Networks of Spiking Neurons Learn to Learn and Remember

(work in progress)

Wolfgang Maass

Institute for Theoretical Computer Science
Graz University of Technology, Austria

Credits

Researchers from our Lab:

Darjan Salaj

Anand Subramoney

Guillaume Bellec

Robert Legenstein

Funding: Human Brain Project of the European Union

Preprint of some of the results:

G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. arXiv:1803.09574, 2018.

Content

- LSNNs: Networks of spiking neurons with long short-term memory
- Computing abilities of LSNNs
- Basic setup of Learning-to-Learn (L2L)
- Learning to learn from rewards
- Learning to learn from a teacher
- Summary

LSNNs: Networks of spiking neurons with long short-term memory

Why is it difficult to produce recurrent neural networks with brain-like components and network dynamics that exhibit powerful computing capabilities?

Research in Machine Learning has shown that **artificial** recurrent neural networks can be computationally very powerful, see e.g.

- *Graves,..., Hinton. Speech recognition with deep recurrent neural networks. IEEE conference 2013*
- *Wu, ..., Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. 2016*

But their ANNs employ special modules for working memory:

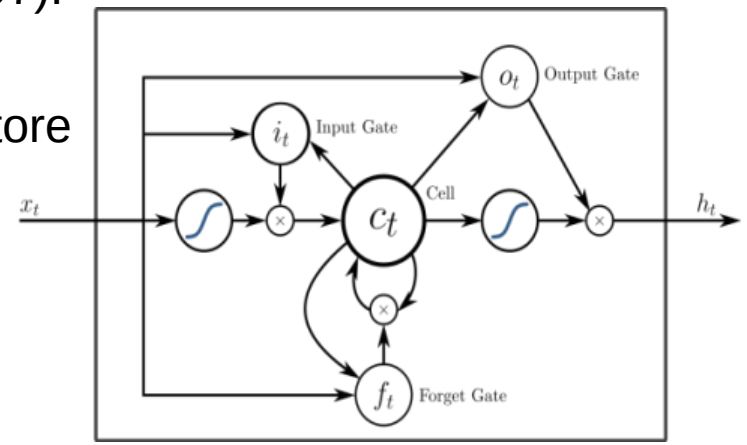
LSTM (Long Short-Term Memory) units.

In contrast, it has been quite hard to demonstrate interesting computational properties of recurrently connected networks of spiking neurons (SNNs) .

LSTM units

Introduced by (Hochreiter and Schmidhuber, 1997).

LSTM units endow ANNs with the capability to store and update information, without disturbance by the inherent dynamics of a recurrent neural network.

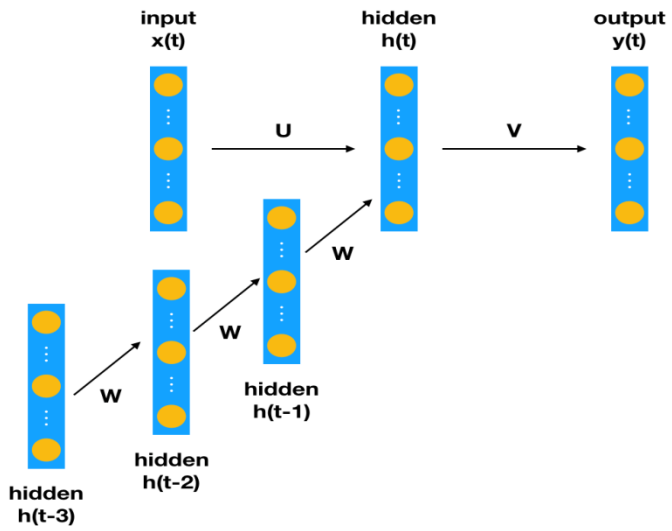


The value c_t in the memory cell of an LSTM units remains unchanged (i.e., the identity function is applied to its content), unless the input gate is opened to let new values flow into it.

Think of adding registers from a digital computer to a neural network, but with trained (rather than programmed) rules for writing into the register or reading from it.

LSTM units support backpropagation through time (BPTT) for training a recurrent network

When one applies BPTT to a recurrent neural network, one unrolls the network of hidden units into a deep feedforward network: the network of hidden units is cloned for each time step s, and all its recurrent connections are replaced by feedforward connections from the clone for step s into the one for step s+1:



When one computes the derivative of the error at the output, one has to propagate the error back through all these cloned layers.

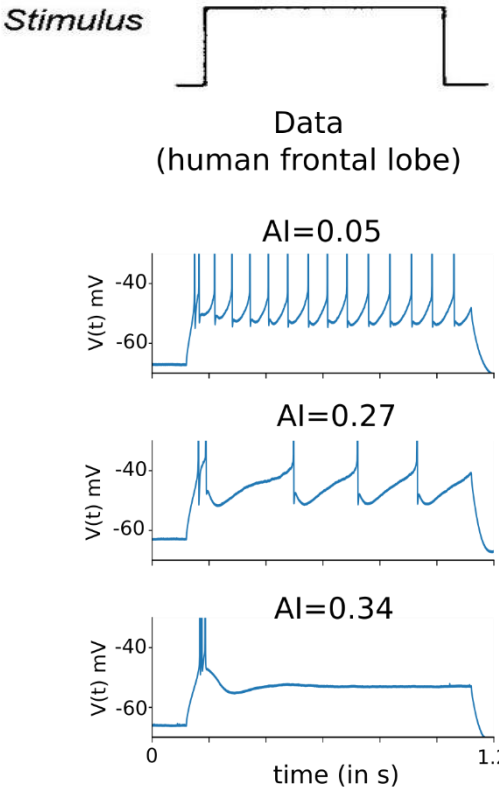
On a backward path of length D, powers of its weights of order D arise.

This leads to exploding or vanishing gradients.

This defect is avoided by a neuron that has a recurrent connection to itself with weight 1 (see the memory cell of an LSTM unit).

Adapting spiking neurons endow SNNs with a similar long short-term memory

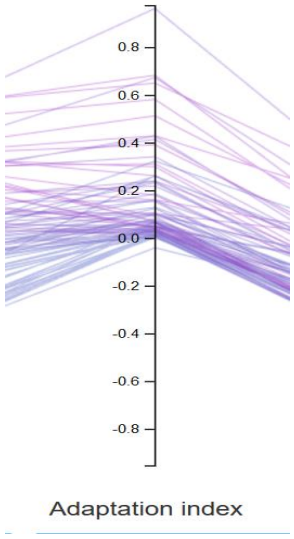
Typical input/output behavior of adapting neurons:



Adaptation index: The rate at which firing speeds up or slows down during a stimulus, defined as:

$$\frac{1}{N-1}\sum_{n=1}^{N-1}\frac{ISI_{n+1}-ISI_n}{ISI_{n+1}+ISI_n}$$

where N is the number of ISIs in the sweep.



Diversity of adapting neurons in the human neocortex according to the Brain Atlas of the Allens Institute

Adapting neurons can easily be modelled by LIF-neurons with an adaptive firing threshold

We assume that the firing threshold B_j of a leaky integrate-and-fire (LIF) neuron j contains a time-varying component b_j(t) that is temporarily increased by each of its spikes $z_j(t)$:

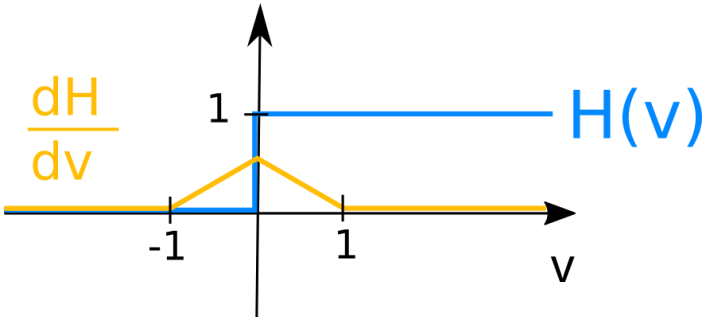
$$B_j(t) = b_j^0 + \beta b_j(t)$$

$$b_j(t+1) = \alpha_j b_j(t) + \big(1 - \alpha_j\big) z_j(t)$$

Backpropagation through time (BPTT) works very well for adaptive spiking neurons

Back propagating gradients through **discontinuous firing events of a spiking neurons:**

- The firing of a neuron is formalized through a binary step function H applied to the scaled membrane voltage v(t) (v crosses the threshold at 0 and resting potential is -1)
- The gradients are propagated through step functions with a **pseudo-derivative** *similarly as in (Courbariaux et al., 2016) (Esser et al., 2016):*



- BPTT propagates well through an adaptive firing threshold, since no spike is involved.

- Furthermore: since adaptive neurons fire less often, they cause fewer artefacts of BPTT.

We arrive in this way at a simple variation of the SNN model: LSNNs

(„L“ indicates the stretch version of the model)

LSNNs contain both adaptive neurons A and regularly firing neurons R (= standard LIF neurons).

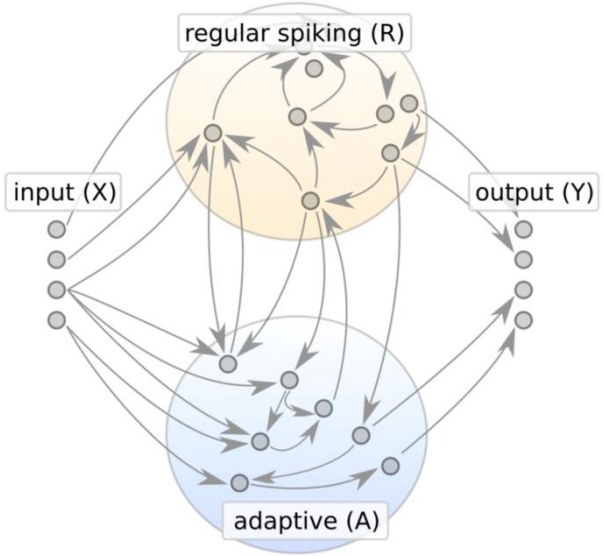
Outputs Y are provided by linear or softmax readout neurons.

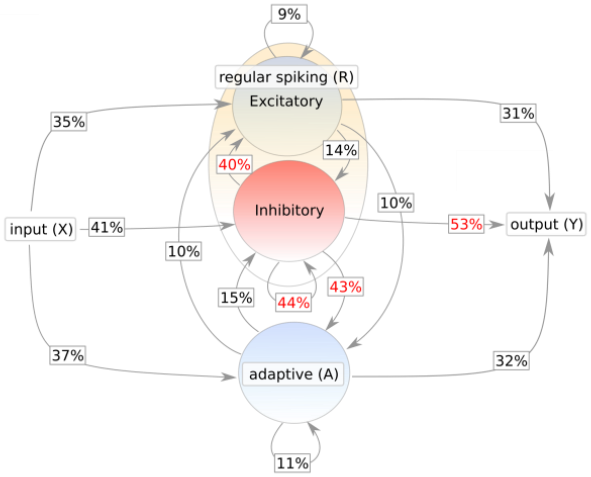
The architecture of LSNNs matters, e.g. complete or random connectivity does not provide the computational power of LSTM units to SNNs

Sparse architectures with larger computational power emerge when one combines weight optimization through BPTT with a biologically inspired rewiring heuristics, as proposed in

Kappel et al, eNeuro 2018, Bellec at al., ICLR 2018

(see preceding talk by Robert Legenstein)

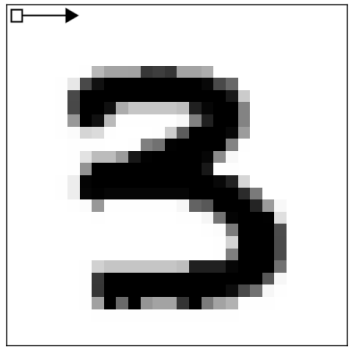




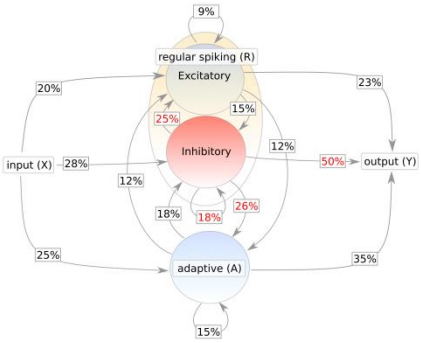
A glimpse at the computational power of LSNNs

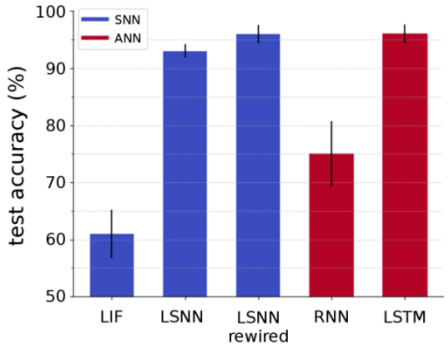
A standard test of temporal integration capability: sequential MNIST

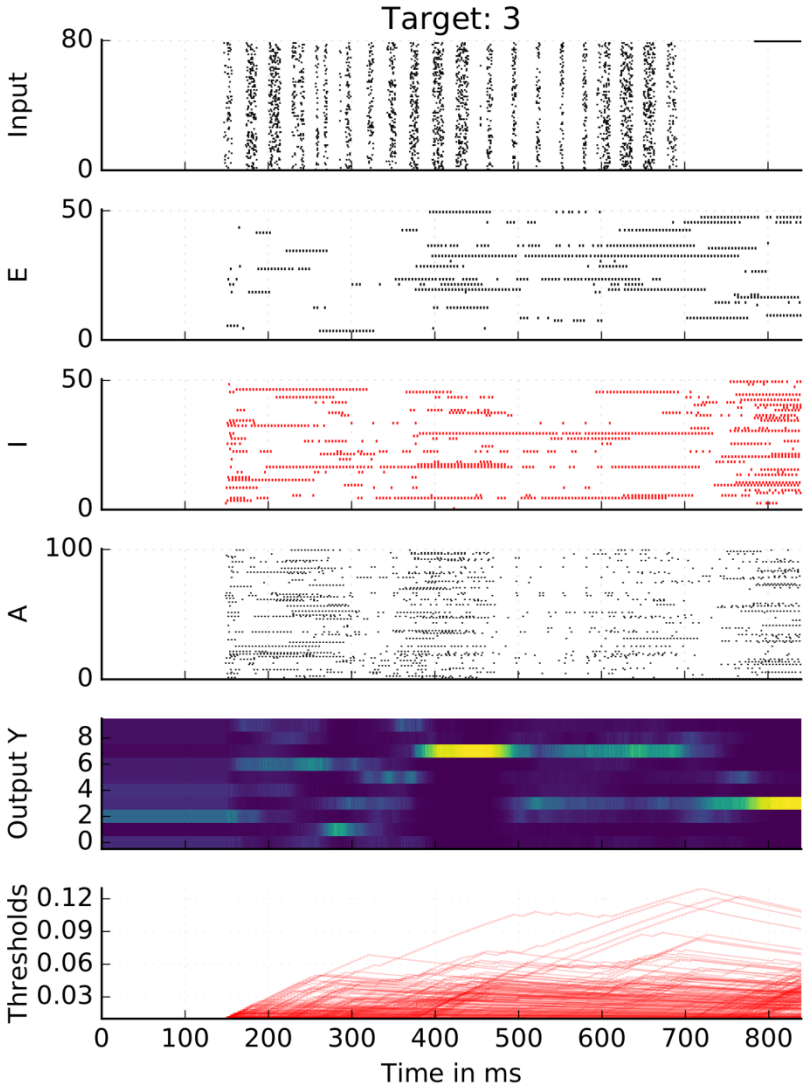
(the pixels of each handwritten digit are presented sequentially, here one pixel every ms)



LSNNs achieve for this task a similar performance as LSTM networks:







Basic setup of Learning-to-Learn (L2L)

Motivation for investigating L2L for SNNs

- Virtually all existing demos of learning in SNNs are arguably biologically unrealistic,since they start with a tabula rasa network, that has no „**innate knowledge**“ from evolution, and no **transfer knowledge** from previously learned tasks.

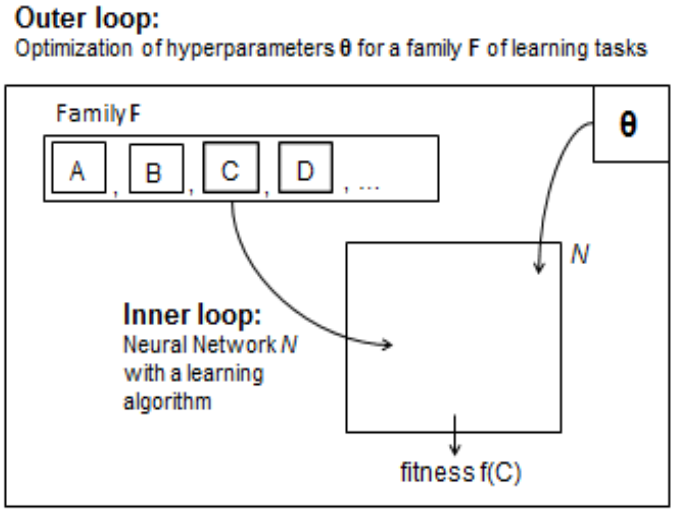
- Brains are able to learn many things from single or few examples („**one-shot learning**“), and we need to achieve this also for SNNs

- Josh Tenenbaum et al. have pointed out that the capability to extract and use **abstract** knowledge is essential for that

- An unrelated motivation: We need to see more SNNs that can carry out powerful computations, e.g., control some stepwise behaviour.

- L2L had so far not been applied to SNNs

L2L framework in modern ML



Instead of a single learning task, we define a **family F of many learning tasks** (F is in general infinitely large).

Specify a set of hyperparameters (hp's) **Θ** of the NN *N* in the inner loop.

Define --for any values **Θ** of the hp's-- the **fitness** f(C) of *N* in learning task C from F. The hp's Θ remain fixed while *N* learns the task C (in the „inner loop").

One optimizes the hp's **Θ** through some „outer loop" optimization (BPTT. ES, ...) so that *N* has high fitness f(C) for randomly drawn tasks C from F.

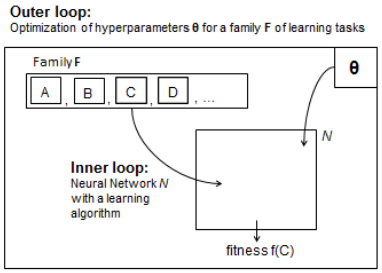
Note: When we use BPTT for the outer loop, we use in addition the errors of *N* on the task C, but accumulate backpropagated errors over several tasks C before applying them.

Essential difference to the standard evaluation of learning in ML:

Testing is **not** carried out for new examples from the **same learning task**, but for new examples from a **new learning task** from the **same family F.**

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., ... & Botvinick, M. (2016). Learning to reinforcement learn. arXiv preprint arXiv:1611.05763.

One can apply this L2L approach just as well to SNNs *N* in the inner loop



Define F as the set of learning tasks which the SNN should be able to learn very fast.

One can choose hp's Θ arbitrarily: Covenient choice: Let them control all aspects of the SNN and its learning about which one is not sure, such as

- time constants and other parameters of neuron and synapses
- **some** or **all** of the synaptic weights and connections of *N*
- plasticity rules and their parameters.

I will show two demos for the case where **all synaptic connections and weights of *N* were hp's, and hence determined by the outer loop** (so that they could **not** be used for learning a particular task from F), like in the talk of Matt Botvinick.

Choice of demos

I focus on **benchmark challenges** that were previously proposed for L2L applied to **non-spiking** recurrent ANNs (LSTM networks), for RL and supervised learning:

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., ... & Botvinick, M. (2016). Learning to reinforcement learn. arXiv preprint arXiv:1611.05763

Wang, J.X, ..., Botvinick, M. (2018) Prefrontal cortex as a meta-reinforcement learning system; biorxiv (Nature Neuroscience, in press)

Hochreiter, S., Younger, A. S., & Conwell, P. R. Learning to learn using gradient descent. ICANN 2001.

.

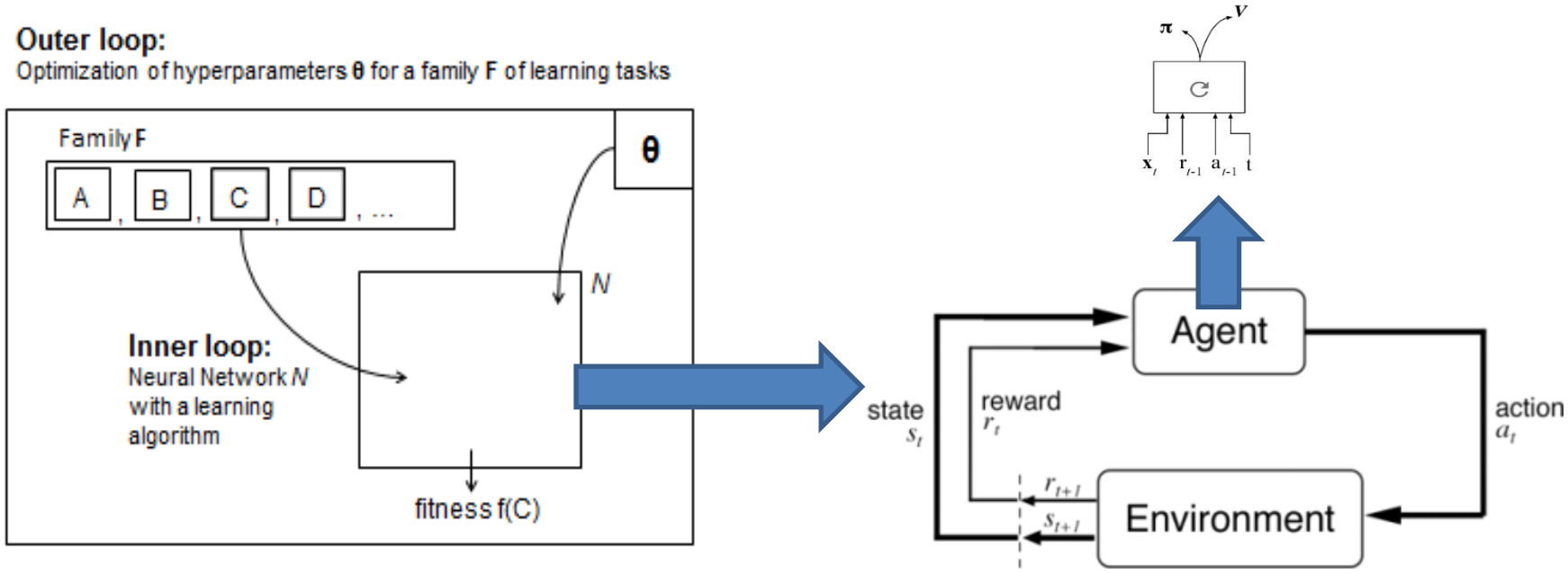
Learning to learn from rewards („Meta-RL“)

- The fitness f(C) reflects here the the sum of **rewards accumulated for task C.** We use the advantage actor-critic algorithm (policy gradient) in combination with deep learning (BPTT), like in (Mnih et al., 2016).

- *N* gets as input the current state, and the action and reward for the **preceding** step

- *N* needs to store in its dynamics the action/reward history while **learning** a task C, and produce a policy.

- The **weight vector W** of the LSNN encodes **the RL-algorithm** (!) which it applies.

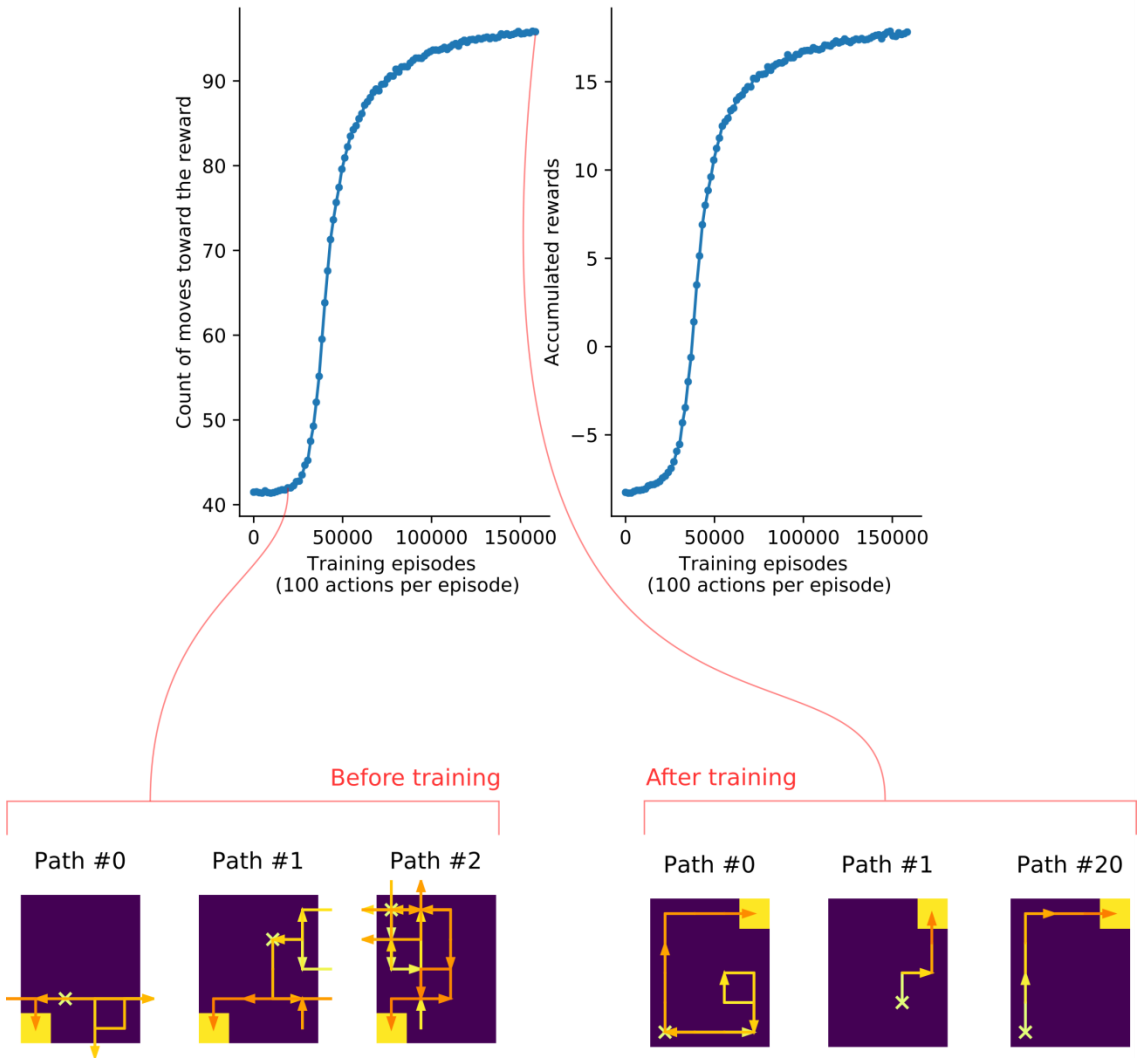


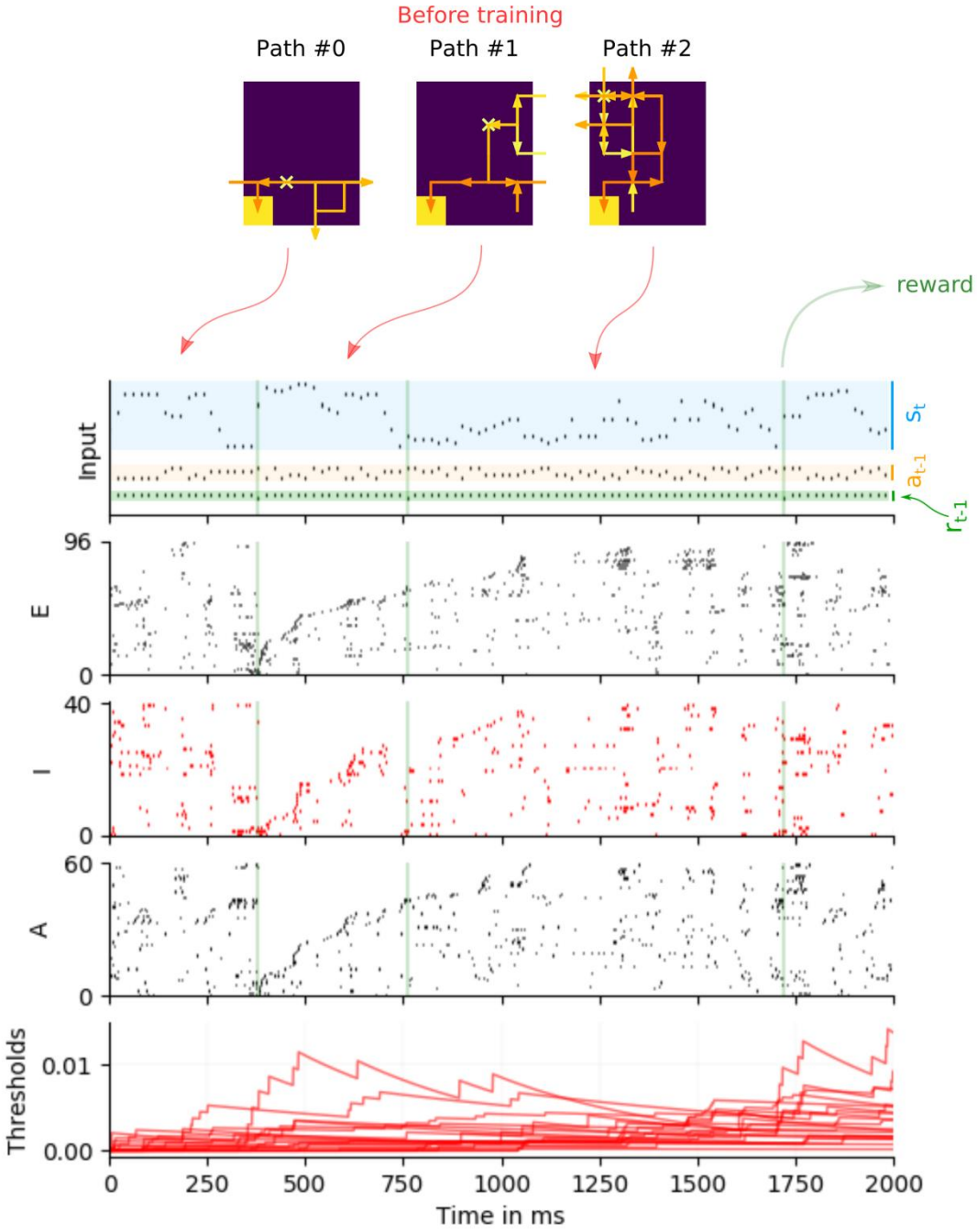
Learning to learn navigation in a maze

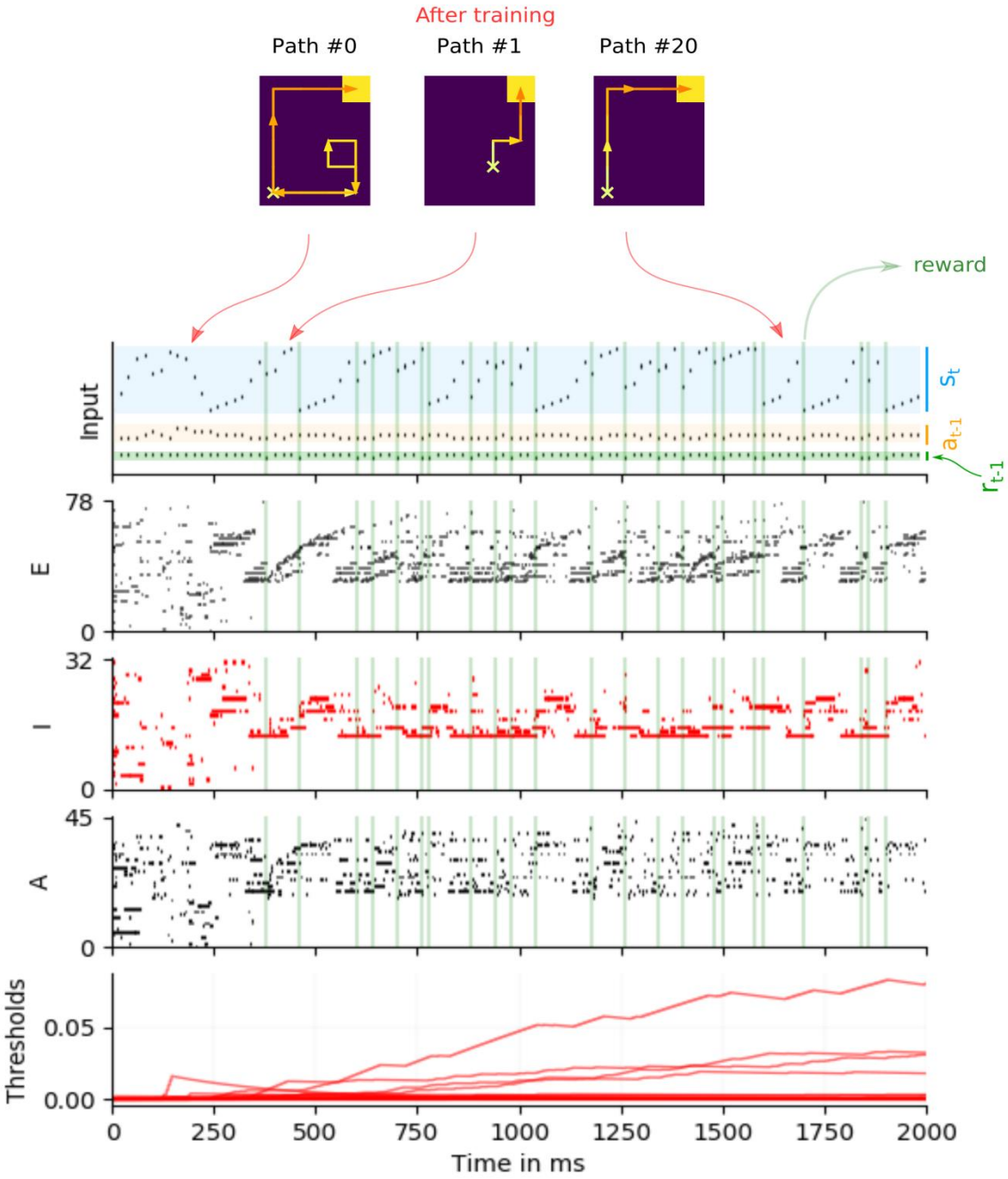
Family F of tasks: Each task is a 4 x 5 maze with a goal in a random corner. The agent starts at a random position and gets 100 steps to collect rewards.

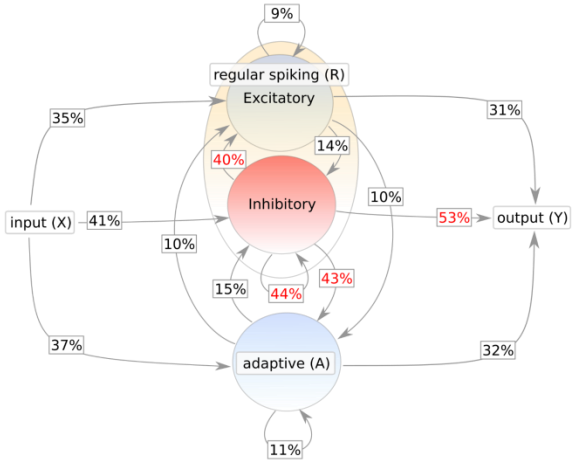
Rewards: 1 for reaching goal (afterwards agent gets reset to a random location)

-0.1 for all other moves




Before training
Path #0
Path #1
Path #2
reward
Input
s_t
a_{t-1}
r_{t-1}
E
96
0
I
40
0
A
60
0
Thresholds
0.01
0.00
0
250
500
750
1000
1250
1500
1750
2000
Time in ms





The weights and connections of the resulting LSNN encoded a clever exploration strategy, and the capability to efficiently exploit the knowledge gained through exploration.

It is very difficult to build a SNN by hand that can control **any** nontrivial sequential behavior.

Relation to the work of Botvinick et al

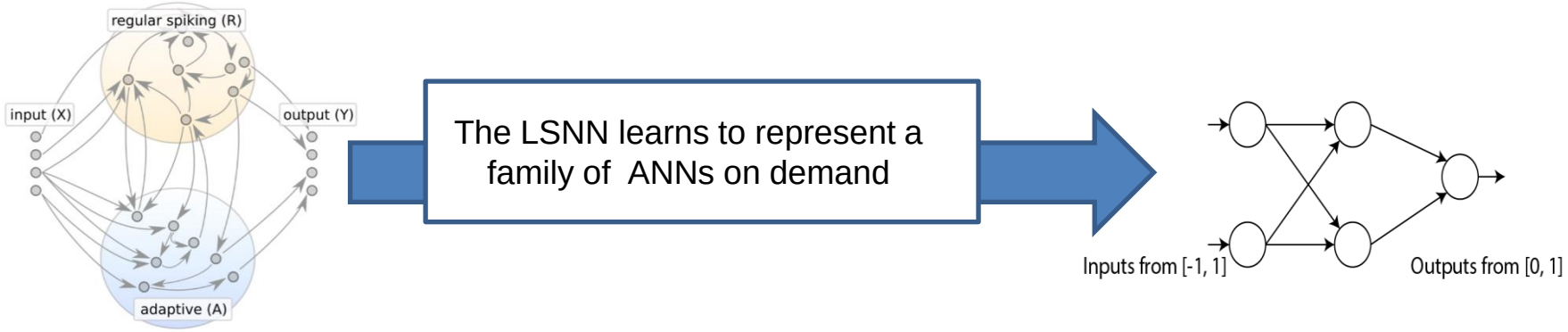
- We have also verified (see our arxiv paper) that transfer learning can be demonstrated for SNNs in a similar manner as in their work with LSTM networks

- Whereas it is not clear how the activity in LSTM networks can be related to neural activity in the brain, the implementation with excitatory and inhibitory neurons of LSNNs provides a clear link to experimental data to neural recordings from the brain.

Learning to learn from a teacher

In this demo the challenge for the LSNN is to find a learning algorithm that has the functionality of backprop (BP)

- Let the family F consist of all functions G that can be represented by a 2 layer ANN („target network“) with sigmoidal neurons and weights from [-1, 1]:



- The LSNN is fed with a new sample to classify < x_1, x_2 > and the target output $G(x_1', x_2')$ for the input <x_1', x_2' > at the **preceding** sample (hence this is **supervised learning**).

- If it wants, the SNN can store the preceding input <x_1', x_2' > and compute the error and the error gradient after the next trial (similar to backprop).

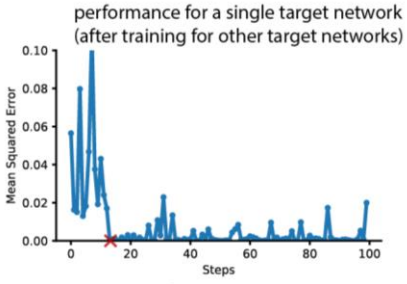
- ***But it does not have to do that. The SNN can choose any learning algorithm that can be performed within its dynamics.***

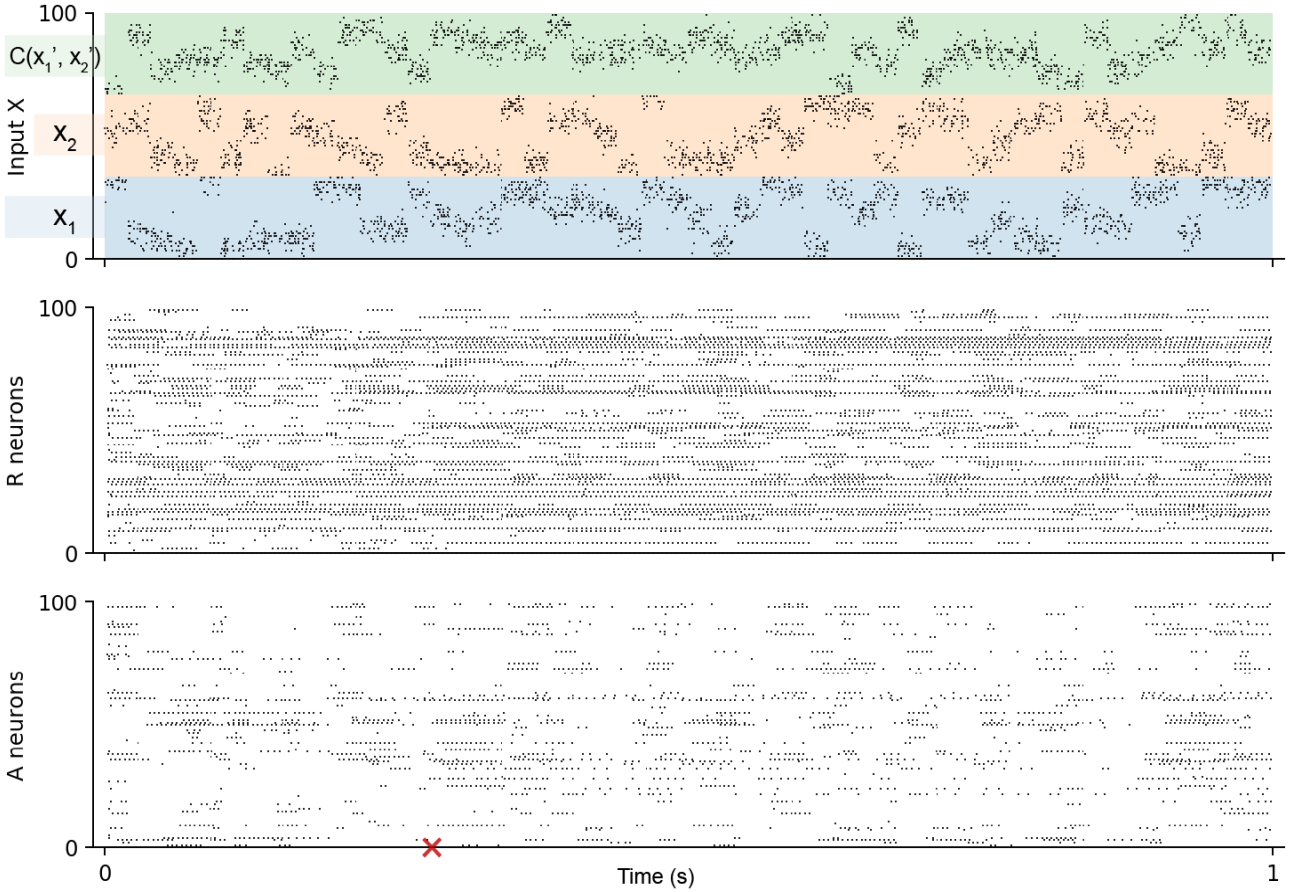
- Note that in principle, all learning rules are defined by smooth functions, and can therefore be approximated by NNs.

A typical learning episode for a new function G defined by a random 2-layer target network

(after fixing all parameters of the LSNN through the outer loop of L2L)

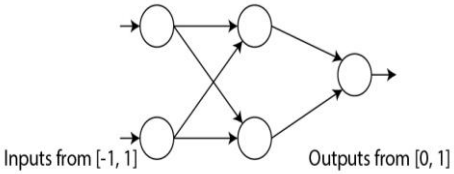
The LSNN learns in this episode to reproduce the input/output behaviour of G after about 15 trials:



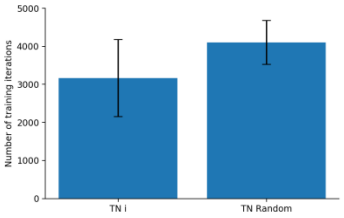


The emergent learning algorithm in LSNNs for 2-layer ANNs appears to differ from BP: is has Bayesian features

- We selected 4 target networks that appeared quite often during training in the outer loop



- During testing, when the synaptic weights were fixed, these 4 target networks were learned faster, in spite of the fact that new network inputs were drawn randomly:



- Hence the learning algorithm that was encoded by the synaptic connections and weights of the LSNN had a Bayesian feature.

Summary

- The integration of adapting neurons into SNNs enables us to port at least some of the powerful computing and learning capabilities of recurrent LSTM networks into spiking neural networks

- Deep Learning becomes (for now) an essential tool for producing computationally powerful LSNNs

- The resulting LSNNs compute with spike patterns, rather than rates. Hence they provide **new paradigms for coding and computing with spikes**.

- We have shown that the Meta-RL approach of Botvinick et al can be ported from LSTM networks into networks of excitatory and inhibitory neurons, thereby supporting a biological interpretation

- We have shown that SNNs that are able to learn from a teacher with seeminly **new learning algorithms, implemented through local rules in the LSNN**.

- R**everse engineering** of the resulting LSNNs and their new learning properties becomes a nontrivial but very interesting new research topic.