Introduction to LSNN model

26.06.2018
Darjan Salaj
salaj@igi.tugraz.at

TU Graz

Institute for Theoretical Computer Science
Graz University of Technology, Austria

Motivation

- Artificial recurrent neural networks
  - show powerful computational capabilities
  - achieve state of the art performance in:
    - machine translation, speech processing, video prediction, etc.
  - always employ specialized network architecture
    - LSTM, GRU

- Spiking recurrent neural networks
  - hard to demonstrate comparable computational properties

Artificial RNNs

- Specialized architecture
- Stable memory through **cell state**

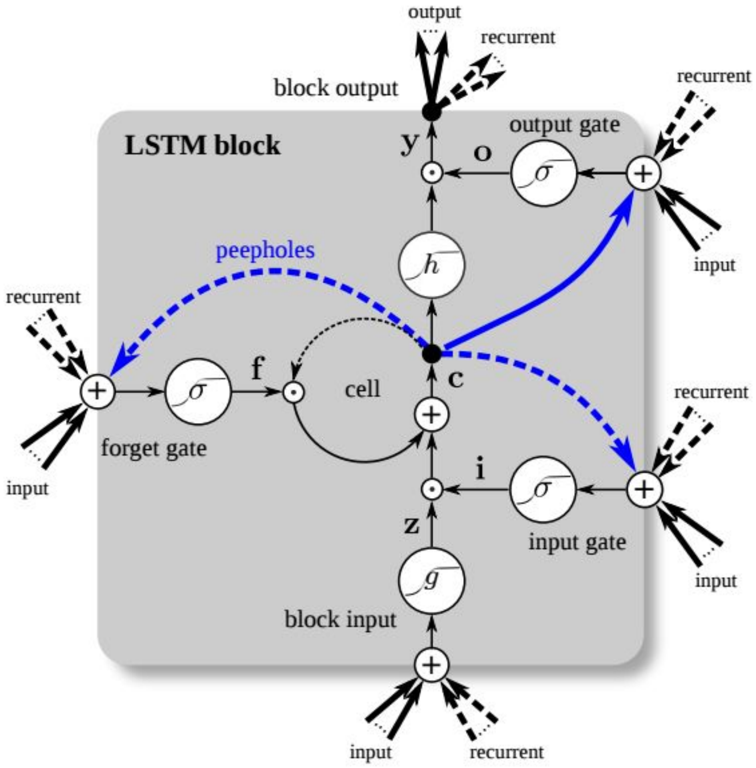


Figure: Greffet al., LSTM: A search space odyssey, CoRR abs/1503.04069 (2015)

Cell state in SNN?

- Neural adaptation
  - Allen Institute: Adaptation index of neurons in neocortex of mouse and humans
- Simple model for adaptation:

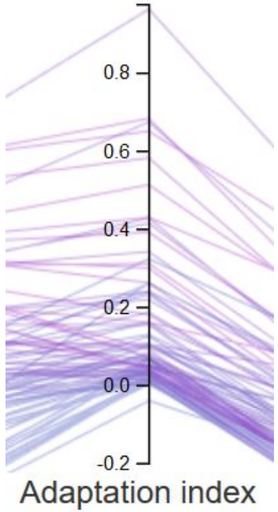
The firing threshold B_j of an adaptive neuron contains a time-varying component $b_j(t)$ that is temporarily increased by each of its spikes $z_j(t)$ and decays slowly.

$$z_j(t) \quad \text{— spike train of neuron } j$$

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

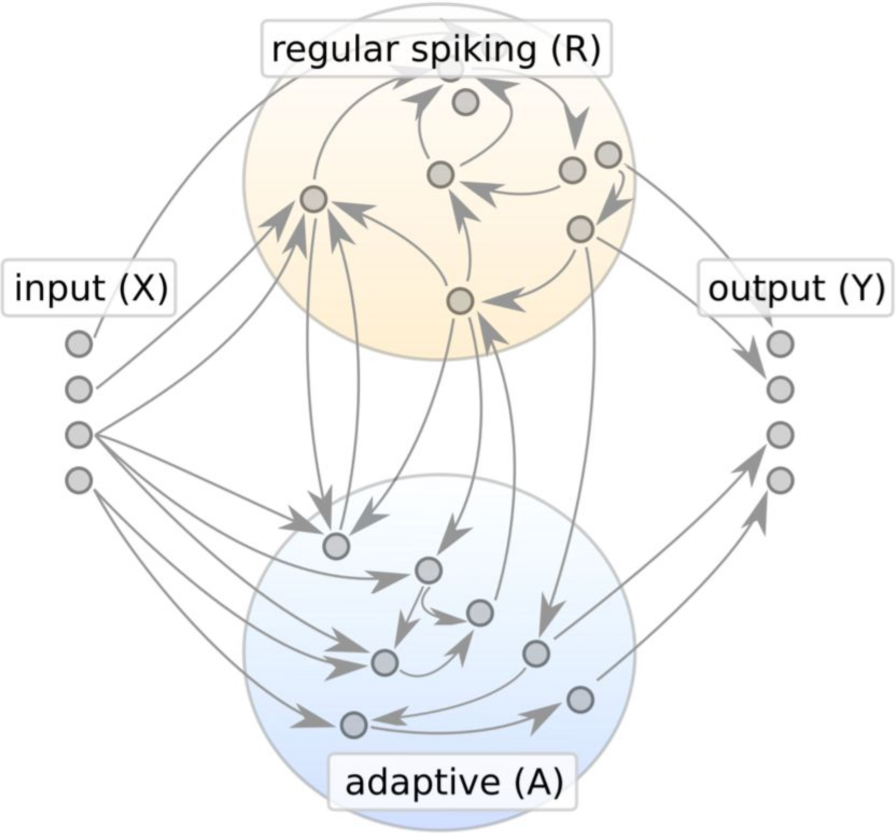
$$b_j(t+\delta t) = \rho_j b_j(t) + (1-\rho_j) z_j(t)$$

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



LSNN model

- Two neuron types:
  - LIF neurons
  - LIF neurons with adaptive threshold
- Architecture

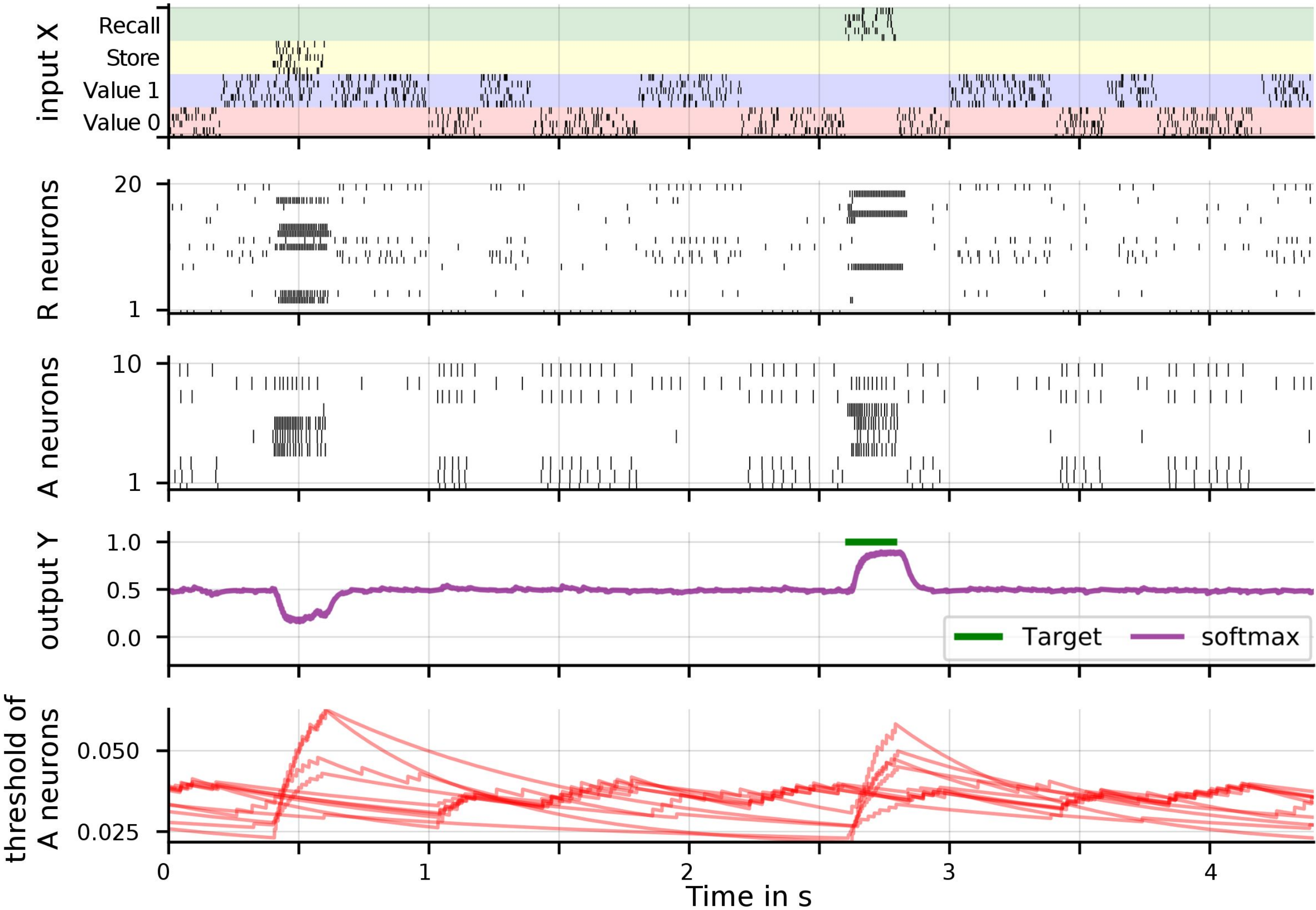


- Discrete time simulation (1ms step)
- Trained using BPTT (Bellec et al.)

Computational capabilities of LSNNs

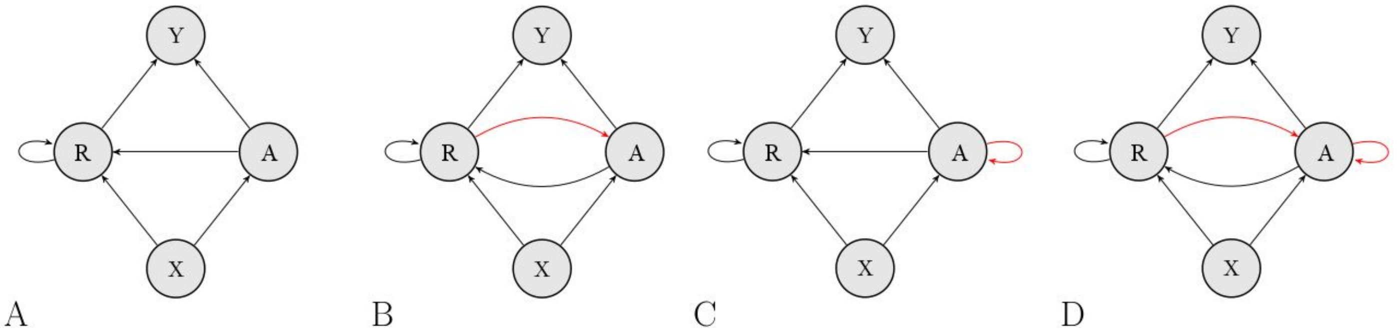
Store-recall task: memorize and recall a single bit

Bit presented for 200ms

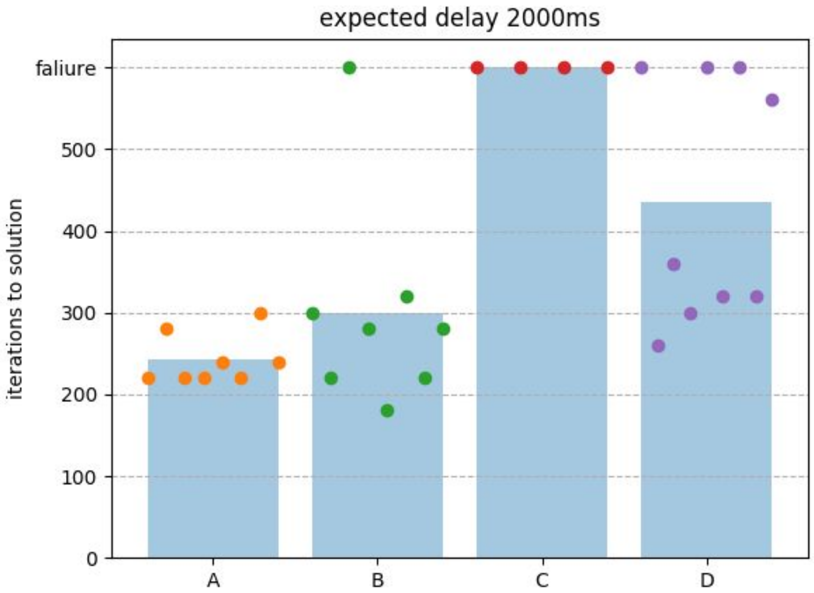


Importance of architecture

Proposed architectures:



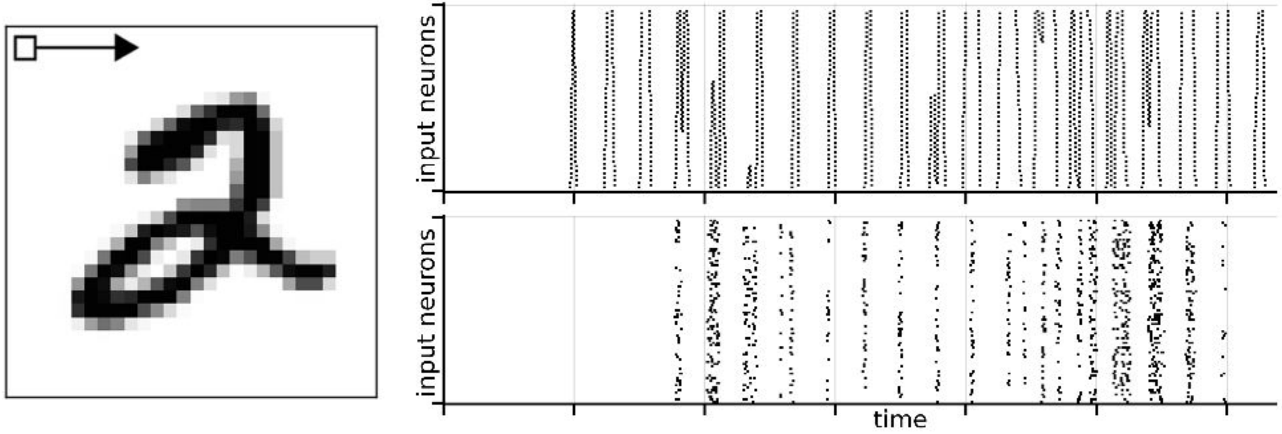
Time to convergence (error < 5%):



Computational capabilities of LSNNs

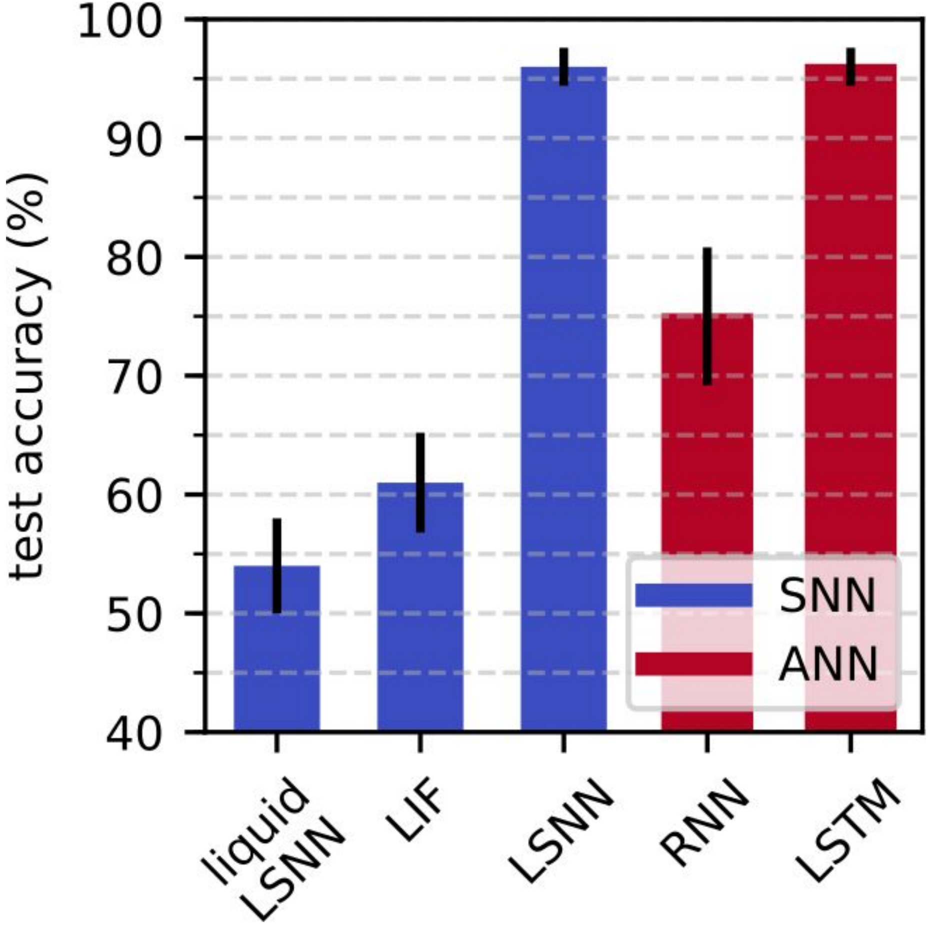
Sequential MNIST:

- Classify images with pixels presented in sequential manner
- Analog to spike encoding

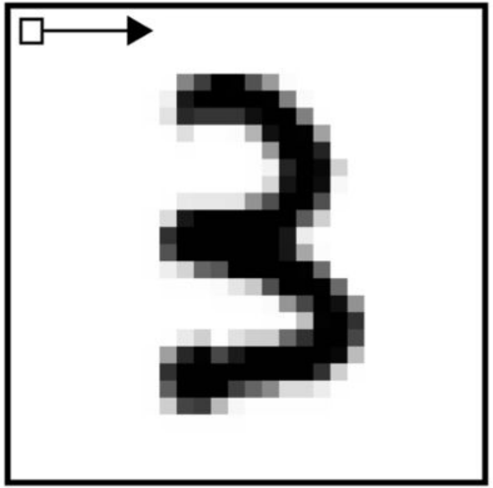


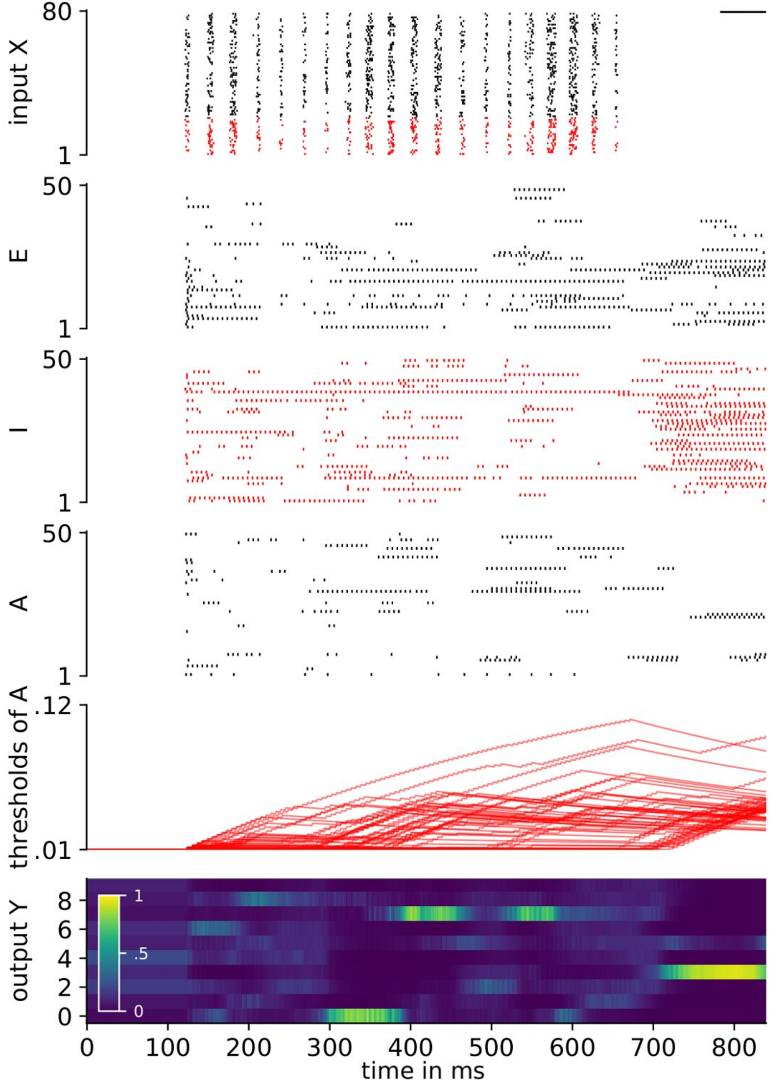
Sequential MNIST results

Architecture learned using
DEEP R (Bellec et al. 2018)



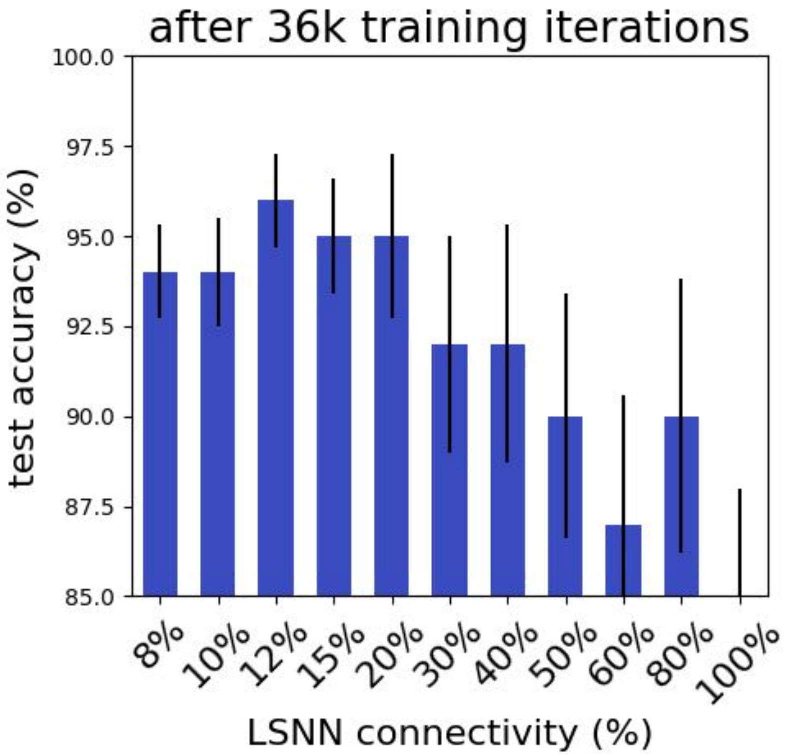
Sequential MNIST results



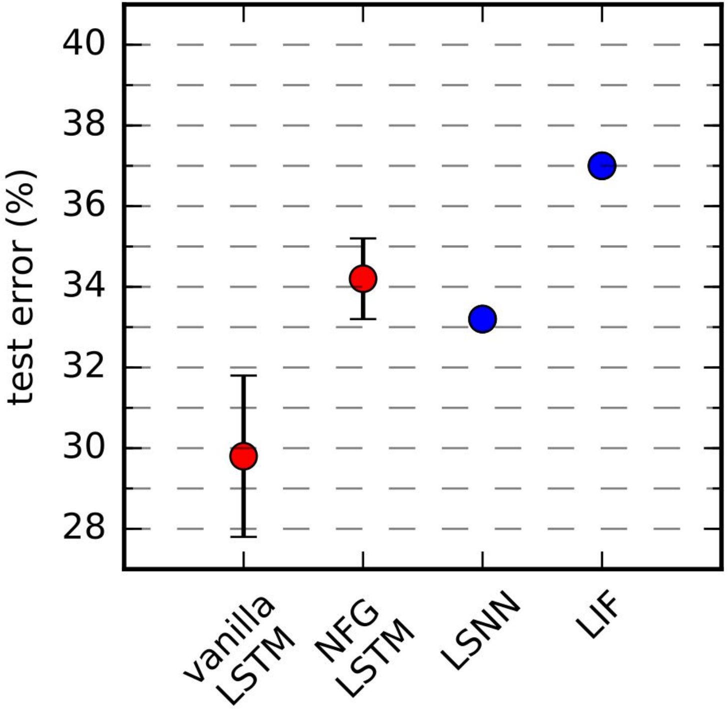


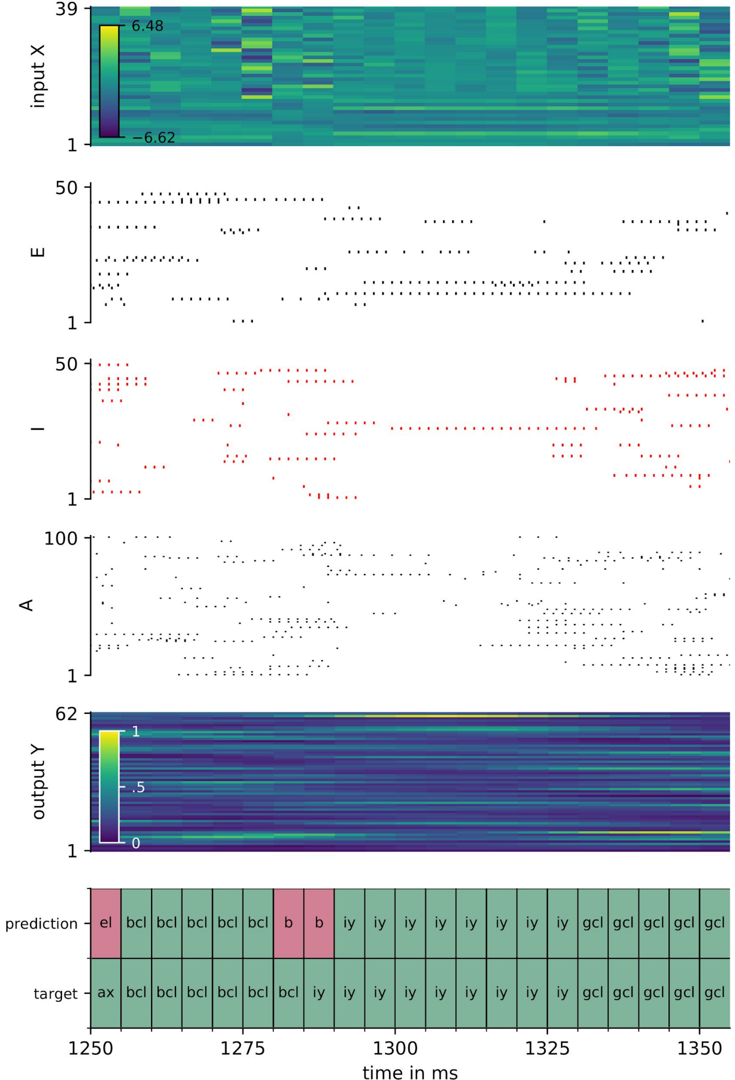
Sequential MNIST results

Impact of LSNN architecture
on the performance



TIMIT results





Klaus Greff, Rupesh K Srivastava, Jan Koutnik, Bas R Steunebrink, and Jurgen Schmidhuber. LSTM: A search space odyssey.
IEEE transactions on neural networks and learning systems, 2017.

LIF with adaptive threshold in discrete time (details)

Membrane potential: $V_j(t)$

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

Input current (weighted sum of spikes): $I_j(t)$

$$V_j(t + \delta t) \quad = \quad \alpha V_j(t) + (1 - \alpha) R_m I_j(t) - B_j(t) z_j(t) \delta t$$

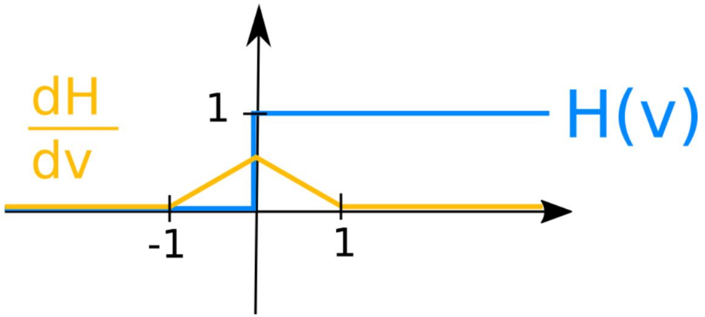
Spikes: $z_j = H\left(\frac{V_j - B_j}{B_j}\right)\frac{1}{\delta t}$

where ***H*** is a binary step function

Backpropagation through time (BPTT) in LSNN

Backpropagating gradient through **spikes**:

- Step function is not differentiable
- The gradients are propagated through step functions with a **pseudo-derivative**



normalized membrane potential $v_j(t) = \frac{V_j(t) - B_j(t)}{B_j(t)}$

$$\frac{dz_j(t)}{dv_j(t)} := \gamma \max\{0, 1 - |v_j(t)|\}$$

Back propagating through many timesteps is subject to exploding-vanishing gradients:

- The **slow** dynamics of the adaptive threshold solves this issue
- similar to the memory units of LSTM

Thank you!

Pseudo-derivative of spiking neuron output

$$\text{normalized membrane potential } v_j(t) = \frac{V_j(t) - B_j(t)}{B_j(t)}$$

$$\frac{dz_j(t)}{dv_j(t)} := \gamma \max\{0, 1 - |v_j(t)|\}$$

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. arXiv preprint arXiv:1602.02830, 2016.

Steven K. Esser, Paul A. Merolla, John V. Arthur, Andrew S. Cassidy, Rathinakumar Appuswamy, Alexander Andreopoulos, David J. Berg, Jeffrey L. McKinstry, Timothy Melano, Davis R. Barch, Carmelo di Nolfo, Pallab Datta, Arnon Amir, Brian Taba, Myron D. Flickner, and Dharmendra S. Modha. Convolutional networks for fast, energy-efficient neuromorphic computing. Proceedings of the National Academy of Sciences, 113(41):11441–11446, November 2016.

LSNN model

$$
\begin{aligned}
I_j(t) &= \sum_i W_{ji}^{in} x_i(t - d_{ji}^{in}) + \sum_i W_{ji}^{rec} z_i(t - d_{ji}^{rec}) \\
V_j(t + \delta t) &= \alpha V_j(t) + (1 - \alpha) R_m I_j(t) - B_j(t) z_j(t) \delta t \\
b_j(t + \delta t) &= \rho_j b_j(t) + (1 - \rho_j) z_j(t)
\end{aligned}
$$

$$
\begin{aligned}
B_j(t) &= b_j^0 + \beta b_j(t) \\
\alpha &= \exp(-\frac{\delta t}{\tau_m}) \\
\rho_j &= \exp(-\frac{\delta t}{\tau_{a,j}})
\end{aligned}
$$

Minimal LSNN solving store-recall task

Store-recall:
memorize single bit