Learning in cortical microcircuits with multi-compartment pyramidal neurons

Johannes Gille

Thesis notes
January 26, 2023

M.Sc. Kognitive und integrative Systemneurowissenschaften

Philipps-Universität Marburg

Supervisors:

Prof. Dr. Dominik Endres, Philipps-Universität Marburg

Dr. Johan Kwisthout, Radboud University

0.1   utilities

One-sided exponential decay kernel

$$\kappa(t) = H(t)e^{-t/\tau_\kappa} \tag{1}$$

$$H(t) = \begin{cases} 1 & \text{if } t > 0 \\ 0 & \text{if } t \leq 0 \end{cases} \tag{2}$$

Antiderivatives:

$$\int_{-\infty}^{x} H(t)dt = tH(t) = max(0,t) \tag{3}$$

Convolution:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau$$

For functions f, g supported on only $[0,\infty]$ (as one-sided decay kernels and spiketrains are), integration limits can be truncated:

$$(f * g)(t) = \int_{0}^{t} f(\tau)g(t-\tau)d\tau$$

Plasticity:

$$\frac{dW_{ij}}{dt}(t) = F(W_{ij}(t), s_i^*(t), s_j^*(t), V_i^*(t)) \tag{4}$$

$$F[s_j^*, V_i^*] = \eta\kappa * (V_i^* s_j^*) \tag{5}$$

$$\text{with } V_i^* = (s_i - \phi(V_i))h(V_i), \tag{6}$$

$$s_j^* = \kappa_s * s_j. \tag{7}$$

For an event-based plasticity we need:

$$\Delta W_{ij}(t,T) = \int_{t}^{T} dt' F[s_j^*, V_i^*](t') \tag{8}$$

$$= \int_{t}^{T} dt' \eta\kappa * (V_i^* s_j^*) \tag{9}$$

$$= \eta \int_{t}^{T} dt' \int_{0}^{t'} dt'' \; \kappa(t'-t'')V_i^*(t'')s_j^*(t'') \tag{10}$$

$$= \eta \int_{0}^{t} dt' \int_{t''}^{t'} dt'' \; \kappa(t'-t'')V_i^*(t'')s_j^*(t'') \tag{11}$$

$$\tag{12}$$

Starting with the complete Integral from $t = 0$.

$$\Delta W_{ij}(0,t) = \eta \int_0^t dt' \int_0^{t'} dt'' \; \kappa(t' - t'')V_i^*(t'')s_j^*(t'')$$

$$= \eta \int_0^t dt'' \int_{t''}^t dt' \; \kappa(t' - t'')V_i^*(t'')s_j^*(t'')$$

$$= \eta \int_0^t dt'' \; [\tilde{\kappa}(t - t'') - \tilde{\kappa}(0)] \, V_i^*(t'')s_j^*(t'')$$

With $\tilde{\kappa}$ being the antiderivative of κ:

$$\kappa(t) = \frac{\delta}{\delta t}\tilde{\kappa}(t)$$

$$\tilde{\kappa}(t) = -e^{-\frac{t}{t_\kappa}}$$

The above can be split up into two separate integrals:

$$\Delta W_{ij}(0,t) = \eta \left[-I_2(0,t) + I_1(0,t)\right]$$

$$I_1(t_1, t_2) = -\int_{t_1}^{t_2} dt' \; \tilde{\kappa}(0)V_i^*(t')s_j^*(t')$$

$$I_2(t_1, t_2) = -\int_{t_1}^{t_2} dt' \; \tilde{\kappa}(t_2 - t')V_i^*(t')s_j^*(t')$$

Which implies the identities

$$I_1(t_1, t_2 + \Delta t) = I_1(t_1, t_2) + I_1(t_2, t_2 + \Delta t)$$

$$I_2(t_1, t_2 + \Delta t) = e^{-\frac{t_2 - t_1}{\tau_\kappa}} I_2(t_1, t_2) + I_2(t_2, t_2 + \Delta t)$$

$$I_2(t_1, t_2 + \Delta t) = -\int_{t_1}^{t_2 + \Delta t} dt' \; \tilde{\kappa}(t_2 + \Delta t - t')V_i^*(t')s_j^*(t') \tag{13}$$

$$= -\int_{t_1}^{t_2} dt' \left[-e^{-\frac{t_2 + \Delta t - t'}{\tau_\kappa}}\right] V_i^*(t')s_j^*(t') - \int_{t_2}^{t_2 + \Delta t} dt' \left[-e^{-\frac{t_2 + \Delta t - t'}{\tau_\kappa}}\right] V_i^*(t')s_j^*(t') \tag{14}$$

$$= -e^{-\frac{\Delta t}{\tau_\kappa}} \int_{t_1}^{t_2} dt' \left[-e^{-\frac{t_2 - t'}{\tau_\kappa}}\right] V_i^*(t')s_j^*(t') - \int_{t_2}^{t_2 + \Delta t} dt' \left[-e^{-\frac{t_2 + \Delta t - t'}{\tau_\kappa}}\right] V_i^*(t')s_j^*(t') \tag{15}$$

Using this we can rewrite the weight change from t to T as:

$$\Delta W_{ij}(t,T) = \Delta W_{ij}(0,T) - \Delta W_{ij}(0,t)$$

$$= \eta[-I_2(0,T) + I_1(0,T) + I_2(0,t) - I_1(0,t)]$$

$$= \eta[I_1(t,T) - I_2(t,T) + I_2(0,t)\left(1 - e^{-\frac{T-t}{\tau_\kappa}}\right)]$$

The simplified **?** case would be:

$$\frac{dW_{ij}}{dt} = \eta(\phi(u_i) - \phi(\hat{v}_i))\phi(u_j)$$

$$\Delta W_{ij}(t,T) = \int_t^T dt' \, \eta \, (\phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}})) \, \phi(u_j^{t'})$$

$$\Delta W_{ij}(t,T) = \eta \int_t^T dt' \, (\phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}})) \, \phi(u_j^{t'})$$

$$V_i^* = \phi(u_i^{t'}) - \phi(\widehat{v_i^{t'}})$$

$$s_j^* = \kappa_s * s_j$$

Where s_i is the postsynaptic spiketrain and V_i^* is the error between dendritic prediction and somatic rate and $h(u)$. The additional nonlinearity $h(u) = \frac{d}{du} ln \, \phi(u)$ is ommited in our model **TODO: should it though?** .

$$\tau_l = \frac{C_m}{g_L} = 10 \tag{16}$$

$$\tau_s = 3 \tag{17}$$

Writing membrane potential to history (happens at every update step of the postsynaptic neuron:

```
UrbanczikArchivingNode< urbanczik_parameters >::write_urbanczik_history(
    Time t, double V_W, int n_spikes, int comp)
{
  double V_W_star = ( ( E_L * g_L + V_W * g_D ) / ( g_D + g_L ) );
  double dPI = ( n_spikes - phi( V_W_star ) * Time::get_resolution().get_ms
    () )
      * h( V_W_star );
}
```

I interpret this as:

$$\int_{t_{ls}}^T dt' \, V_i^* = \int_{t_{ls}}^T dt' \, (s_i - \phi(V_i))h(V_i),$$

$$\int_{t_{ls}}^T dt' \, V_i^* = \sum_{t=t_{ls}}^T (s_i(t) - \phi(V_i^t)\Delta t)h(V_i^t)$$

```
for (t = t_ls; t< T; t = t + delta_t)
{
    minus_delta_t = t_ls - t;
    minus_t_down = t - T;
    PI = ( kappa_l * exp( minus_delta_t / tau_L ) - kappa_s * exp(
    minus_delta_t / tau_s ) ) * V_star(t);
    PI_integral_ += PI;
    dPI_exp_integral += exp( minus_t_down / tau_Delta_ ) * PI;
}
// I_2 (t,T) = I_2(0,t) * exp(-(T-t)/tau) + I_2(t,T)
PI_exp_integral_ = (exp((t_ls-T)/tau_Delta_) * PI_exp_integral_ +
    dPI_exp_integral);
```

```
11 W_ji = PI_integral_ - PI_exp_integral_;
12 W_ji = init_weight_ + W_ji * 15.0 * C_m * tau_s * eta_ / ( g_L * ( tau_L -
      tau_s ) );
13
14 kappa_l = kappa_l * exp((t_ls - T)/tau_L) + 1.0;
15 kappa_s = kappa_s * exp((t_ls - T)/tau_s) + 1.0;
16
```

$$\int_{t_{ls}}^{T} dt' s_j^* = \tilde{\kappa}_L(t') * s_j - \tilde{\kappa}_s(t') * s_j$$

I_1 in the code is computed as a sum:

$$I_1(t, T) = \sum_{t'=t}^{T} \left(s_L^*(t') - s_s^*(t') \right) * V^*(t') \tag{18}$$

0.2 steady-state potentials in Sacramento (2018)

$$u_k^p = \frac{g_B}{g_{lk} + g_B + g_A} v_{B,k}^P + \frac{g_A}{g_{lk} + g_B + g_A} v_{A,k}^P$$

$$\hat{v}_{B,k}^P = \frac{g_B}{g_{lk} + g_B + g_A} v_{B,k}^P$$

$$\hat{v}_k^I = \frac{g_B}{g_{lk} + g_B} v_k^I$$

$$\lambda = \frac{g_{som}}{g_{lk} + g_B + g_{som}}$$

1. In the torch implementation, there no persistence between timesteps at all. Input is fed into the network and processed feedforward and feedback. Output is read and weights (+biases) are updated. Rinse and repeat.

2. to what extent should dendritic and somatic compartments decay?

3. Can (should) we transfer the learned bias from the torch model?

4. I can "cheat" the apical voltage constraint for self prediction by increasing apical leakage conductance. How does this influence my model?

5. Is there some analytical approach to identifying why synaptic weights deviate from their intended targets?

6. I think that lambda needs to be scaled in dependence on g_{lk}, such that current inputs, spike inputs and leakage cancel each other out.

7. How do we deal with population size dependence?

0.3 Parameter study

- Transfer function ϕ
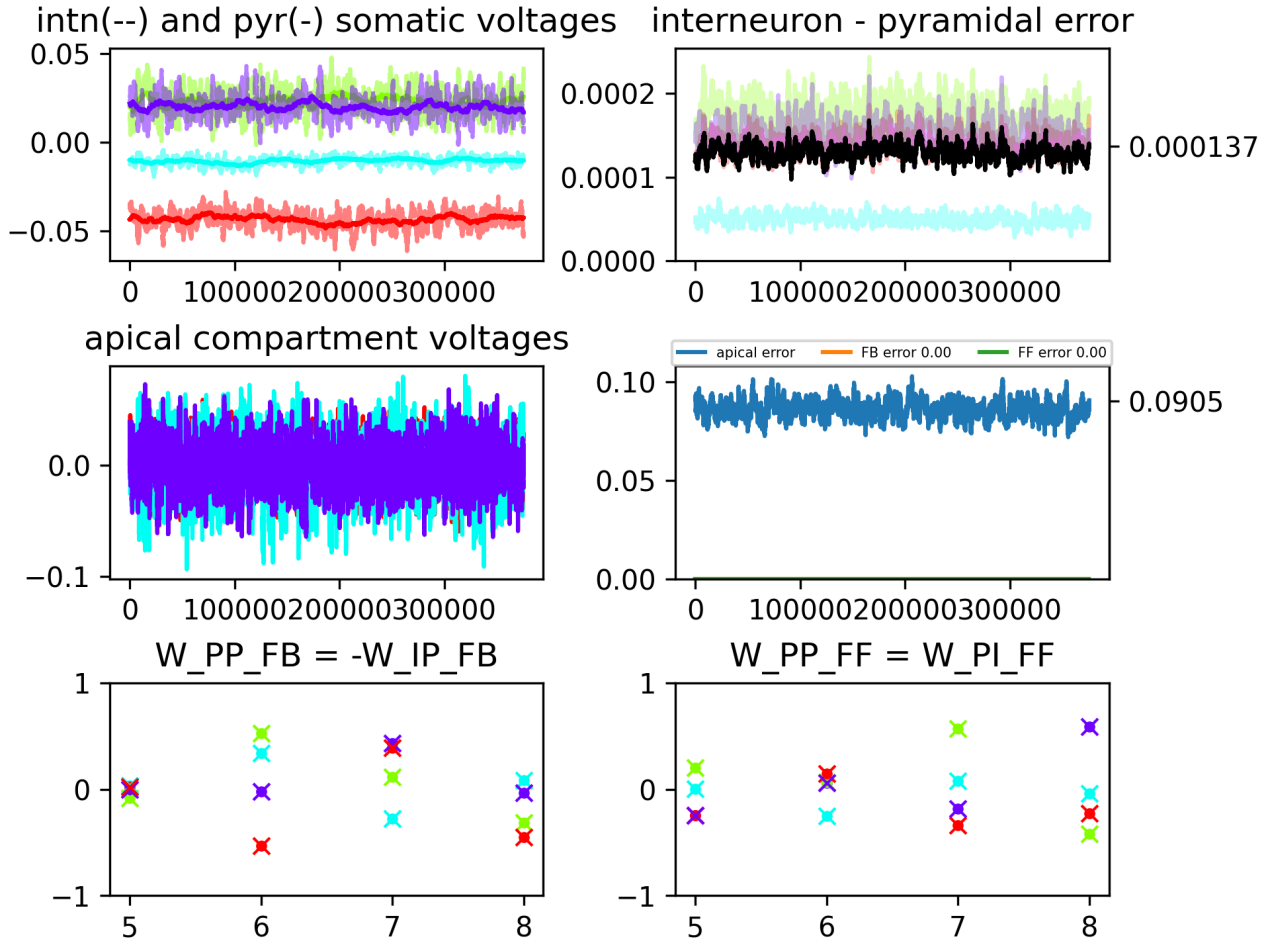
- interneuron mixing factor λ

Figure 1: Self-predicting initialization without plasticity

- injected current I_e

- dendritic leakage conductance $g_{lk,d}$

- somatic leakage conductance $g_{lk,s}$

- Learning rate η

- synaptic time constants τ_{delta}

- noise level σ

- Simulation time $\mathbb{T}$

- plasticity onset after the network relaxes

- compartment current decay τ_{syn}

Observations

- In self-predicting paradigm, Apical errors stay constant, despite interneuron error steadily increasing.

- Interneuron error (between neuron pairs) is proportional to absolute somatic voltage in self-predicting paradigm.
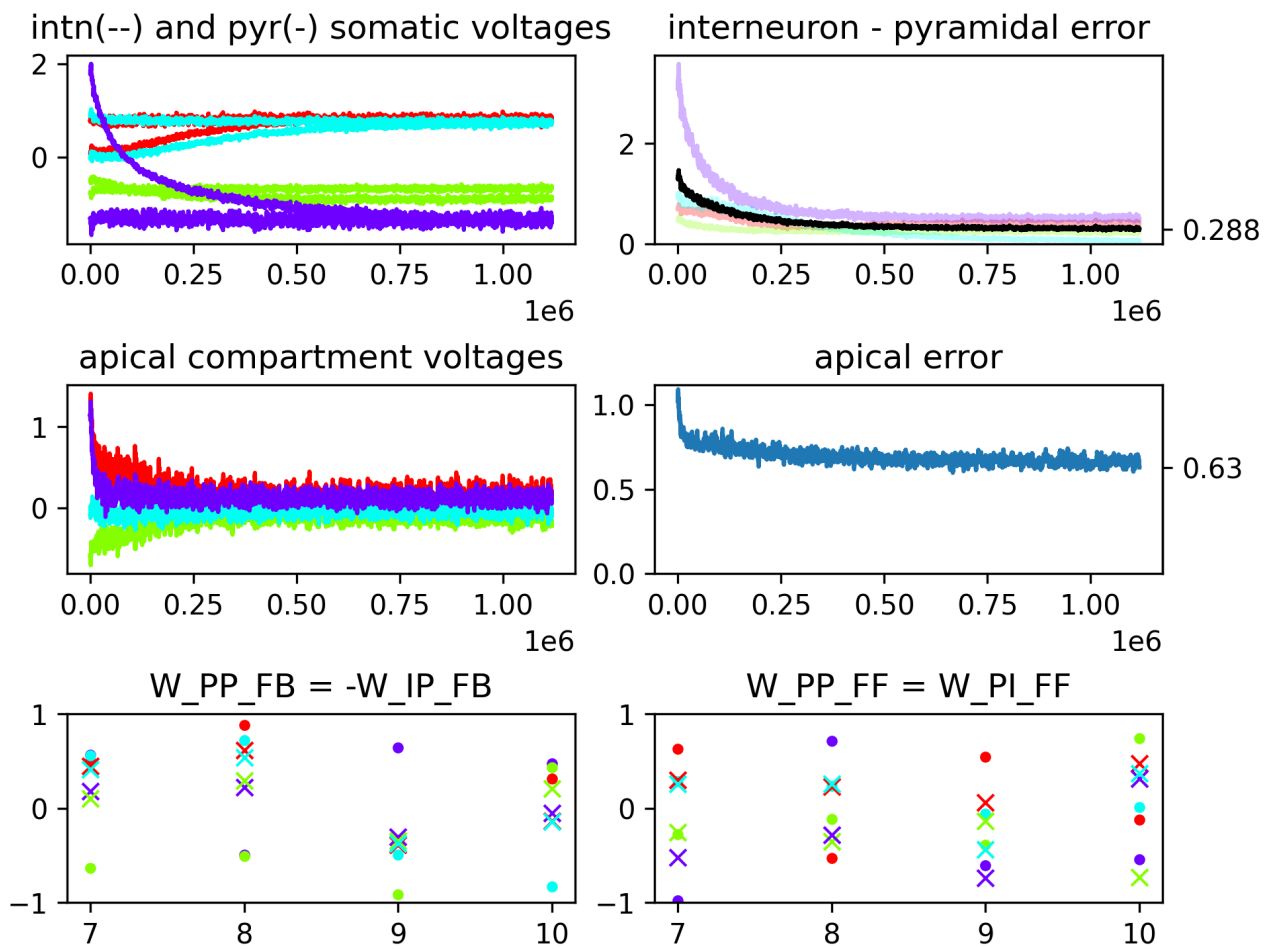
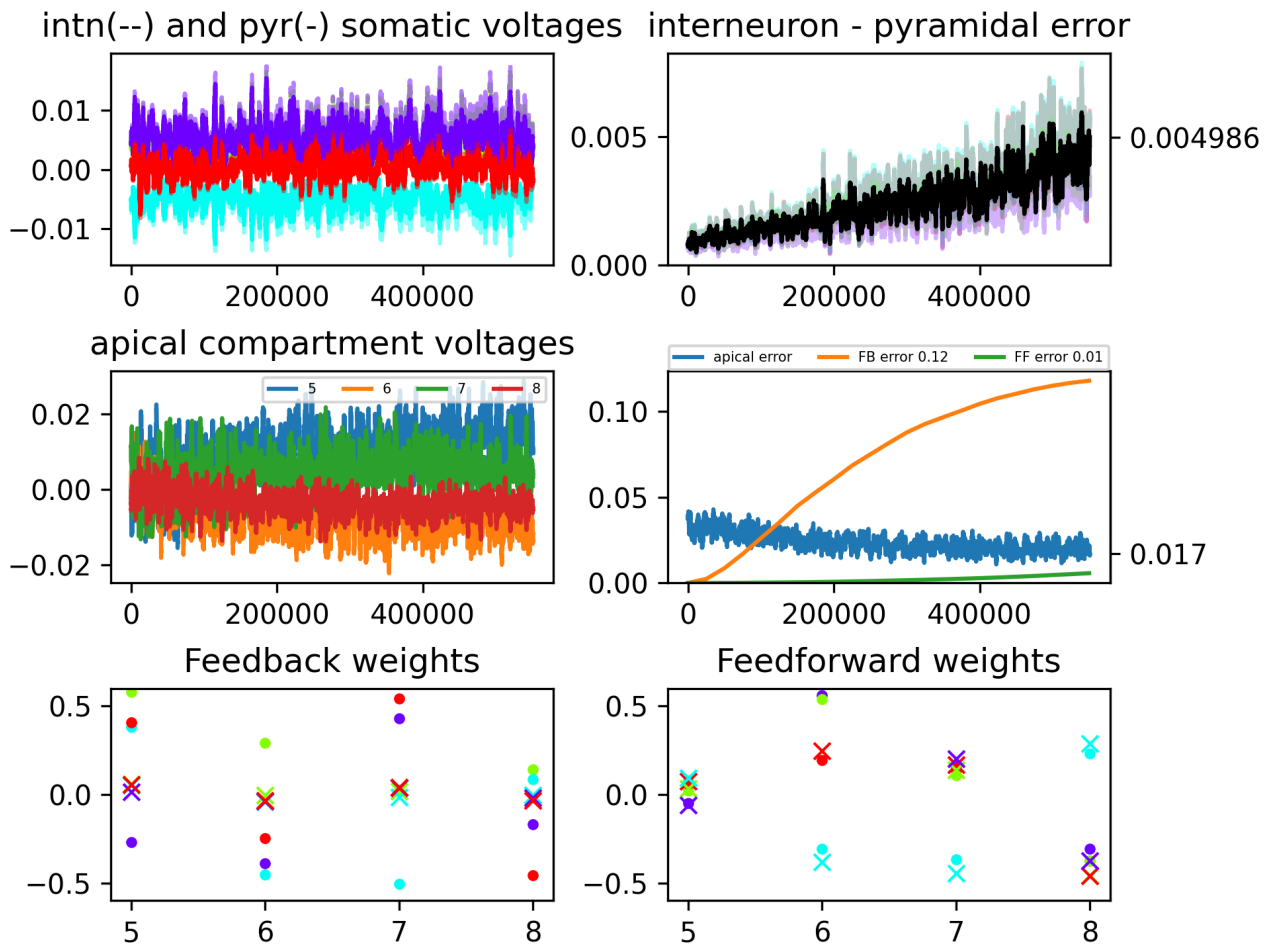**Figure 2:** Random initialization with plasticity enabled

Figure 3: Self-predicting initialization with plasticity enabled. Feedback weights collapse pretty quickly; feedforward weights remain shockingly stable. This seems to be neuron-specific. In other simulations of equal parametrization, only 1/4 remained stable compared to 3/4 as seen here. Apical voltage decreases over time, which somehow makes sense as feedback int→pyr weights are reduced to zero. Yet these would ideally precisely counteract feedback pyr weights.

- abs interneuron voltage is always higher than abs pyramidal voltage. This kind of makes sense, as interneurons receive direct current input proportional to pyramidal voltage in addition to feedforward input. This discrepancy disappears when setting λ to 0 as expected.

- When plasticity is enabled from a random starting configuration, apical error **sometimes** converges to better values than can be achieved in both self-predicting paradigms. I believe this to be a huge issue: the self-predicting state does not cause minimal apical voltage, and completely decayed feedback weights are preferable to perfectly counteracting feedback weights.

- feedforward weights tend to increase absolutely, i.e. drift towards the closest extreme. *This only happens since I re-implemented the second exponential term in the pyr_synapse.* Yet they do not simply explode to the nearest extremum, but will traverse a zero weight to reach the maximum with equal sign as the weight they are supposed to match.

- feedback weights tend to decay to around zero. Yet they appear to remain close to zero in the direction they are supposed to be.

- Idea: I think that the somatic nudging is handled as straight currents being sent to the neuron, instead of the difference between actual and desired somatic voltage.

- In the paper and Mathematica code, Feedback learning rate is 2-5 times higher than feedforward lr. In my model, for learning to happen on similar time scales, feedback lr has to be 100 times lower than feedforward lr. An indicator that my plasticity is messed up.

- The simulation is likely producing way too few spikes (5-20 per 1000ms iteration). Could adapting the activation function yield better results?

- In the Mathematica solution, leakage conductance is greater than 1! ($\delta U_i = -(g_L + g_D + g_{SI})U_i + g_D V_{BI} + g_{SI}U_Y$) with $g_L + g_D + g_{SI} = 1.9$

Chapter 1

Preliminary structural components

1.1 Synaptic delays

Where I will inspect the implications of synaptic delays inherent to the NEST simulations on the model and plasticity rule. In particular, I will look at the biological necessity for this type of delay and discuss why any model attempting to replicate neuronal processes must be resilient to these delays.

1.2 Literature review - Backpropagation in SNN

Where I will review other attempts at implementing biologically plausible Backpropagation alternatives and contrast them to the current model.

1.3 NEST Urbanczik-senn implementation

1.4 My neuron model

- Low pass filtering

- multi-compartment computation

- Imprecision of the ODE

- abuse of the somatic conductance

1.4.1 NEST rate neuron shenanigans

Given how long I worked on a rate neuron implementation in NEST, some pages should be devoted to this effort.

1.5 My synapse model

Where I discuss the synapse implementation with regard to multi-compartment neurons, urbanczik-archiving and in particular the issues with timing that arise from NEST delays.

1.6 The relation between the pyramidal microcircuit and actual microcircuits

Where I can finally use the shit that has been on my whiteboard for half a year...

This will also serve as valuable insight into how plausible this microcircuit actually is, and might give some insight into possible model extensions.

1.6.1 Interneurons and their jobs

1.7 Does it have to be backprop?

Where I will explain my concerns regarding the usefulness of approximating backpropagation in light of the substantial one-shot learning capability of the brain and the active inference model.

1.8 Discrepancies between mathematica and NEST model

1. Weights deviate slightly. This difference can be alleviated by exposing a single stimulus for a longer duration before switching.

1.9 Transfer functions

Where I will discuss the sensitivity of this entire simulation to minor changes in the parametrization and style of transfer function being used.

Chapter 2

The weight-leakage tradeoff

Where I will discuss the issue, that decreasing both synaptic weights and dendritic leakage conductance lead to more stability in the dendritic voltage, while at the same time requiring longer exposure per iteration.

TODOs

- Prove that the network is stable in the self-predicting state and at the end of learning

- Show the limits of learning capability (i.e. how big of a network it can match)

- Test the network on a real-world dataset (mnist)

- prove/find literature on why the poisson process is a rate neuron in the limit.

- Does the network still learn when neurons have a refractory period?

- Comparison to other spiking backprops

- what can we learn from this? does it describe part of the brain