Spike-based models for cognitive computations and robust training of memristive neural networks

PhD candidate: Ceca Kraišniković
Advisor: Dr. Robert Legenstein
July 21, 2023

PhD thesis outline

- **Spike-based models for cognitive computations**
  - Solving cognitive tasks involving sequences of symbols and rules
  - Using biologically plausible models: Recurrent Spiking Neural Networks (SNNs)
  - **Motivation:** insights into mechanisms of the brains
    - By solving cognitive, rather than sensory processing tasks
    - Testing computational capabilities of SNNs

- Brain-inspired computing
  Robust training of memristive neural networks
  - Using *memristors* as artificial synapses
  - To deal with non-ideal behaviors of memristors
  - **Motivation:** memristors are a promising technology for implementing learning in hardware, and *in-memory* computing

Part 1:
Spike-based models for solving cognitive tasks

Salaj*, D., Subramoney*, A., **Kraisnikovic*, C.**, Bellec, G., Legenstein, R., & Maass, W. (2021). Spike frequency adaptation supports network computations on temporally dispersed information. *Elife*.
* joint first authors

Kraisnikovic, C., Maass, W., & Legenstein, R. (2022). Spike-based symbolic computations on bit strings and numbers. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*. IOS Press.

Working memory and neural codes

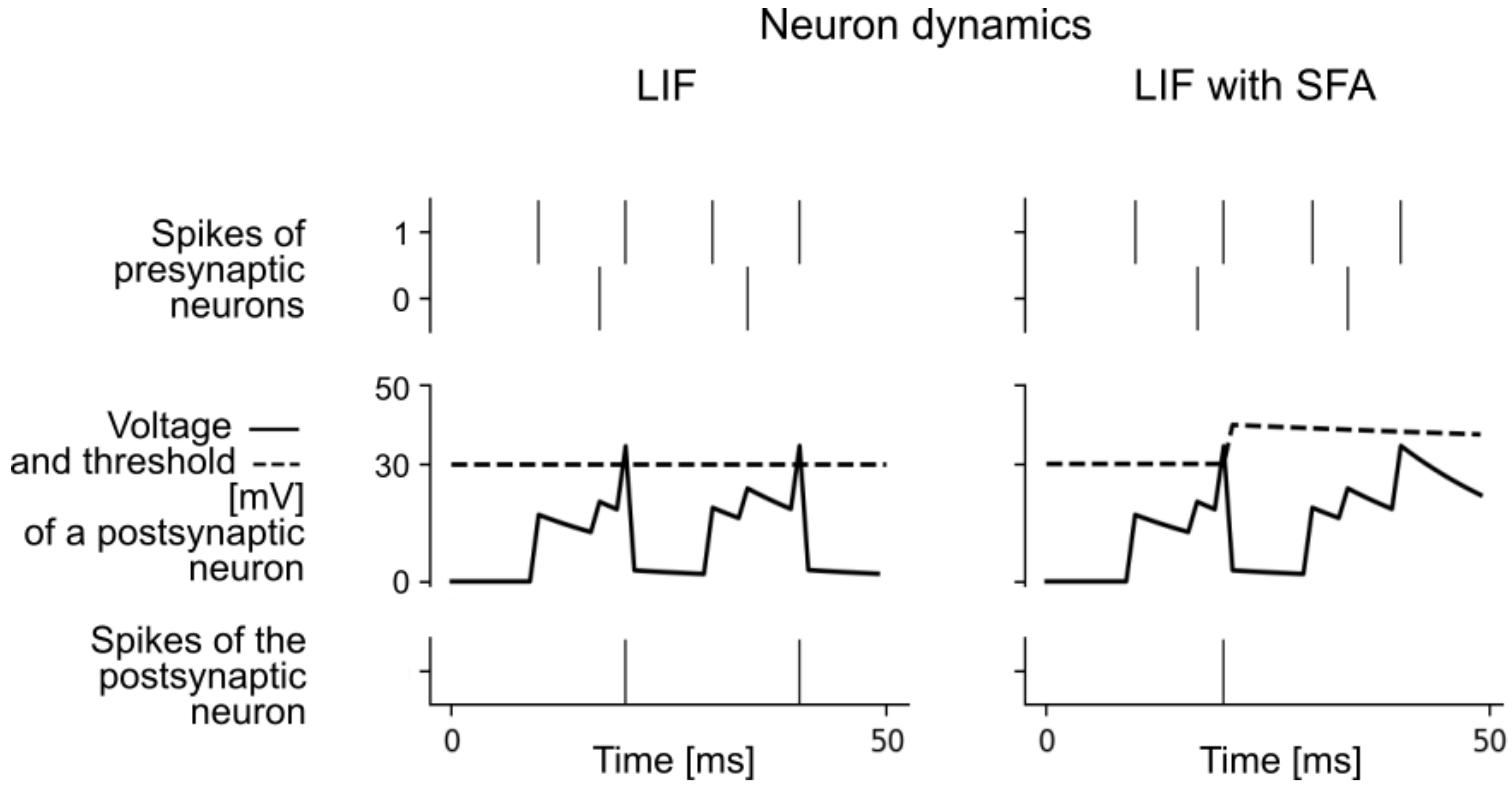
- Information encoding, maintenance and manipulation
- Studied through neural codes

- **Neural adaptation** [1, 2]
  - Response decays upon repeated or prolonged stimulation
  - Reflects context and history dependence
  - Spike Frequency Adaptation **(SFA)** [3, 4]
    - On a single neuron level
    - Preceding firing activity of a neuron transiently increases its threshold

[1] Weber, A. I., & Fairhall, A. L. (2019). The role of adaptation in neural coding. *Current opinion in neurobiology.*
[2] Benda, J. (2021). Neural adaptation. *Current Biology.*
[3] Allen Institute. 2018b. Cell Feature Search. http://celltypes.brain-map.org/data
[4] Pozzorini et al., 2015. Automated high-throughput characterization of single neurons by means of simplified spiking models. *PLoS computational biology.*

Neuron model

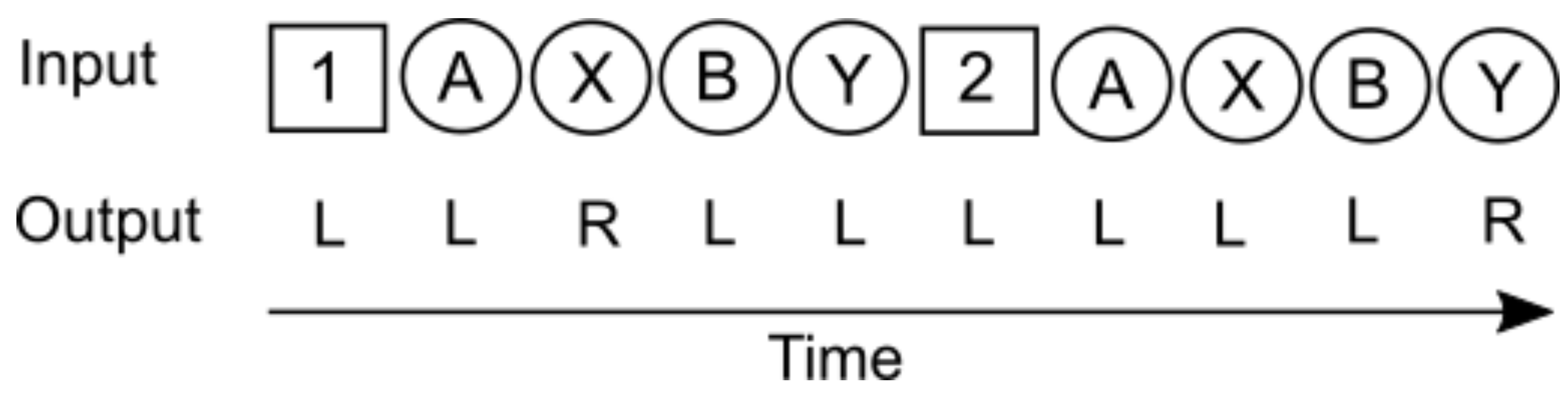
- **Generalized Leaky Integrate and Fire (LIF) neuron model** [5, 6] **and Spike Frequency Adaptation (SFA)**

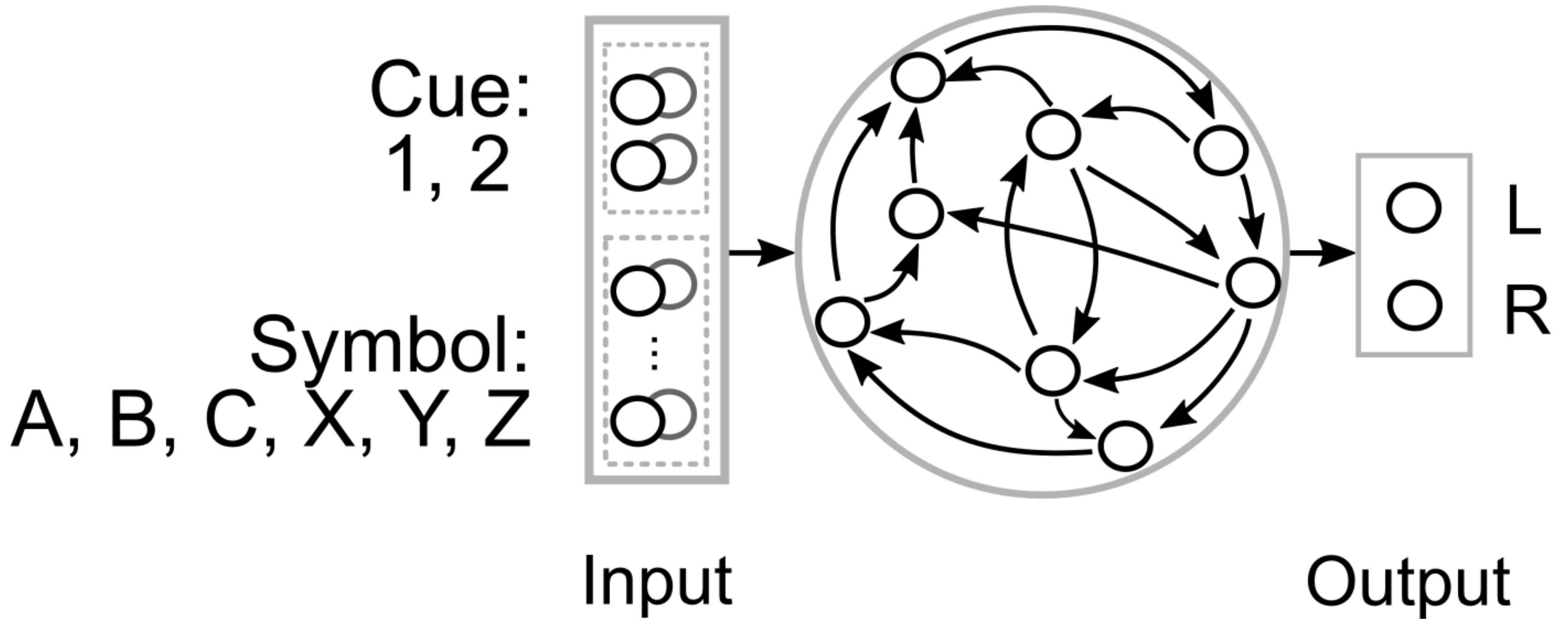


[5] Teeter et al., 2018. Generalized leaky integrate-and-fire models classify multiple neuron types. *Nature communications*.
[6] Allen Institute. 2018a. Allen Cell Types Database Technical White Paper: Glif models.

SNNs with SFA learn the 12AX task

The 12AX task (simple version) [7, 8]





Under the context of:
- rule 1, observe if A is followed by X.
- rule 2, observe if B is followed by Y.

When the sequence 1-A-X or 2-B-Y is detected, press R (right button), otherwise L (left button).

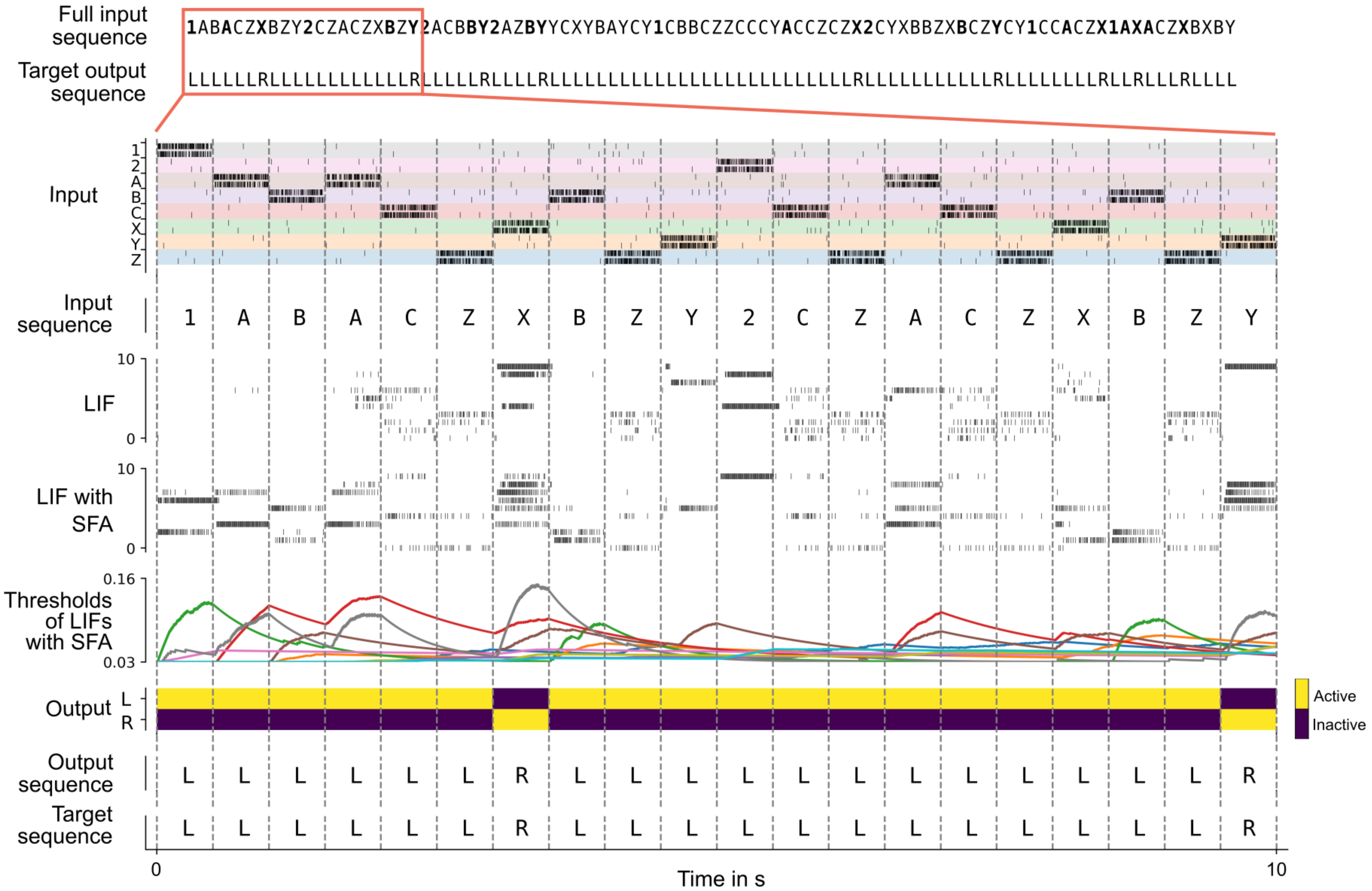
- Increasing difficulty by adding distractor letters C and Z between relevant letters
- With distractor letters, information has to be maintained on two levels:
  - Most recent relevant digit
  - Most recent relevant letter

[7] Frank et al., 2001. Interactions between frontal cortex and basal ganglia in working memory: a computational model. *Cognitive, Affective, & Behavioral Neuroscience*.
[8] O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*.

SNNs with SFA learn the 12AX task

Performance: 97.79% of test trials with all correct outputs L and R.

No need for hierarchical WM, as e.g., in [8].

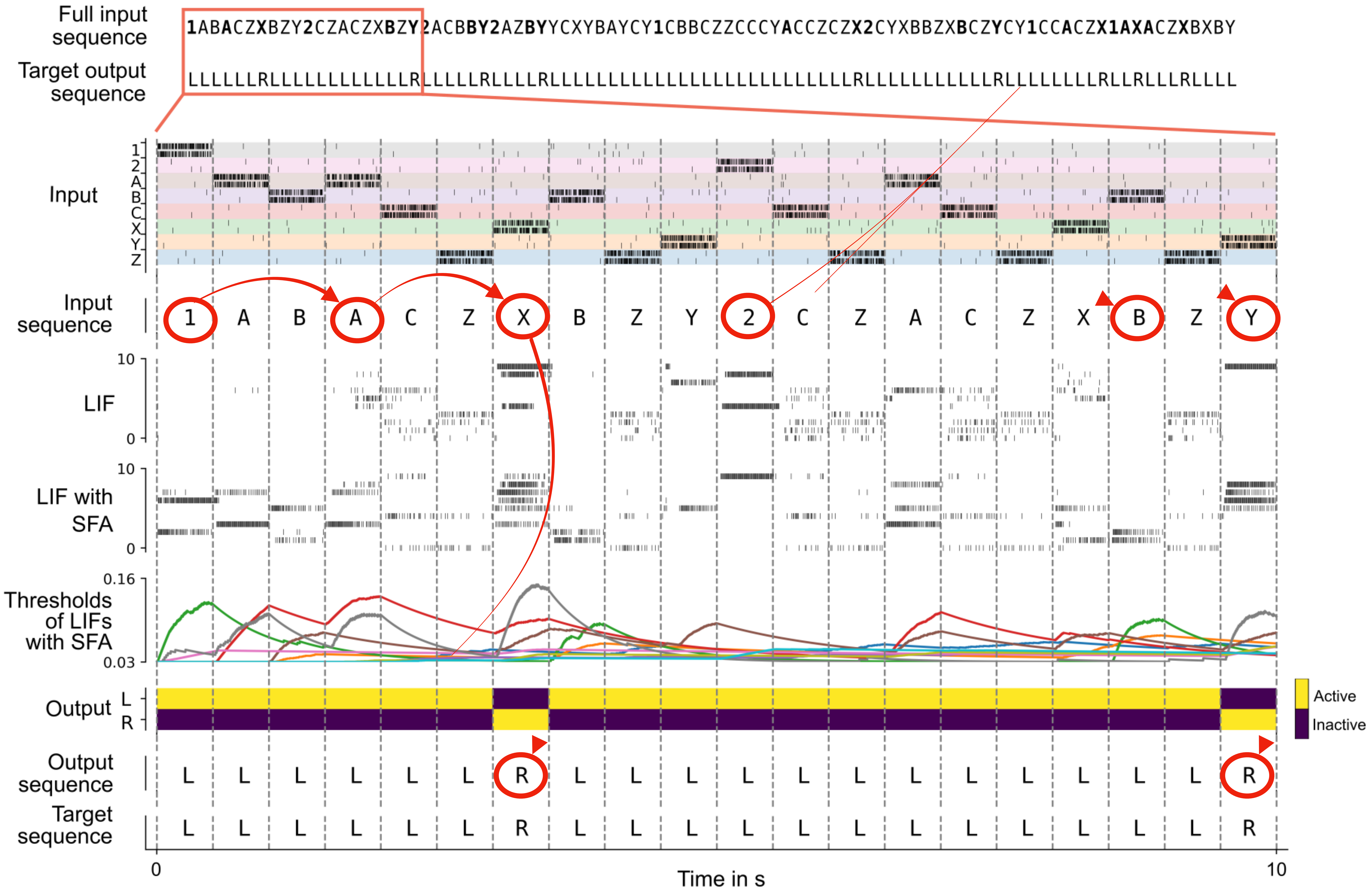


[8] O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*.

SNNs with SFA learn the 12AX task

Performance: 97.79% of test trials with all correct outputs L and R.

No need for hierarchical WM, as e.g., in [8].

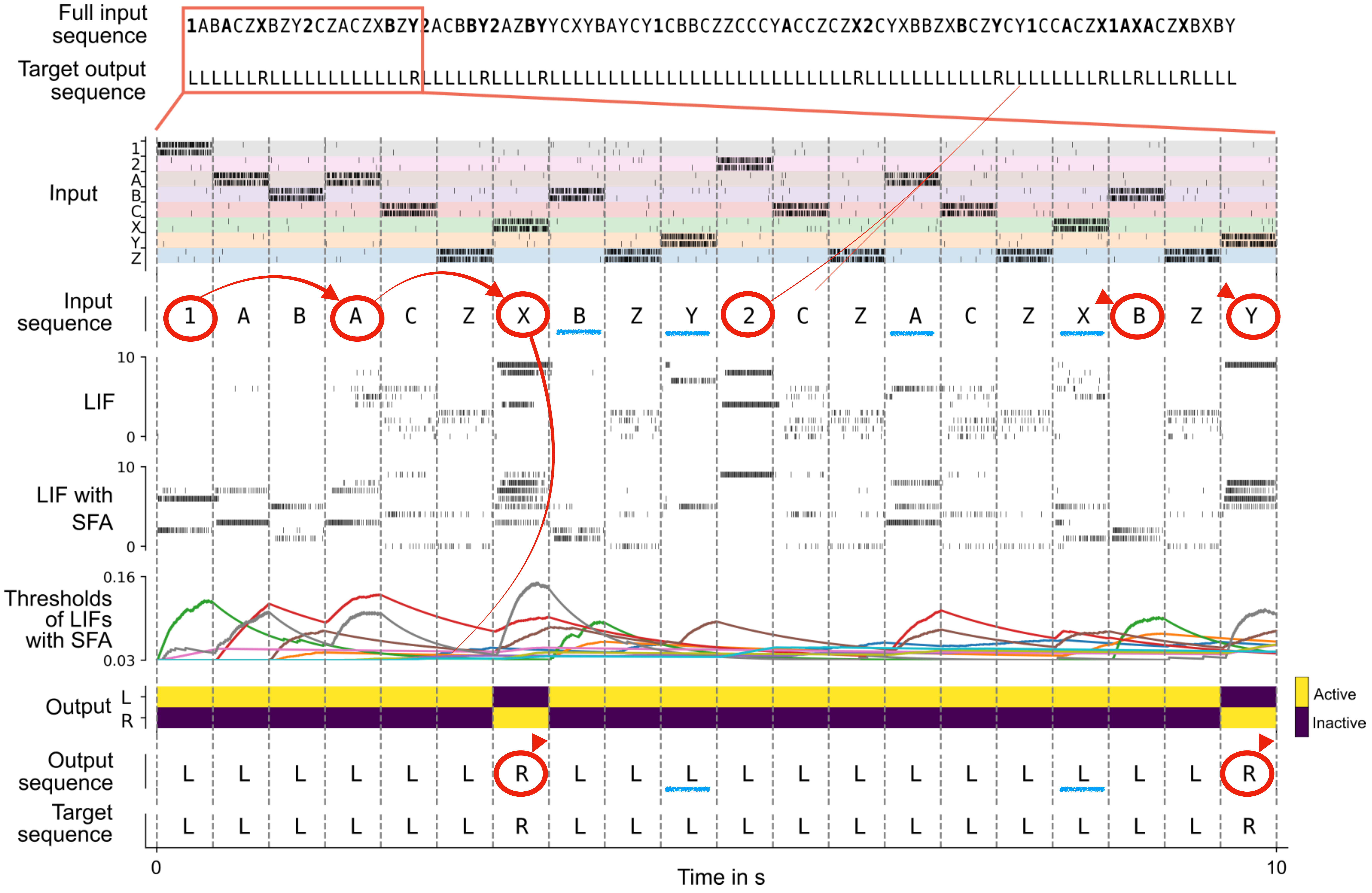


[8] O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*.

SNNs with SFA learn the 12AX task

Performance: 97.79% of test trials with all correct outputs L and R.

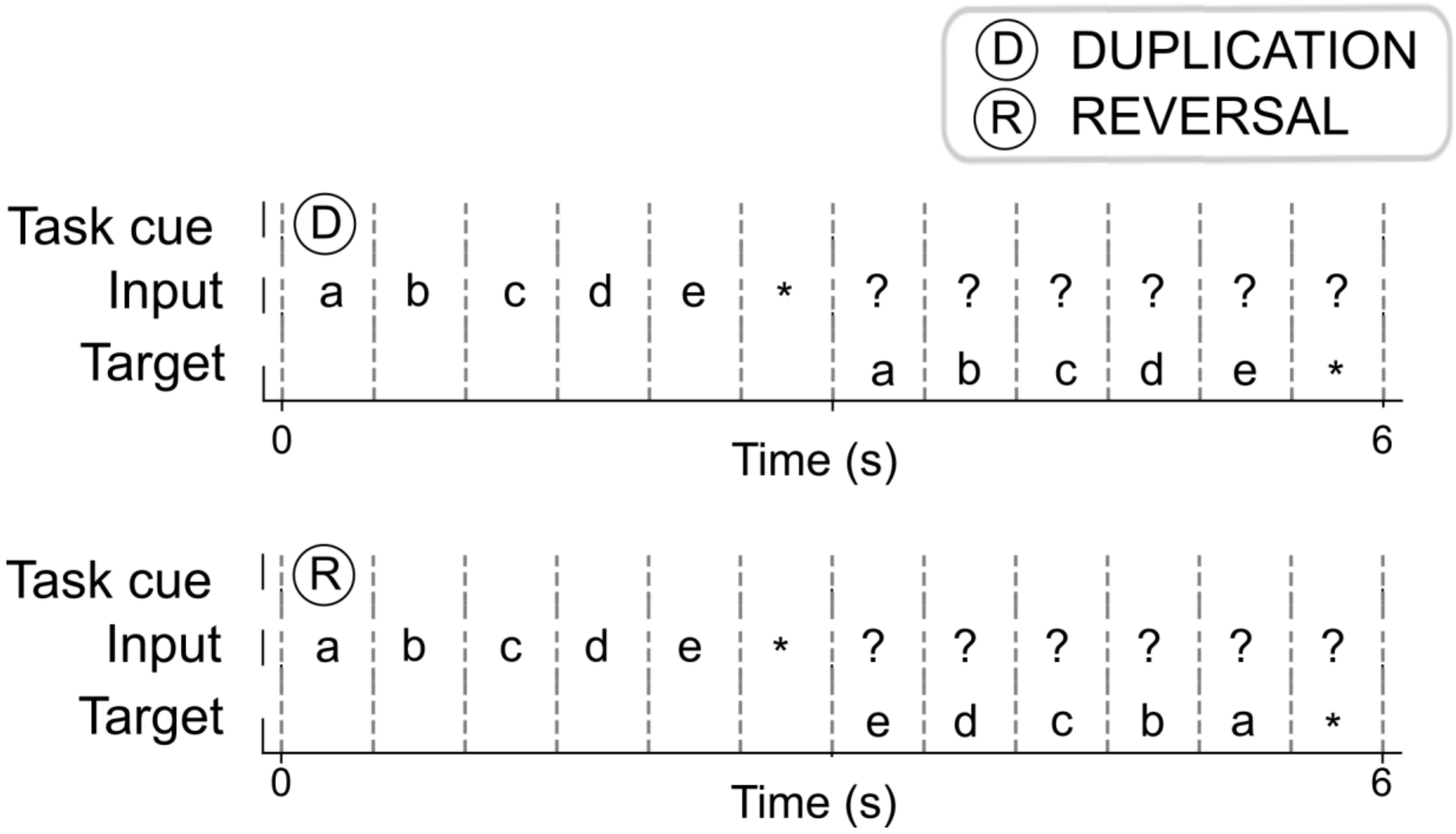
No need for hierarchical WM, as e.g., in [8].

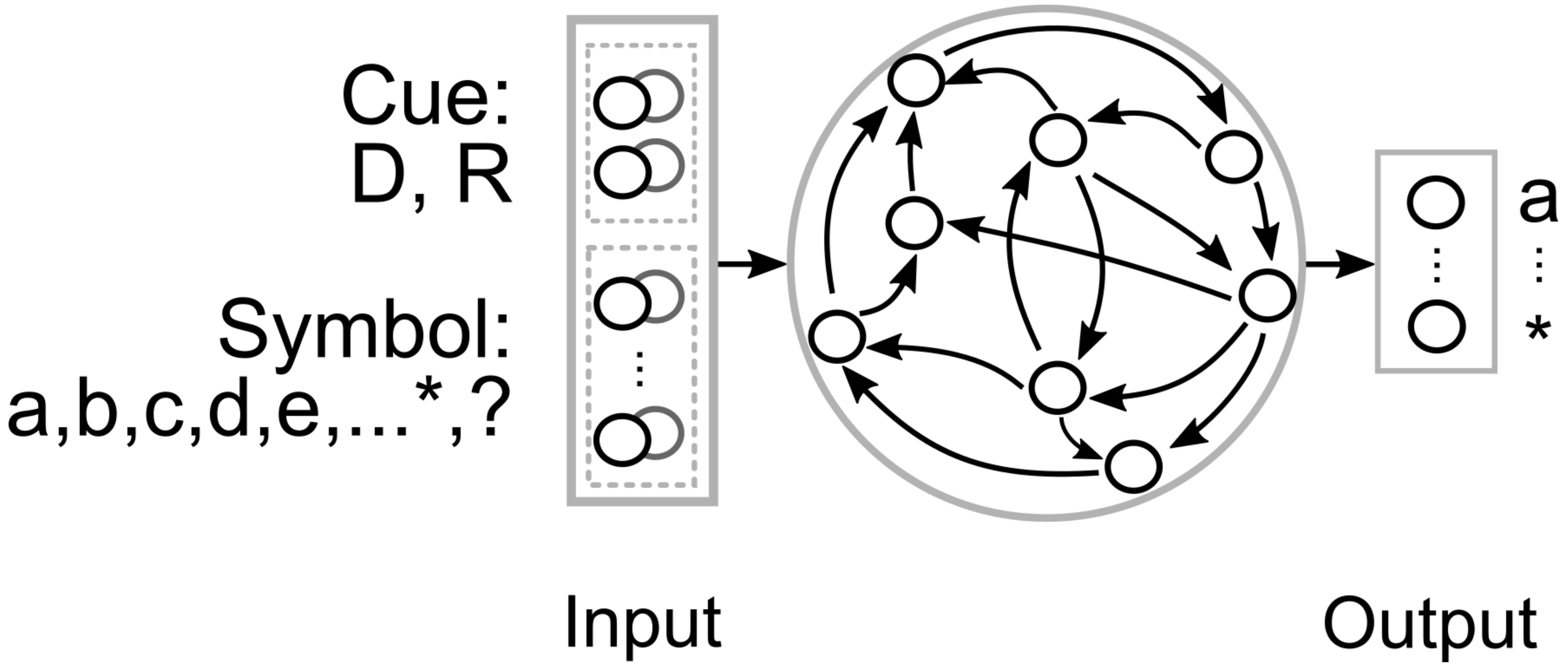


[8] O'Reilly, R. C., & Frank, M. J. (2006). Making working memory work: a computational model of learning in the prefrontal cortex and basal ganglia. *Neural computation*.

SNNs with SFA learn complex operations on sequences of symbols

Sequence manipulation tasks [9, 10]





Performance: 95.88% of test sequences (unseen during training) fully correct.

[9] Marcus, G. F. (2003). The algebraic mind: Integrating connectionism and cognitive science. *MIT press.*
[10] Barone, P., & Joseph, J. P. (1989). Prefrontal cortex and spatial sequencing in macaque monkey. *Experimental brain research.*

SNNs with SFA learn complex operations on sequences of symbols

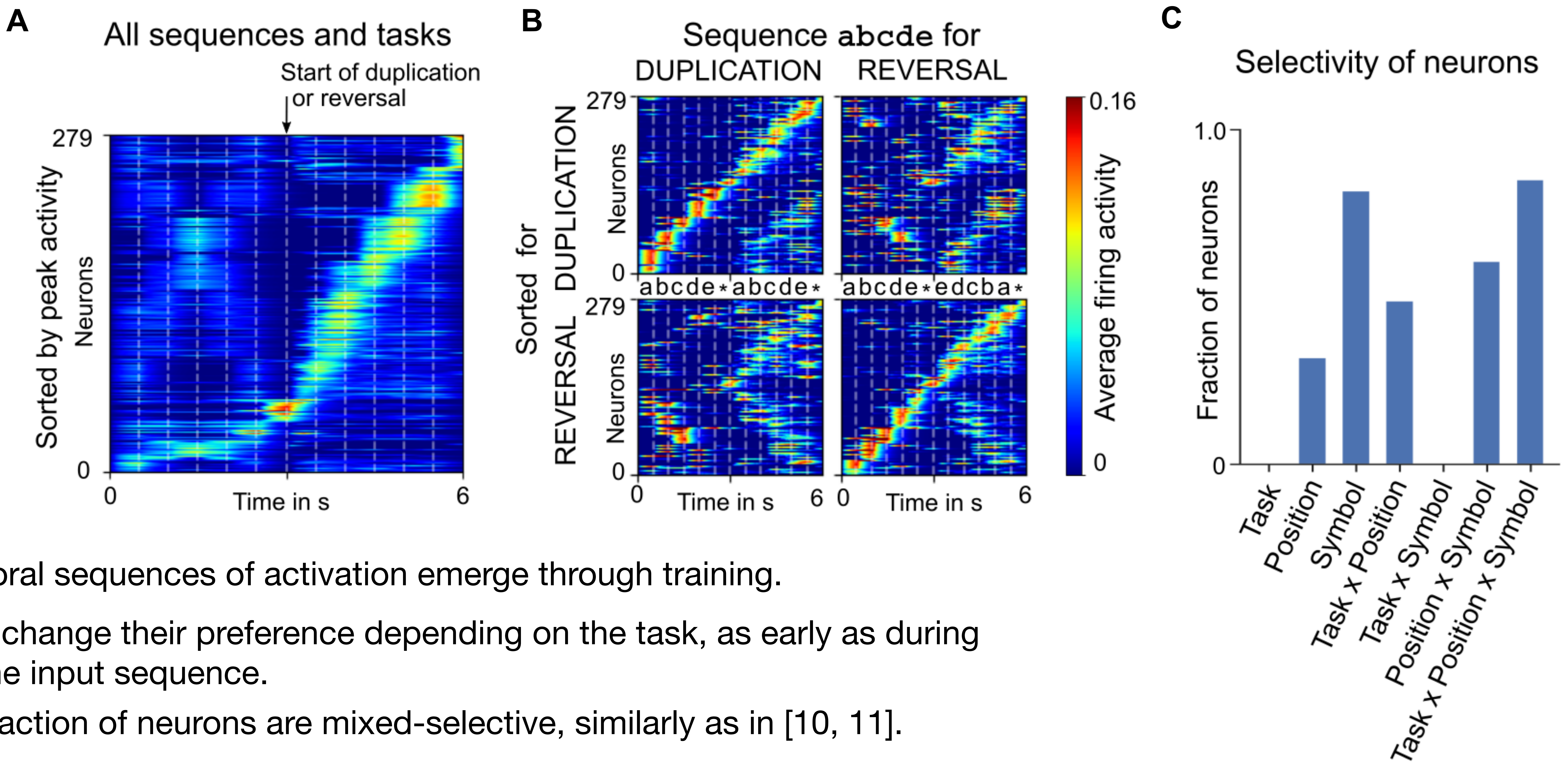


Fig. **A, B**: Temporal sequences of activation emerge through training.

Fig. **B**: Neurons change their preference depending on the task, as early as during the loading of the input sequence.

Fig. **C**: A large fraction of neurons are mixed-selective, similarly as in [10, 11].

[10] Barone, P., & Joseph, J. P. (1989). Prefrontal cortex and spatial sequencing in macaque monkey. *Experimental brain research.*
[11] Rigotti et al., 2013. The importance of mixed selectivity in complex cognitive tasks. *Nature.*

Conclusion

- SFA enables SNNs to integrate information into ongoing network computations (on time scale of seconds)
  - Without SFA mechanisms, SNNs achieved performance of 0%
- A generic rather than specific neural circuitry suffices for implementing a form of hierarchical working memory
- No special-purpose circuits for performing arithmetics (*not covered in this presentation*)
- Implementation of working memory (*not covered in this presentation*)
- Emergent neural codes of a trained SNN resemble neural codes from experiments with monkeys
  - Diversity of neural codes emerged (mixed-selectivity)
  - A possibility for uncovering the computational primitives of the brain

Part 2:
Fault pruning: Robust training of neural networks with memristive weights

Kraišniković, C., Stathopoulos, S., Prodromakis, T., & Legenstein, R. (2023). Fault pruning: Robust training of neural networks with memristive weights. In *International Conference on Unconventional Computation and Natural Computation.*

Memristors [12, 13, 14]

- Analog non-volatile devices (“memory resistors”)
- Resistance R serves as a probed state variable

- **Advantages**
  - Can be integrated with ultra-high density
  - Suited for implementation of matrix-vector multiplications
  - Low-power consumption

- **Challenges**
  - Fabrication, operational constraints
  - Limited endurance of the devices
  - Yield and repeatability issues

[12] Chua, L. (1971). Memristor-the missing circuit element. *IEEE Transactions on circuit theory*.
[13] Strukov et al., 2008. The missing memristor found. N*ature*.
[14] Indiveri et al., 2013. Integration of nanoscale memristor synapses in neuromorphic computing architectures. *Nanotechnology.*

Memristive neural network training

- **Faulty behavior of memristors** [15]
  - Stuck memristors
  - Faulty updates
    - Concordant faults
    - Discordant faults
  - Negative consequences on network performance

- **Our approach**
  - Analyze impact of faulty memristor behavior on neural network training
  - Strategy: use *Fault pruning*

    Fault pruning: Detection of faults during training and pruning of connections on the fly

[15] Xia et al., 2018. Fault-tolerant training enabled by on-line fault detection for RRAM-based neural computing systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Memristive neural network training

- Mapping resistance $R_i \in [R_{\text{min}}, R_{\text{max}}]$ to weight $w_i \in [w_{\text{min}}, w_{\text{max}}]$:

$$w_i = \alpha\left(\frac{1}{R_i} - \frac{1}{R_{\text{C}}}\right)$$

- Inverse mapping from weight to resistance:

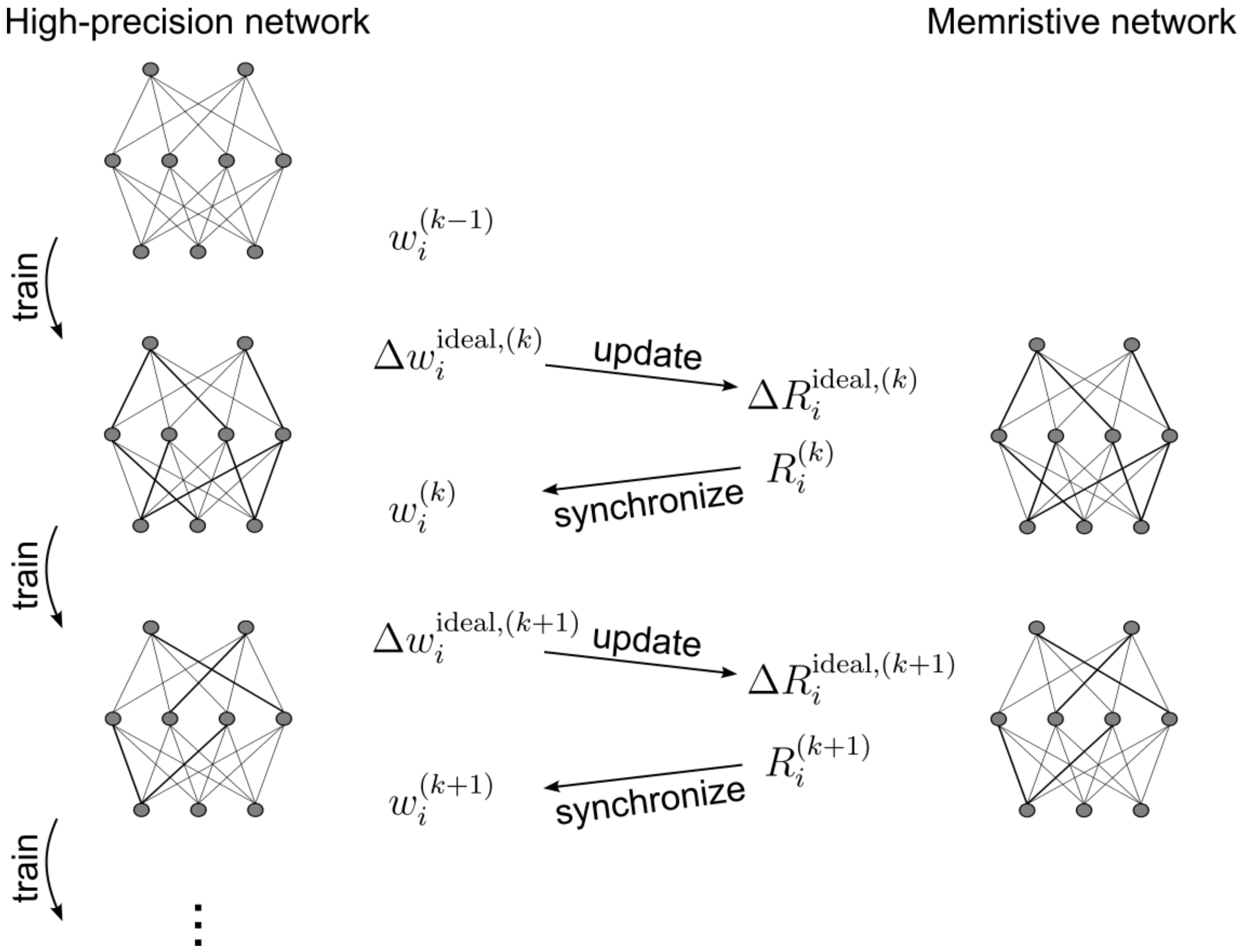
$$R_i = \frac{1}{\frac{1}{R_{\text{C}}} + \frac{w_i}{\alpha}}$$

- Weight and resistance updates:

$$\Delta w_i = w_i^{(k)} - w_i^{(k-1)}$$

$$\Delta R_i = R_i^{(k)} - R_i^{(k-1)}$$

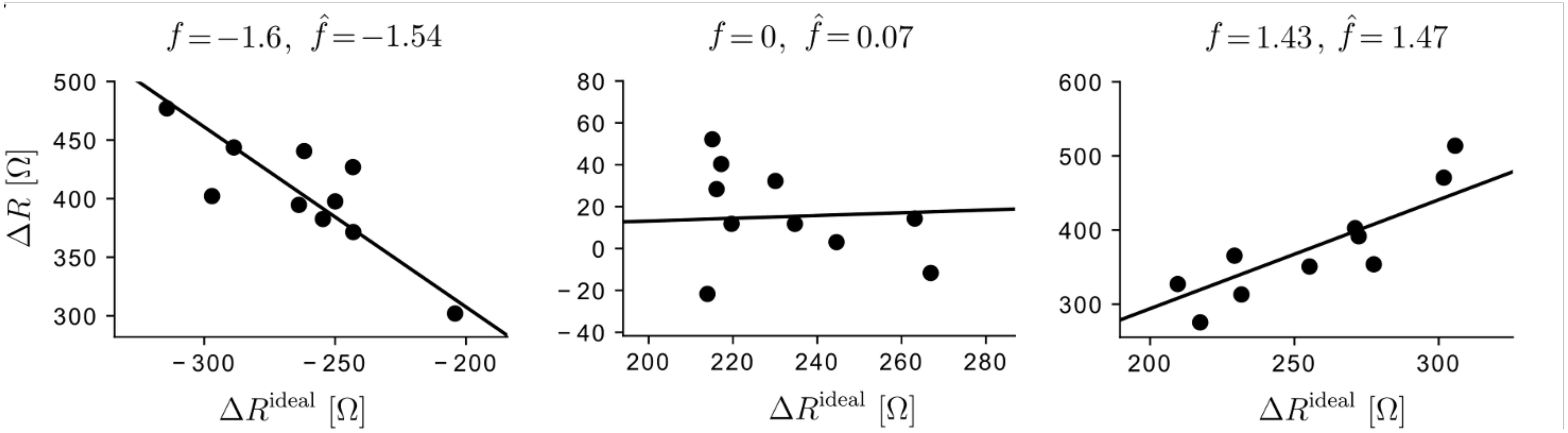
In-the-loop training
[16, 17]



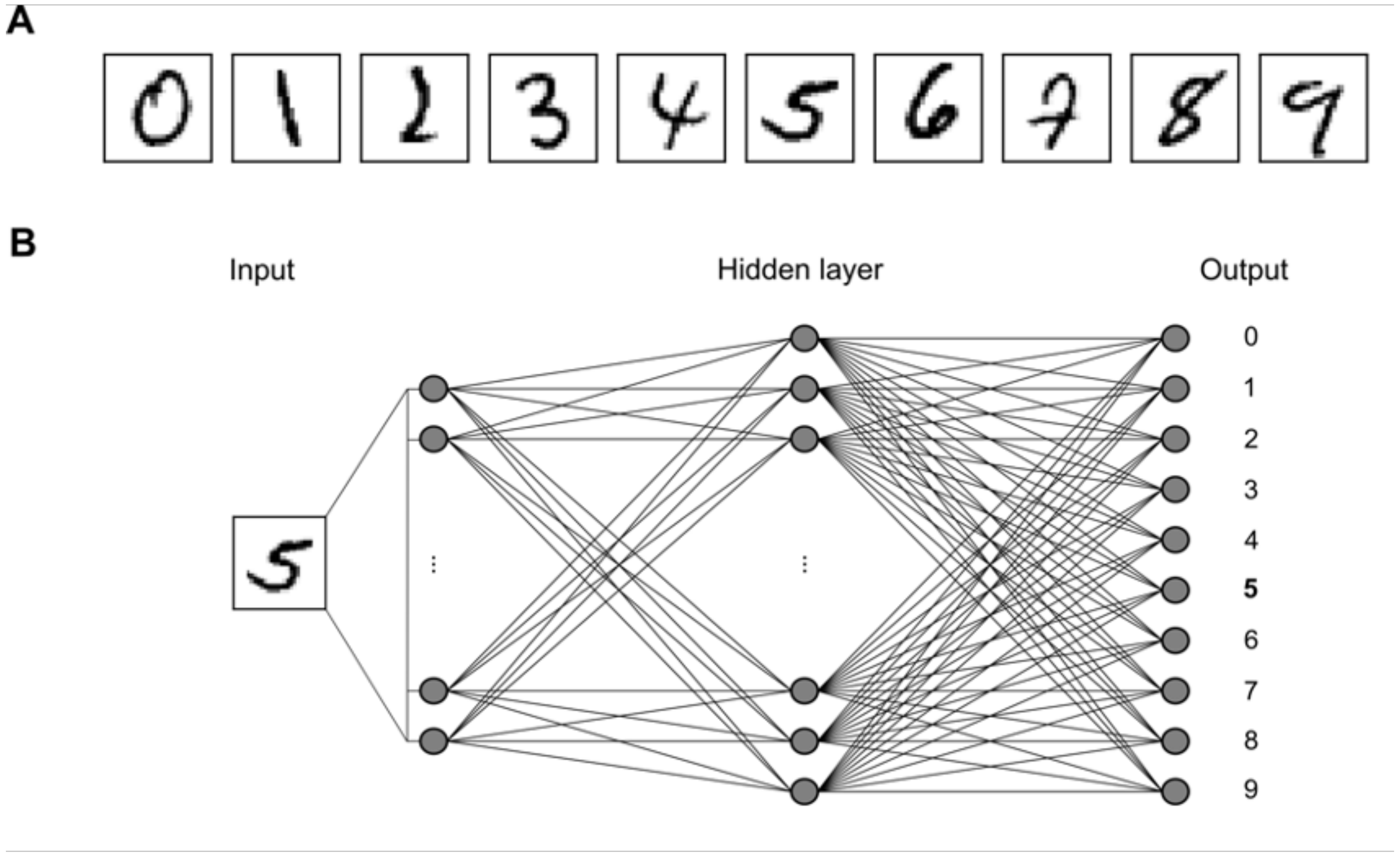
[16] Schmitt et al., 2017. Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system. *IJCNN.*
[17] Woźniak et al., 2020. Deep learning incorporating biologically inspired neural dynamics and in-memory computing. *Nature Machine Intelligence*.

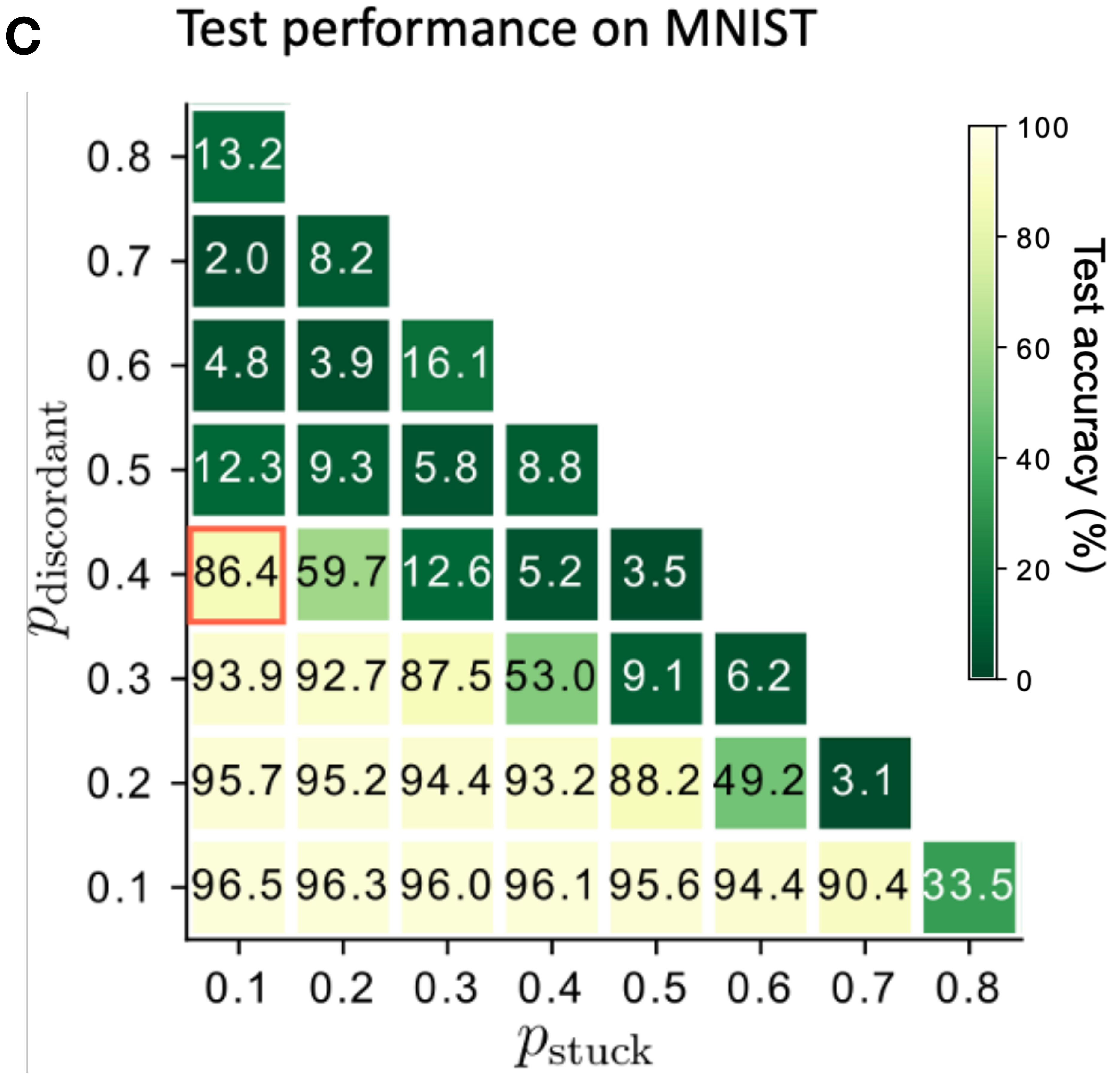
Model of imperfect memristor

- Memristor faults modeled by fault factor f_i
  - Modulates resistance change: $\Delta R_i^{(k)} = f_i \cdot \Delta R_i^{\text{ideal},(k)} + \eta_i^{(k)}$
  - Stuck memristors: $f_i = 0$
  - Concordant faults: $f_i > 0$
  - Discordant faults: $f_i < 0$



The MNIST task





Discordant memristive changes are detrimental.

Neural networks can be pruned significantly and achieve little loss in accuracy [18, 19], hence we asked if one can prune faulty memristive connections.

[18] Bellec et al., 2017. Deep rewiring: training very sparse deep networks. In *International Conference on Learning Representations (ICLR).*
[19] Liu et al., 2019. Rethinking the value of network pruning. In *International Conference on Learning Representations (ICLR).*

Fault pruning algorithm

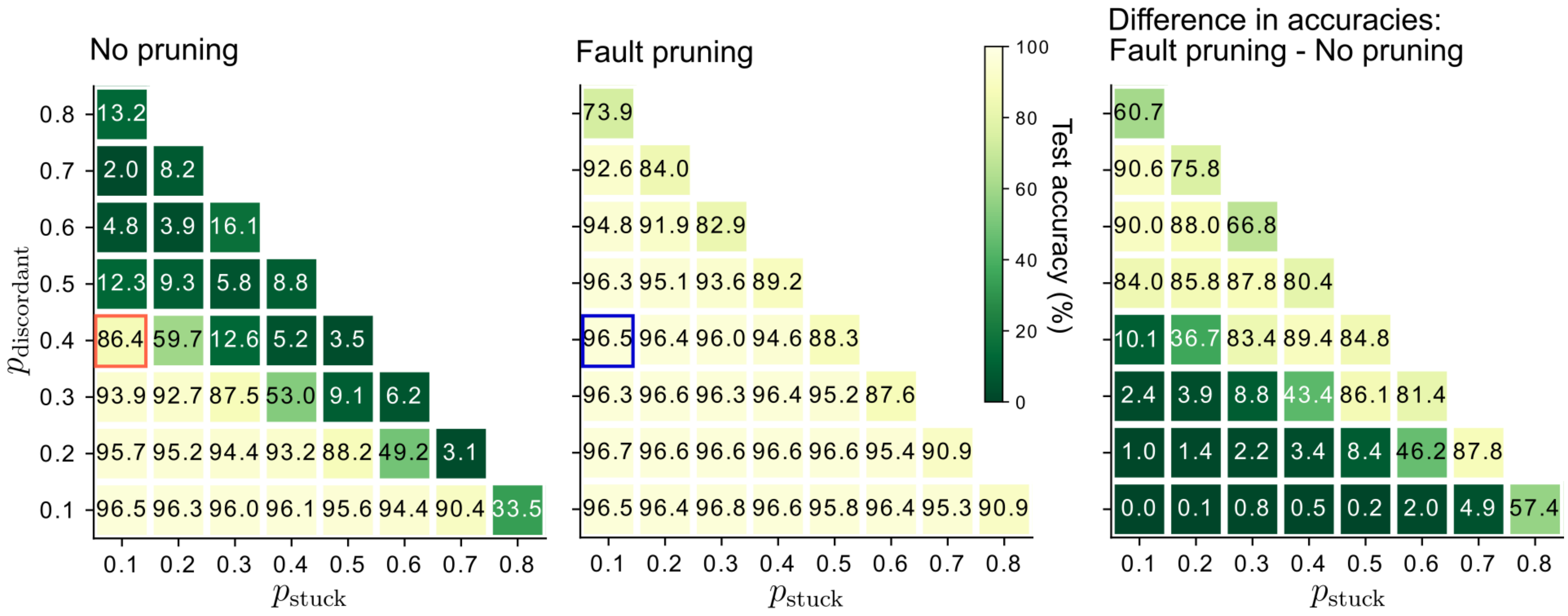
- Estimate fault factor over a window of previous updates:

$$\hat{f}_i = \frac{\sum_l \Delta R_i^{\mathrm{ideal},(l)} \Delta R_i^{(l)}}{\sum_l \left(\Delta R_i^{\mathrm{ideal},(l)}\right)^2}$$

- Remove detected unreliable memristors from the network if $\hat{f}_i < \theta$, and we set $\theta = 0.1$

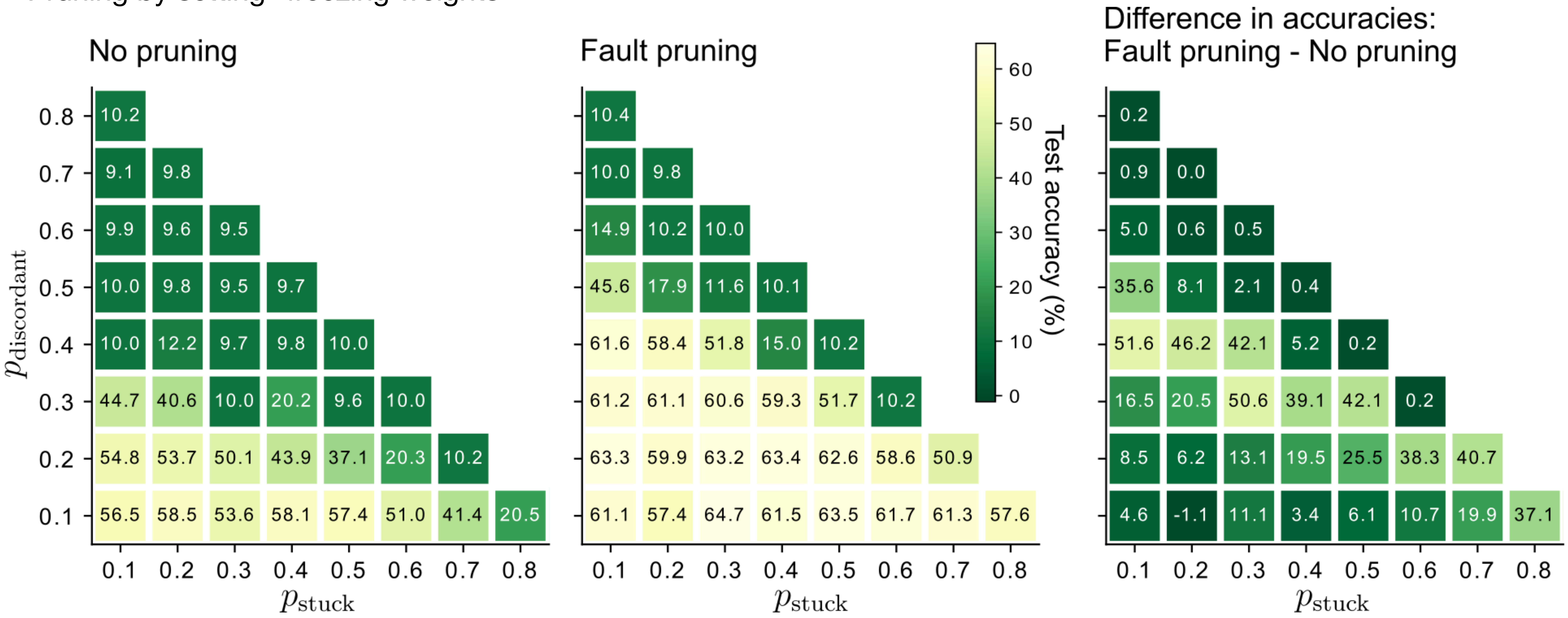
- Two variants of the algorithm
  - Variant 1: Prune faulty weights (set to zero)
  - Variant 2: Don’t update faulty weights (keep last weight)

Results on MNIST



Results on CIFAR-10

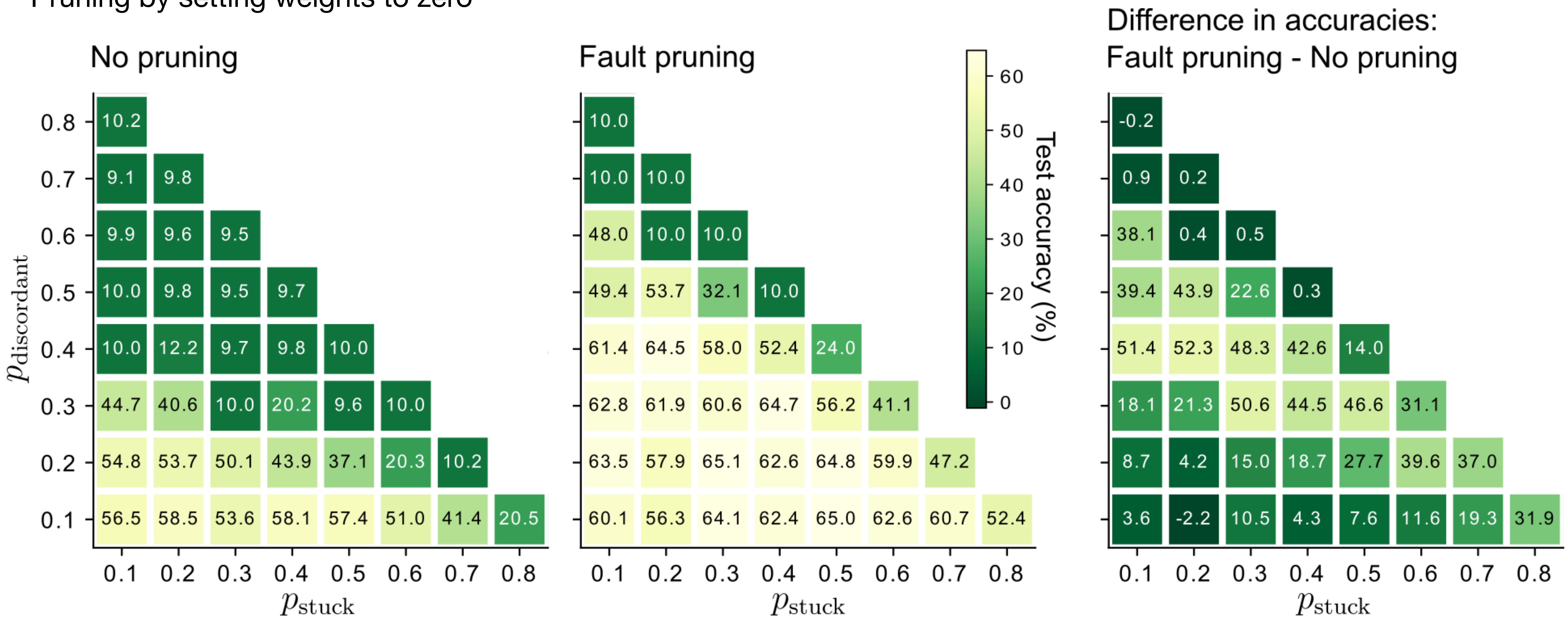
- Convolutional Neural Network (CNN), LeNet [20]
- Pruning by setting “freezing weights”



[20] LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks.*

Results on CIFAR-10

- Convolutional Neural Network (CNN), LeNet [20]
- Pruning by setting weights to zero



[20] LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks.*

Conclusion

- Fault pruning managed to preserve very good performance
- General approach, independent of the network structure and trained tasks
- Estimation of faults on the fly, and acting accordingly
- A simple linear regression for estimation of faults
    - Can be substituted by more advanced approaches

Summary and contributions of the thesis

Spike-based models for cognitive computations and robust training of memristive neural networks

- Spike-based models can solve complex cognitive tasks
- Such models can then be used to ask questions about the brain

- Robust training will likely be crucial as integration density of memristive crossbars increases

- **PhD thesis contribution**: a (preliminary) step towards the implementation of (cognitive) brain-like neural networks *in silico*

Thank you for listening! Questions?

Cognitive processing in the brain

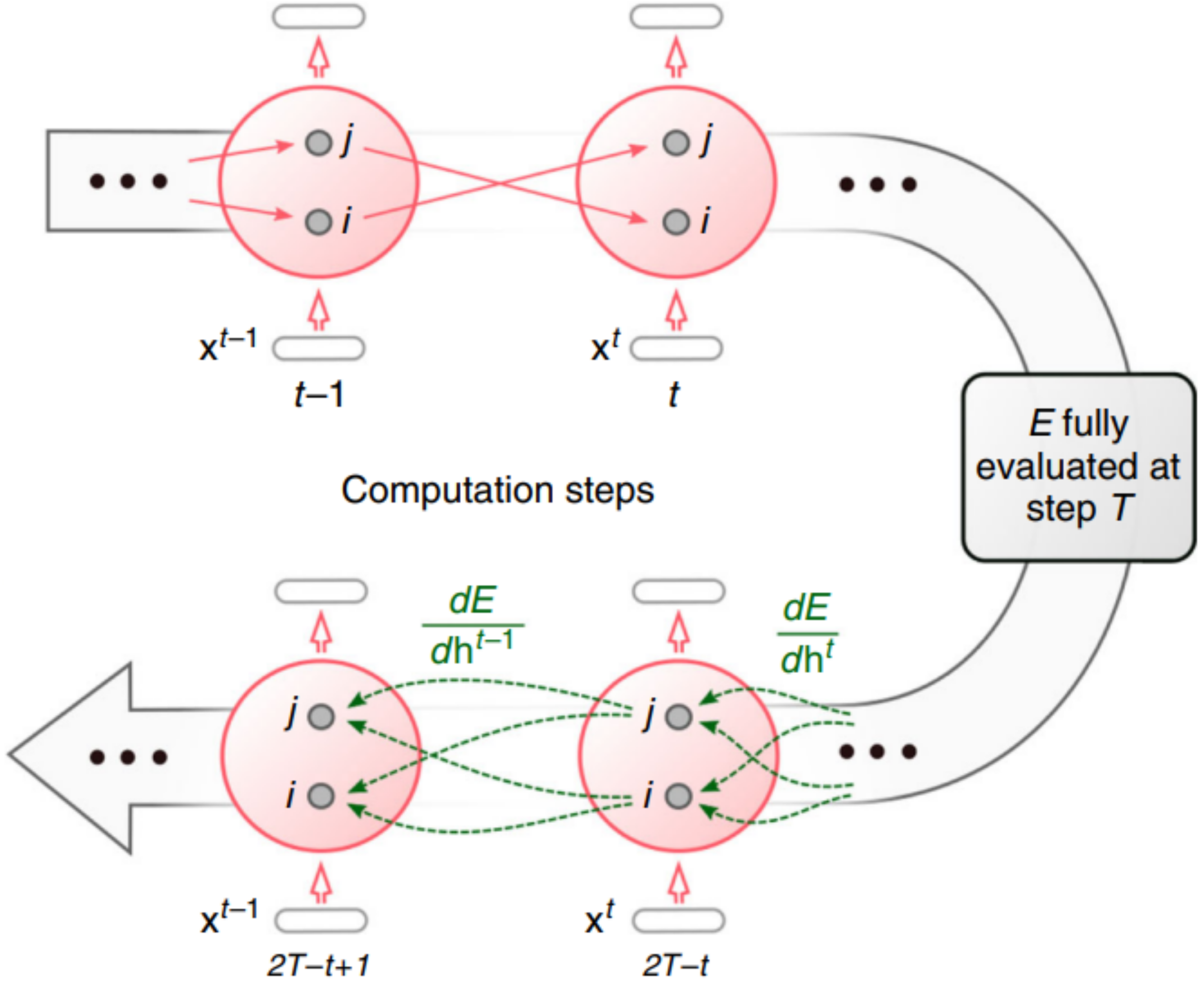
- **Working memory**
  - Supported by the prefrontal cortex

  - Information encoding, maintenance and manipulation
  - Dynamic and flexible

  - Context-dependent processing

Training algorithms

BPTT



e-prop

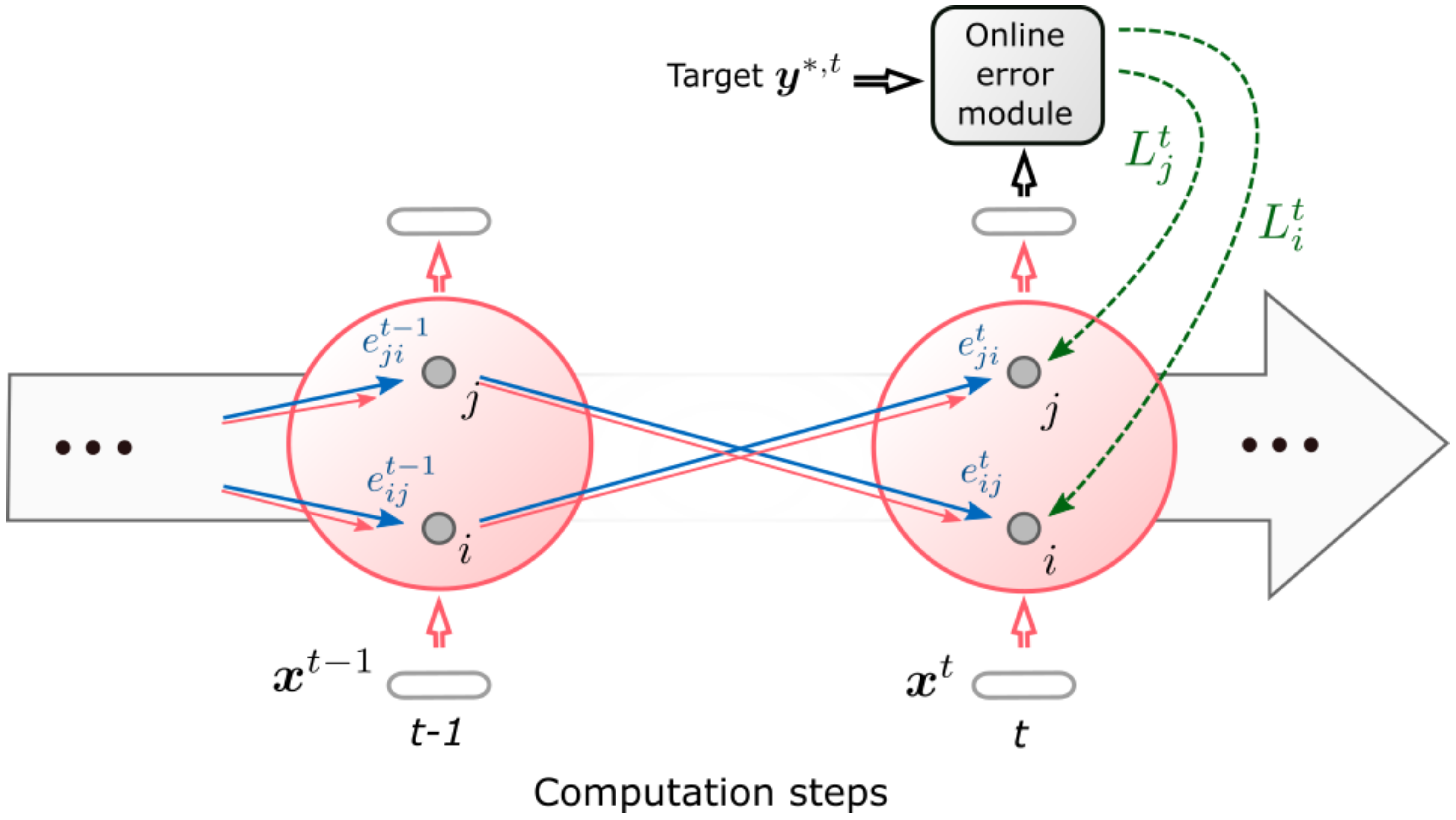


Image source: Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (2020). A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature communications*, *11*(1), 3625.

LIF with SFA neuron model

LIF neuron dynamics:

$$\tau_m \dot{V}_j(t) = -V_j(t) + R_m I_j(t)$$

Adaptive threshold:

$$A_j(t) = v_{\text{th}} + \beta a_j(t),$$

$$a_j(t+1) = \rho_j a_j(t) + (1-\rho_j) z_j(t),$$

$$\rho_j = \exp\left(\frac{-\delta t}{\tau_{a,j}}\right)$$

In discrete time:

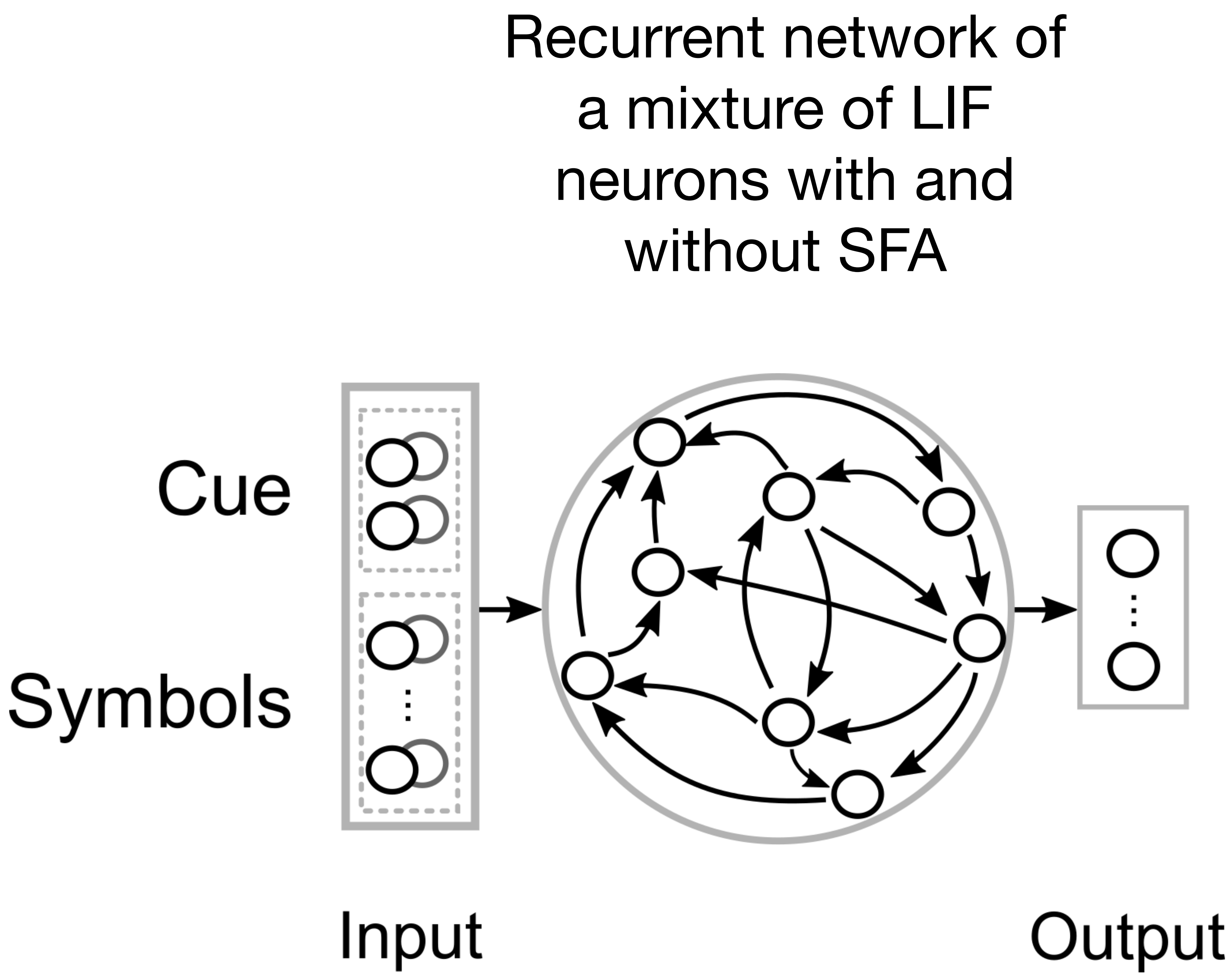
$$V_j(t+\delta t) = \alpha V_j(t) + (1-\alpha) R_m I_j(t) - v_{\text{th}} z_j(t) \delta t,$$

$$\alpha = \exp\left(-\frac{\delta t}{\tau_m}\right),$$

$$z_j(t) = H\left(\frac{V_j(t) - v_{\text{th}}}{v_{\text{th}}}\right)\frac{1}{\delta t},$$ with $H(x) = 0$ if $x < 0$ and 1 otherwise.

$$I_j(t) = \sum_i W_{ji}^{\text{in}} x_i(t - d_{ji}^{\text{in}}) + \sum_i W_{ji}^{\text{rec}} z_i(t - d_{ji}^{\text{rec}})$$

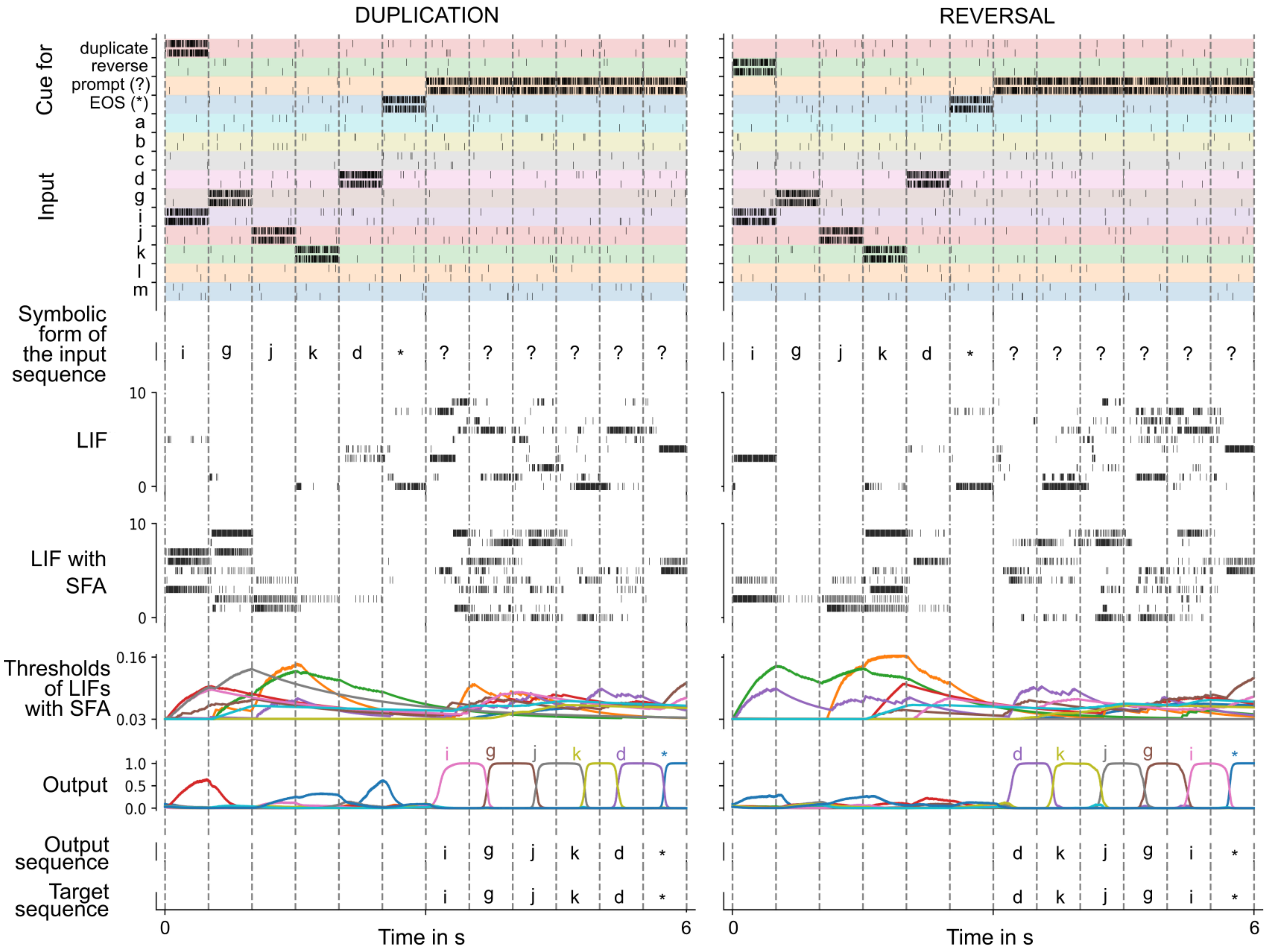
Neural network architecture



Recurrent network of spiking neurons, because:

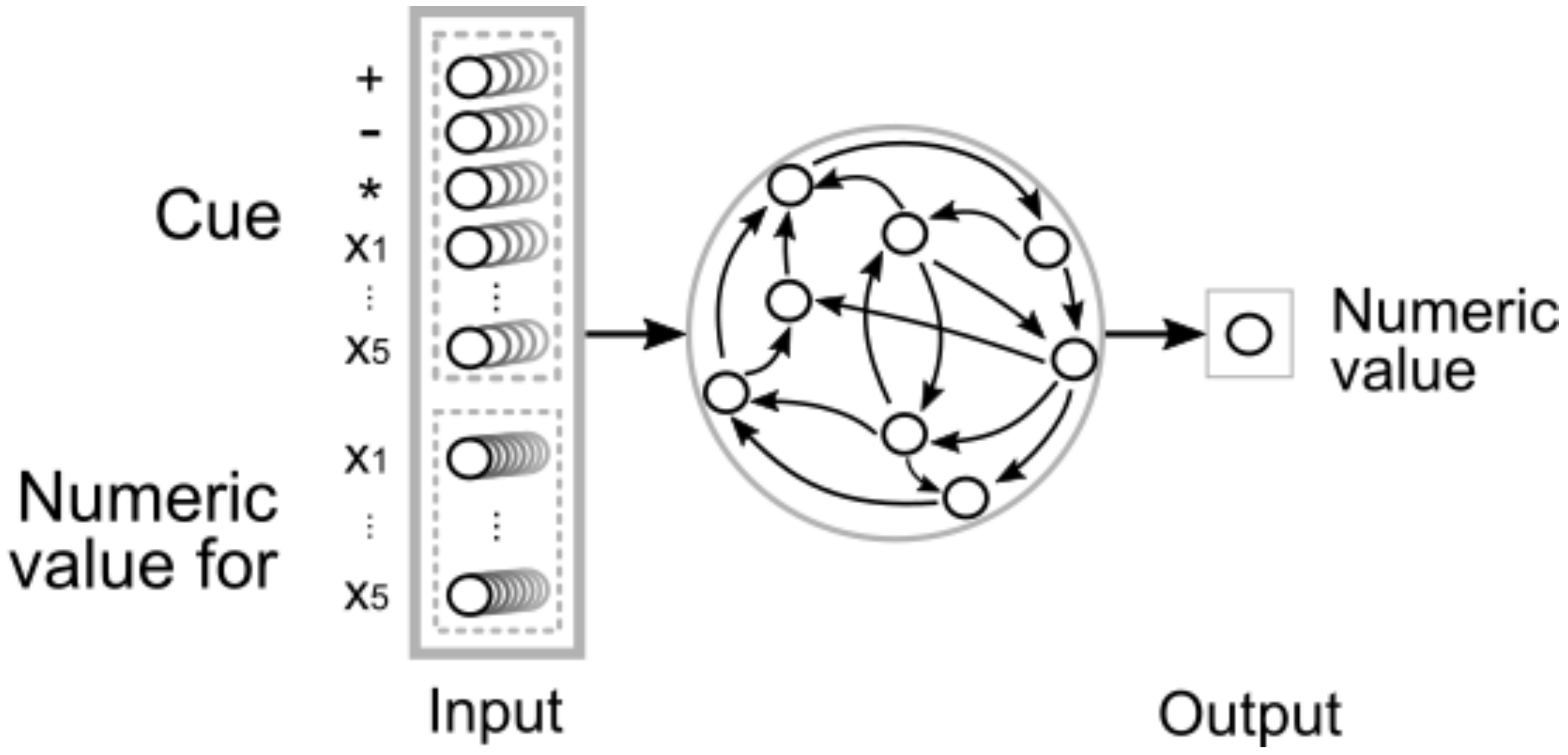
- we are solving temporal computing tasks
- neural networks in the brain are highly recurrent
- spike-based communication of the brain is energy-efficient → investigating what would be feasible to implement in energy-efficient, neuromorphic hardware

Spike Frequency Adaptation supports network computations on temporally dispersed information



SNNs learn to evaluate nested arithmetic expressions

- Variables and numbers encoded into spiking activity
- The network learns the binding “variable - concrete value”, and evaluates given expressions in real time



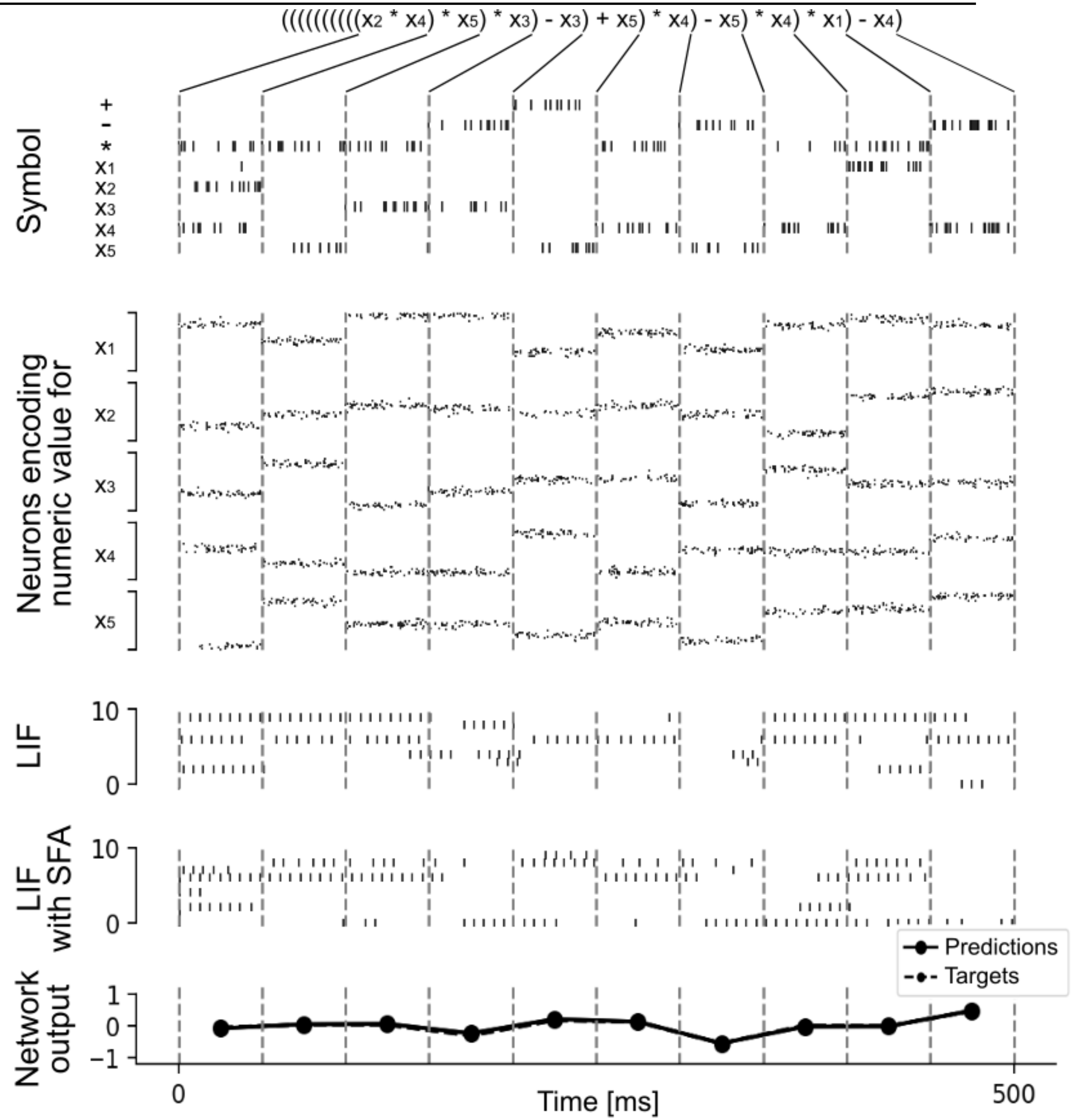
An example trial: $(((((x_2 * x_4) * x_5) * x_3) - x_3) + x_5)$

Input	$x_2 * x_4$ 0.4999*(-0.1113)	$r_1 * x_5$ (-0.0794)*(-0.71)	$r_2 * x_3$ 0.051*0.8209	$r_3 - x_3$ 0.0629-0.3314	$r_4 + x_5$ (-0.2308)+0.4717
Output	$= r_1$ -0.0794	$= r_2$ 0.051	$= r_3$ 0.0629	$= r_4$ -0.2308	$= r_5$ 0.2096
Target	-0.0553	0.0564	0.0419	-0.2685	0.2409

0 Time [ms] 250

SNNs learn to evaluate nested arithmetic expressions

- Variables and numbers encoded into spiking activity
- The network learns the binding “variable - concrete value”, and evaluates given expressions in real time



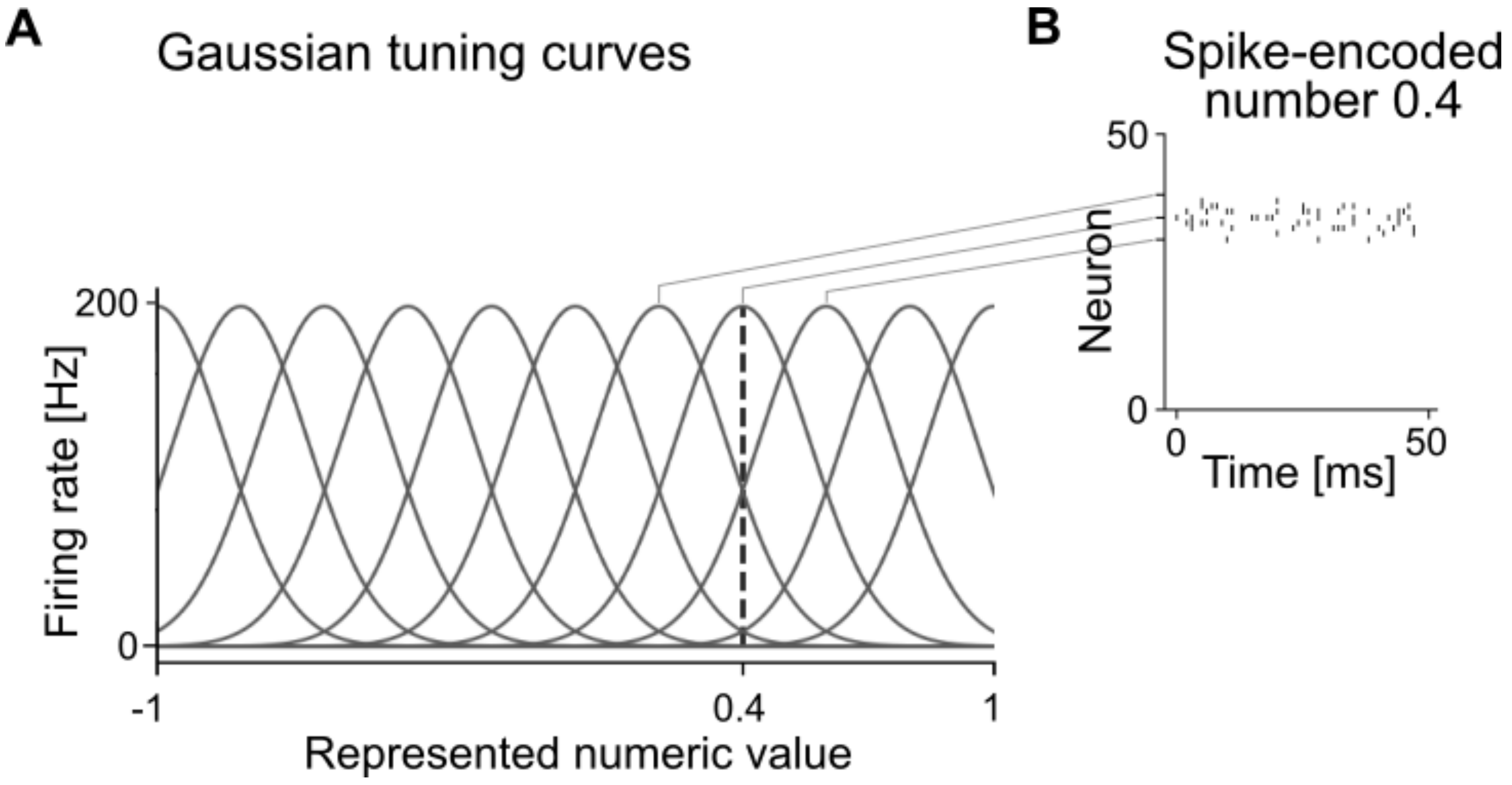
Performance:
MSE: 0.0341, MAE: 0.1307

SNNs learn to evaluate nested arithmetic expressions

Step i	Arith. expression	Numeric expression	Output r_i	Target	Absolute error
1.	$x_2 * x_4$	0.4966 * (-0.1113)	-0.0794	-0.0553	0.0241
2.	$r_1 * x_5$	(-0.0794) * (-0.7100)	0.0510	0.0564	0.0054
3.	$r_2 * x_3$	0.0510 * 0.8209	0.0629	0.0419	0.0210
4.	$r_3 - x_3$	0.0629 - 0.3314	-0.2308	-0.2685	0.0377
5.	$r_4 + x_5$	(-0.2308) + 0.4717	0.2096	0.2409	0.0313
6.	$r_5 * x_4$	0.2096 * 0.7014	0.1304	0.1470	0.0166
7.	$r_6 - x_5$	0.1304 - 0.6650	-0.5417	-0.5346	0.0071
8.	$r_7 * x_4$	(-0.5417) * (-0.0165)	-0.0222	0.0089	0.0311
9.	$r_8 * x_1$	(-0.0222) * (-0.8182)	-0.0128	0.0182	0.0310
10.	$r_9 - x_4$	(-0.0128) - (-0.4727)	0.4739	0.4599	0.0140

Table 1. Evaluation of nested arithmetic expressions with an SNN.

SNNs learn to evaluate nested arithmetic expressions



50 input neurons encode uniformly distributed numeric values from [-1, 1]. They fire with a particular mean on that analog value, and std of 0.08:

$$f_i = f_{\text{max}} \exp\left(-\frac{(m_i - z)^2}{2\sigma^2} \right),$$

$f_{\text{max}} = 200$ Hz, m_i the value for which the neuron is responsible, and z_i a value from the input domain.

Motivation

- **Artificial Intelligence (AI):** large amounts of data processed, demands on computing speed and efficiency
- **Neuro-inspired chips**
  - Main features: neuron-synapse structure, in-memory computation, learning capabilities
  - Information is stored in the form of synaptic weights
  - Synaptic plasticity: ability to increase or decrease synaptic weights by means of changes in conductance

Motivation

- **Key metrics for performance evaluation**
  - Computing density
  - Energy-efficiency
  - Computing accuracy: influenced by non-idealities of devices
  - Learning capabilities: off-chip, on-chip, hybrid
- **Our focus**
  - Resistive Random Access Memory devices (RRAM, “**memristors**”)
  - Improving energy-efficiency
  - Learning “in-the-loop”
    - Robust training of neural networks with memristive weights
    - Detection of faulty memristors
    - Improving computing accuracy

Estimation of the fault factors $\hat{f}_i$

$$\Delta R_i = \hat{f}_i \cdot \Delta R_i^{\text{ideal}} + \epsilon$$

- Estimated from $N = 10$ data points $(\Delta R_i^{\text{ideal},(l)}, \Delta R_i^{(l)}),\ l \in \{k - N + 1, k - N + 2, \ldots, k - 1, k\}$
- The least-squares estimator of $\hat{f}_i$ minimises the error:

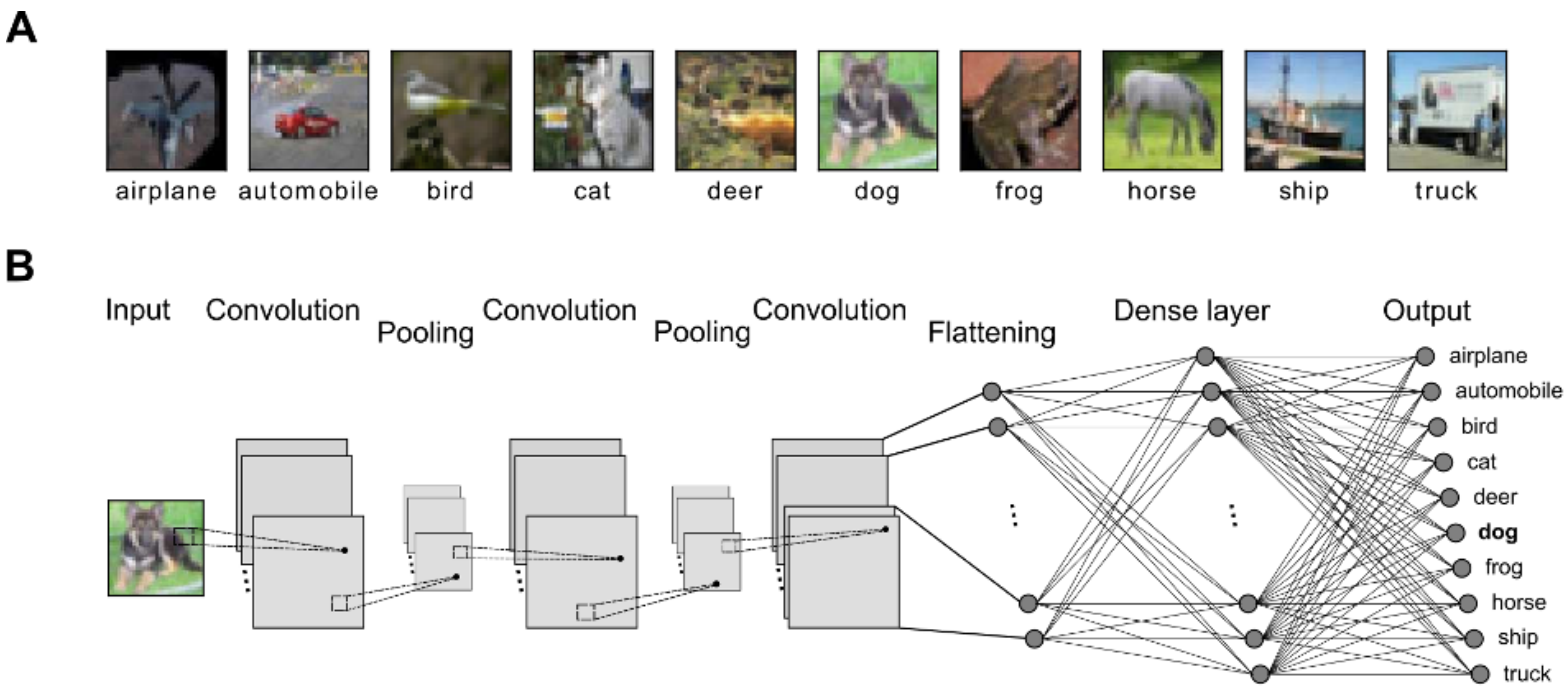
$$\mathscr{L}(\hat{f}_i) := \sum_l \left(\Delta R_i^{(l)} - \hat{f}_i \cdot \Delta R_i^{\text{ideal},(l)}\right)^2,$$

$$\frac{\partial \mathscr{L}}{\partial \hat{f}_i} = 2 \sum_l \left(\Delta R_i^{(l)} - \hat{f}_i \cdot \Delta R_i^{\text{ideal},(l)}\right)\left(-\Delta R_i^{\text{ideal},(l)}\right) \stackrel{!}{=} 0$$

$$\hat{f}_i \sum_l \left(\Delta R_i^{\text{ideal},(l)}\right)^2 = \sum_l \Delta R_i^{(l)} \Delta R_i^{\text{ideal},(l)}$$

$$\hat{f}_i = \frac{\sum_l \Delta R_i^{(l)} \Delta R_i^{\text{ideal},(l)}}{\sum_l \left(\Delta R_i^{\text{ideal},(l)}\right)^2}$$

Results on CIFAR-10



LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.

Connectivity in the network after pruning