Fast classification using sparsely active spiking networks

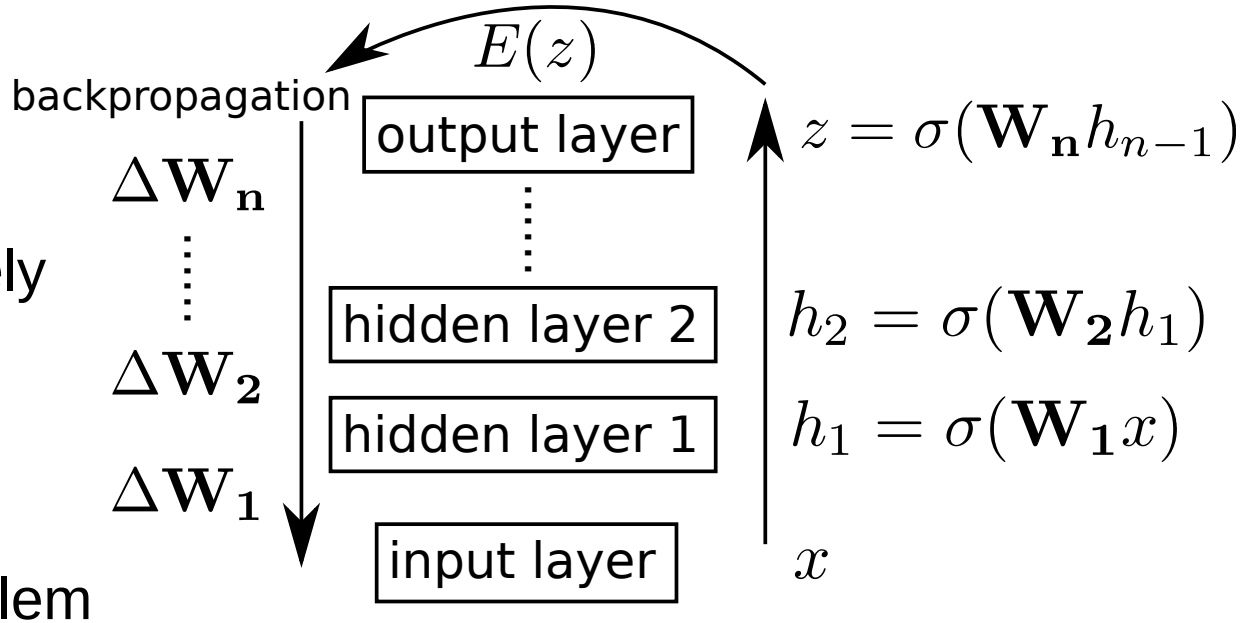
Hesham Mostafa
Institute of neural computation, UCSD

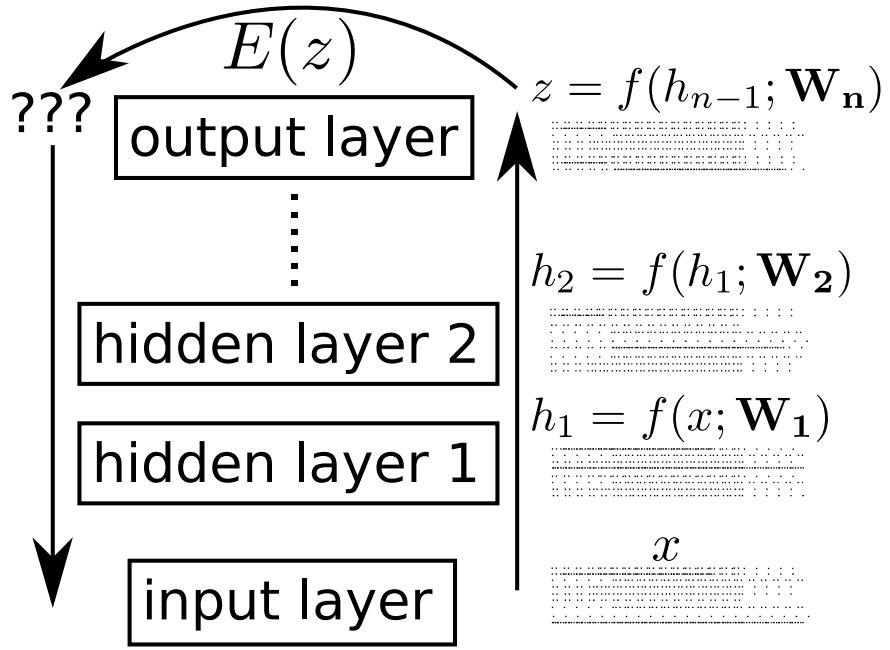
Artificial networks vs. spiking networks

Multi-layer networks are extremely powerful function approximators.

Backpropagation is the most effective method we know of to solve the credit assignment problem in deep artificial networks

<u>How do we solve the credit assignment problem in multi-layer spiking networks?</u>

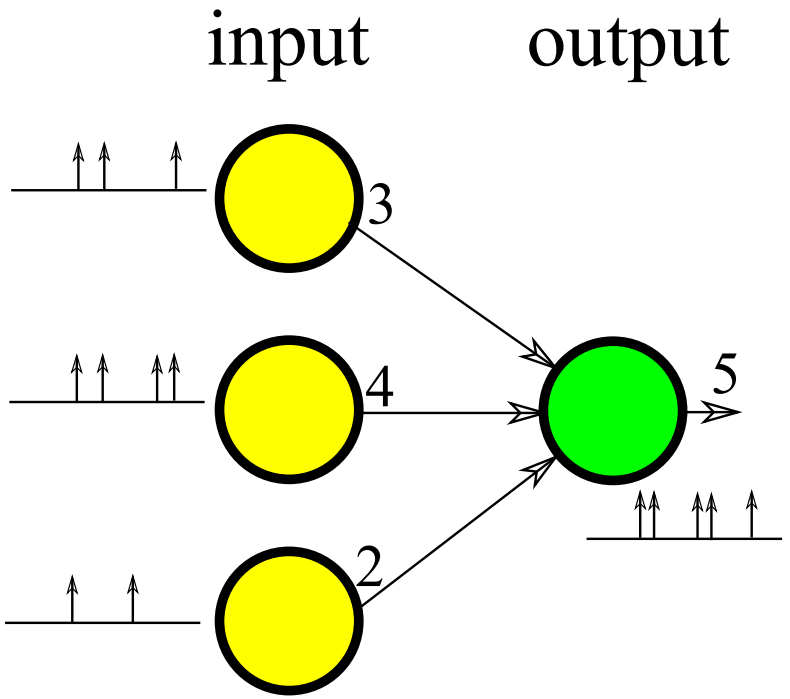




Neural codes and gradient descent

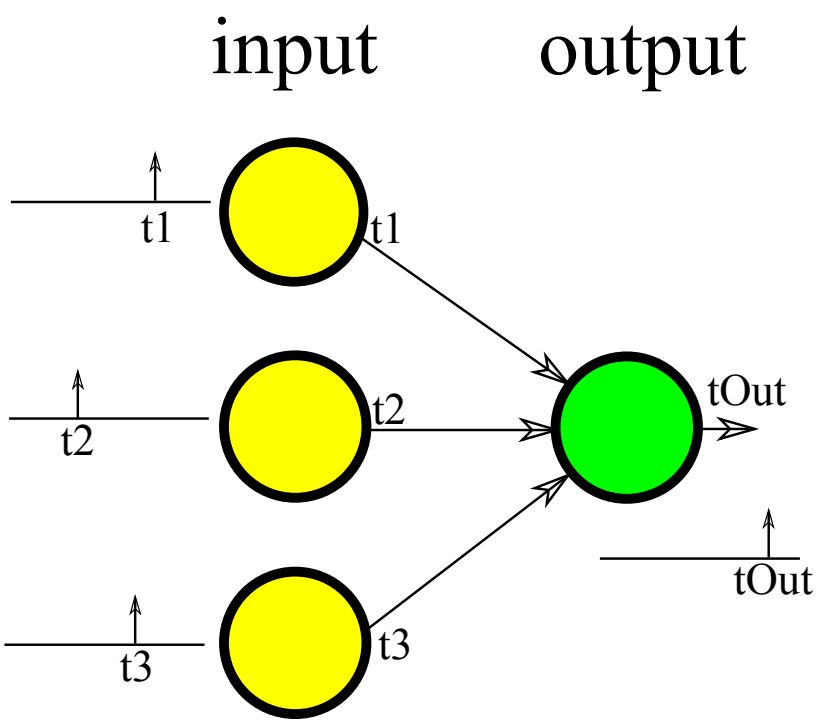
Rate coding:

- Spike counts/rates are discrete quantities
- Gradient is zero almost everywhere
- Only indirect or approximate gradient descent training possible



Temporal coding

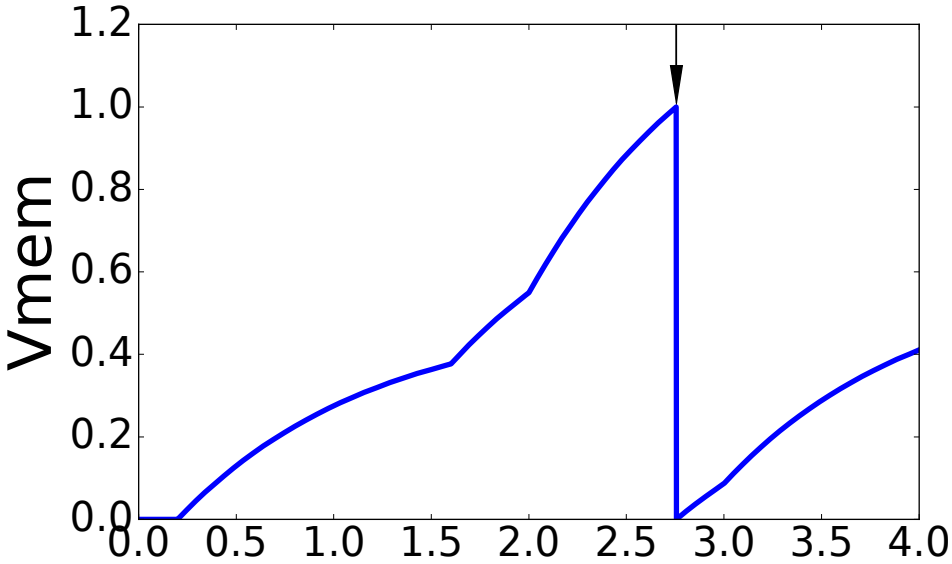
- Spike times are analog quantities
- Gradient of output spike time w.r.t input spike times is well-defined and non-zero
- Direct gradient descent training possible



The neuron model

Non-leaky integrate and fire neuron

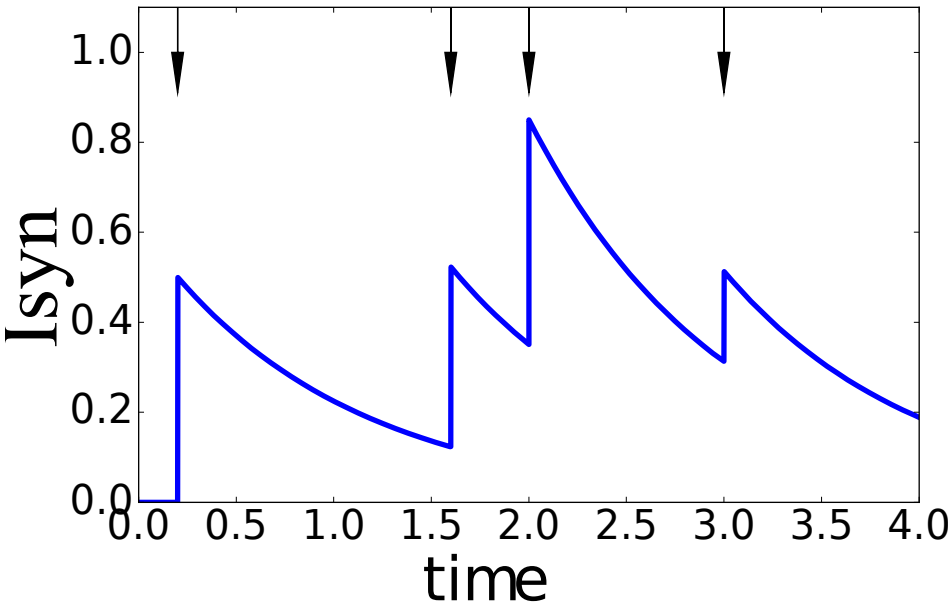
$$\frac{dVmem(t)}{dt} = Isyn(t)$$ (firing threshold is 1)



Exponentially decaying synaptic current

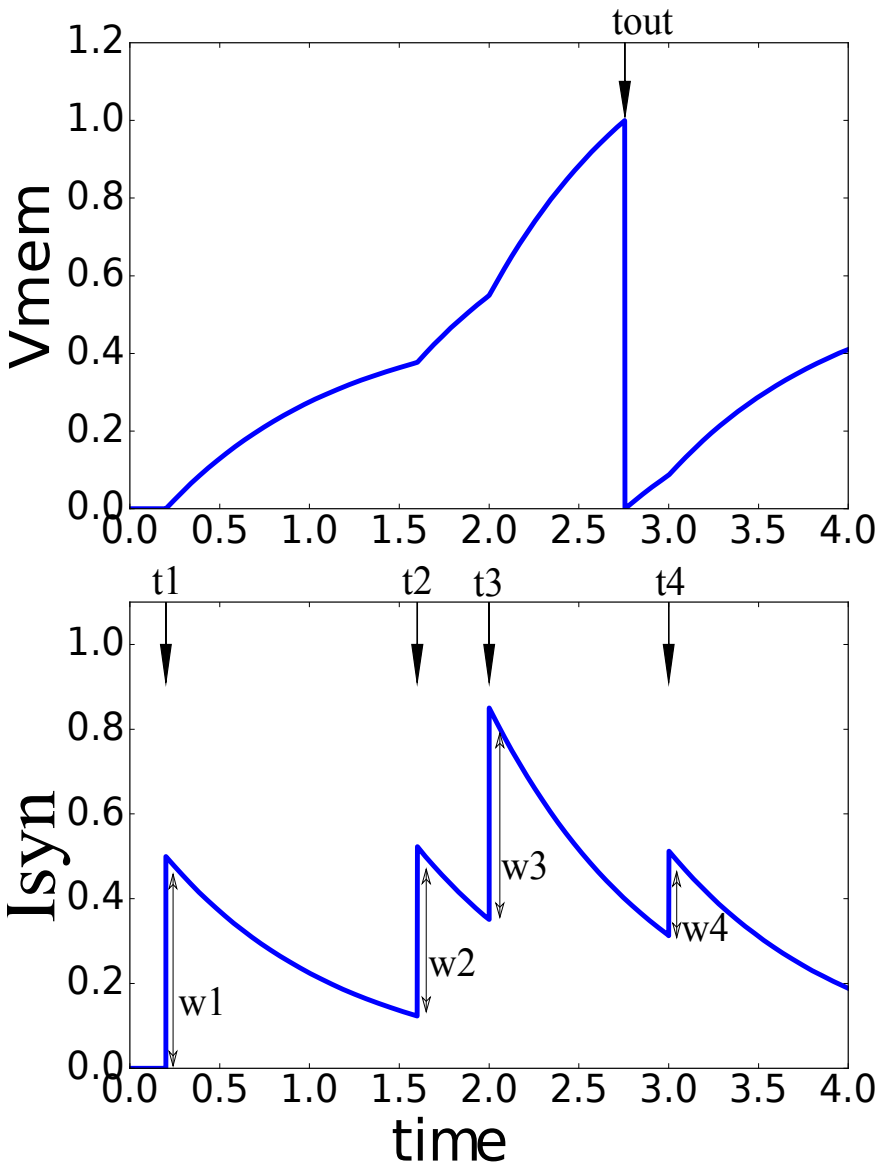
$$Isyn(t) = \sum_i w_i \exp(-(t - t_i))\Theta(t - t_i)$$

$\Theta(t - t_i)$: Step function



The neuron's transfer function

$$exp(t_{out}) = \frac{w_1 exp(t_1) + w_2 exp(t_2) + w_3 exp(t_3)}{w_1 + w_2 + w_3 - 1}$$



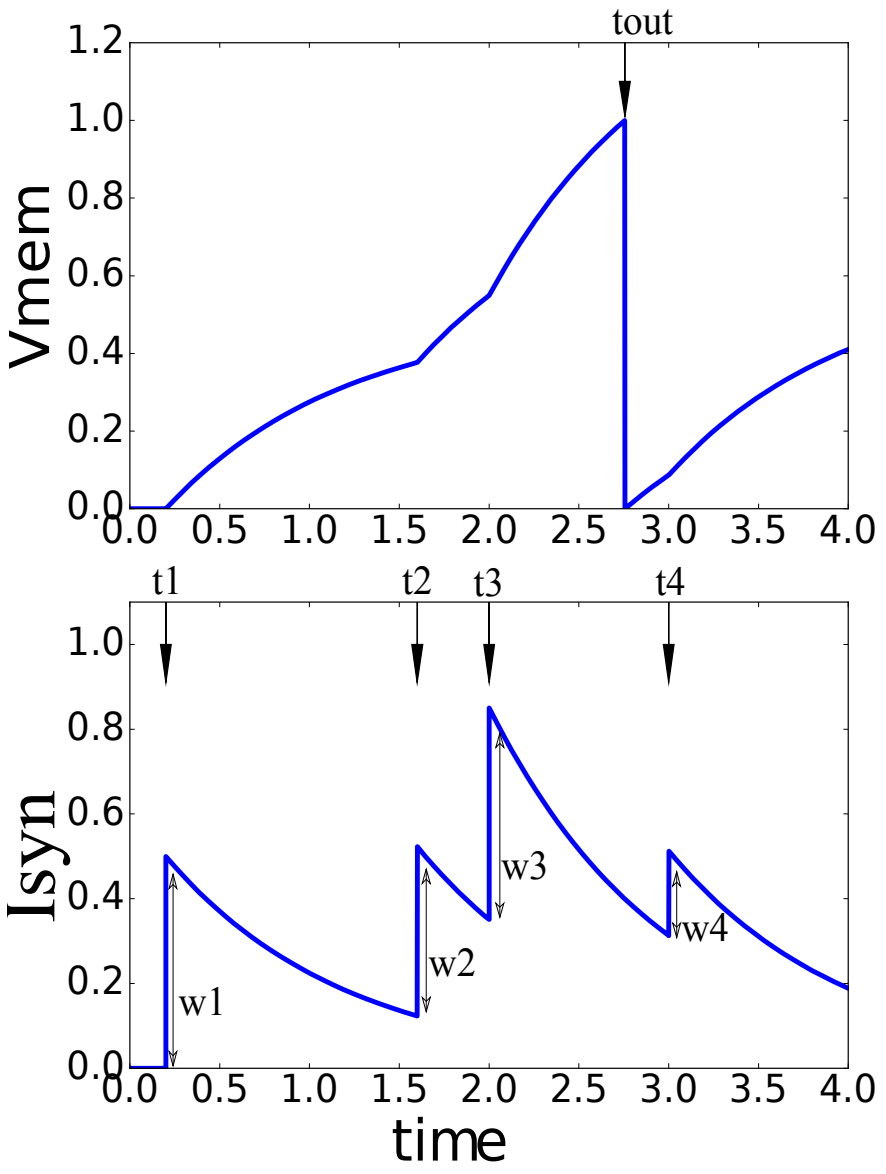
The neuron's transfer function

$$exp(t_{out}) = \frac{w_1 exp(t_1) + w_2 exp(t_2) + w_3 exp(t_3)}{w_1 + w_2 + w_3 - 1}$$

In general:

$$exp(t_{out}) = \frac{\sum_{i \in C} w_i exp(t_i)}{\sum_{i \in C} w_i - 1}$$

Where C is the causal set of input spikes (input spikes that arrive before output spike)



The neuron's transfer function

$$exp(t_{out}) = \frac{w_1 exp(t_1) + w_2 exp(t_2) + w_3 exp(t_3)}{w_1 + w_2 + w_3 - 1}$$

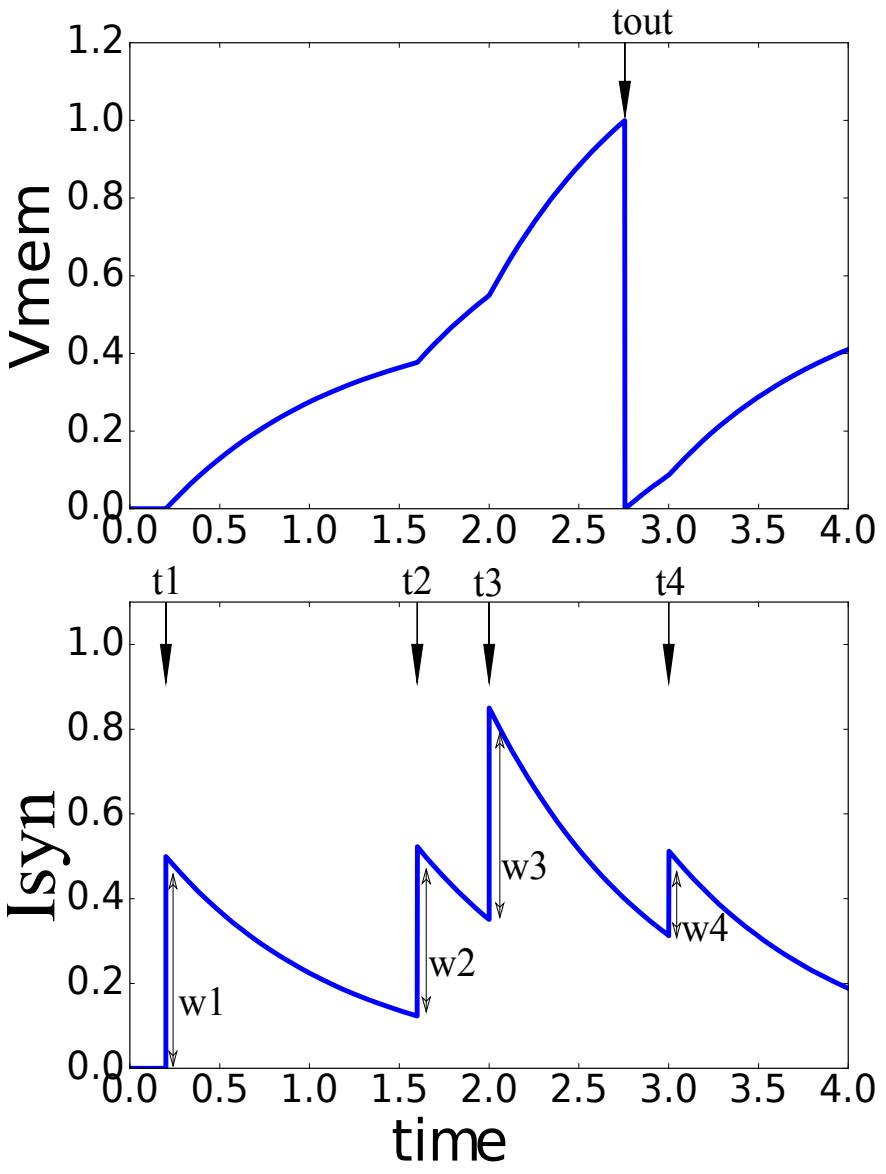
In general:

$$exp(t_{out}) = \frac{\sum_{i \in C} w_i exp(t_i)}{\sum_{i \in C} w_i - 1}$$

Where C is the causal set of input spikes (input spikes that arrive before output spike)

Time of the L^{th} output spike:

$$exp(t^L_{out}) = \frac{\sum_{i \in C} w_i exp(t_i)}{\sum_{i \in C} w_i - L}$$



Change of variables

$$exp(t_x) \rightarrow z_x$$

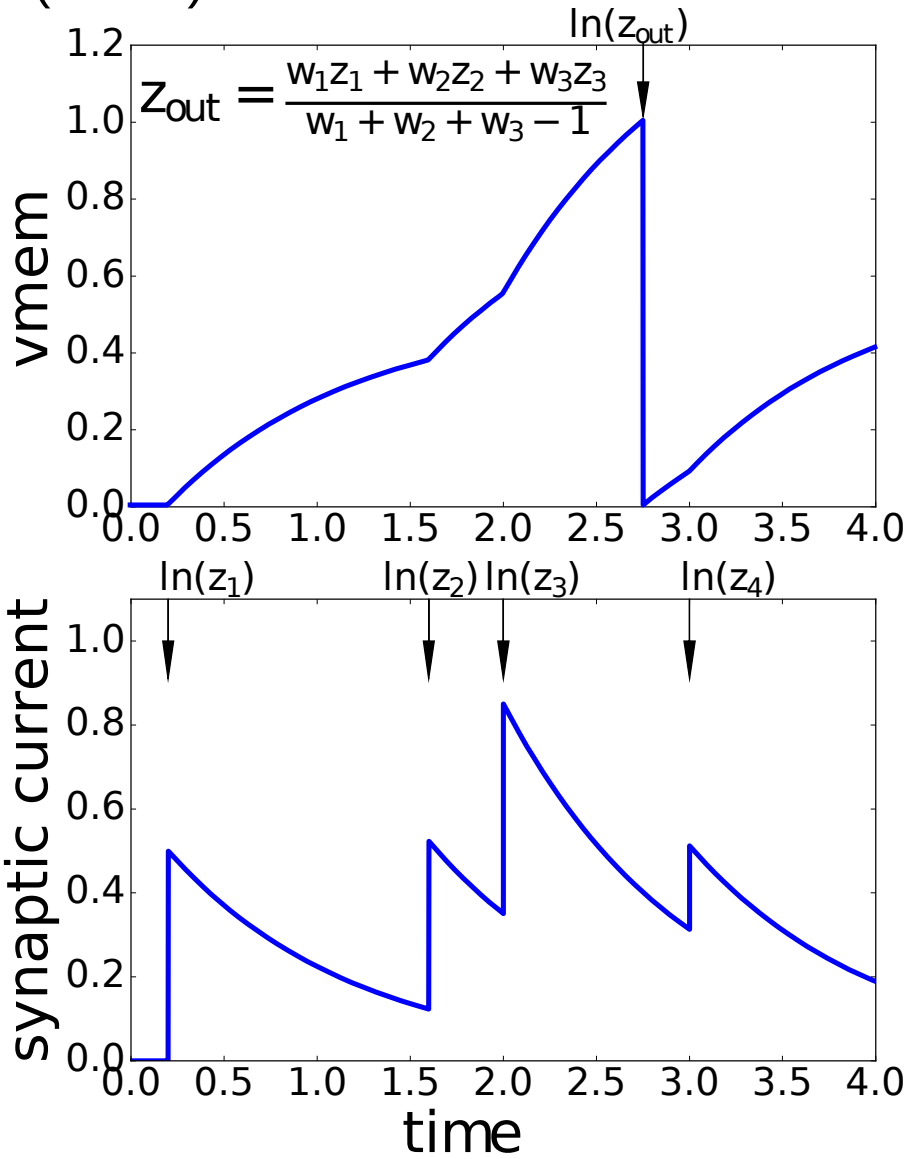
The neuron's transfer function then becomes piece-wise linear in the inputs (but not the weights):

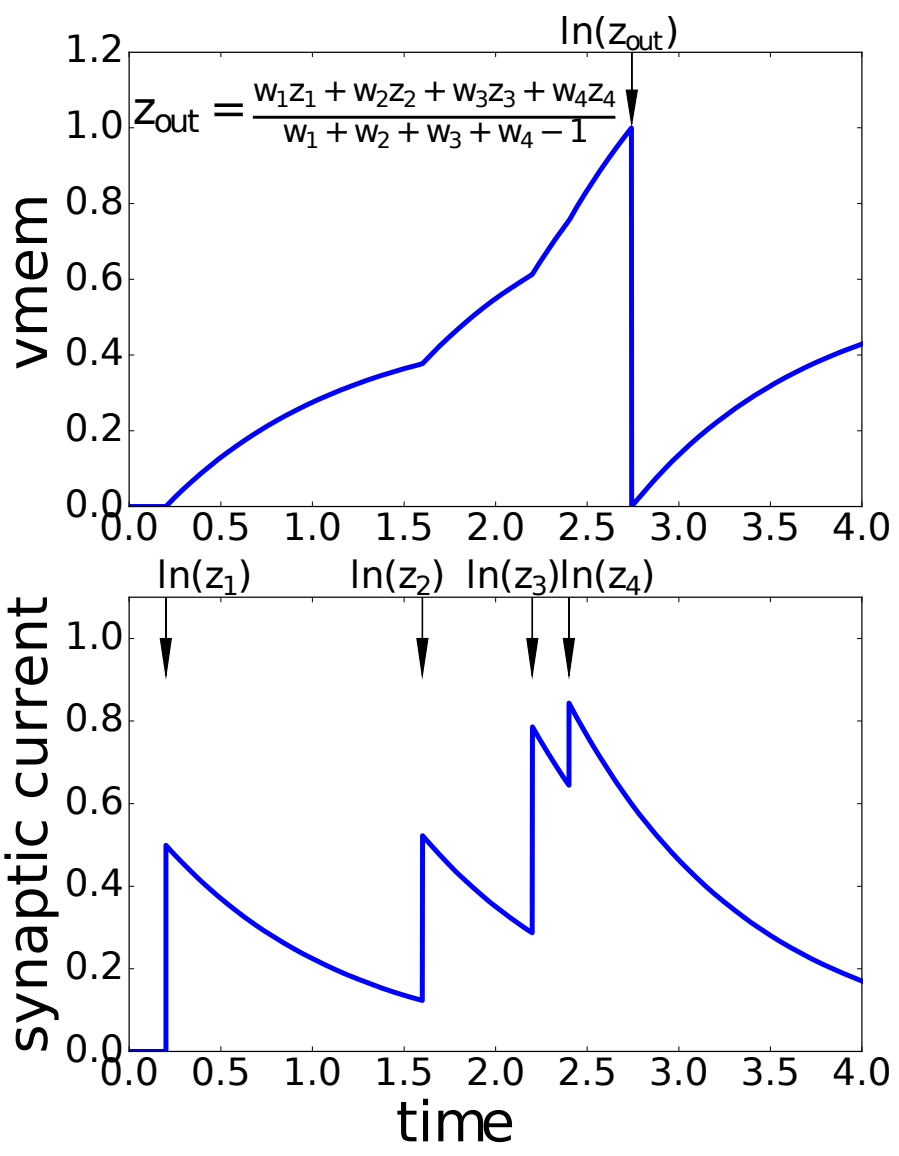
$$z_{out} = \frac{\sum_{i \in C} w_i z_i}{\sum_{i \in C} w_i - 1}$$

Where is the non-linearity?

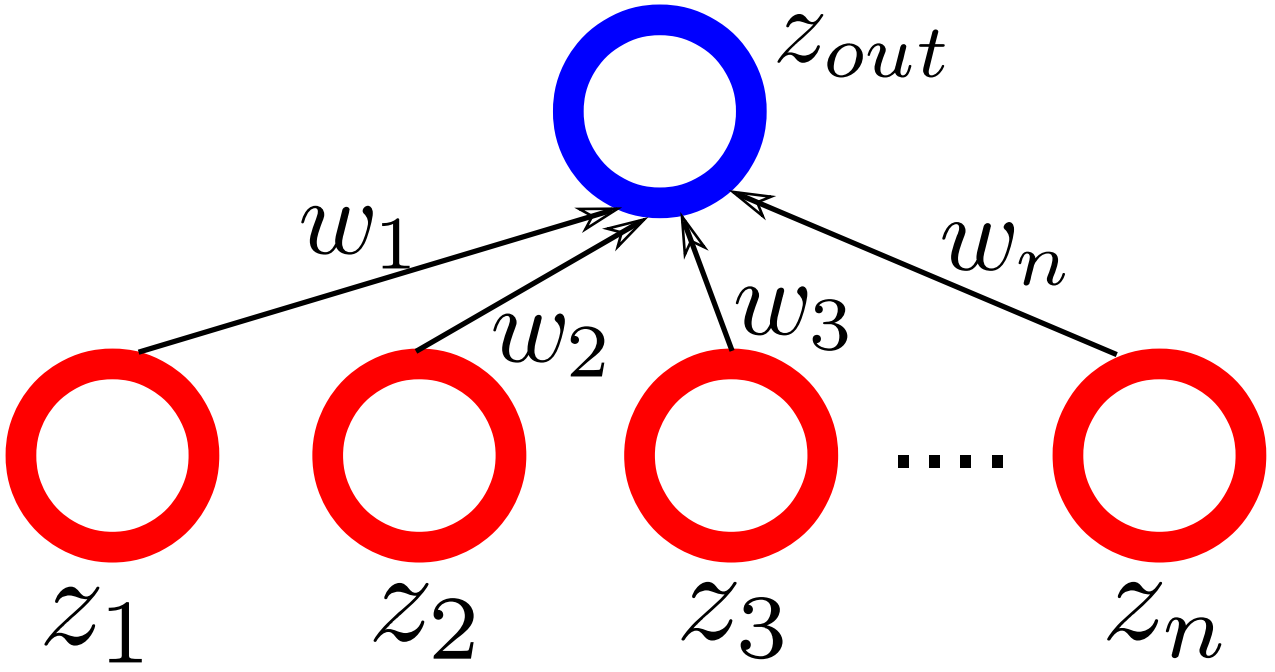
Non-linearity arises due to the input dependence of the causal set of input spikes

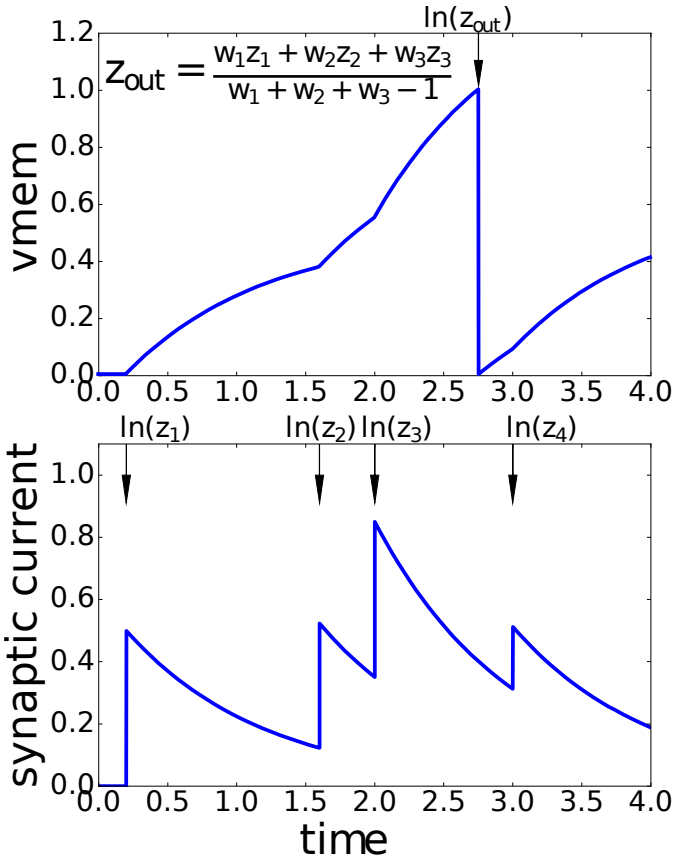
The piecewise linear input-output relation is reminiscent of Rectified Linear Units (ReLU) networks





What is the form of computation implemented by the temporal dynamics?





- To compute zout:
  - Sort {z1,z2,..,zn}
  - Find the causal set, C, by progressively considering more early spikes
  - Calculate $z_{out} = \sum_{i \in C} w_i z_i / (\sum_{i \in C} w_i - 1)$

Can not be reduced to the conventional ANN neuron: $z_{out} = f\left(\sum_i w_i z_i\right)$

Backpropagation

$$z_{out} = \frac{\sum_{i \in C} w_i z_i}{\sum_{i \in C} w_i - 1}$$

To use backpropagation to train a multi-layer network, we need the derivatives of the neuron's output w.r.t:

Weights

$$\frac{dz_{out}}{dw_p} = \begin{cases} \frac{z_p - z_{out}}{\sum_{i \in C} w_i - 1} & \text{if} \quad p \in C \\ 0 & \text{otherwise} \end{cases}$$

Inputs

$$\frac{dz_{out}}{dz_p} = \begin{cases} \frac{w_p}{\sum_{i \in C} w_i - 1} & \text{if} \quad p \in C \\ 0 & \text{otherwise} \end{cases}$$

Time of first spike encodes neuron's value. Each neuron is allowed to spike only once in response to an input pattern:

- Forces sparse activity. Training has to make maximum use of each spike
- Allows quick classification response

Classification Tasks

- We can relate the time of any spike differentiably to the times of all spikes that caused it
- We can impose any differentiable cost function on the spike times of the output layer and use backpropagation to minimize cost across training set
- In a classification setting, use a loss function that encourages the output neuron representing the correct class to spike first
- Since we have an analytical input-output relation for each neuron, training can be done using conventional machine learning packages (Theano/Tensorflow)

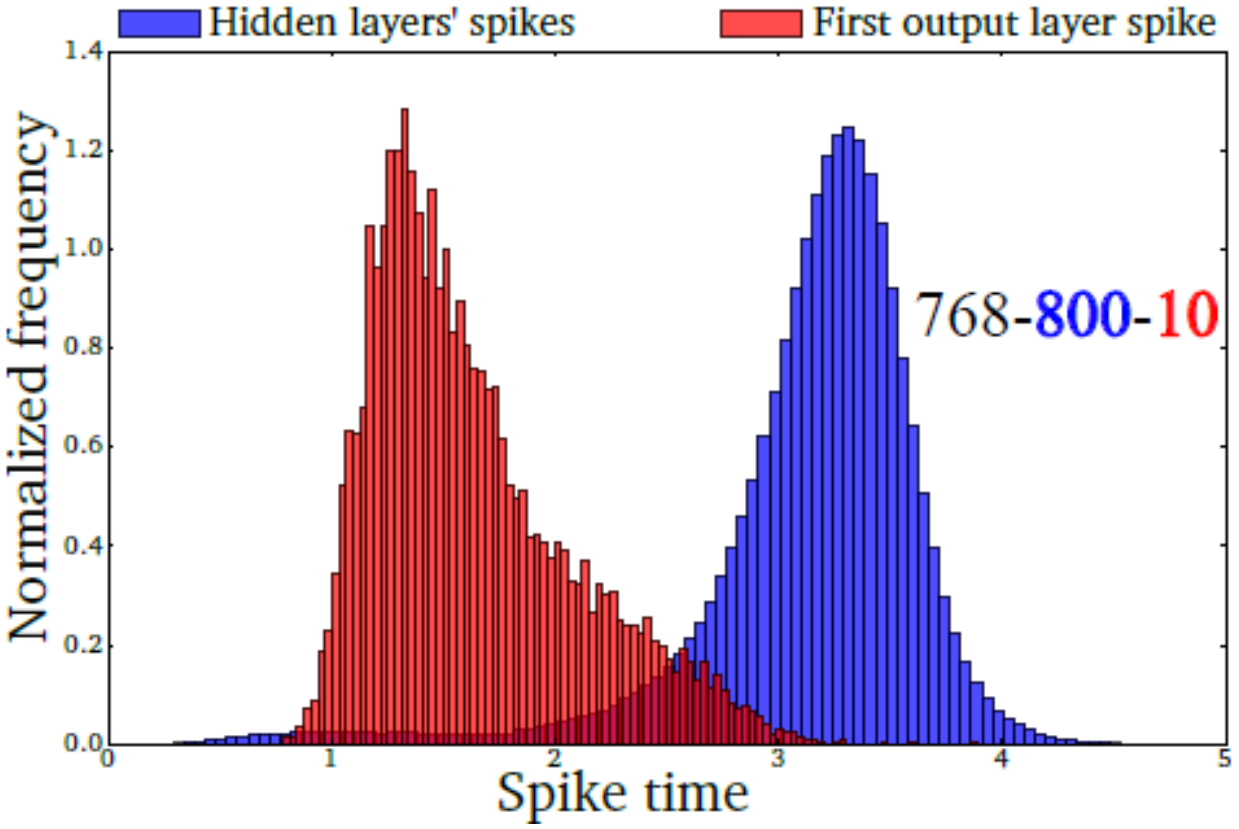
MNIST task

- Pixel values were binarized.
- High intensity pixels spike early
- Low intensity pixels spike late

Table 1: Performance results for the permutation-invariant MNIST task

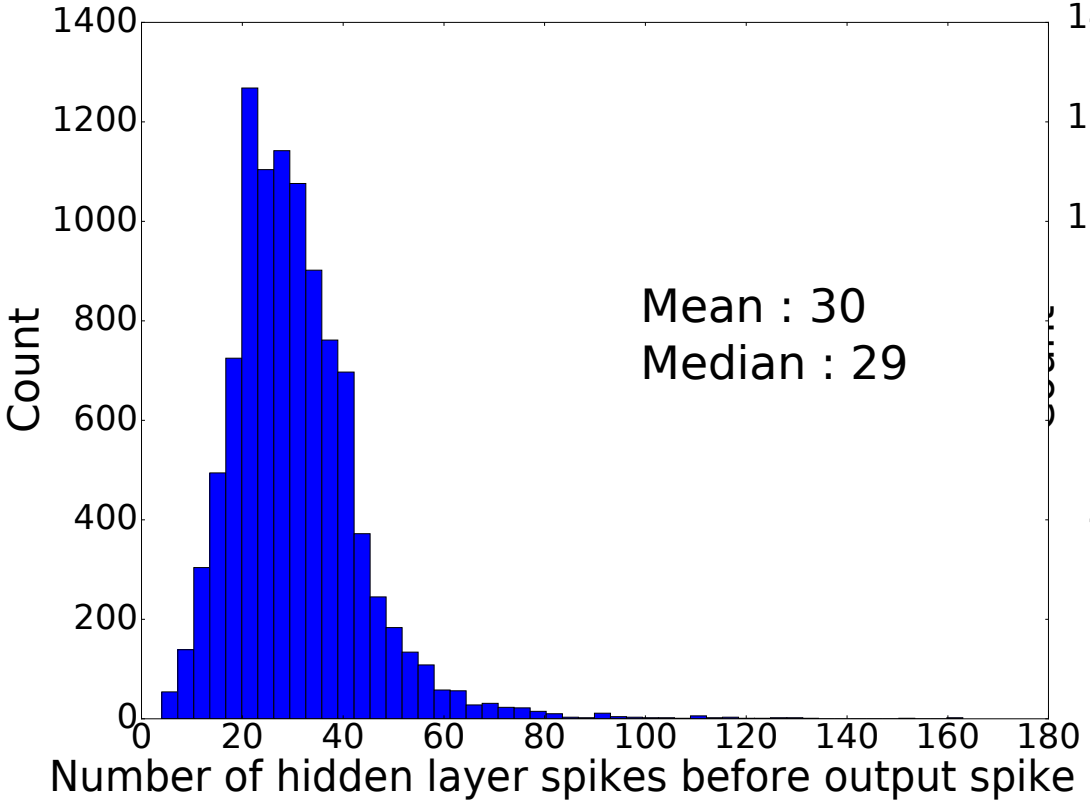
Network	Training set error	Test set error
784-800-10 (non-noisy training input)	0.013%	2.8%
784-800-10 (noisy training input)	0.005%	2.45%

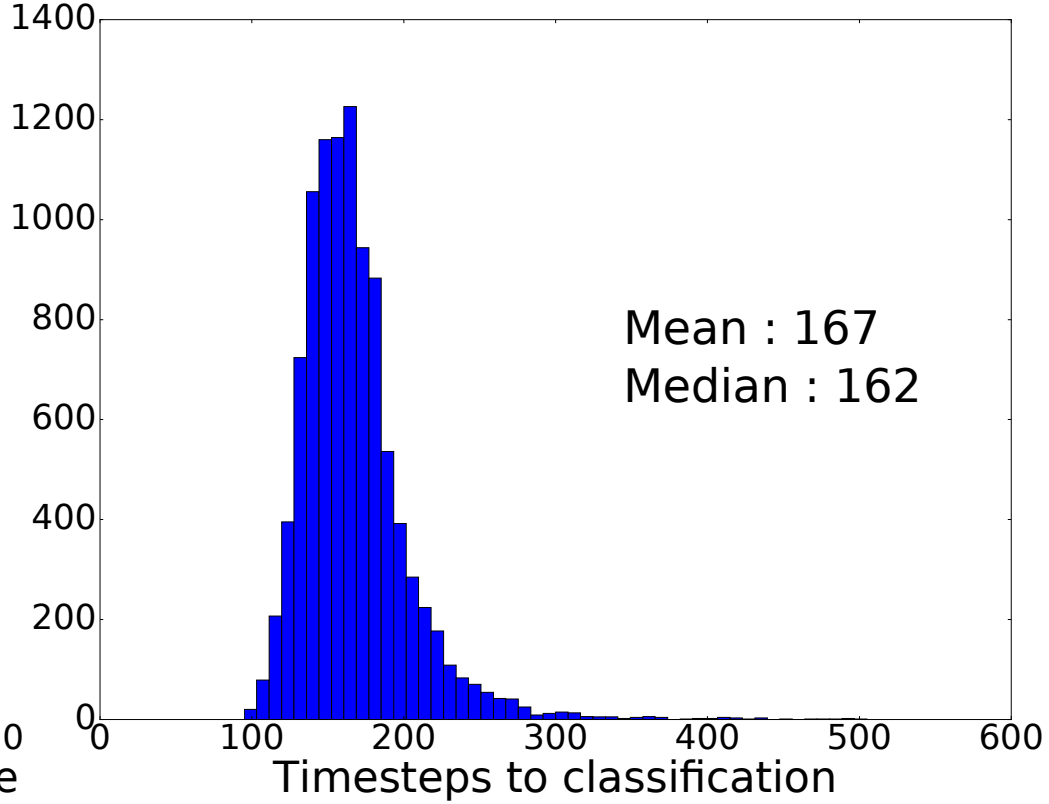
Classification is extremely rapid



- A decision is made when the first output neuron spikes
- A decision is made after only 25 spikes (on average) from the hidden layer in the 768-800-10 network, i.e, only 3% of the hidden layer neurons contribute to each classification decision

FPGA prototype





- 97% test set classification accuracy on MNIST in a 784-600-10 network (8-bit weights)
- Average number of spikes until classification: 139
- Only 13% of input to hidden weights are looked up
- Only 5% of hidden to output weights are looked up

Acknowledgements

University of Zurich[UZH]

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Giacomo Indiveri
Tobi Delbruck

Institute of neuroinformatics

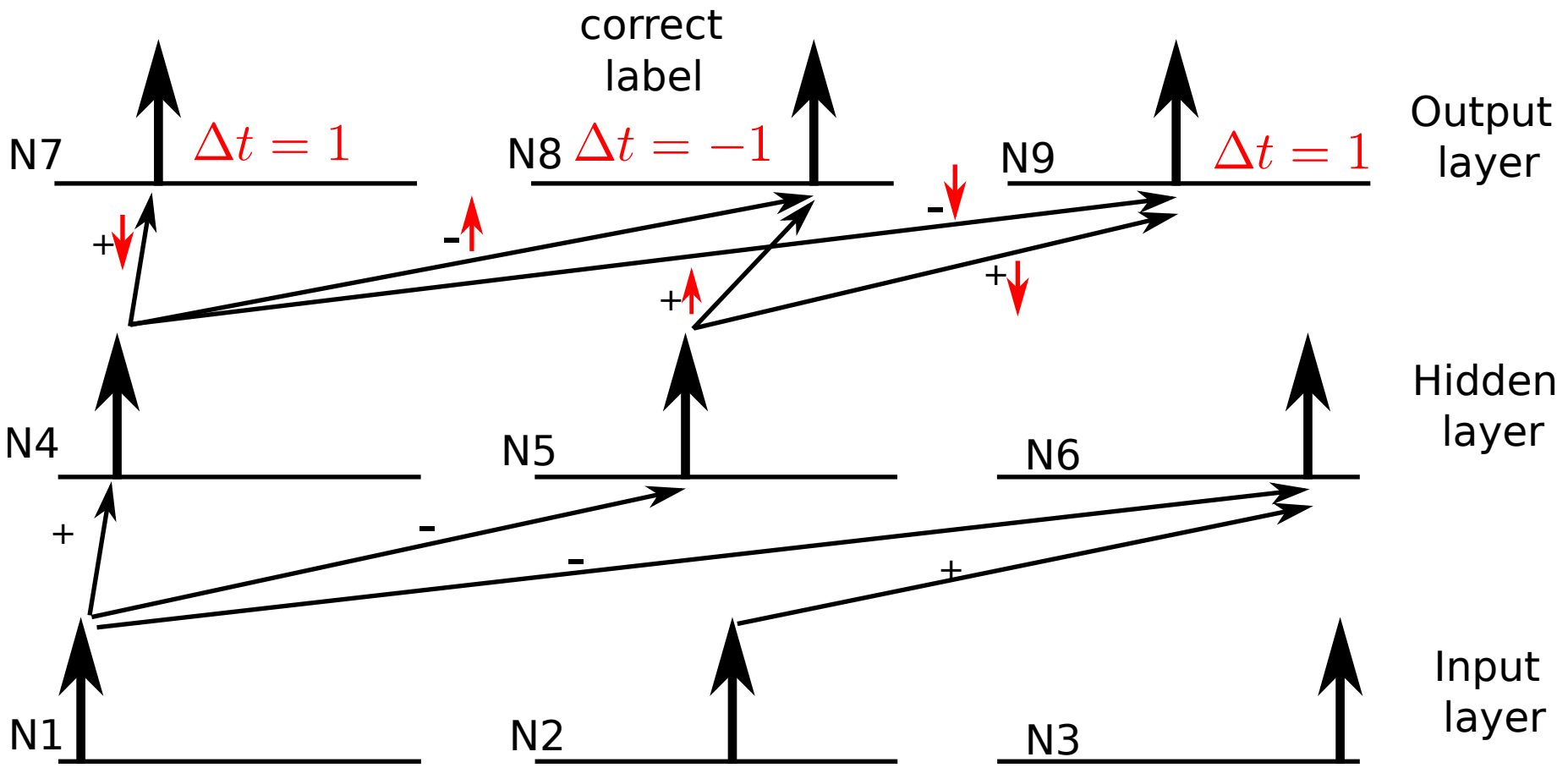
FNSNF
Fonds National Suisse
Schweizerischer Nationalfonds
Fondo Nazionale Svizzero
Swiss National Science Foundation

UCSD

INC

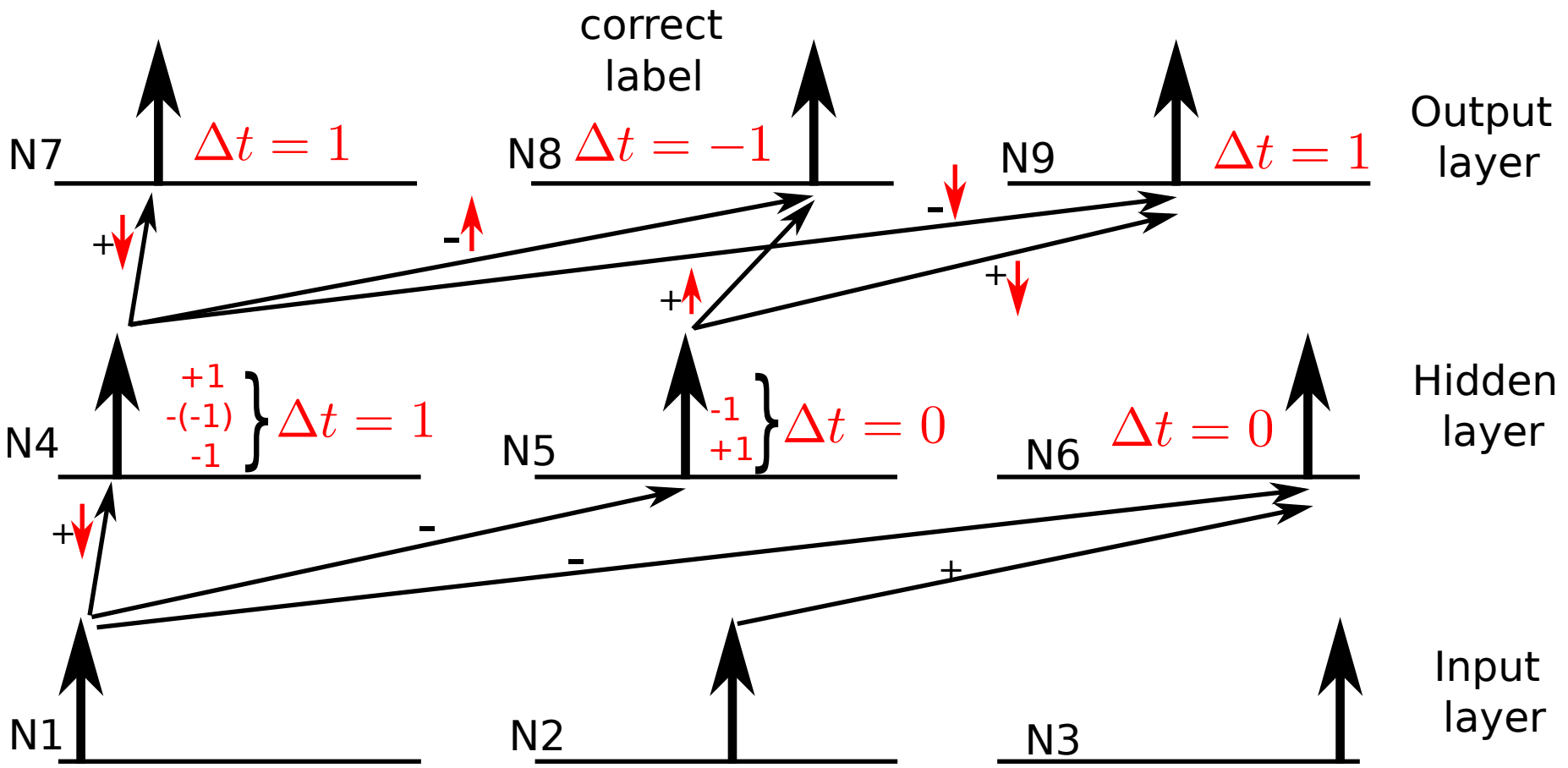
Gert Cauwenberghs
Sadique Sheik
Bruno Pedroi

Approximate learning



- Update Hidden→Output weights to encourage the right neuron to spike first
- Only update weights that actually contributed to output timings

Approximate learning



- Backpropagate time deltas using only the sign of the weights
- The final time delta at a hidden layer neuron can be obtained using 2 parallel popcount operations (count 1s in a bit vector) and a comparison.