ReScience

[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, 2018
Accepted Feb, 1, 2018
Published Feb, 1, 2018

Competing Interests:
The authors have declared that no competing interests exist.

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 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 finding from spike pair repetition and triplet experiments [9], as from 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 V1 simple cells. Therefore, they implemented a homeostatic mechanism to control the amount 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 rate-coded 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 (v2.7) with the help of the neuro-simulator ANNarchy [10], numpy (v1.11.0)

and matplotlib (v1.5.1). 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 (see **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, the leak conductance is g_L and E_L is 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 $2ms$ after a spike is emitted, they used the so called 'resolution trick'. For this, they simulated once the complete change in the membrane potential through a spike with high precision and integrated. For simulations, they used the integrated number and clamped the membrane potential for $2ms$.
This means, the membrane potential is set to $29.4mV$ after a spike, one millisecond later to $29.4mV + 3.462mV$ and another millisecond later to $E_L + 15mV + 6.0984mV$.

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

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

The hyperpolarization current described by w_{ad} (see **Eq.** 3). After a spike, w_{ad} is increased by the amount b and decays down 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 the neuron spiked and will decay to $V_{T_{rest}}$ (see **Eq.** 4).

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

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

Synaptic model

The proposed learning rule consists of two terms: a long term potentiation (LTP) term (**Eq.** 5) controls increases in synaptic efficiency:

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

Therefore, A_{LTP} is the learning rate for the LTP term. The parameters θ_+ and θ_- are plasticity thresholds for the membrane potential (u), respectively for the over time averaged version ($\bar{u}_+$ (see **Eq.** 6)). The original paper they did not mentioned the meaning of both thresholds. With $\theta_+ = -45.3mV$ it is above the spiking threshold. This suggest, that the threshold avoid the occurrence of LTP if the postsynaptic neuron

not spiked already. With $\theta_+ = E_L$, LTP only occurs if the membrane potential is above the resting potential.

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

On every spike of the presynaptic neuron, the spike train $\bar{x}_i$ is increased by one and will decay over time with a time constant τ_x (see **Eq.** 7). If the presynaptic neuron spikes to a time point t, the spike counter X_i is 1, 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 when the postsynaptic neuron spikes shortly after the presynaptic neuron or if the membrane potential is high long enough, so that $\bar{u}_-$ exceeds θ_-.

The second term is the long term depression (LTD) term (**Eq.** 8) and governs decreases in synaptic efficiency:

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

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

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

Therefore, the homeostatic variable $\bar{\bar{u}}$ is computed over the quadratic difference between the postsynaptic membrane potential and the resting potential (E_L). With a higher activity, $\bar{\bar{u}}$ increases, leading to a higher amount of LTD and a weight decrease. In contrast, a lower activity decreases the amount 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 the positive LTP and the negative LTD terms (**Eq.** 10):

$$\frac{dw}{dt} = -LTD_{Term} + LTP_{Term} \tag{10}$$

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 **Tab.** 2 and **Tab.** 1.

Reproduction of experiments

In the original publication, the authors reproduce spike timing triplet experiments in the visual cortex of rats [9]. Furthermore, they investigate the emerged structure of the connectivity 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 emergent of receptive fields by presenting natural scenes (**Fig.** 7 in [4]). In the available Matlab source code, they investigate the stable learning of weights using 500 input neurons and one postsynaptic neuron. The firing rate of these neurons follow a Gaussian distribution and the spike timing a Poisson process. This task is reimplemented as well. With this analysis, the functionality of the reimplementation is shown on the main feature of this learning rule.

One experiment what is not reproduce, is the experiment with ten excitatory and three inhibitory and stochastic Poisson input (**Fig.** 5 in [4]). In the original publication, they presented the emergent of a stable receptive fields and that the strength of synapses depends on the input firing rate. Here, both features are presented with two different tasks. The second experiments what is not reproduce, is the experiment with 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. The synapse weight development is reproduced by another task. The moving receptive field are not reproduced here, but by reproducing of receptive fields generally we assume that moving receptive fields would be emerge with the here proposed reimplementation.

The experiment protocols are based on the description on the publication of Clopath et al. [4]. The implementation of the learning rule was mainly made 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. Despite ANNarchy make it easy to write down the model equations to build up networks, the processing order of the equations is strictly given by ANNarchy [10]. Because of this, the execution order for differential equations and non-differential equations, or for equations of the synapses or neurons are different from the order mentioned in the published Matlab source code. Further, the changes are calculated separately for the neurons and synapses variables. After that,changes are added and then it will be checked if a postsynaptic event appeared. At the end of every simulation step is the recording happening [10].

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

To reproduce the STDP learning window (see **Fig.** 1 left), we create a list of discrete time points where the pre- or postsynaptic neurons should emit spikes. The presynaptic neuron spikes every $50ms$. The postsynaptic neurons spikes in a range from $1ms$ to $15ms$ before or after the presynaptic neuron. For the repetition frequency experiment or the triplet experiment (see **Fig.** 1 right), the number of pre- and postsynaptic spike pairs increases from a pair frequency of $0.1Hz$ to $50Hz$. The time between a pre- and postsynaptic spike of a pair is $10ms$. To reproduce this experiments, it was necessary to set $\bar{u}$ to a fixed value as mentioned in the original publication [4]. The parameter changes are shown in **Tab.** 2.

To analyze the connectivity depending the number of spikes, a small network with ten neurons connected with each other is build. Every neuron receives input from one additional neuron, with Poisson-distributed spike patterns. The firing rate of each

Poisson neuron is increased from 2Hz to 20Hz, influencing the firing rate of the 10 corresponding neurons in the network.

The reimplementation of the model is mainly based on the Matlab source code from modelDB. Besides of experiments in 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 neuron. For every trial (125ms) 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 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 **Tab.** 1.

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}	$0.8 * 10^{-3}$

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	$\bar{\bar{u}}$	$120mV^2$

Reimplementation

The reimplementation was done with Python 2.7 and the neural-simulator ANNarchy [10] (v.4.6.4). With ANNarchy, it is possible to implement the description of the learning and the neuronal behavior by define the mathematical equations and it cares about the temporal execution of the equations. Therefore, variables are mostly written as first-order ordinary differential equations, which are solved by ANNArchy. This supports the implementation of more complex models and bigger neuronal networks. Therefore, ANNarchy supports rate based and spiking learning rules, and it provides a way to combine both kinds of neuronal networks. The definition of learning rules, the neurons and the network structure is done in the easy understanding Python language. To archive a good performance on the execution of the model, the Python code is compiled in C++. With that, ANNarchy make it easy to implement the a neuronal network model and show good performances [10].

As mentioned in the original publication, to reproduce the voltage-clamp experiment, the pairing repetition task and the STDP learning window we set $\bar{baru} = u_{ref}$. The implementation of the network for this tasks can be found in **net_fix.py**. For the connectivity experiments and for the emergent of stable weights, the homeostatic mechanism dynamic as described in the original publication [4]. The network with a dynamic homeostatic mechanism can be found in **net_homeostatic.py**. The following explanation of the network implementation is from the **net_homeostatic.py**.

Network implementation

To achieve a correct behavior of the learning rule, the correct implementation of the membrane potential behavior, specially after a spike, is necessary. Therefore, the here presented reimplementation orientates on the original source code written in Matlab. The usage of ANNarchy make it easy to define the behavior of synapses and neurons to create larger networks. But the internal execution order of the equations are under the control of ANNarchy and make it difficult to ensure an equal execution order. In the original Matlab source code exist 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 $32.863mV$. One step later, the membrane potential is set to $-49.5mV$ Additionally the differential equation for one time step. Please note, $dt = 1ms$ in the Matlab source code.

```
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 neuron model 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 that we can add the necessary 3.462 on the membrane potential one step after the spike, and set to $-49.5mV$ after the second time step with the additionally changes.

```
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 string variable ('neuron_eqs'). The variable 'vm' describes the membrane potential u, 'vmean' the homeostatic variable $\bar{baru}$, 'umeanLTD' and 'umeanLTP' are the equations for u_-, respectively u_+. 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 spikes to recreate 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)""",
                    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, what 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'). With that, we reproduce the behavior of the neuron as described in the published Matlab source code.

```
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 spiking learning are defined by strings of the differential equations. The 'ltdTerm' describes the LTD_{Term} and the 'ltpTerm' the LTP_{Term}. Variables of the pre- or post-synaptic neuron, they are define in the neuron model, can be addressed with the prefix 'pre.', respectively 'post.'. With the 'if w>wMin' statement in the 'ltdTerm', the weight only decreases if the weight is above the lower bound. In the 'ltpTerm' a analogous term is implement to avoid, that weights exceeds the upper bound. The parameters are defined in a string, analogous to the parameters of the neuron model. Therefore, the parameter 'urefsquare' is the homeostatic reference parameter u_{ref}^2. The learning rates A_{LTD} and A_{LTP} are 'aLTD', respectively 'aLTP'. And the threshold θ_- is defined by 'thetaLTD' and θ_+ by 'thetaLTP'. The parameter 'transmit' is zero or one, dependent if 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, what expects a 'parameters' argument, the 'equations' and a description if the pre-synaptic neuron spikes ('pre_spike'). After a pre-synaptic spike, the input current of the post-synaptic neurons increases by the value of the synaptic weight. Therefore, 'g_target' is the target variable on the post-synaptic side what should be increased. The target variable is defined in the **Projection** object (see below). Additionally, a description for a post-synaptic spike is possible.

Implementation of the Experiments

The implementation of the different experiments are in the current 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. The population can be give a unique name, optionally. Besides that, it exist a set of predefined **Population** objects in ANNarchy, for example the **PoissonPopulation**. This object provides a population of spiking neurons, which spiking behavior follows a Poisson distribution. As for the **Population** object with the 'geometry' argument is the size or the spatial geometry of the population. The argument 'rates' defines the mean firing rate of the population neurons. A name can be given, optionally.

```
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 define the pre-synaptic population and the 'post' argument the post-synaptic population. Both arguments expects a **Population** object. The 'target' argument defines the target variable of the post-synaptic 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',
    synapse= ffSyn
).connect_all_to_all(weights = 0.1,allow_self_connections=True)
```

A further description of the experiment implementations can be found in the corresponding python files.

Recording variables

With the **Monitor** object provides ANNarchy a easy possibility to record variables from projections and populations.

```
    dendrite = projV1_V1.dendrite(0)
    m_d = Monitor(dendrite, ['w','deltaW','ltdTerm','ltpTerm'])
```

Results

To prove the correctness of the here proposed reimplementation, different experiments of the original paper are reproduced. Although most results are reproduced successfully, the some experiments could not be absolutely reproduced or show small differences. All weight changes are shown relatively to the initial weight value, 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

To reproduce the voltage clamp experiment, the postsynaptic membrane potential changes from a fix values of $-80mV$ to $0mV$. Resulting changes in the learning rule

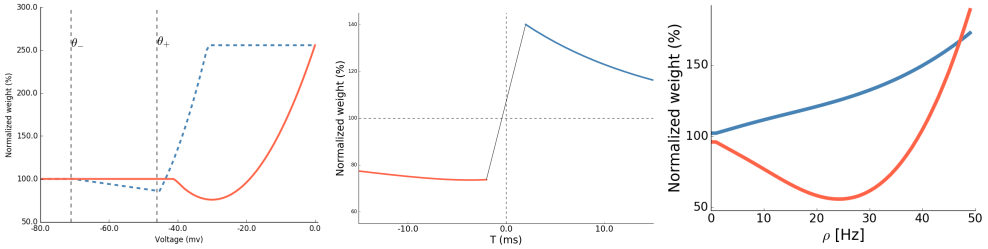

Figure 1: Replication of experimental findings. **Left**, weight change in the Voltage clamp experiment. The blue line presents the weight change with the parameter set for the visual cortex. The red line presents the weight change with the parameter set for the hippocampus. **Middle**, the classic STDP 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.

are implemented as mentioned in the original publication. Further, the presynaptic neuron spikes with a firing rate of $25Hz$ for $50ms$. Further details can be found in the **ann_clamp.py** file. The results are shwon 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.

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 presented one in the original publication. A small difference can be seen in the higher positive and negative change. In the original publication is the normalized weight at a time difference of $-10ms$ around 70%. In our result, the weight is around 80%. This could be caused by a different internal processing of ANNarchy, as mentioned above.

The analysis of the pairing repetition frequency task is shown on **Fig. 1 right**. With lower repetition frequency, post-pre pairs (red line) lead to LTD. At a repetition frequency around $30Hz$, 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 $50Hz$, 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] study nonlinearity of STDP by using protocols of spike bursts. In the first task, they changed the number of postsynaptic spikes from one up to three, with $+10ms$ and with $-10ms$ between the presynaptic and the first postsynaptic spike. The postsynaptic neuron fires with $50Hz$. The results from the reimplementation is shown in **Fig. 2 upper left**. The upper marks represents for the weight change with $+10ms$, the lower marks the weight change with $-10ms$ between the presynaptic and 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

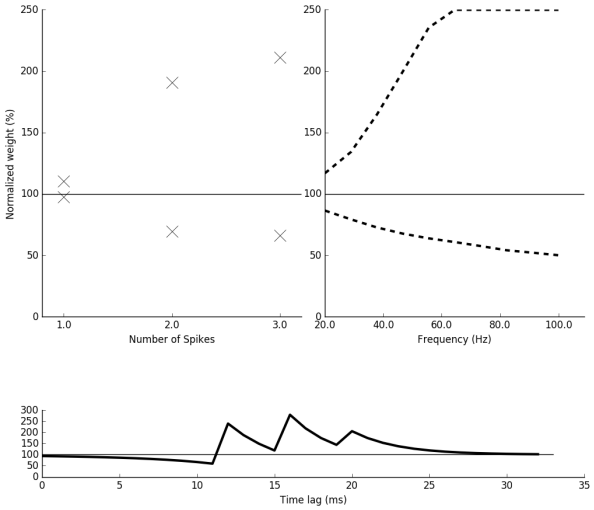

Figure 2: Burst experiments Upper left, weight change as a function of the numbers of postsynaptic spikes. **Upper right**, weight change as a function of the frequency between three postsynaptic spikes. **Down**, weight change as a function of the time between the first of three postsynaptic spikes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

postsynaptic spikes (**Fig. 2 upper right**). As shown in Clopath et al. [4], a higher frequency leads to a higher change of the synaptic weight.

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

Despite of the label, what is orientated on the publication from Clopath et al. [4], the graphs show the weight changes as mentioned in the original experimental paper by Nevian and Sakmann [5].

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 connection structure if neurons fire with different frequencies. Here, the color scheme is similar to the original publication. Weak connections (above $\frac{3}{4}$ of the maximum activity) are blue. Yellow represents strongly unidirectional connections while red represents strong bidirectional connections. 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 emerged to neurons with a low firing under $5Hz$ rate. If the postsynaptic neuron firing with a rate above $5Hz$, strong unidirectional weights emerge. This is in line with the connection pattern presented in the original paper (see **Fig. 4** in [4]).

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

Receptive fields

In the published Matlab source code, they demonstrated the emergent of stable weights. Therefore, they presented a one dimensional input with 500 input values. At every

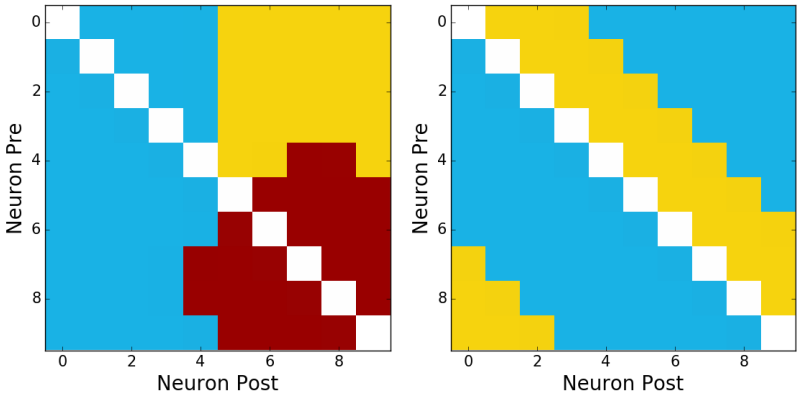

Figure 3: Different connectivity patterns. Depending on the spiking activity, different connectivity patterns emerge between the neurons. The color scheme is similar to them in the original publication. Weak connections are blue, strong unidirectional connections are yellow and red are strong bidirectional connections. **Left**, Neurons with similar high firing rates develop strong bidirectional connections. **Right**, connection pattern follows the temporal order of the occurred spikes.

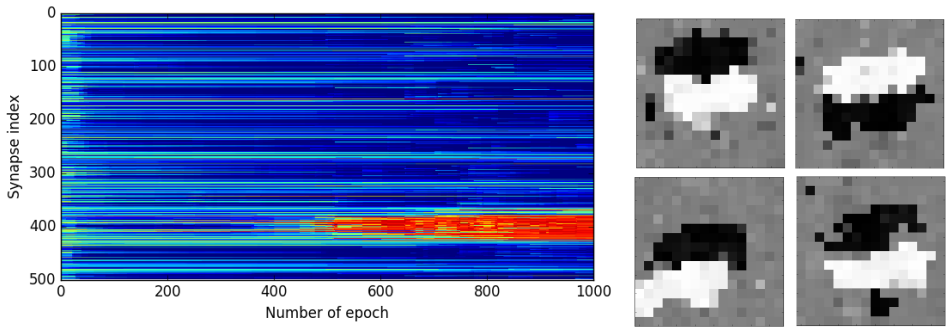

Figure 4: Stable weights on Poisson distributed Input. Right Colors show the weight value at the end of the current epoch from the presynaptic neuron to a single postsynaptic neuron. Blue are weight values around zero and red are weight values around the maximum weight value of 3. On the y-Axis is the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

time step, a subset of near to each other lying neurons are active. The emergent stable weights are shown in **Fig. 4, right**. After 500 epochs, a spatial 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 emergent of a cluster of strong synaptic weights can be interpreted as a receptive fields, what defines the selectivity of the neuron. Therefore, Clopath et al. [4] demonstrated the appearance of receptive fields, similar to them of 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 16×16 pixel input, divided in an ON-part for the positive and and OFF-part for the negative values. The 200000 presented patches are cut out randomly of ten natural scenes [7], and each patch presented for $200ms$. The emergent receptive fields are shown in **Fig. 4, left**.

Was war für die Implementierung des Modelles in ANNArchy wichtig ? Was für Probleme gab es ? Welche Schritte waren notewendig um es zu implementieren? Was für die implementierung relevantes stand im Paper und was musste selbst 'entschlüsselt' werden?

Conclusion

Our reimplementation of voltage based STDP learning rule from Clopath et al. [4] is able to reproduce most of the experimental data and the emergent connectivity structures proposed in the original paper as well as the emergent of orientation selective receptive fields, like them in the primary visual cortex. In comparison to the graphs in the original publication, the here presented reimplementation shows some 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. Further, initial weight values are not given in the original publication, this complicates the reproduction of the experiments. Initial weight are here given in the corresponding tables. Please note, the weights can change from task to task to achieve the proportional values for the figures.

ANNarchy as a framework make 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 to little differences in the results. Further, the change of some parameters for the most tasks was necessary. But this seems to be a regular problem by reimplementing models [3].

Acknowledgment

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

References

[1] A. Artola, S. Bröcher, and W. 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.