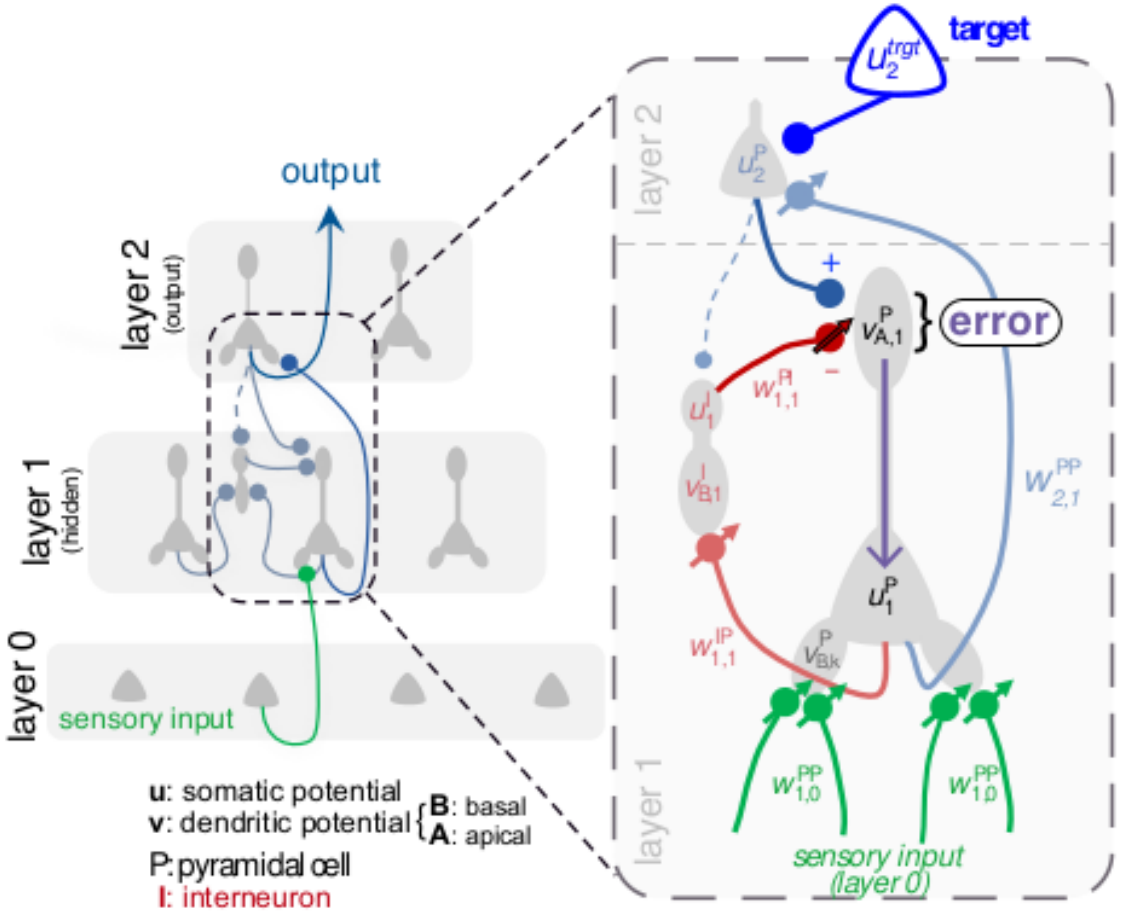
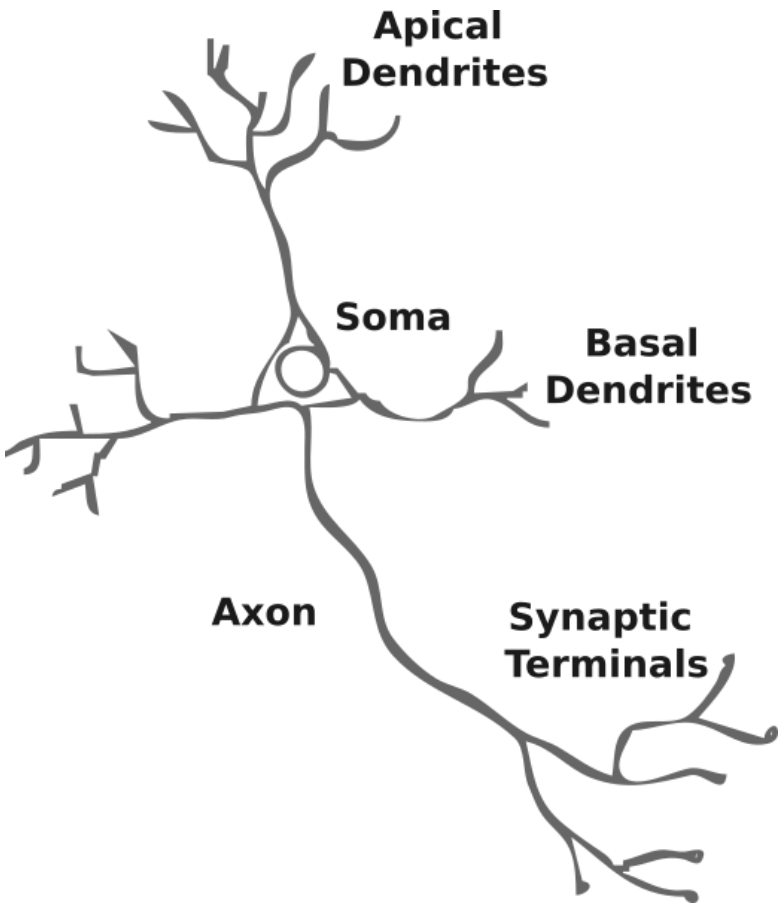
Dendritic cortical microcircuits approximate the backprob algorithm (Sacramento 2017)

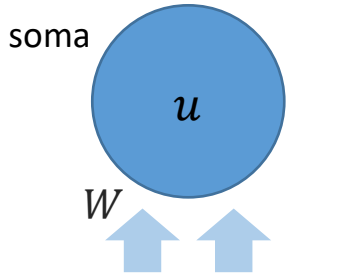


- reproduce lost code (→ repository with source and examples will be available online)
- How does the model work? Any subtleties?

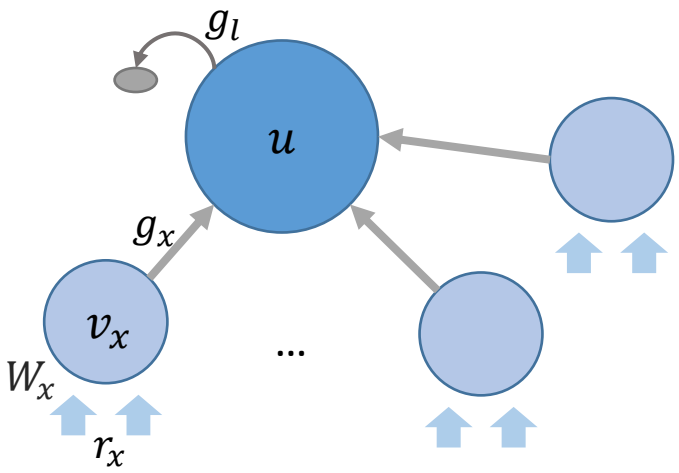
(future: bring to Hardware)

From point neurons to multi-compartment neurons





presynaptic input r



$$\dot{u} = -g_l u + \sum_x g_x(v_x - u)$$

$$v_x = \sum_i W_{xi} r_{xi}$$

$$r = \phi(u)$$

$$\tau\dot{u} = -u + \tau \sum_x g_x v_x$$

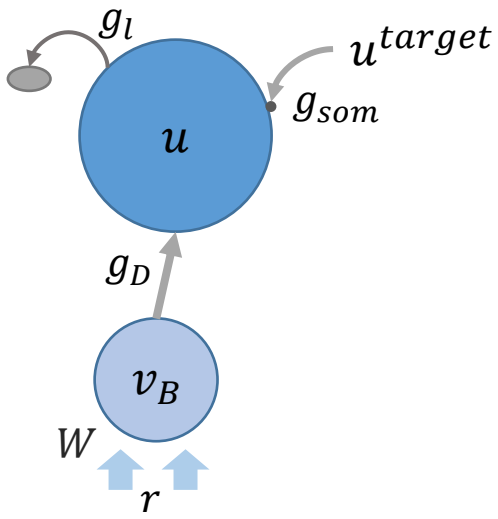
$1/\tau = g_{tot}$
$= g_l + \sum g_x$

$T \gg \tau$

$$\tilde{u} = \tau \sum_x g_x v_x$$

steady-state solution

Learning by dendritic prediction of somatic spiking



$$\hat{v}_B \equiv \tilde{u} = \frac{g_D}{g_l + g_D} v_D = \frac{g_D}{g_l + g_D} W \cdot r$$

with nudging $\tilde{u} = (1-\lambda)\hat{v}_B + \lambda u^{target}, \quad \lambda = \frac{g_{som}}{g_l + g_D + g_{som}}$

dendritic prediction error: $e \equiv \phi(u) - \phi(\hat{v}_B)$

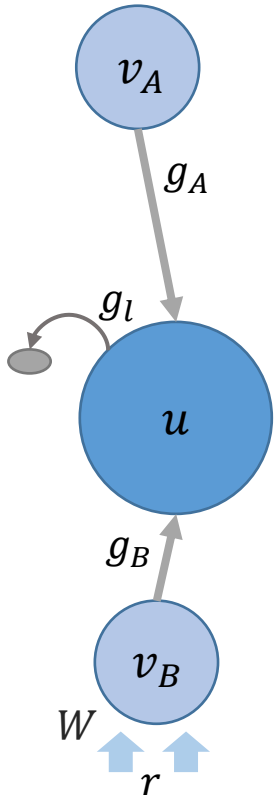
 $\boxed{\dot{W} = \eta\, e\, r^T}$  until $\hat{v}_B = u^{target}$

in praxis: $\tau_W \dot{\Delta} = -\Delta + e\, r^T$

$\dot{W} = \eta \Delta$

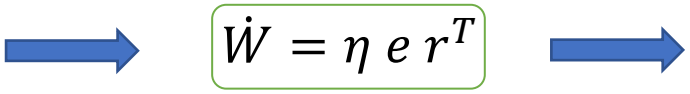
Supplements; Urbanczik 2014

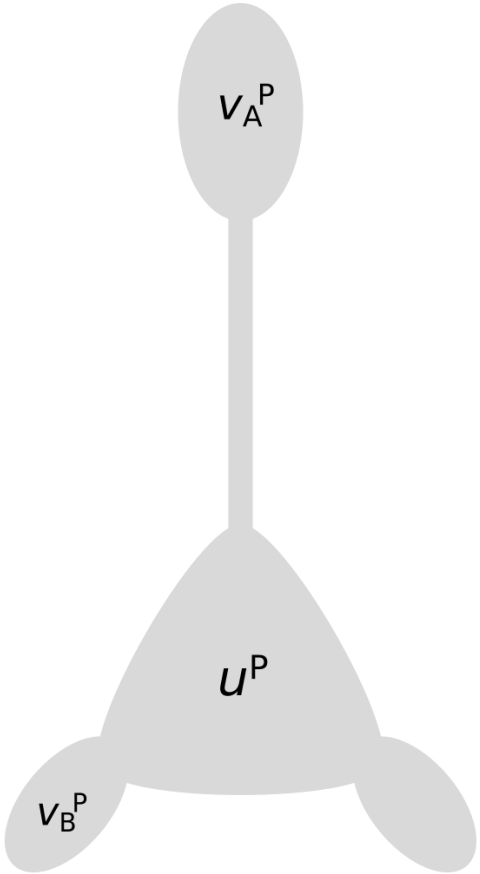
Pyramidal neurons



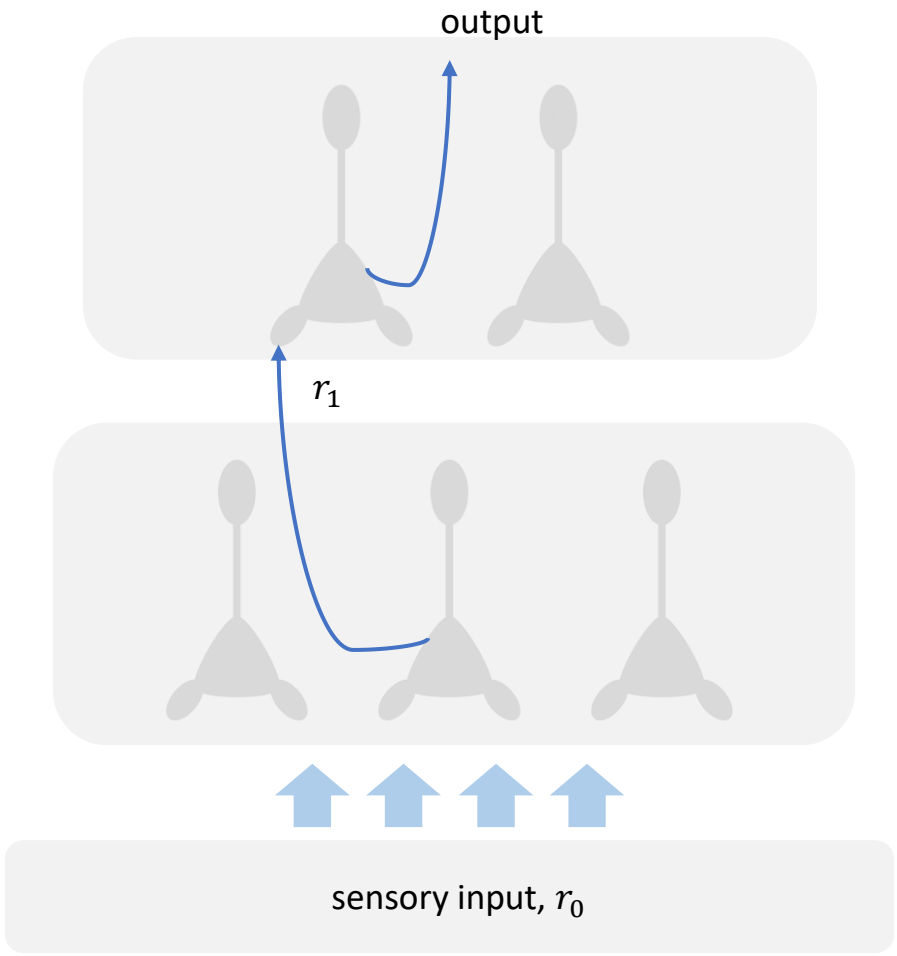
$$v_A \begin{cases} = 0 \Rightarrow \hat{v}_B \equiv \tilde{u} = \dfrac{g_B}{g_l + g_B + g_A}\, v_B = \dfrac{g_B}{g_l + g_B + g_A} W \cdot r \\ \neq 0 \Rightarrow \tilde{u} = \hat{v}_B + \lambda\, v_A\,, \quad \lambda = \dfrac{g_A}{g_l + g_B + g_A} \end{cases}$$

dendritic prediction error: $e \equiv \phi(u) - \phi(\hat{v}_B)$





Deep cortical microcircuits (naïve)



apply the learning rule in the output layer

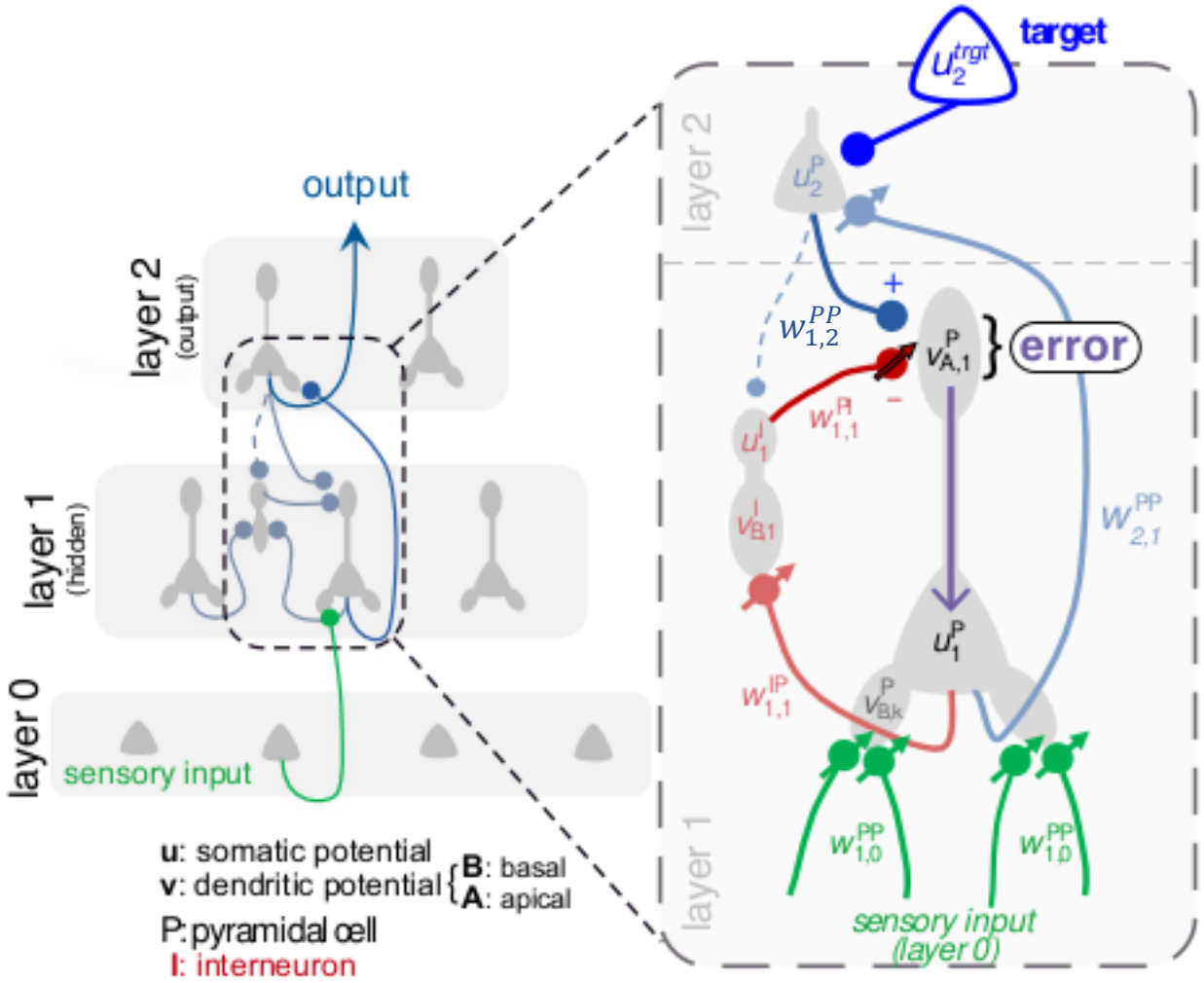
→ learns mapping $r_1(r_0) \rightarrow u^{target}$

How can the hidden layer support learning?
How should it adjust $r_1(r_0)$?

propagate output error back to hidden layer via $r_1(r_0)$

→ Implement a feedback-loop

Deep cortical microcircuits



lateral plasticity → self-predicting state:

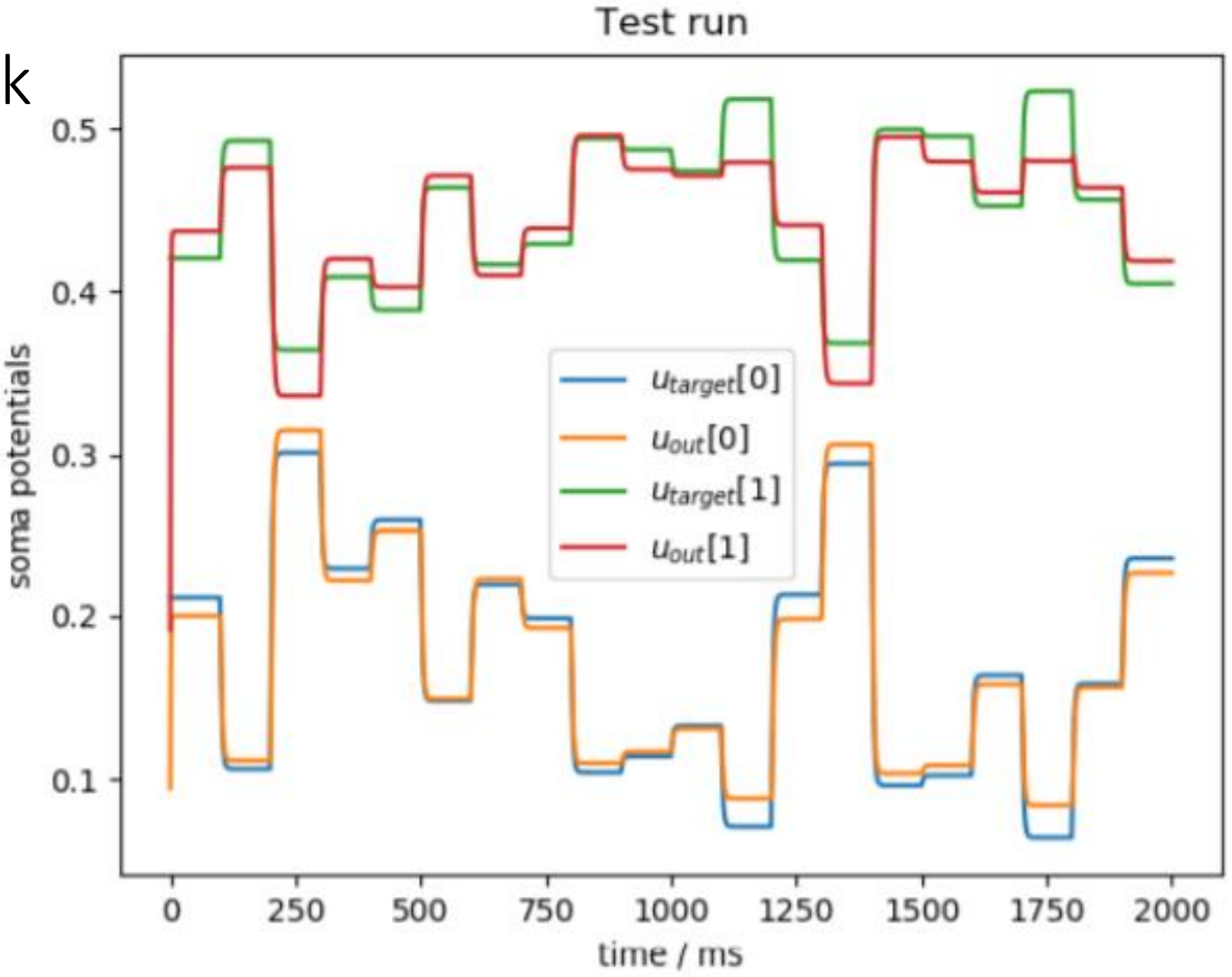
$$W_{2,1}^{PP} = W_{1,1}^{IP}$$

$$W_{1,1}^{PI} = -W_{1,2}^{PP}$$

$$\Rightarrow \quad v_A^P \approx 0, \quad u_1^I \approx u_2^P$$

Mimic-regression-task

task: 2x3x2 network learns to mimic the output of a randomly initialized network of same size



Steady-State Approximation

100 ms pattern time ≫ 0.1 ms time step dt → training is slow

idea: approximate the settling process

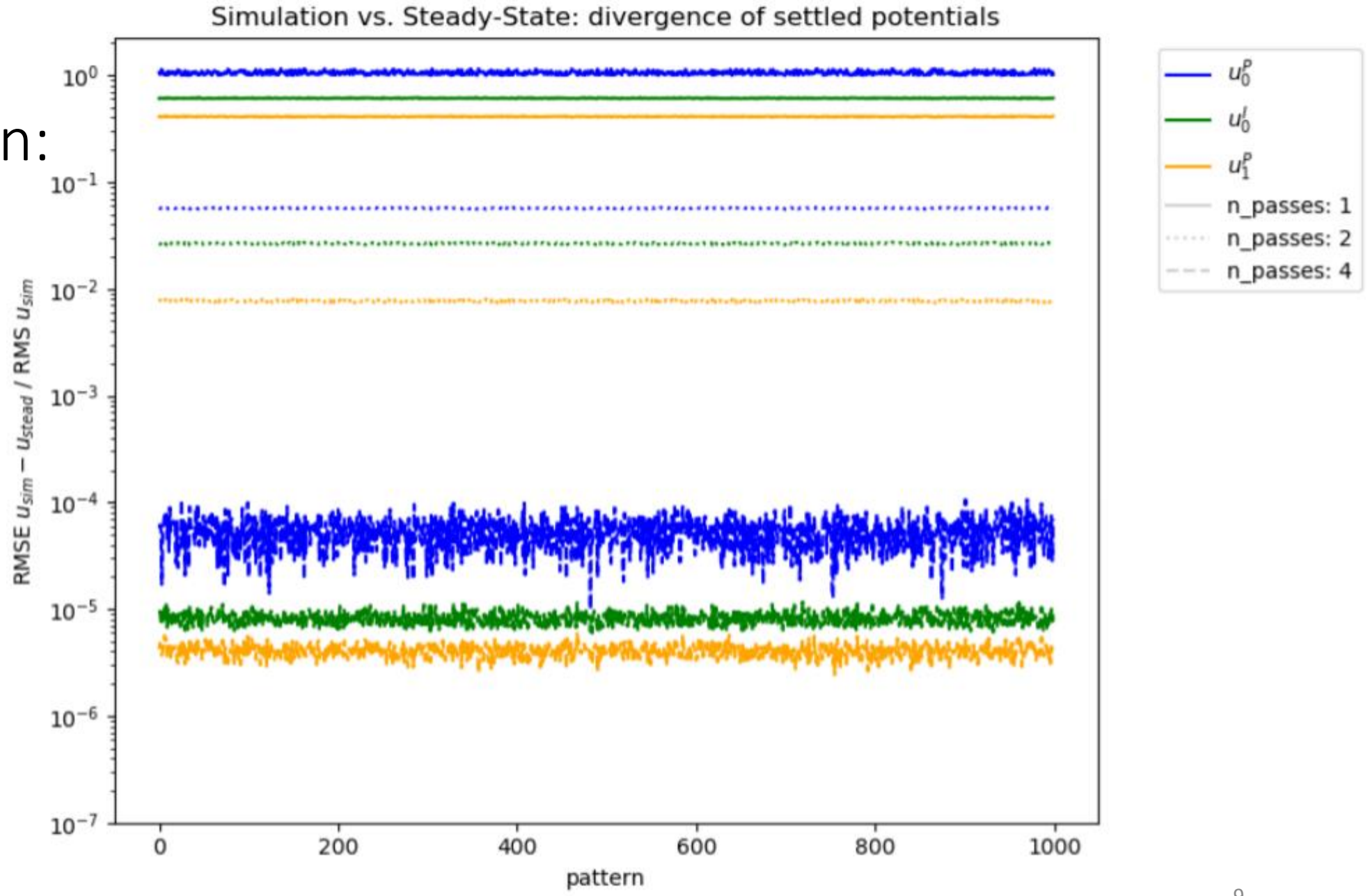
1. propagate the input upwards, layer by layer with dendritic prediction / steady-state solution:
2. propagate corrections downwards with full steady-state solution:
3. repeat n_{passes} times → (hopefully) converge to steady-state of entire network

Plasticity:

- solve $\dot{W} = \eta\, e\, r^T$ with time step $dT \gg dt$
- update potentials after each weight update

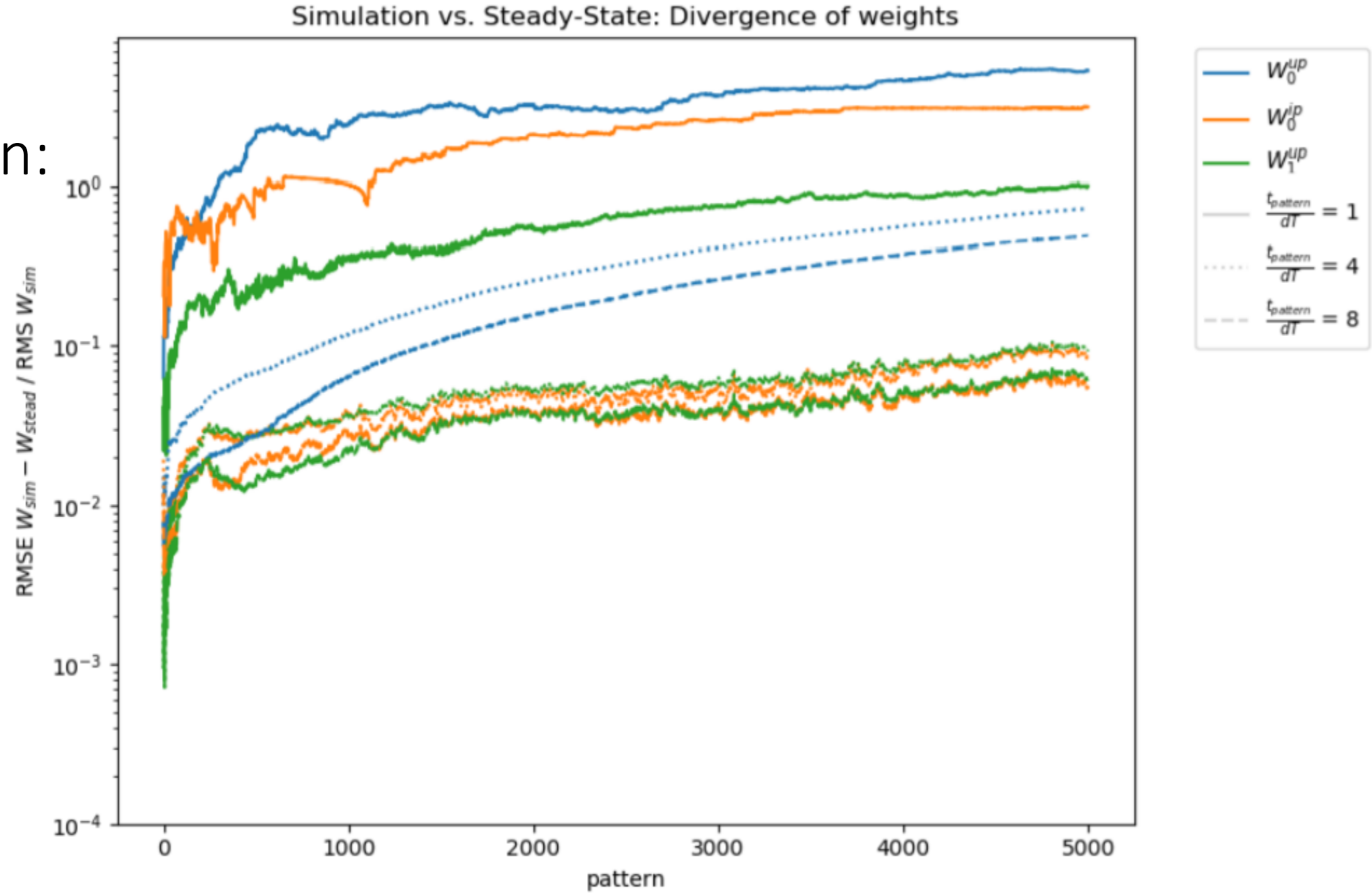
What do we hope to gain? Approximation might not be accurate, but good approach for hyperparameter search!

Steady-State Approximation: Validity



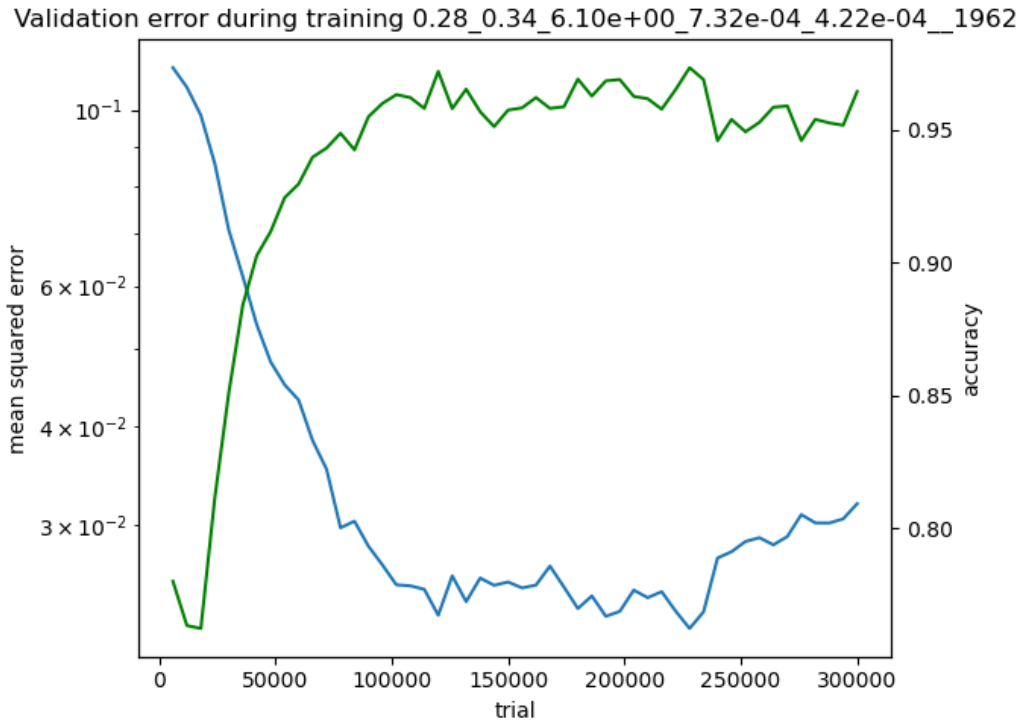
relative rms deviation of soma potentials, when plasticity is switched off

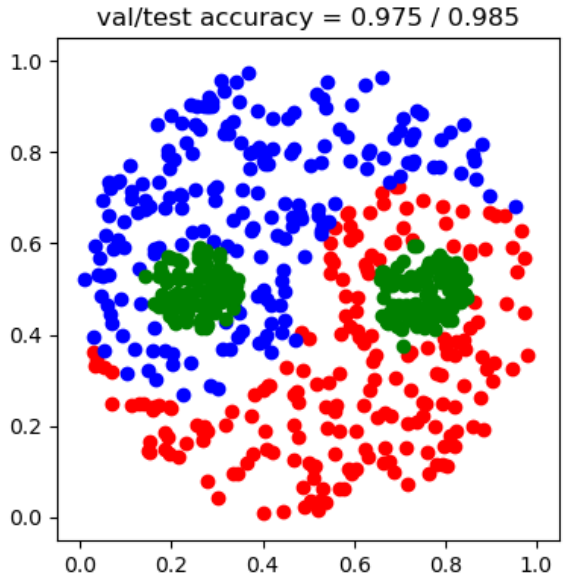
Steady-State Approximation: Validity

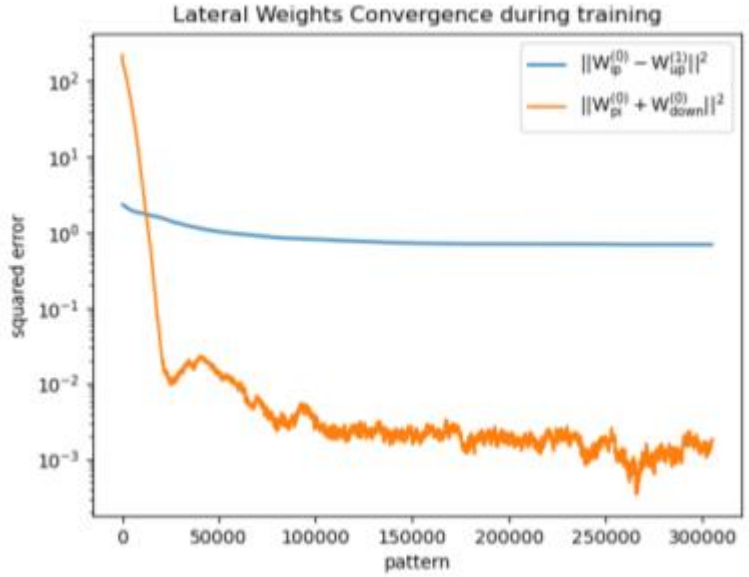


relative rms deviation of weights

Yin-Yang results





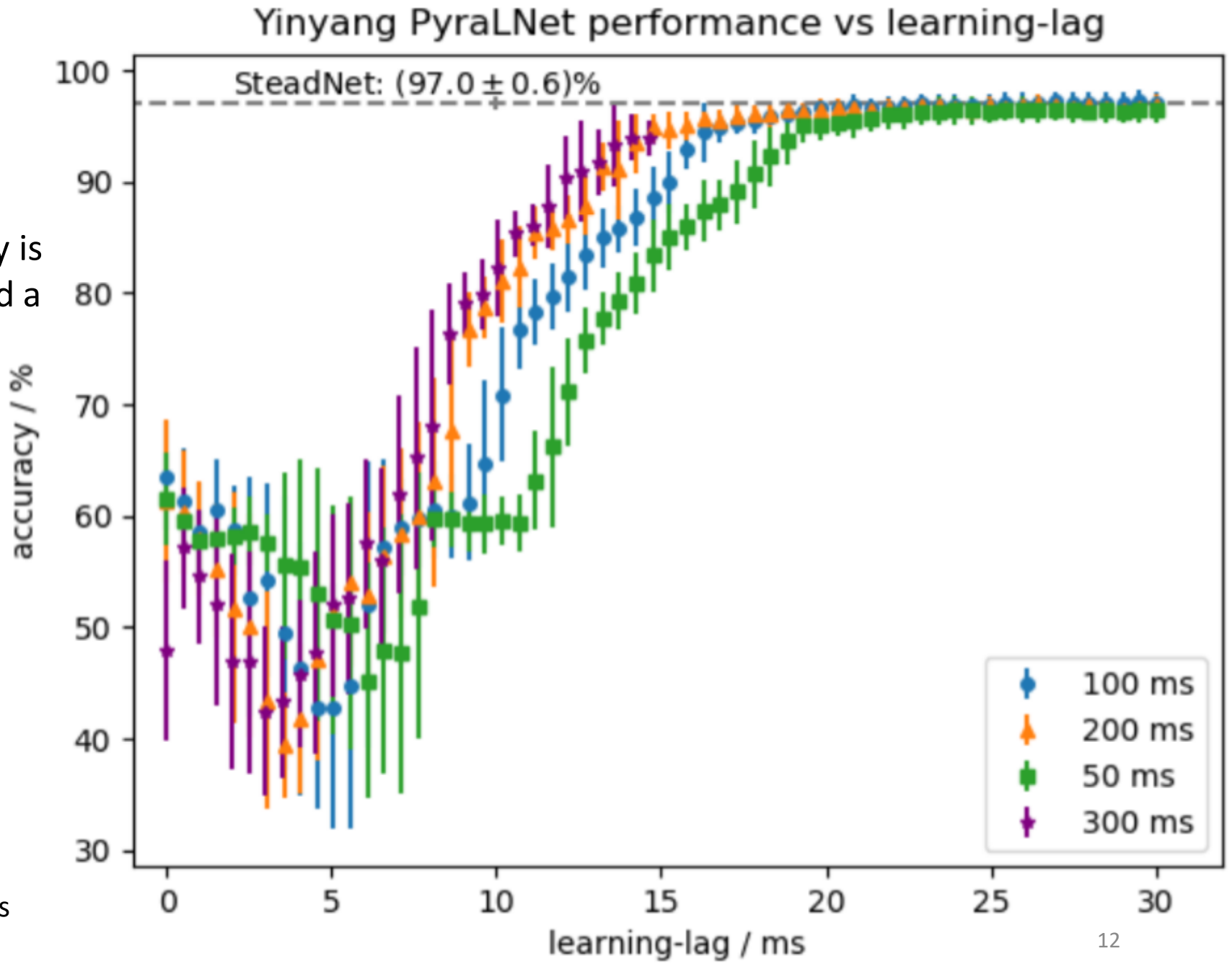


trained on 55 epochs a 6000 samples: $(97.2 \pm 1.1)\%$ accuracy

Yin-Yang results: learning-lag

during settling unwanted plasticity is induced → chokes learning beyond a certain point

➡ introduce learning-lag: no plasticity for fixed time after each new pattern



*the network is initialized in the self-predicting state and PI-plasticity is switched of
*trained on 6000 samples for 45 epochs

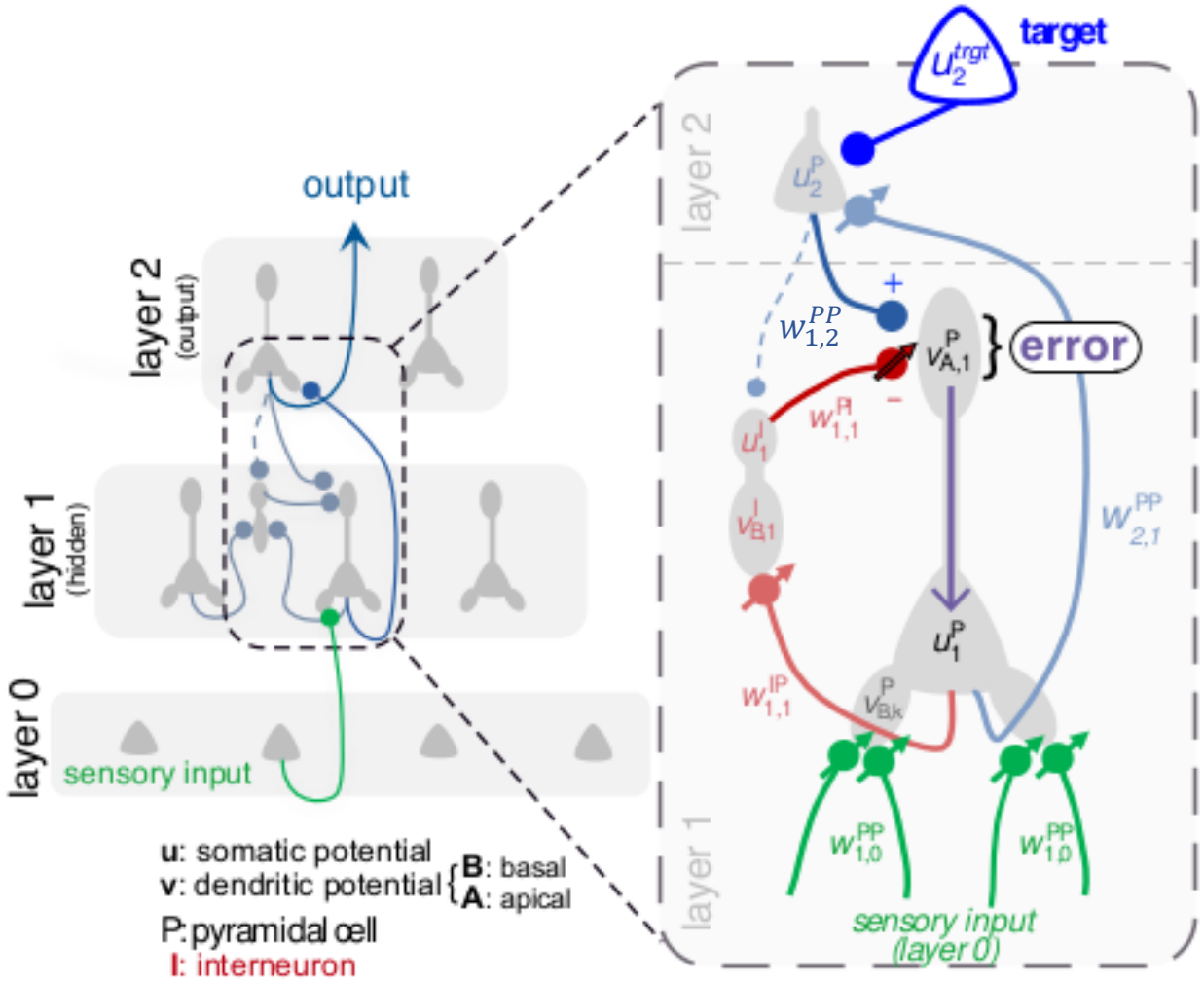
Summary

- Code for simulation (PyraLNet) and steady-state approximation (SteadNet) is available
- model is able to solve the Yin-Yang problem effectively once learning-lag is introduced
- SteadNet is able to solve MNIST

However:

- SteadNet results cannot directly be generalized to full simulation (since unwanted plasticity during settling might cause problems)

Deep cortical microcircuits



$$u_2^P = -g_{tot}u_2^P + \mathrm{g_B W_{2,1}^{PP}}\phi(u_1^P) + g_{som}u^{target}$$

$$u_1^I = -g_{tot}u_1^{\mathrm{I}} + \mathrm{g_D W_{1,1}^{IP}}\phi(u_1^P) + g_{som}u_2^P$$

$$u_1^P = -g_{tot}u_1^P + \mathrm{g_B W_{1,0}^{PP}} r_0 + g_A v_{A,1}^P$$

$$v_{A,1}^P = W_{1,1}^{PI}\phi(u_1^I) + W_{1,2}^{PP}\phi(u_2^P)$$

$$\dot{W}_{1,1}^{PI} = -\eta^{PI} v_{A,1}^P \ , \quad \dot{W}_{1,2}^{PP} = 0$$

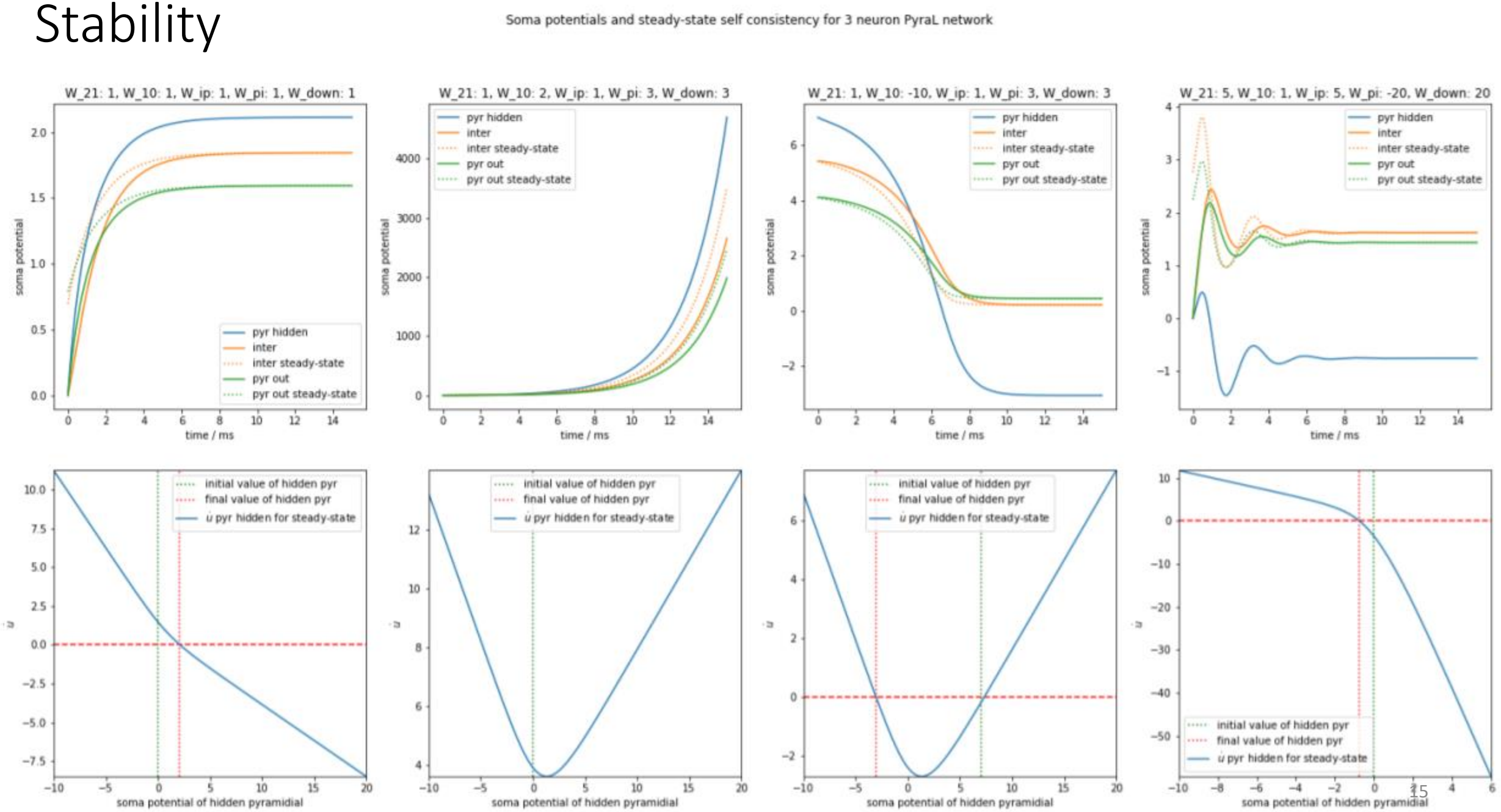
lateral plasticity → self-predicting state:

$$W_{2,1}^{PP} = W_{1,1}^{IP}$$

$$W_{1,1}^{PI} = W_{1,2}^{PP}$$

$$\Rightarrow \quad v_A^P \approx 0, \quad u_1^I \approx u_2^P$$

Stability

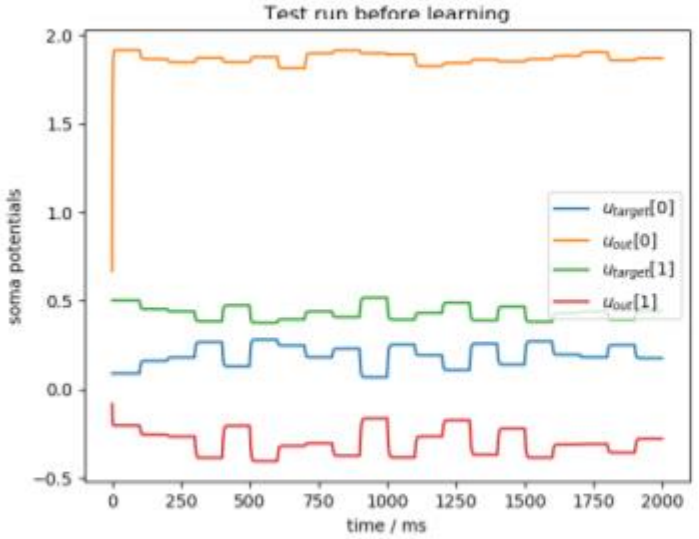

Soma potentials and steady-state self consistency for 3 neuron PyraL network
W_21: 1, W_10: 1, W_ip: 1, W_pi: 1, W_down: 1
W_21: 1, W_10: 2, W_ip: 1, W_pi: 3, W_down: 3
W_21: 1, W_10: -10, W_ip: 1, W_pi: 3, W_down: 3
W_21: 5, W_10: 1, W_ip: 5, W_pi: -20, W_down: 20
soma potential
time / ms
pyr hidden
inter
inter steady-state
pyr out
pyr out steady-state
initial value of hidden pyr
final value of hidden pyr
$\dot{u}$ pyr hidden for steady-state
$\dot{u}$
soma potential of hidden pyramidial

Mimic-regression-task

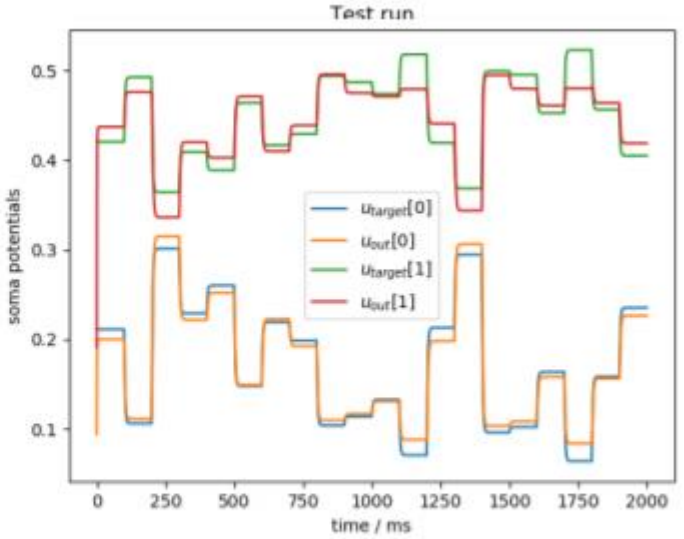
task: 2x3x2 network learns to mimic the output of a randomly initialized network of same size

Details:

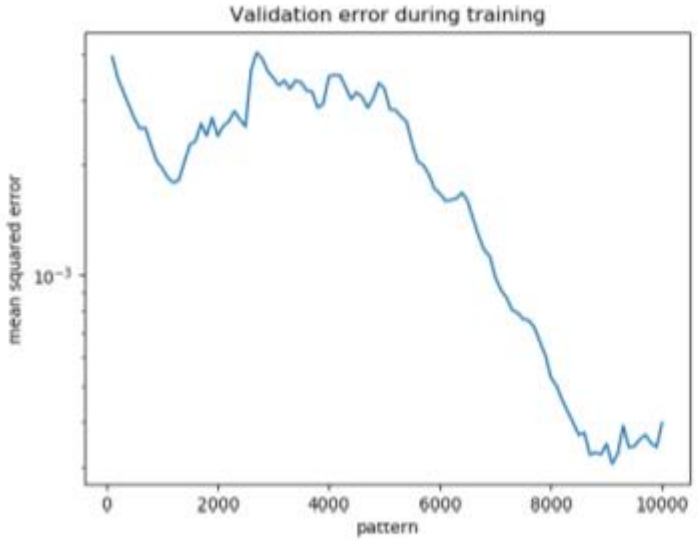
- $g = 0.1 - 1, \ \tau \approx 0.5$ ms
- weights of order 1 (elements)
- input, output patterns are presented for 100 ms and are smoothly transitioned in between (3 ms)
- plasticity time constant $\tau_W = 30$ ms
- simulation time step: 0.1 ms



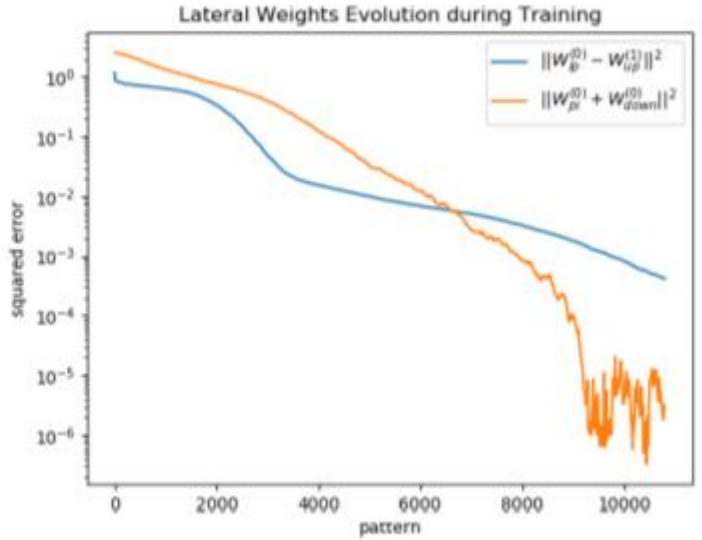
(a) Output layer soma potentials and target potentials before training.



(b) Output layer soma potentials and target potentials after training.

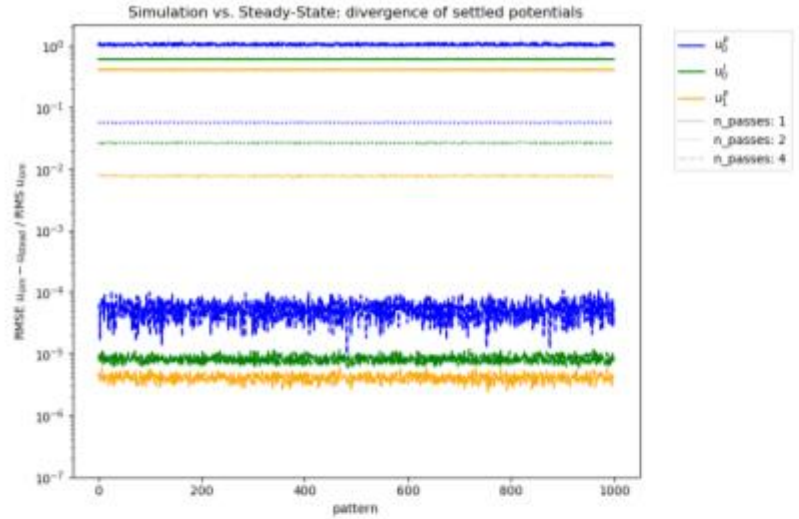


(c) MSE validation error during training.

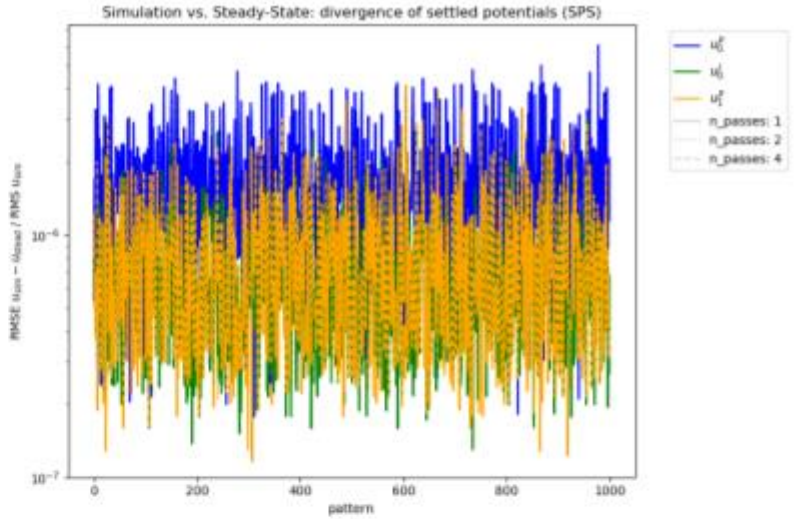


(d) Emergence of self-predicting-state during training starting from randomly initialized connections.

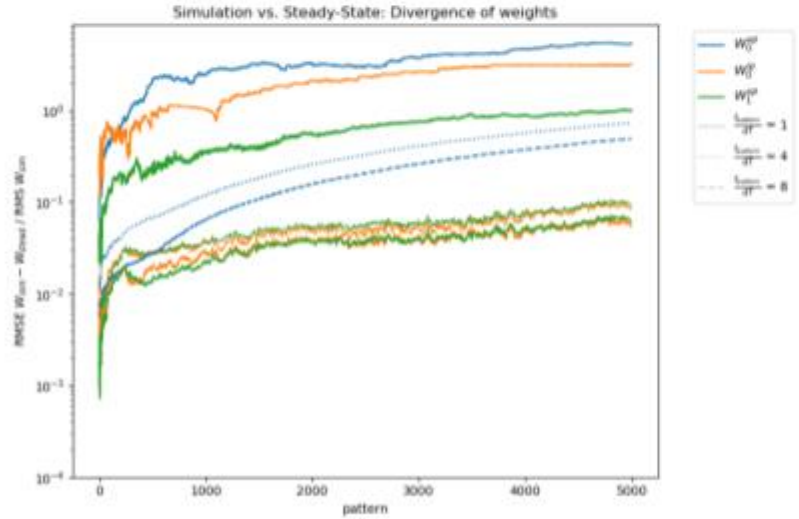
Steady-State Approximation: Validity



(a) Deviation[5] of the simulated from the approximation soma potentials, when plasticity is switched off.



(b) Deviation of the simulated from the approximation soma potentials, when plasticity is switched off, but the network is initialized in the self-predicting state.



(c) Deviation of the simulated from the approximation weights, when plasticity is switched on.

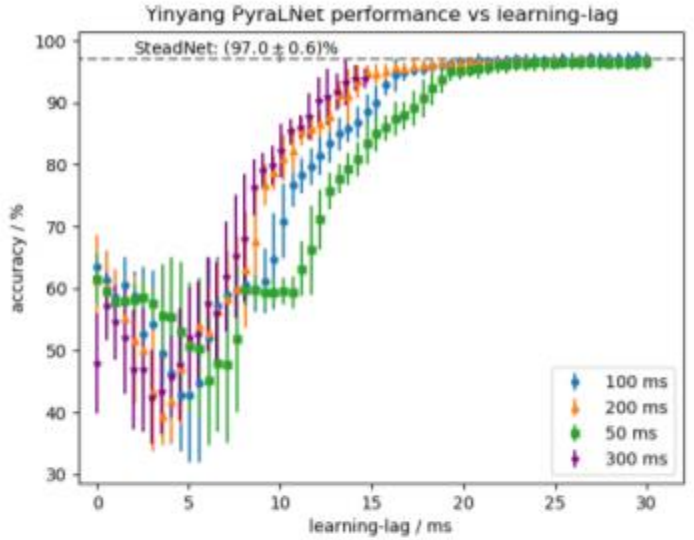
(d) Deviation of the simulated from the approximation weights, when plasticity is switched on, but the network is initialized in the self-predicting state.

Yin-Yang results: learning-lag

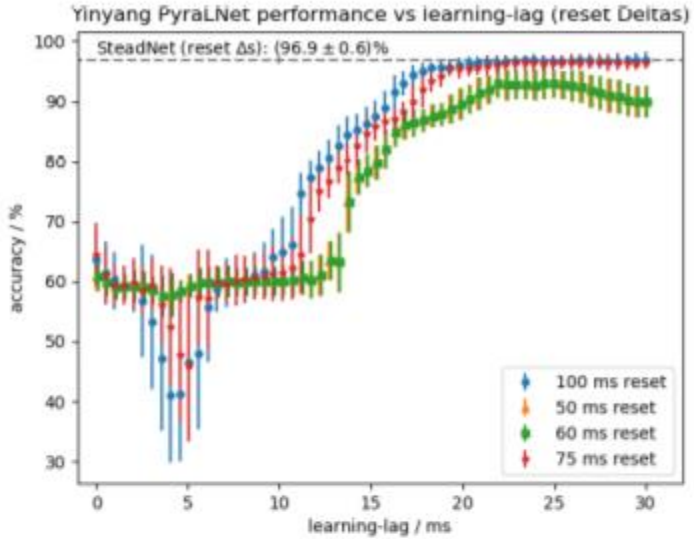
during settling unwanted plasticity is induced → chokes learning beyond a certain point

➡ introduce learning-lag: no plasticity for fixed time after each new pattern

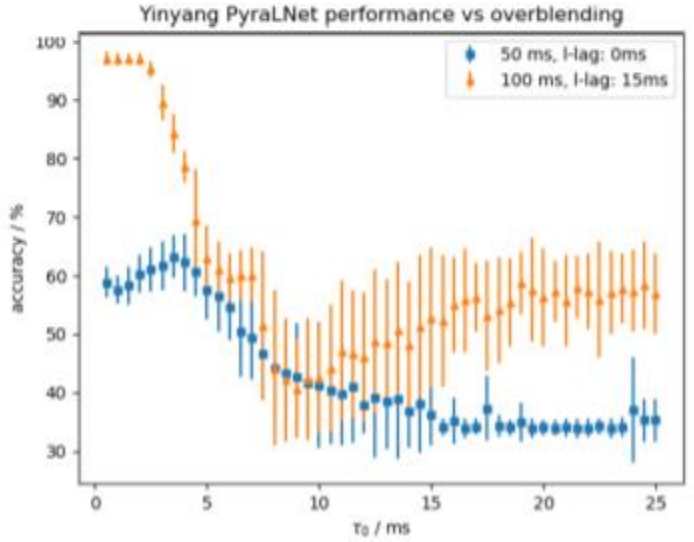
*the network is initialized in the self-predicting state and PI-plasticity is switched of
*trained on 6000 samples for 45 epochs



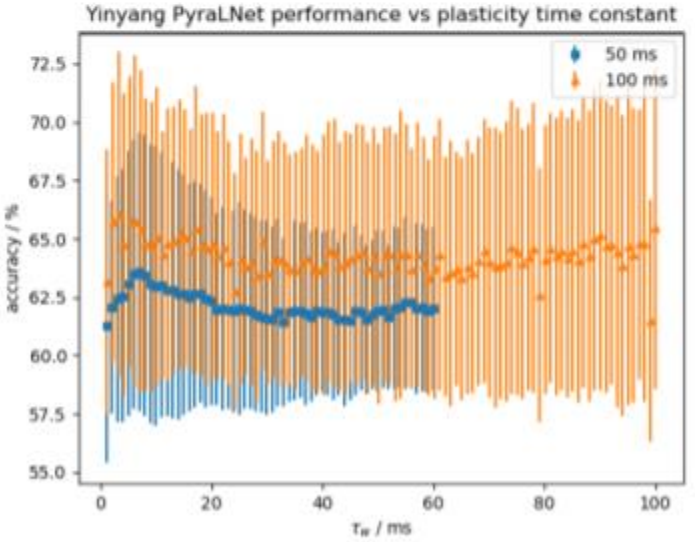
(a) Test set accuracy versus learning-lag for different $t_{pattern}$. See main text for parameter details.



(b) Test set accuracy versus learning-lag for different $t_{pattern}$, but with *reset_deltas* switched on.



(c) Test set accuracy versus pattern transition time constant τ_0.



(d) Test set accuracy versus plasticity time constant τ_w.