Low-Pass RNN using Sigma-Delta Neurons on Loihi

Alpha Renner, Gauthier Boeshertz, Manu Nair

A Loihi Implementation of Backpropagation using Gated Synfire Chains

Alpha Renner, Forrest Sheldon, Anatoly Zlotnik, Louis Tao, Andrew Sornborger

LA-UR-21-21175

3 Approaches to Learning on Loihi

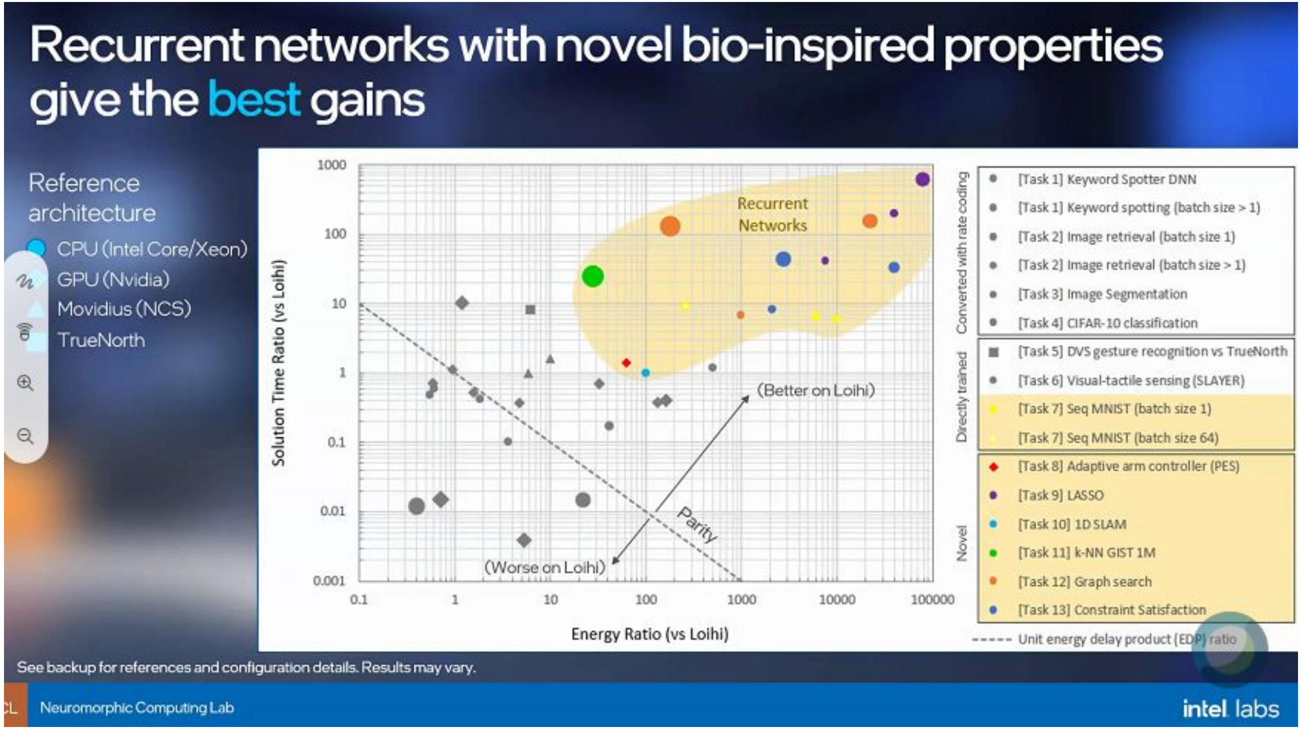
	On-chip learning	Discussion
Conversion ANN->SNN mapping (NxTF, NengoDL, SNN toolbox, LSNN, ΣΔ,...) Offline SNN training (SLAYER, STDB)	✗	• Works well (slightly reduced accuracy) • Once deployed, network does not learn anymore • Training on a GPU is very costly, especially relevant on mobile devices
Non gradient approaches Few shot learning, Associative Memory, ...	✓	• Not much theory • Networks are mostly hand tuned and only contain learning at specific locations • Questionable scalability and real-world usability
Online gradient-based Delta-rule, DECOLLE, EMSTDP, ...	(✓)	• So far, only last layer on-chip training or use of feedback alignment (no vanilla backprop) • Mostly usage of embedded CPU necessary

Low-Pass RNN using Sigma-Delta Neurons on Loihi

Alpha Renner[1]
Gauthier Boeshertz[2]
Manu Nair[1,3]

[1] Institute of Neuroinformatics (INI), University of Zurich and ETH Zurich, Switzerland
[2] Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[3] Synthara AG, Zürich, Switzerland

Motivation



Screenshot from
Mike Davies' talk on Monday
INRC Workshop 2021

→ LSNN is the only deep RNN mapping technique for Loihi so far

lpRNN Pipeline

Offline-Training in Tensorflow → Mapping to Loihi → Inference on Loihi

lpRNN Pipeline

Offline-Training in Tensorflow → Mapping to Loihi → Inference on Loihi

Train with BPTT using lpRNN model (not Loihi specific, but quantization constraints are implemented)

lpRNN equation:

$$y_t = \alpha \odot y_{t-1} + (1-\alpha) \odot \sigma(W_{rec} \cdot y_{t-1} + W_{in} \cdot x_t + b)$$

$\alpha \in [0,1]$ - retention ratio
σ - ReLU

Mozer (1992); Beer (1995); Jaeger et al. (2007); Nair (2019)

lpRNN Pipeline

Offline-Training in Tensorflow

Mapping to Loihi

Inference on Loihi

Train with BPTT using lpRNN model

$$W_{loihi} = \frac{W_{ann} \cdot - W_{fb}}{\tau_{loihi}}$$

$$\tau_{loihi} = \frac{\tau_{ANN}}{ts_{snn}}$$

Generate input spikes (currently in Brian2) and feed them into Loihi

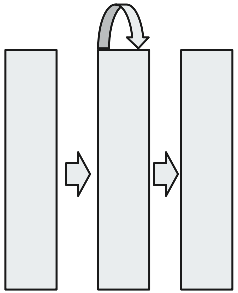
lpRNN Pipeline

Offline-Training in Tensorflow → Mapping to Loihi → Inference on Loihi

Offline-Training in Tensorflow

Train with BPTT using lpRNN model

Example:



dense $\rightarrow$ RNN $\rightarrow$ dense

Mapping to Loihi

$$W_{loihi} = \frac{W_{ann} \cdot - W_{fb}}{\tau_{loihi}}$$

$$\tau_{loihi} = \frac{\tau_{ANN}}{ts_{snn}}$$

Inference on Loihi

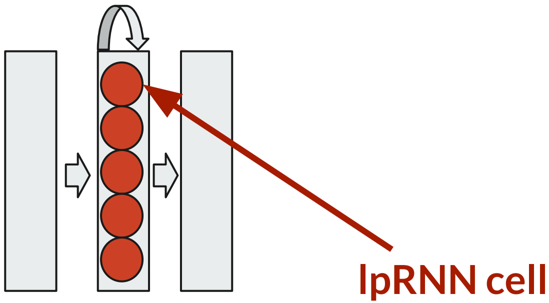
Generate input spikes (currently in Brian2) and feed them into Loihi

lpRNN Pipeline

Offline-Training in Tensorflow

Train with BPTT using lpRNN model

Example:



dense $\rightarrow$ RNN $\rightarrow$ dense

Mapping to Loihi

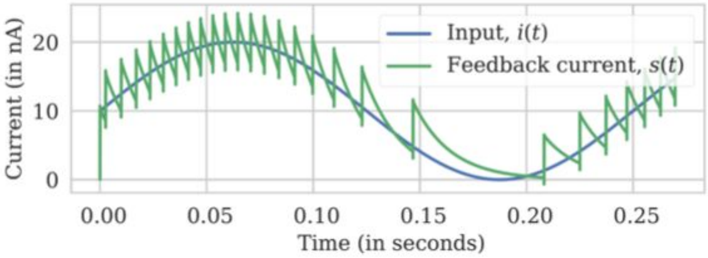
$$W_{loihi} = \frac{W_{ann} \cdot - W_{fb}}{\tau_{loihi}}$$

$$\tau_{loihi} = \frac{\tau_{ANN}}{ts_{snn}}$$

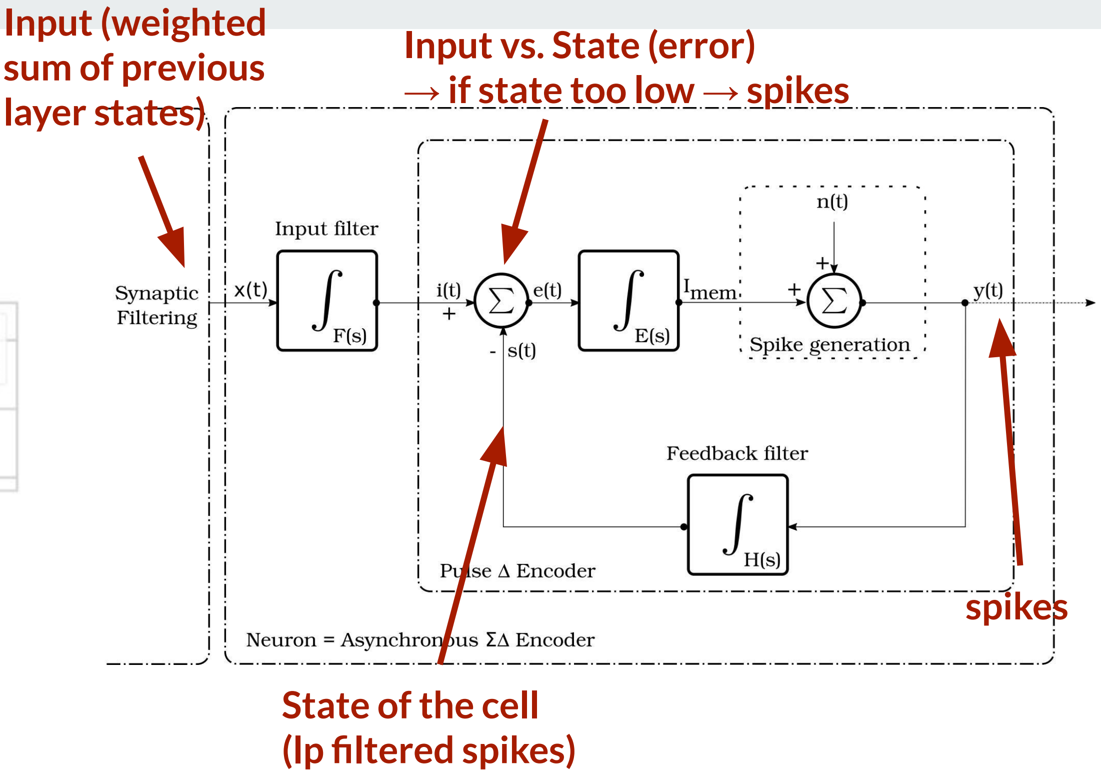
Inference on Loihi

Generate input spikes (currently in Brian2) and feed them into Loihi

lpRNN Cell - Sigma Delta Neuron with Low-Pass Filtering



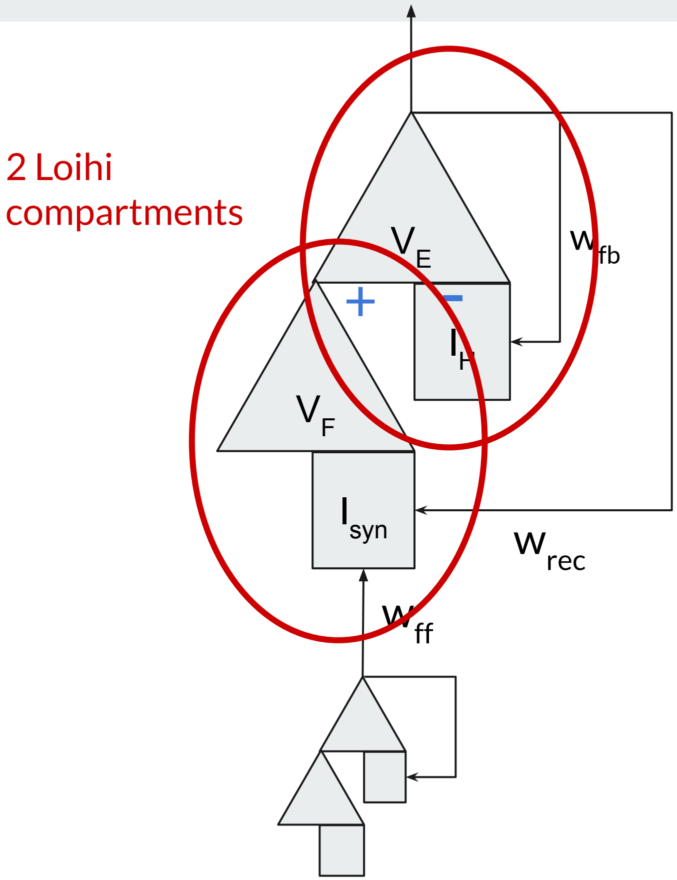
→ The state follows the input current



M. Nair (2019) Mapping high-performance RNNs to in-memory neuromorphic chips

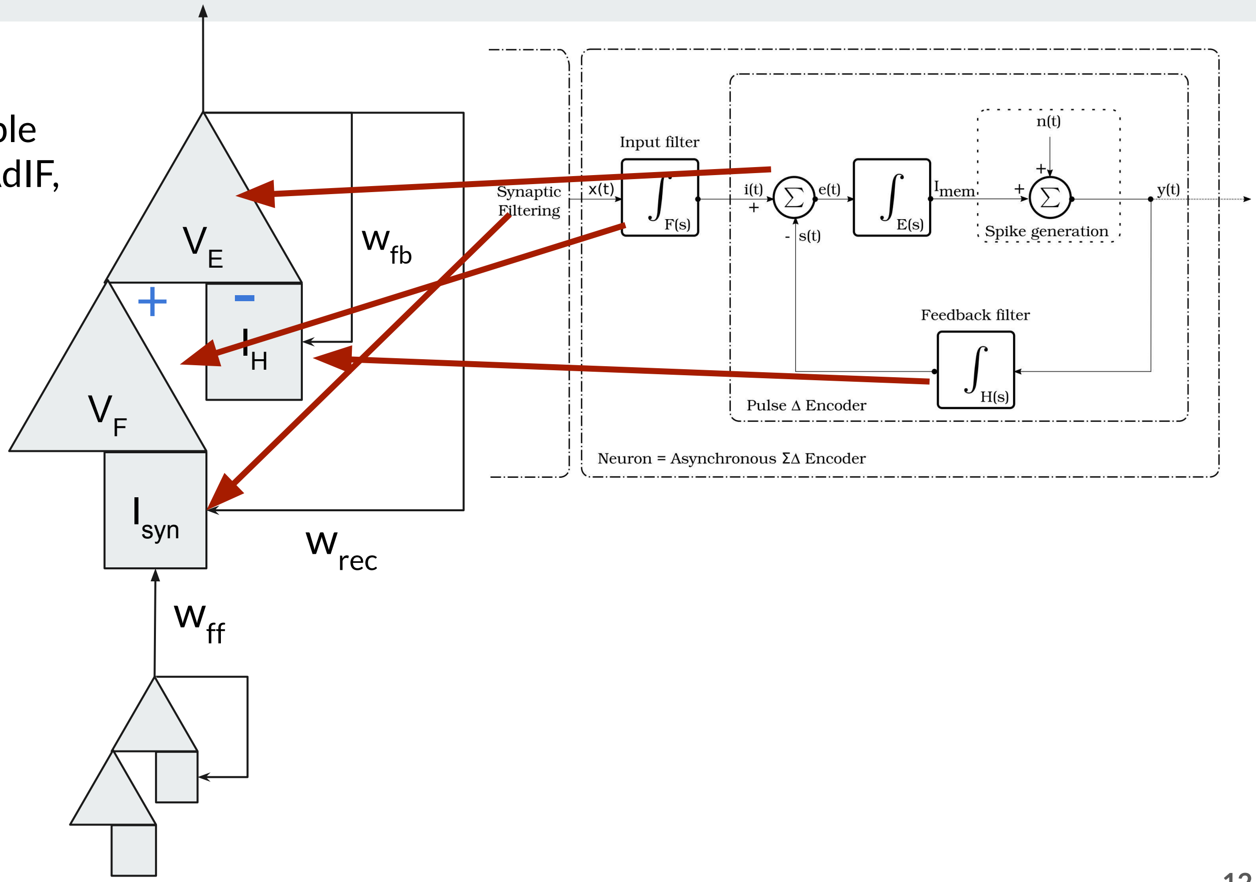
Also read: Gerstner and Kistler (2002), Yoon (2016), Zambrano and Bohte (2016), Yin, Corradi and Bohte (2020)

ΣΔ Neuron Model on Loihi



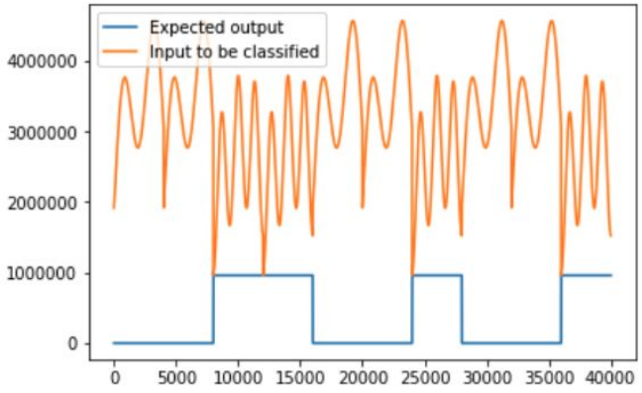
Neuron model on Loihi

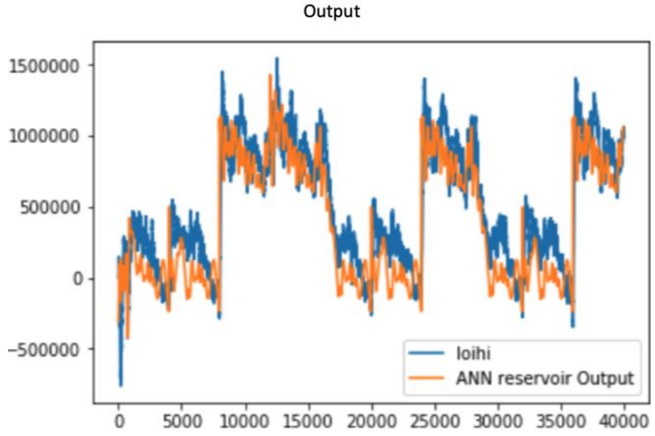
→ Fairly simple biologically plausible 2-compartment neuron on Loihi (AdIF, Gerstner and Kistler, 2002)



Use as a Reservoir Network

Reservoir of 128 neurons
to classify two waveforms

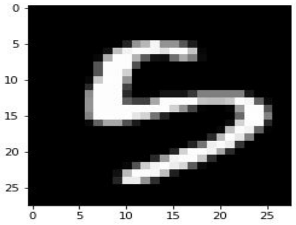




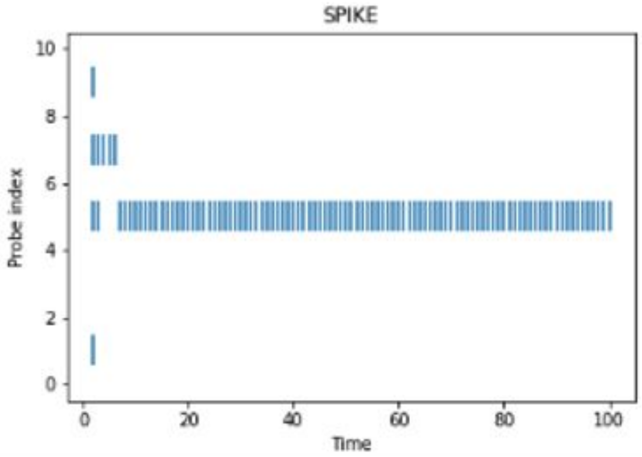
Use as a Feedforward Network (MNIST)

Set τ to 1, so that there is no low pass filtering of the input

5 dense layers (128 neurons)
CPU ANN : 97.9%
Loihi SNN : 93%
(without excessive fine tuning)

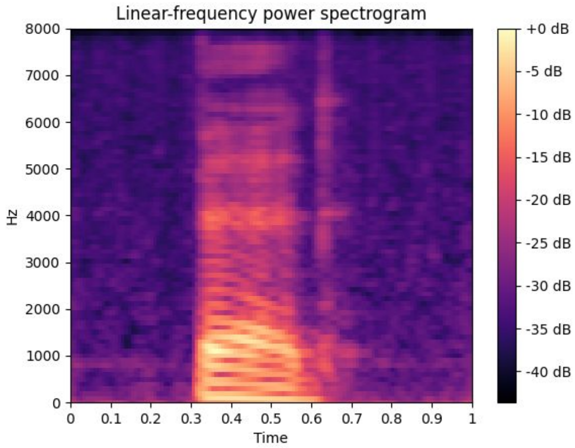


Raster plot of Loihi output layer

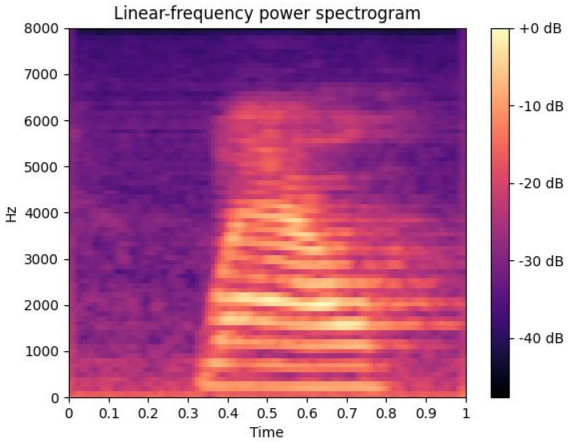


Google Speech Command Task [1] (35 words version)

"down"



"zero"



Unlike CNNs,
the RNN receives
time slices of the input

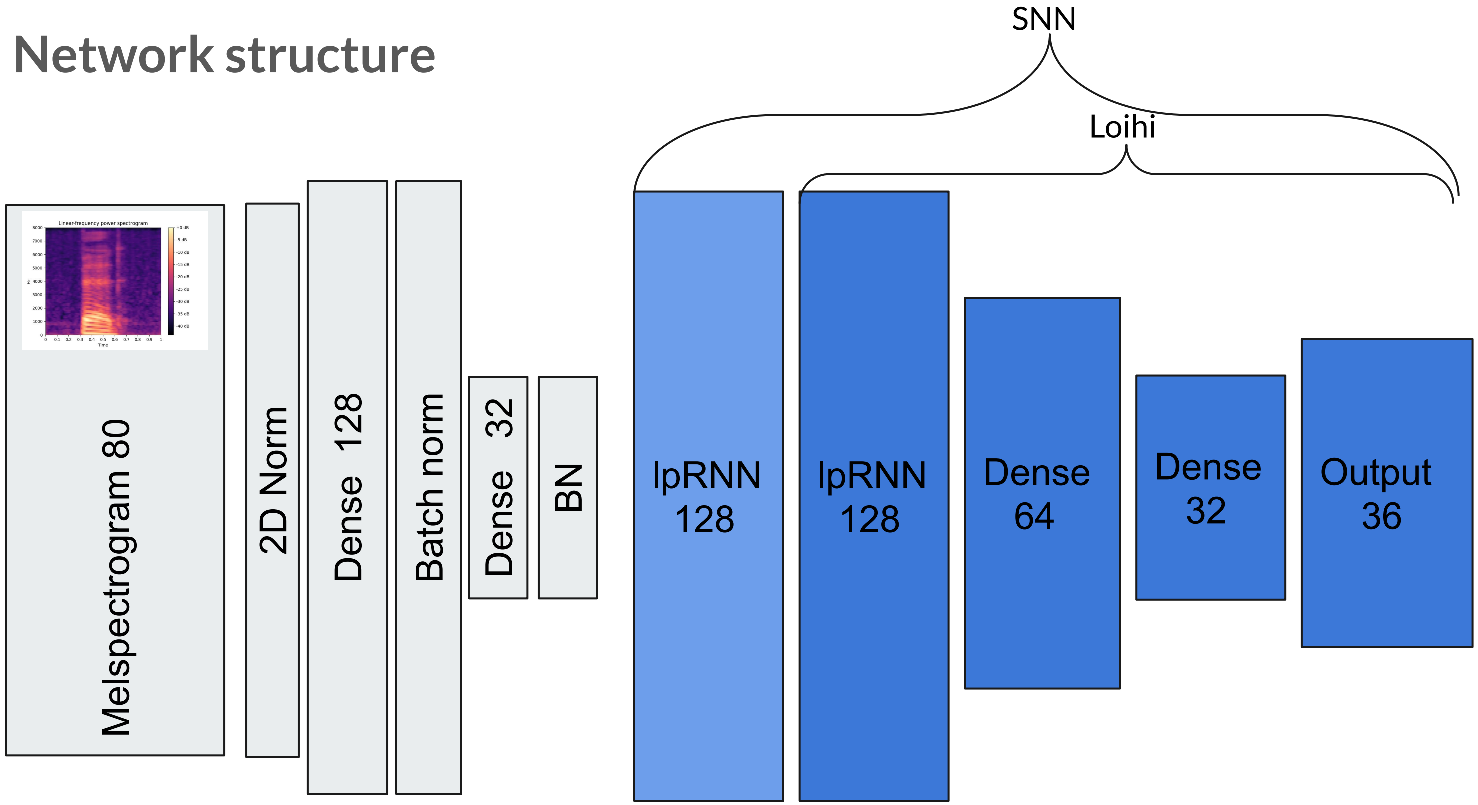
→ real-time operation

Dataset with 35 words:
Backward, Bed, Bird, Cat, Dog, Down, Eight, Five, Follow, Forward, Four, Go, Happy, House, Learn, Left, Marvin, Nine, No, Off, On, One, Right, Seven, Sheila, Six, Stop, Three, Tree, Two, Up, Visual, Wow, Yes, Zero

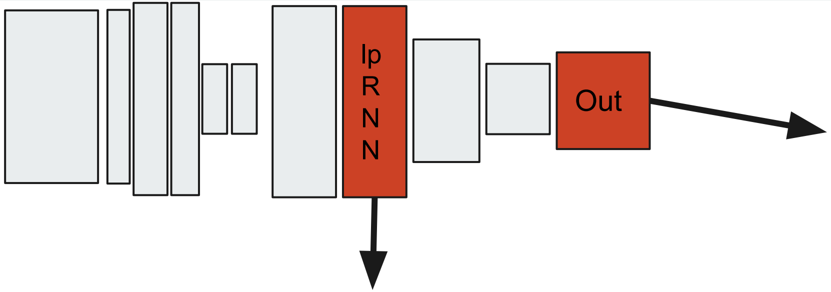
[1] P. Warden Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition

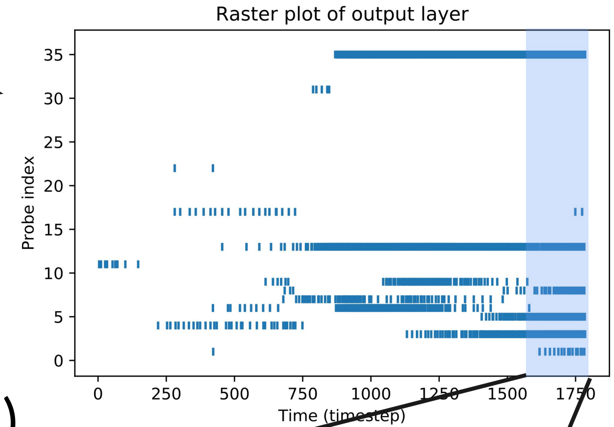
Google Speech Command Task

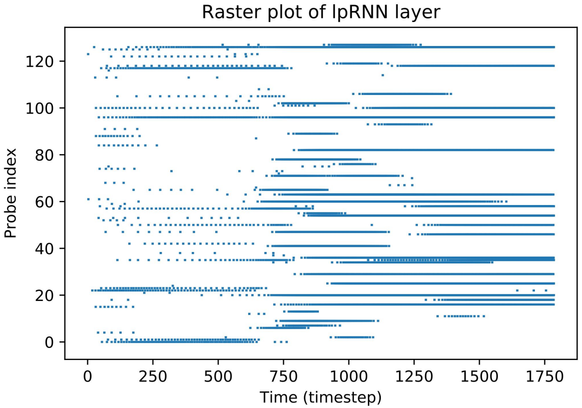
Network structure



Google Speech Command Task

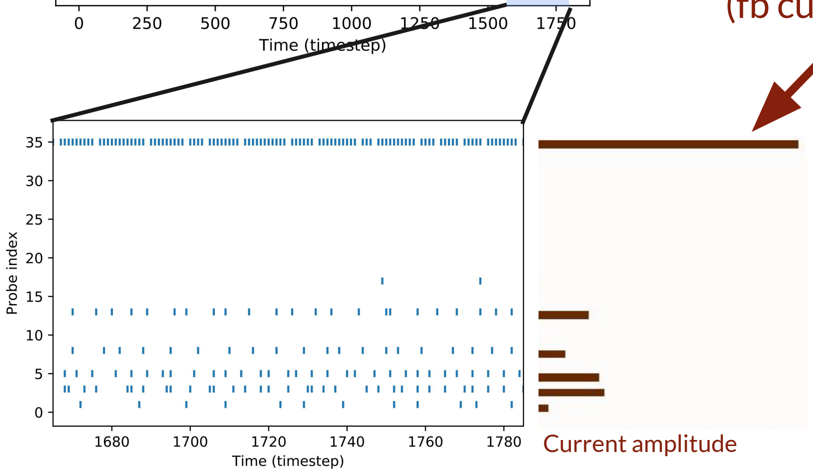






"Zero"
(label 35)

Count spikes or look at output state (fb current)



Sparse RNN state
→ **Advantage for Loihi**

Benchmarking

Task	Vanilla RNN	LSTM	lpRNN	lpRNN on Loihi
Google Commands (Train.)	26%	99.1%	96.8%	-
Google Commands (Test.)	27%	92%	93%	86-87% (~94% ANN match)

lpRNN performs slightly better than LSTM

Caveat: lpRNN performs poorly on text modeling or other usual LSTM tasks

Power Consumption

On Nahuku32 (using 8 cores)
Without other probes, with spikegens

Hardware	Static Power (mW)	Dynamic Power (mW)	Total Power (mW)	Latency (ms)	Energy per inference (mJ)	EDP (uJs)
X86 cores	0.39	42.01	42.39		-	-
Neuron cores	5.39	5.07	10.46		0.55	29
total	898.67	47.08	945.75	52.82	49.81	2630

Possible savings by weight pruning and sparsity regularization

Optimization of I/O

1 second speech recording!
(1750 Loihi timesteps)
→ suited for real-time always-on systems
unlike CNN where time is converted to space

Future Work

- Complete energy profiling
- More benchmarks
- Whole pipeline on Loihi (use raw data instead of spectrogram)
  → Promising simulation results
- Looking forward to Lava and Loihi 2 (with sigma-delta support in the API)
  Accuracy will increase when numerical precision can be distributed better

Take-Home Messages

- lpRNN cell replaces more complicated (and so far not implementable) LSTM/GRU for on-chip inference of natural real-time input data (e.g. audio, biomedical)
- BPTT Training with spikes is not necessary if the spiking neurons are able to represent the state faithfully
- Sigma-delta neurons can achieve this faithful state representation and can be implemented on Loihi with close to state-of-the-art performance

More info (Loihi not yet included):

M.Nair and G. Indiveri (2019), Mapping high-performance RNNs to in-memory neuromorphic chips (arXiv:1905.10692 v4)

LA-UR-21-21175

A Loihi Implementation of Backpropagation using Gated Synfire Chains

Alpha Renner[1], Forrest Sheldon[2], Anatoly Zlotnik[2], Louis Tao[3], Andrew Sornborger[2]

[1] Institute of Neuroinformatics (INI), University of Zurich and ETH Zurich, Switzerland
[2] Los Alamos National Laboratory, Los Alamos, New Mexico, USA
[3] Center for Quantitative Biology, Peking University, Beijing, China

Problems of Spiking On-Chip Backpropagation

Weight transport problem: For correct credit assignment, feedback weights must be the same as feedforward weights.

Backwards computation problem: Forward and backward passes implement different computations.

Gradient storage problem: Error gradients must be computed and stored separately from activations.

Activation storage problem: Forward activations need to be kept in memory for the backward pass.

Differentiability problem: Non-differentiability of spikes .

Hardware constraints problem: Constraints on plasticity mechanisms. Information needs to be local, i.e. only shared between neurons that are synaptically connected. Sufficient weight bit precision is needed for training.

Liao, Leibo and Poggio (2016)

Spiking Backpropagation Approaches

New approaches may help to enable a spiking implementation:

- Lee, J. H., Delbruck, T., & Pfeiffer, M., Frontiers in neuroscience 2016

- **Dendritic cortical microcircuits** – Sacramento, J., Costa, R. P., Bengio, Y., & Senn, W., NeurIPS 2018.

- **Eligibility Propagation** – Bellec, G., Scherr, F., Hajek, E., Salaj, D., Legenstein, R., & Maass, W. (TU Graz), on arxiv, Jan 25, 2019.

- **Surrogate Gradient Learning** – Neftci, E. O., Mostafa, H., & Zenke, F., on arxiv, Jan 28, 2019.

- **Superspike** - Zenke, F., & Ganguli, S.: Neural computation, 2018.

- **Feedback Alignment** - Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J., Nature comm, 2016. Arash, S., Lillicrap, T.P., & Tweed, D.B. Neural computation 2017.

Idea of this Project

Implementation of the (vanilla) backpropagation algorithm in the neural substrate (no SNIPs, just adaptation to hardware constraints)

Based on:

Sornborger, A., Tao, L., Snyder, J., & Zlotnik, A. (2019), NICE.

Backpropagation Algorithm

Update for a single neuron:

$$\Delta w_0^{ij} = \delta_a^i \cdot x^j \cdot r'(z^i)$$

“error” Input Derivative of activation function

Backpropagation Algorithm

Update for a single neuron:

$$\Delta w_0^{ij} = \delta_a^i \cdot x^j \cdot r'(z^i)$$

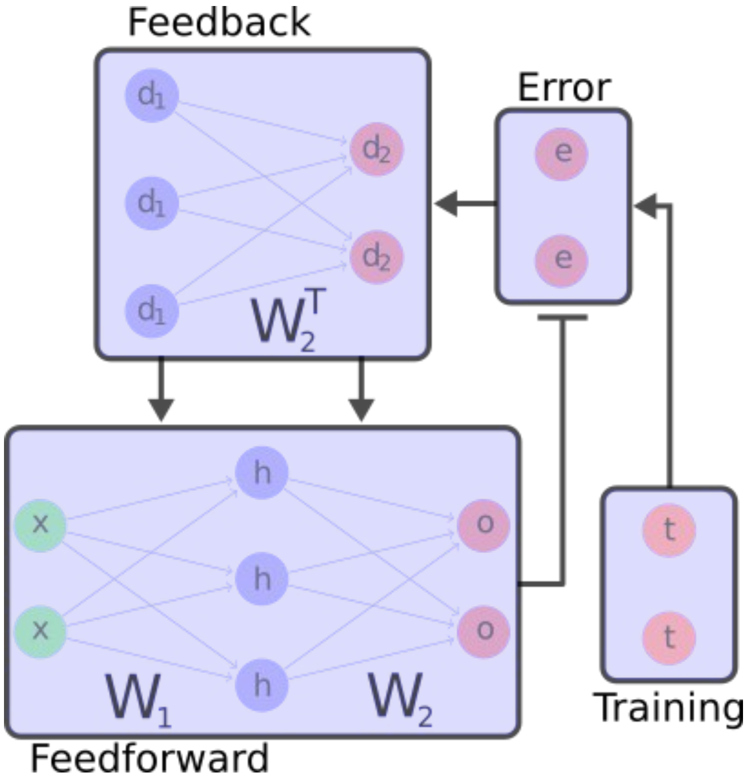
“error” Input Derivative of activation function

Backpropagation to previous layers:

$$\delta_a^i = w_1^{(:,j)T} \cdot \delta_z$$

δ of next layer

Network Schematic and Mechanism



$$\Delta w_0^{ij} = \delta_a^i \cdot x^j \cdot r'(z^i)$$

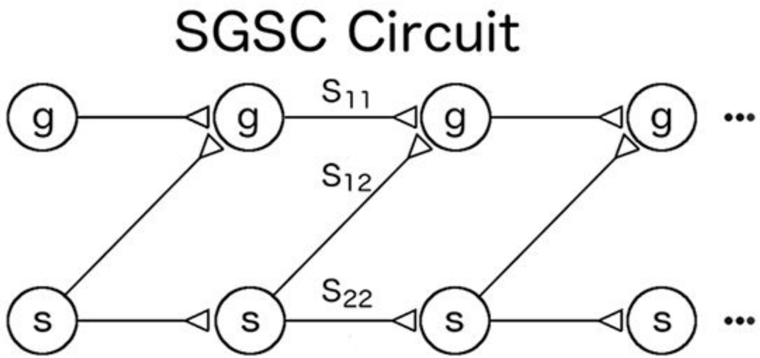
"error" Input Derivative of activation function

$$\delta_a^i = w_1^{(:,j)T} \cdot \delta_z$$

We need to route information through the network!

Synfire-Gated Synfire Chain

Synfire-gated SFC route information/spikes through a network



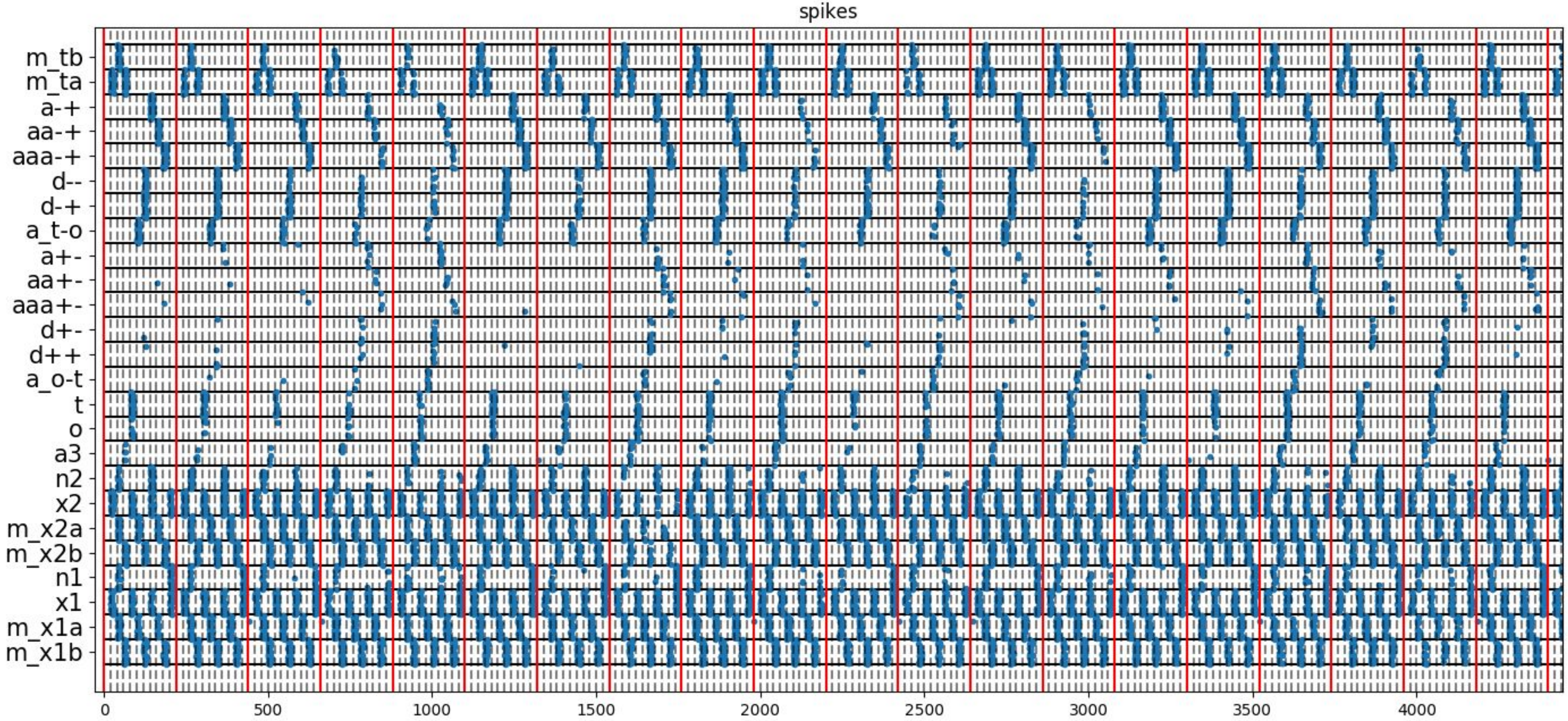
Intuition: All neurons are inhibited globally, the ones allowed to fire are gated on (they only actually fire if they also get additional input from the net though)

"Like switching neurons on and off based on a schedule"

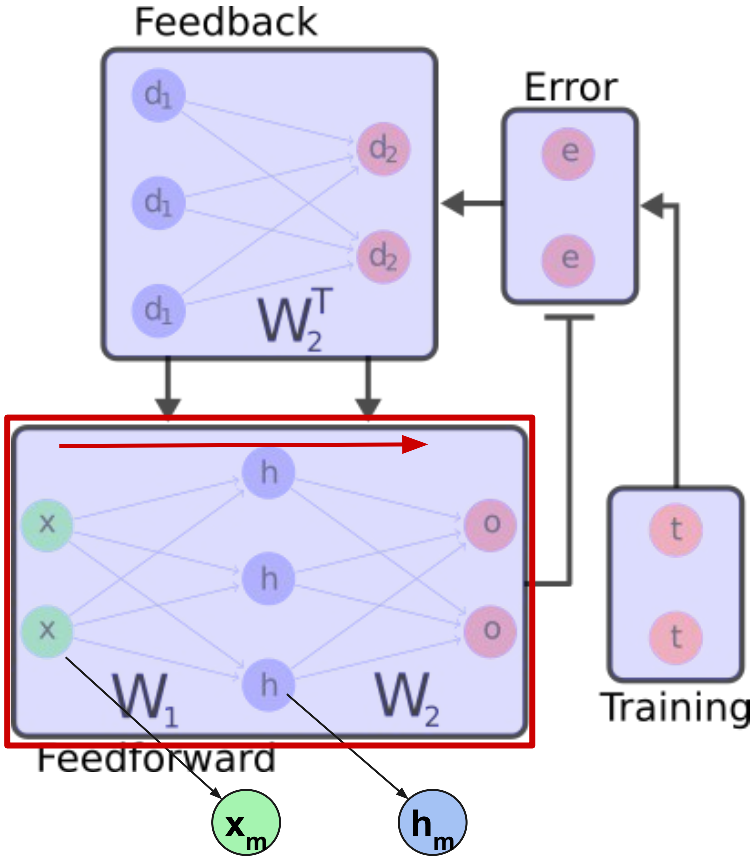
Wang, Z., Sornborger, A. T., & Tao, L. (2016).
Graded, dynamically routable information processing with synfire-gated synfire chains.
PLoS computational biology

Classical SFC Literature:
Abeles (1982, 1991)
Hertz (1997)
Goedeke and Diesmann (2008)
Diesmann et al. (1999)

Raster Plot of Spikes (Pattern)



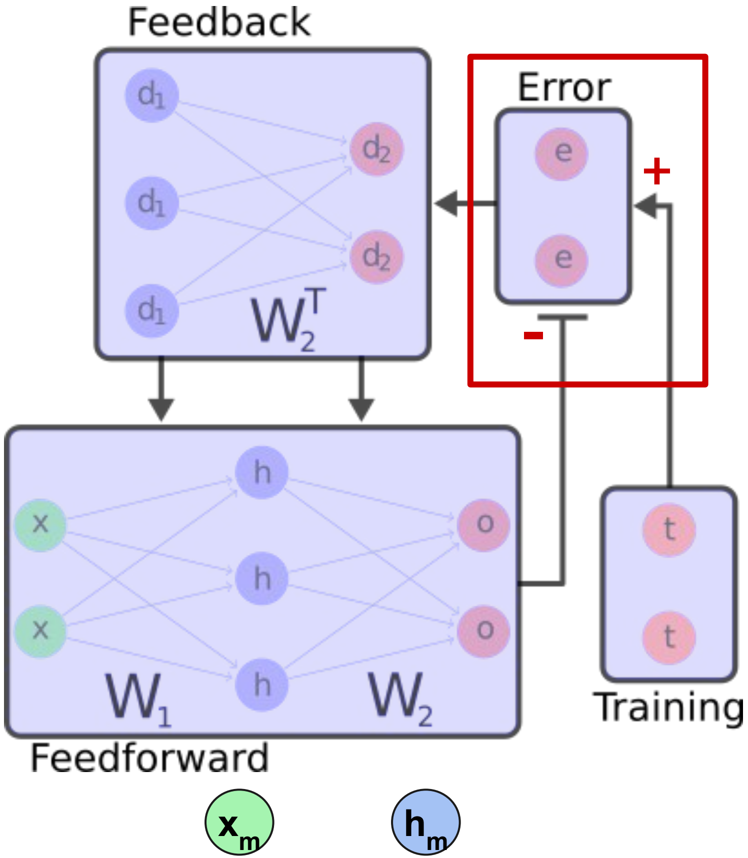
Mechanism - Feedforward



$$\Delta w_0^{ij} = \delta_a^i \cdot x^j \cdot r'(z^i)$$

"error" Input Derivative of activation function

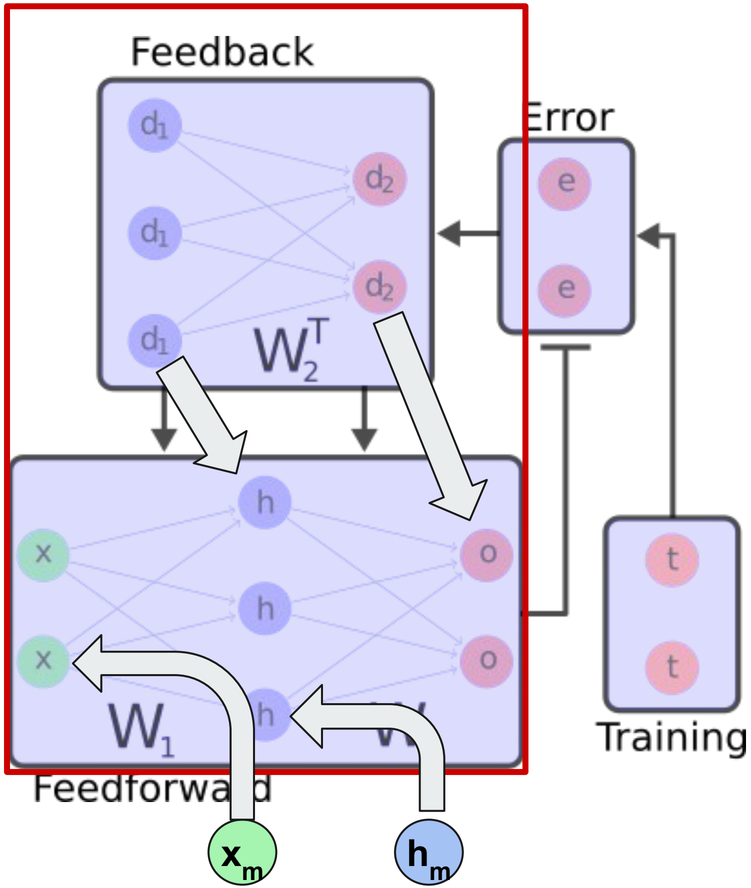
Mechanism - Error



There are twice as many error than output neurons (positive and negative errors)

→ 2 weight update phases (potentiation and depression, governed by “reinforcement channel”)

Mechanism - Backpropagation



$$\Delta w_0^{ij} = \delta_a^i \cdot x^j \cdot r'(z^i)$$

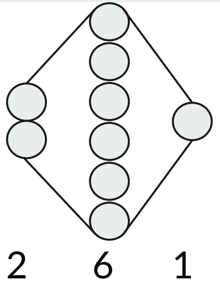
"error" Input Derivative of activation function

Gated Hebbian learning
in 2 phases
(for potentiation and depression)

→ **Can be done in just 14 Loihi timesteps (with binary encoding)**

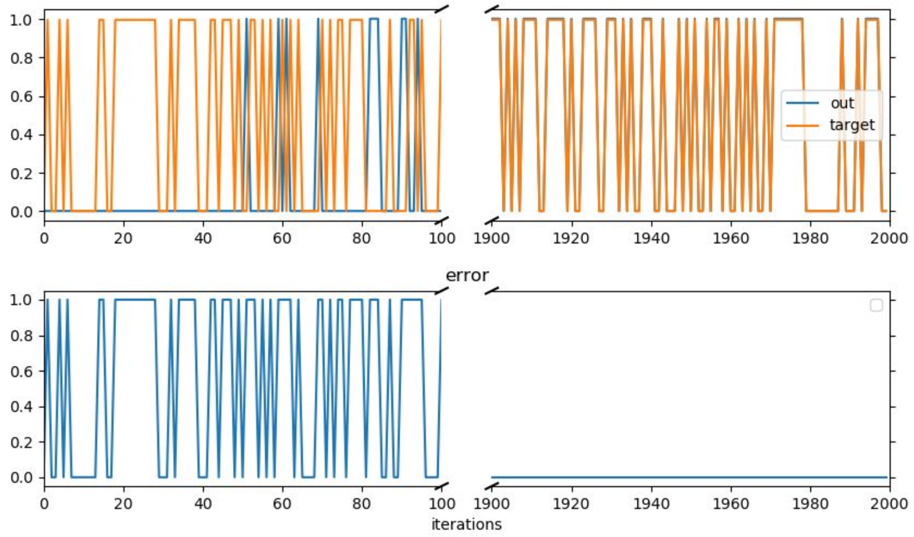
Results: XOR

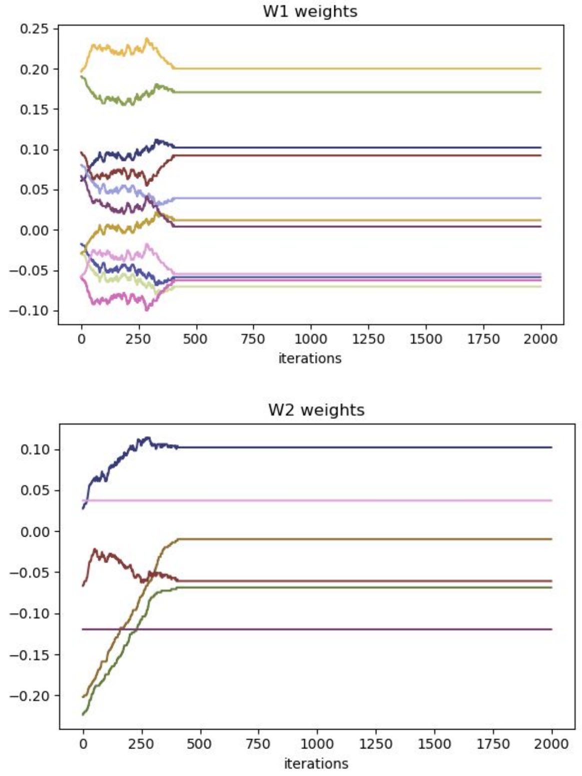
00→ 0
01→ 1
10→ 1
11→ 0



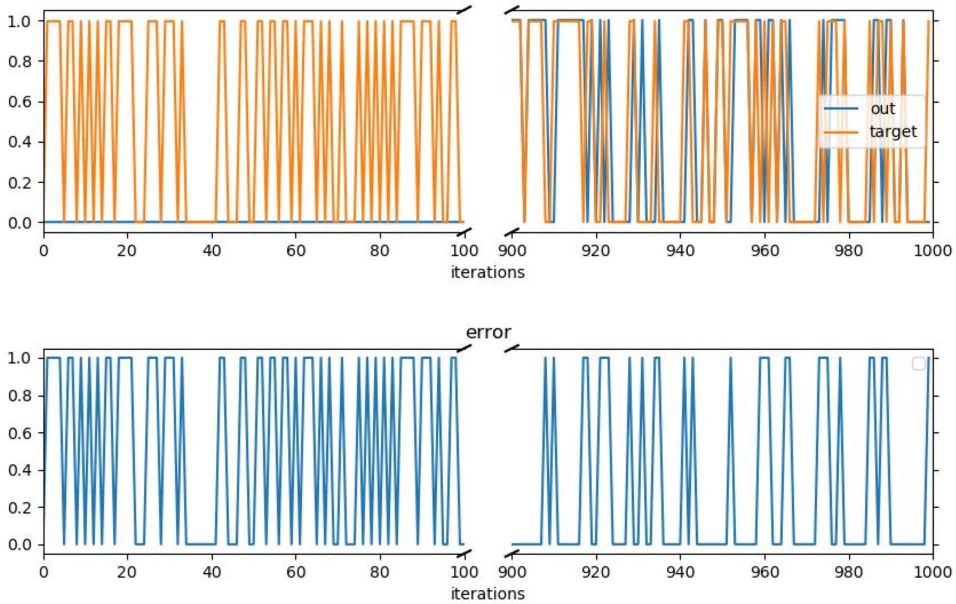
Cannot be solved with a single layer!

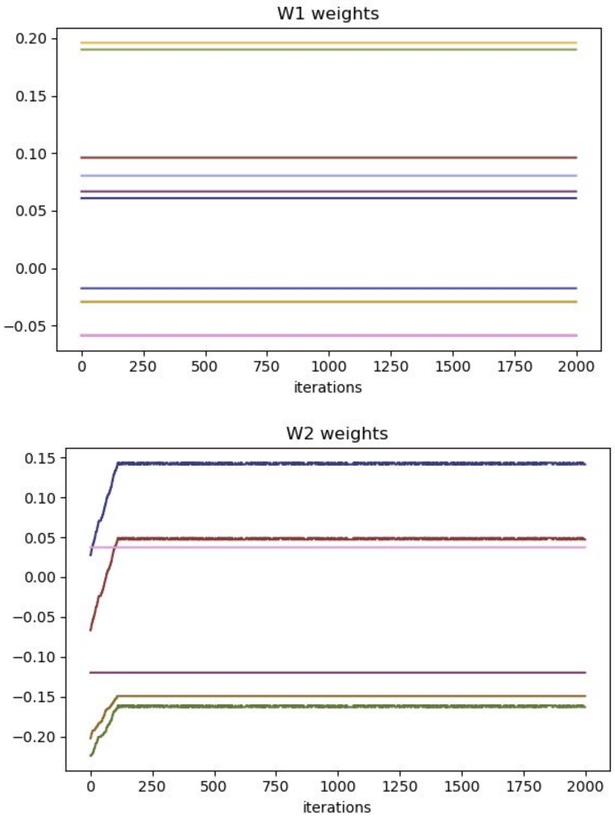
Representative Example:
Weights converge after about 400 it.
error -> 0



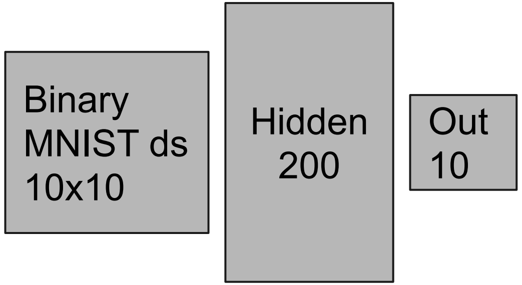


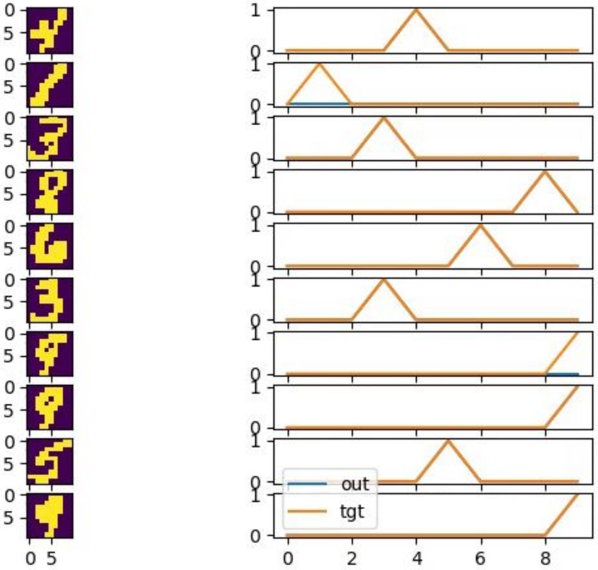
XOR - Ablation of First Layer Learning (Sanity Check)





Preliminary Results: MNIST on Loihi





- So far: 80% train., 70% test
- 3.6 ms per sample (incl. training)

- Without first layer training → stuck at 60%

→ Algorithm works and does backprop!

Conclusion

- Proof of principle of the backpropagation algorithm in a spiking network
- Framework of synfire-gated activity allows for enormous flexibility as we can implement operations that are not otherwise suited for SNN

Further improvements:

- Use STDP (instead of 2 global phases)
- Replace part of the backward network by bidirectional connections

More info:

Renner, A., Sheldon, F., Zlotnik, A., Tao, L., & Sornborger, A. (2020, March). Implementing Backpropagation for Learning on Neuromorphic Spiking Hardware. In *Proceedings of the Neuro-inspired Computational Elements Workshop* (pp. 1-3).

Sornborger, A., Tao, L., Snyder, J., & Zlotnik, A. (2019, March). A pulse-gated, neural implementation of the backpropagation algorithm. In *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop* (pp. 1-9).

Thank you!

Intel Labs for Loihi support and hosting of the workshop.

Collaborators:
Backprop project: Andrew Sornborger, Anatoly Zlotnik, Forrest Sheldon (at LANL)
lpRNN project: Gauthier Boeshertz, Manu Nair

Advisors at INI: Yulia Sandamirskaya, Giacomo Indiveri
Discussions: Yigit Demirag, Melika Payvand

Funding by the Swiss National Science Foundation (SNSF).

END

More questions?

alpren@ini.uzh.ch or on Slack