *Nature Neuroscience* 13.3 (2010), pp. 344–352. ISSN: 1097-6256. DOI: 10.1038/nn.2479. URL: http://www.nature.com/doifinder/10.1038/nn.2479.

[5] Thomas Nevian and Bert Sakmann. "Spine Ca2+ Signaling in Spike-Timing-Dependent Plasticity". In: *Journal of Neuroscience* 26.43 (2006), pp. 11001–11013. DOI: 10.1523/JNEUROSCI.1749-06.2006.

[6] Anaclet Ngezahayo, Melitta Schachner, and Alain Artola. "Synaptic Activity Modulates the Induction of Bidirectional Synaptic Changes in Adult Mouse Hippocampus". In: *Journal of Neuroscience* 20.7 (2000), pp. 2451–2458. ISSN: 0270-6474. DOI: 10.1523/JNEUROSCI.20-07-02451.2000. eprint: http://www.jneurosci.org/content/20/7/2451.full.pdf. URL: http://www.jneurosci.org/content/20/7/2451.

[7] Bruno A. Olshausen and David J. Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6 (1996), pp. 607–609. DOI: 10.1038/381607a0.

[8] Jean-Pascal Pfister and Wulfram Gerstner. "Beyond Pair-Based STDP: a Phenomenological Rule for Spike Triplet and Frequency Effects". In: (2006). Ed. by Y. Weiss, B. Schölkopf, and J. C. Platt, pp. 1081–1088. URL: http://papers.nips.cc/paper/2923-beyond-pair-based-stdp-a-phenomenological-rule-for-spike-triplet-and-frequency-effects.pdf.

[9] Per Jesper Sjöström, Gina G Turrigiano, and Sacha B Nelson. "Rate, Timing, and Cooperativity Jointly Determine Cortical Synaptic Plasticity". In: *Neuron* 32.6 (2001), pp. 1149–1164. ISSN: 0896-6273. DOI: https://doi.org/10.1016/S0896-6273(01)00542-6. URL: http://www.sciencedirect.com/science/article/pii/S0896627301005426.

[10] Julien Vitay, Helge Dinkelbach, and Fred Hamker. "ANNarchy: a code generation approach to neural simulations on parallel hardware". In: *Frontiers in Neuroinformatics* 9 (2015), p. 19. ISSN: 1662-5196. DOI: 10.3389/fninf.2015.00019. URL: https://www.frontiersin.org/article/10.3389/fninf.2015.00019.