Design of Oscillatory Neural Networks by Machine Learning Algorithms

Tamás Rudner,[1] György Csaba,[1] and Wolfgang Porod[2]

[1]Pazmany Peter Catholic University, Faculty of Information Technology and Bionics, Budapest, Hungary

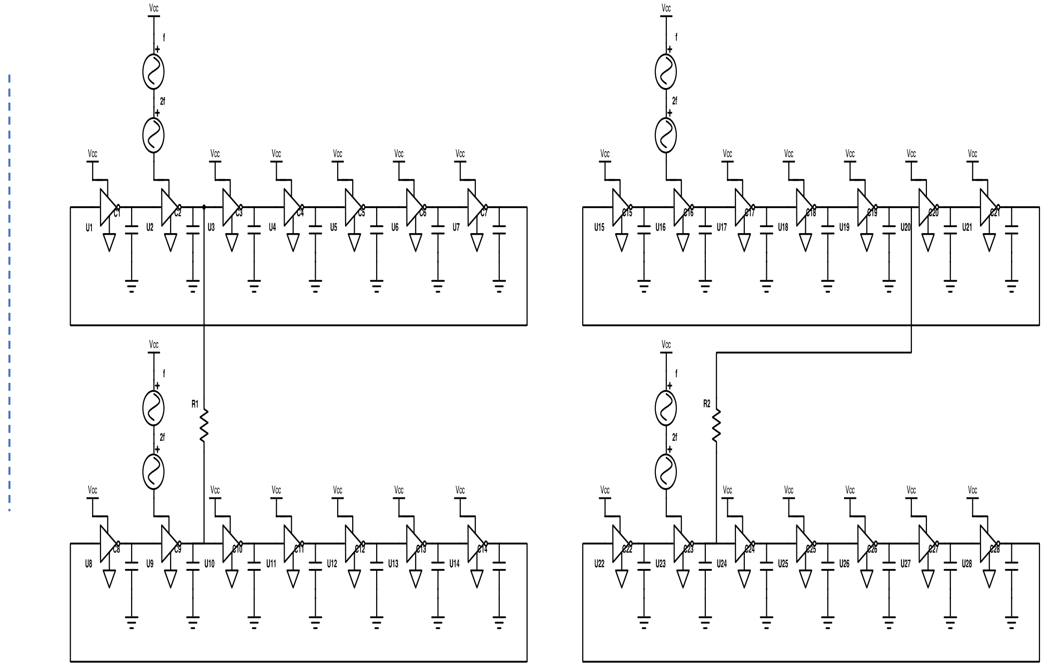
[2]Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, USA

UNIVERSITY OF NOTRE DAME NDnano CENTER FOR NANO SCIENCE AND TECHNOLOGY

Using Physics as a Computer – “Let Physics do the Computing”

- **Complex nonlinear dynamics for device functionality**
  - ***There are no design methods for such devices***
  - **We use machine learning to design the hardware**

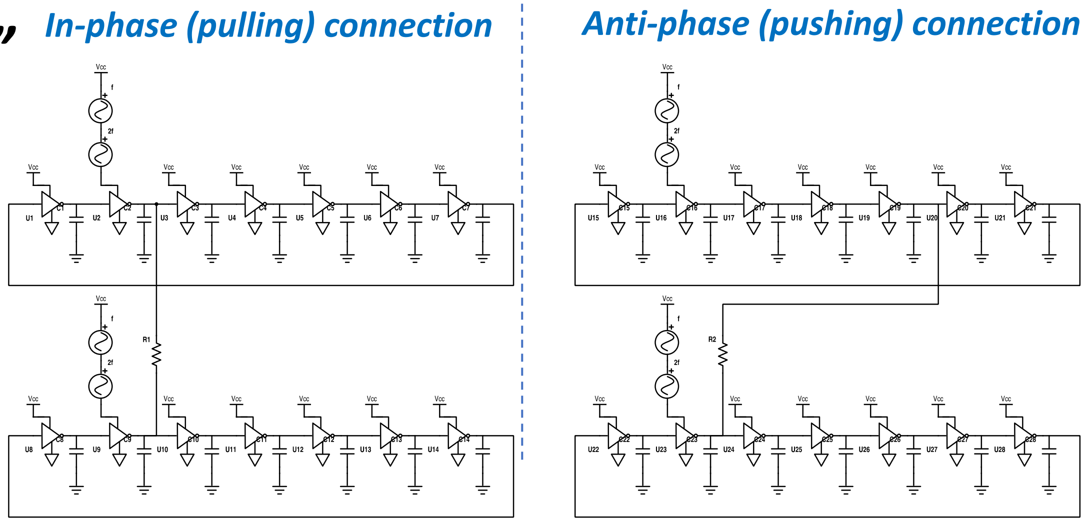
  - **Coupled ring oscillators**



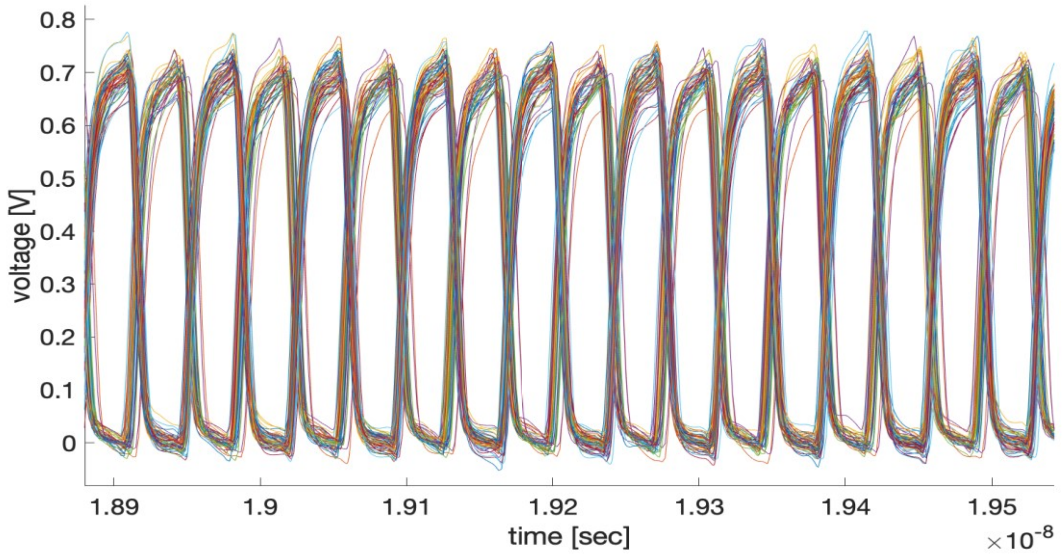
Circuit design by machine learning enables the design of highly energy efficient preprocessors for image processing pipelines

Oscillatory Networks – binary signals represented by phase

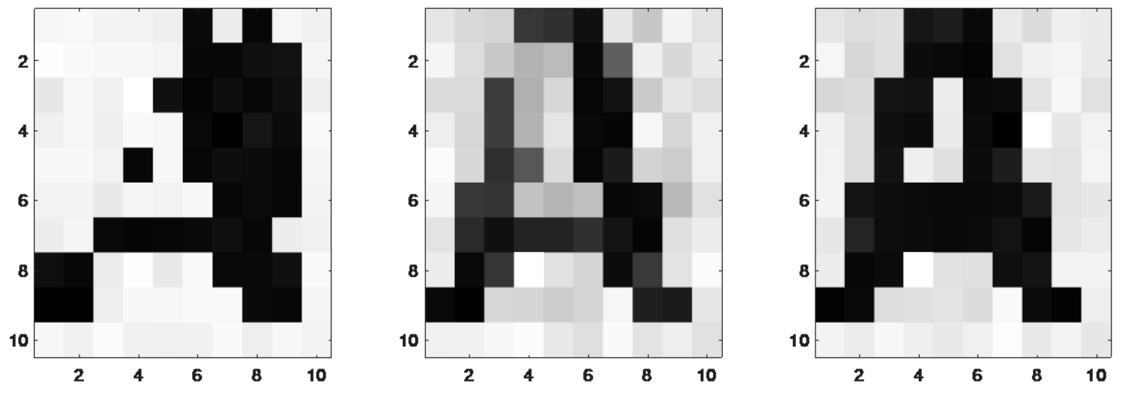
- ***Phase dynamics of oscillators "does the computing"***
- ***In most works, this devices is used as an associative memory, a phase-based Hopfield network***
- ***There is no good learning / design method known to define these couplings***



In-phase (pulling) connection
Anti-phase (pushing) connection



Steady-state phases represent pixel colors in an image



Frank C. Hoppensteadt, and Eugene M. Izhikevich. "Weakly connected oscillators." *Weakly connected neural networks* (1997): 247-293.

Csaba, Gyorgy, and Wolfgang Porod. "Noise immunity of oscillatory computing devices." *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 6, no. 2 (2020): 164-169.

Training based on machine learning

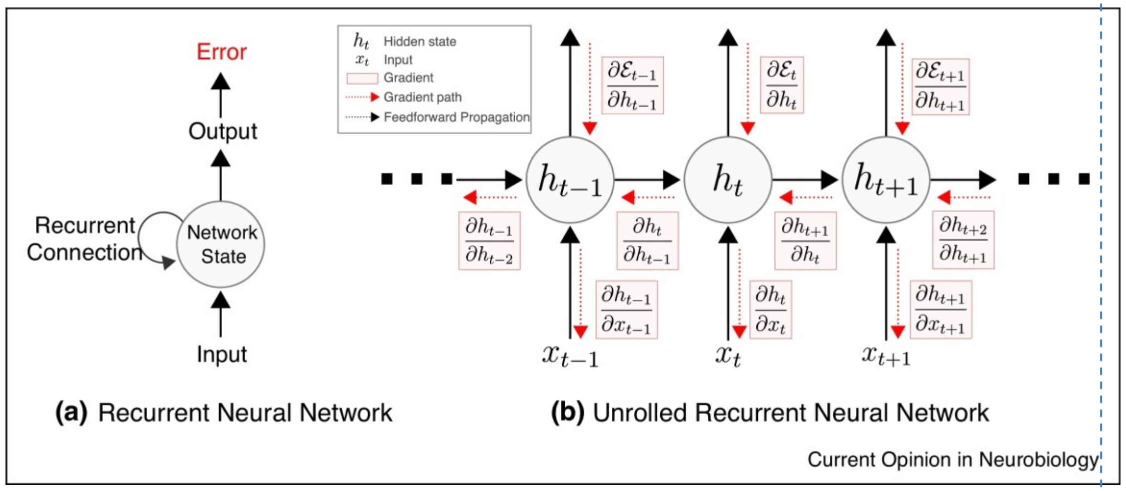
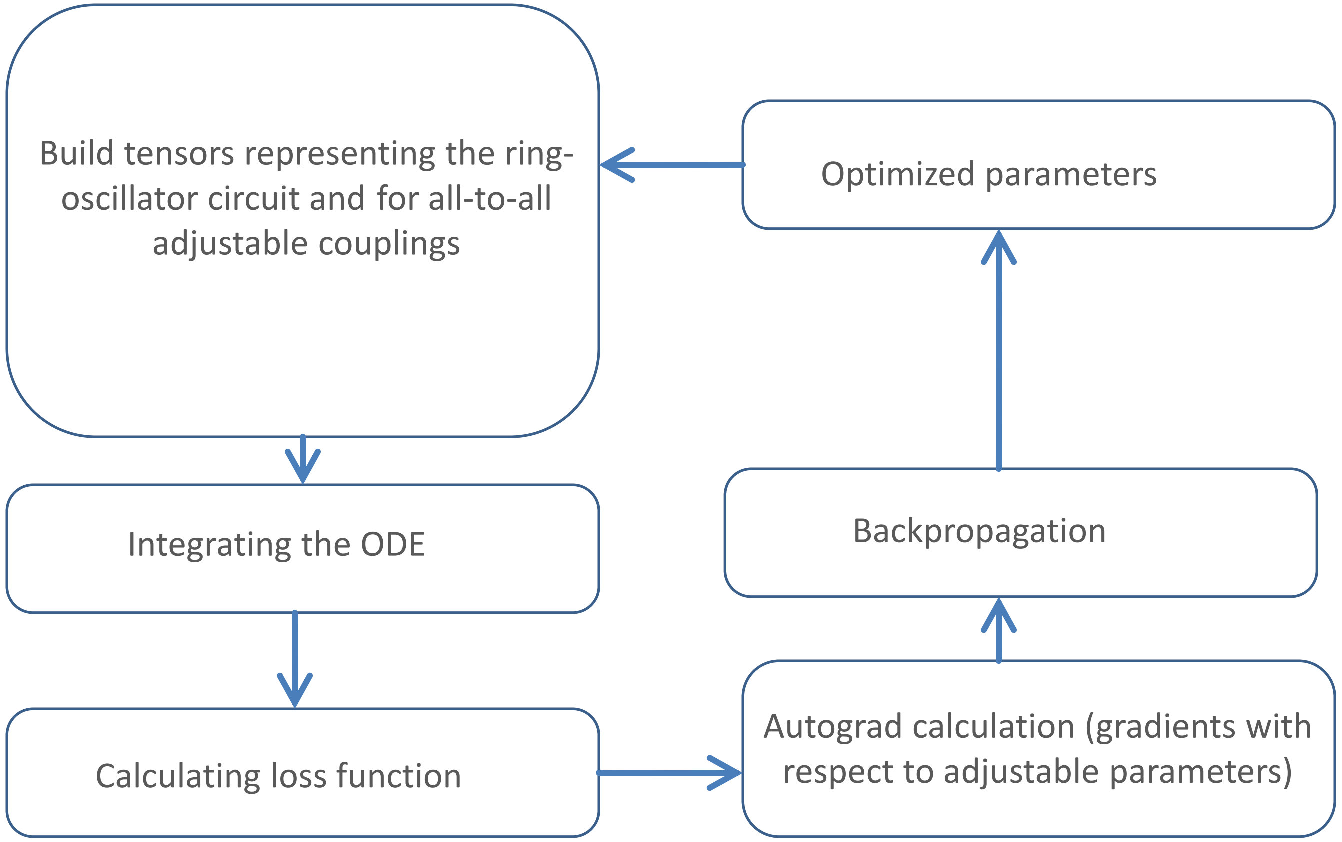
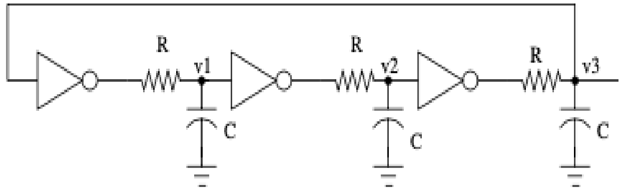

Error
Output
Recurrent Connection
Network State
Input
h_t Hidden state
x_t Input
Gradient
Gradient path
Feedforward Propagation
h_{t-1}
h_t
h_{t+1}
x_{t-1}
x_t
x_{t+1}
$\frac{\partial \mathcal{E}_{t-1}}{\partial h_{t-1}}$
$\frac{\partial \mathcal{E}_t}{\partial h_t}$
$\frac{\partial \mathcal{E}_{t+1}}{\partial h_{t+1}}$
$\frac{\partial h_{t-1}}{\partial h_{t-2}}$
$\frac{\partial h_t}{\partial h_{t-1}}$
$\frac{\partial h_{t+1}}{\partial h_t}$
$\frac{\partial h_{t+2}}{\partial h_{t+1}}$
$\frac{\partial h_{t-1}}{\partial x_{t-1}}$
$\frac{\partial h_t}{\partial x_t}$
$\frac{\partial h_{t+1}}{\partial x_{t+1}}$
(a) Recurrent Neural Network
(b) Unrolled Recurrent Neural Network
Current Opinion in Neurobiology

Illustration of BPTT: Figure from: Lillicrap, Timothy P., and Adam Santoro. "Backpropagation through time and the brain." *Current opinion in neurobiology* 55 (2019): 82-89.

- We used the Pytorch-based torchdiffeq package
- The physical model is a SPICE-like ODE model of a ring oscillator
- Backpropagation through time (BPTT) can be applied to inverse-design or optimize any system that is describable by differential equations.

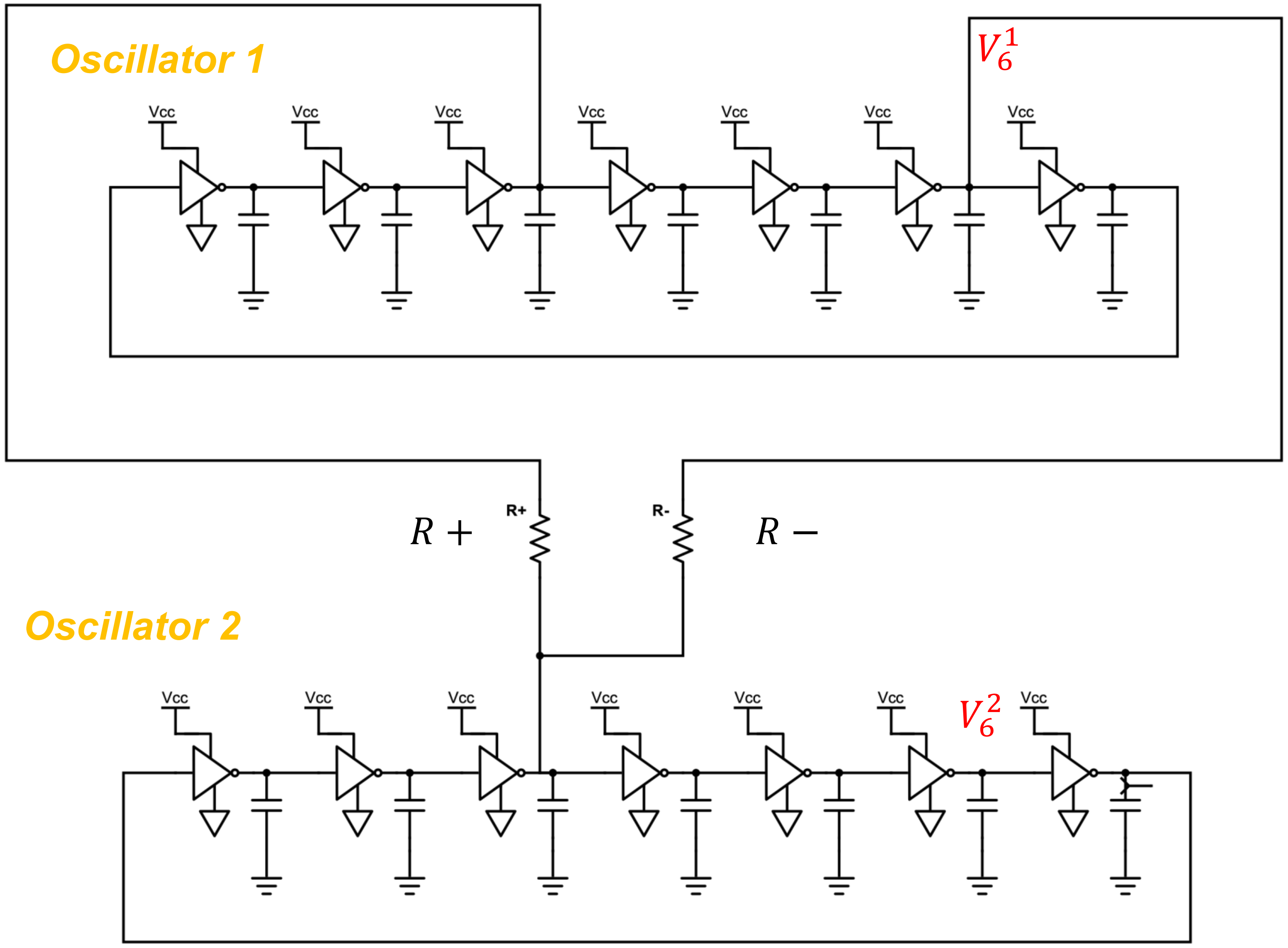

Build tensors representing the ring-oscillator circuit and for all-to-all adjustable couplings
Integrating the ODE
Calculating loss function
Autograd calculation (gradients with respect to adjustable parameters)
Backpropagation
Optimized parameters

$$\dot{v}_1(t) = \frac{f(v_3(t)) - v_1(t)}{RC}$$
$$\dot{v}_2(t) = \frac{f(v_1(t)) - v_2(t)}{RC}$$
$$\dot{v}_3(t) = \frac{f(v_2(t)) - v_3(t)}{RC},$$



Lai, Xiaolue, and Jaijeet Roychowdhury. "Analytical equations for predicting injection locking in LC and ring oscillators." In *Proceedings of the IEEE 2005 Custom Integrated Circuits Conference, 2005.*, pp. 461-464. IEEE, 2005.

A simple two-oscillator model



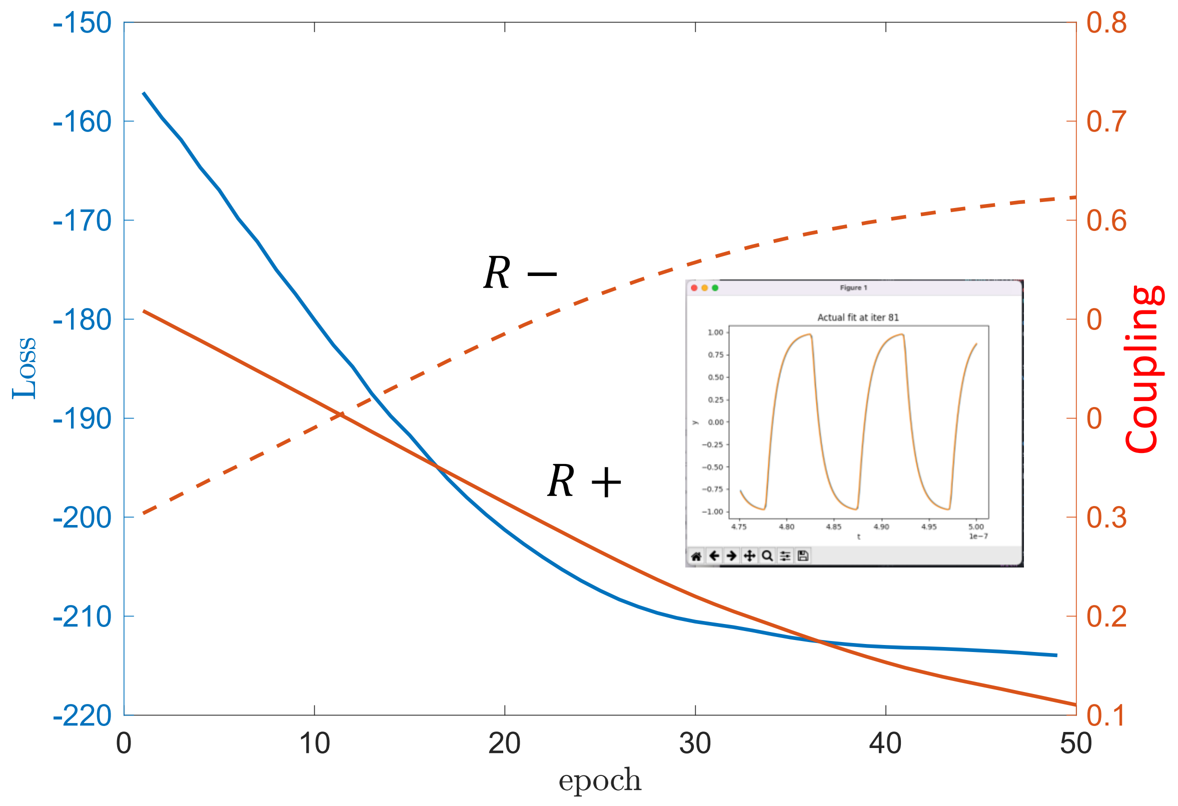
Loss function (objective function):

For in-phase couplings:
Maximize $\mathbf{correlation}(\mathbf{V_6^1}, \boldsymbol{V_6^2})$

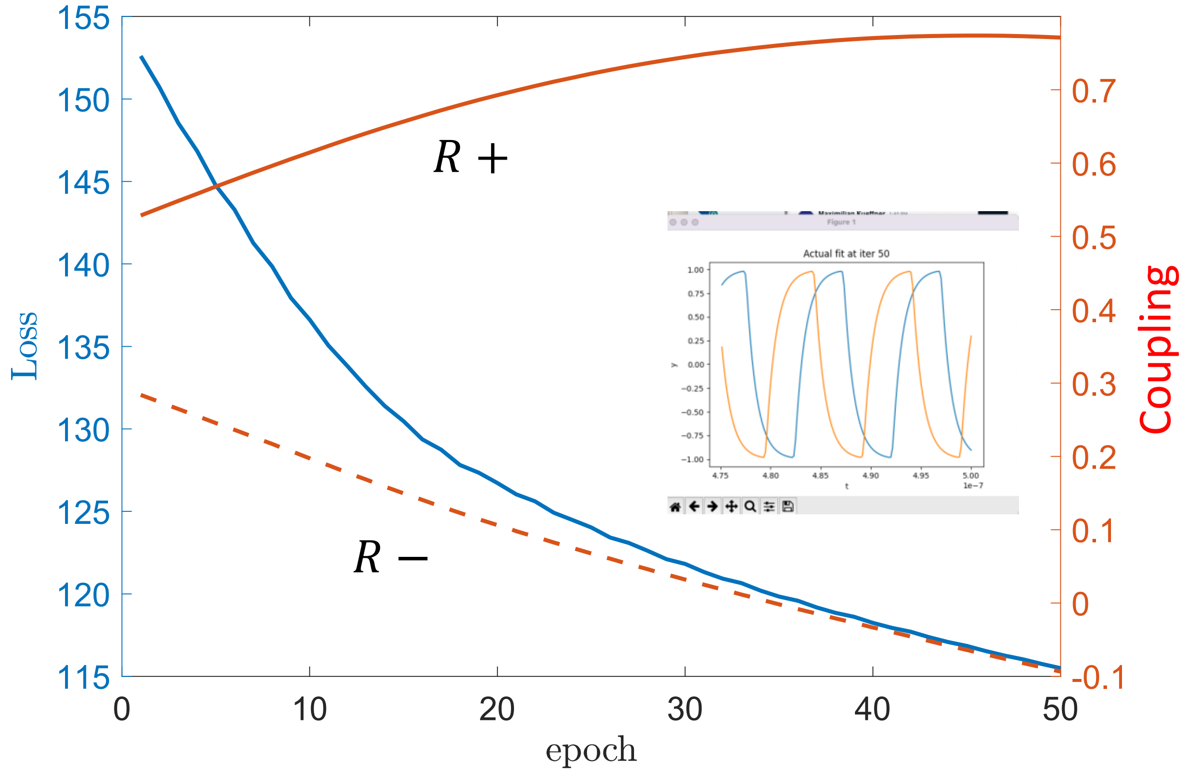
For anti-phase couplings:
Minimize $\mathbf{correlation}(\mathbf{V_6^1}, \boldsymbol{V_6^2})$

Loss function: corr($v_{1,6}$,$v_{2,6}$)
→ in-phase: max
→ anti-phase: min

A simple two-oscillator model



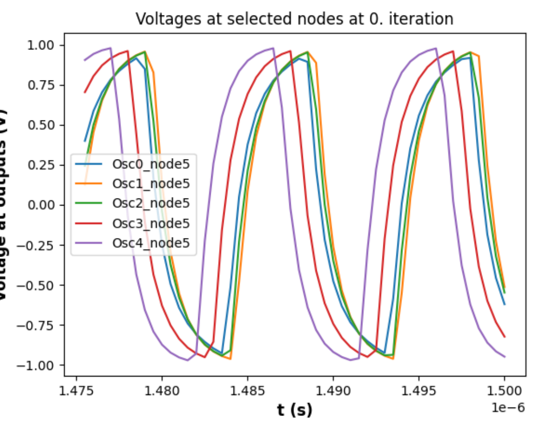
In-phase coupling learned

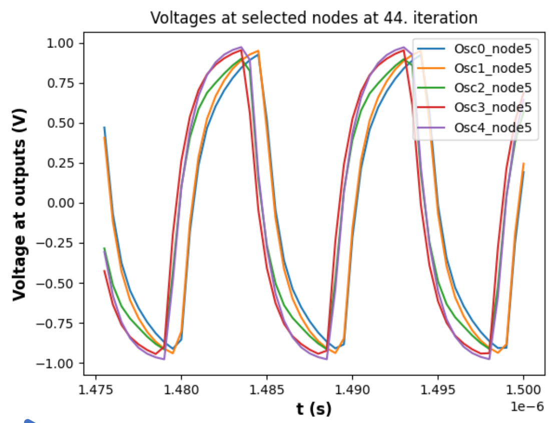


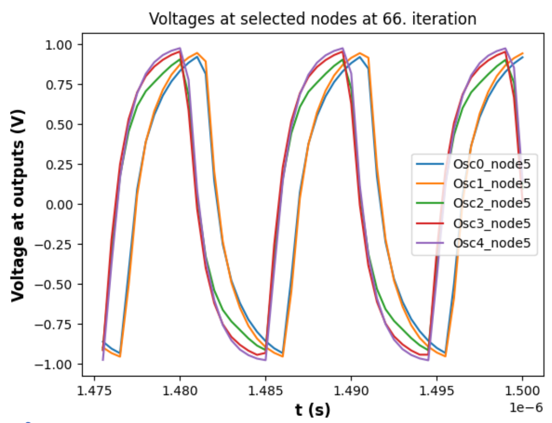
Anti-phase coupling learned

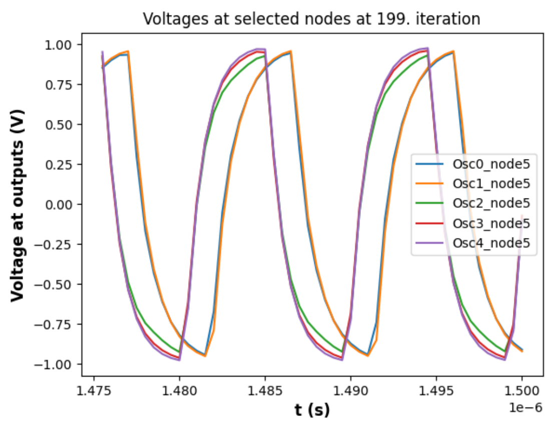
The Pytorch-based solver adjusts the coupling strength to achieve the desired phase pattern.

Convergence to more complex patterns (5 ROs)



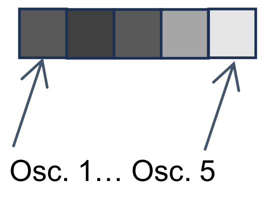


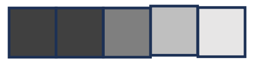


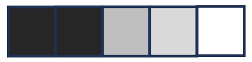


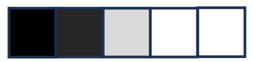
Initial state: arbitrary phases

Final epoch: converged state



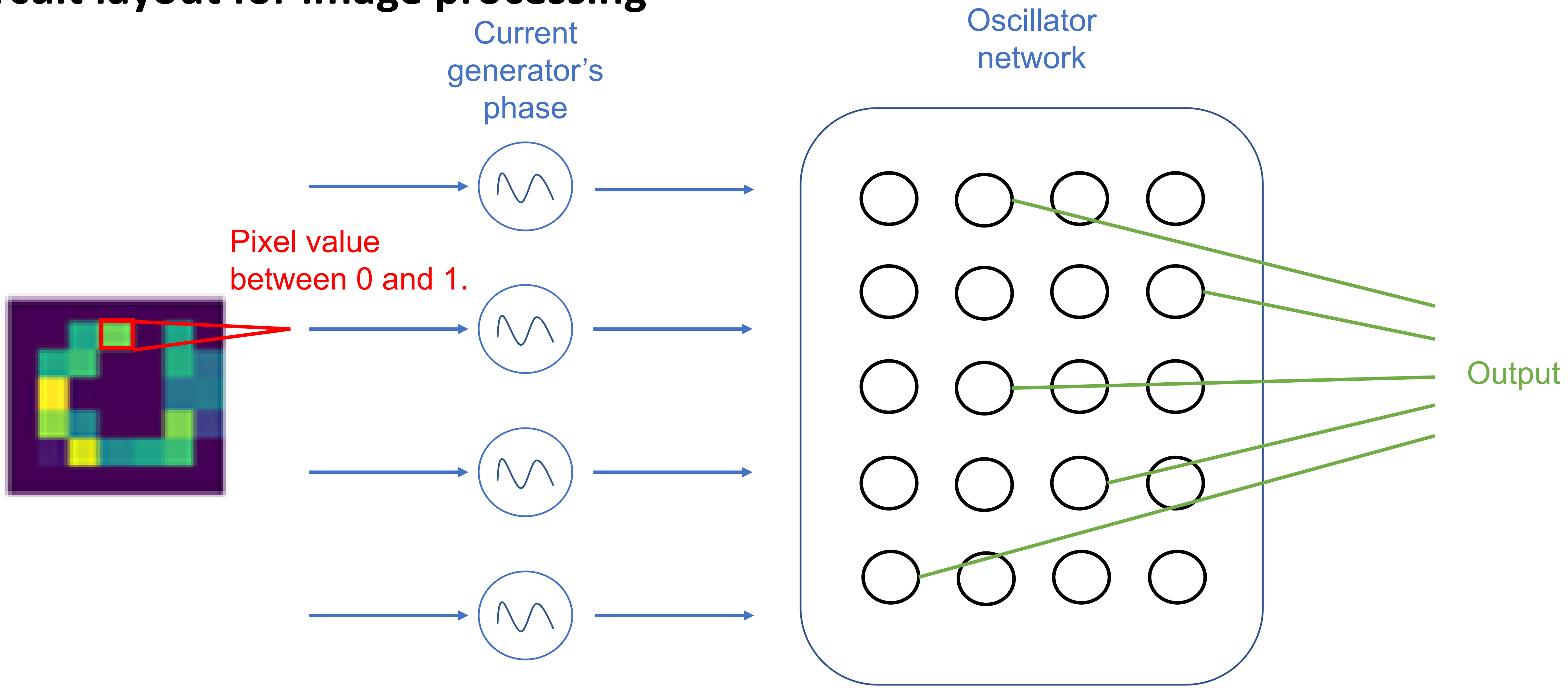






Loss function is defined as mean squared error of the ground truth (corresponding to the binary pattern 1,1,0,0,0). Phases are referenced to one of the oscillators. The pixel colors correspond to images, and we achieve convergence to specified image patterns.

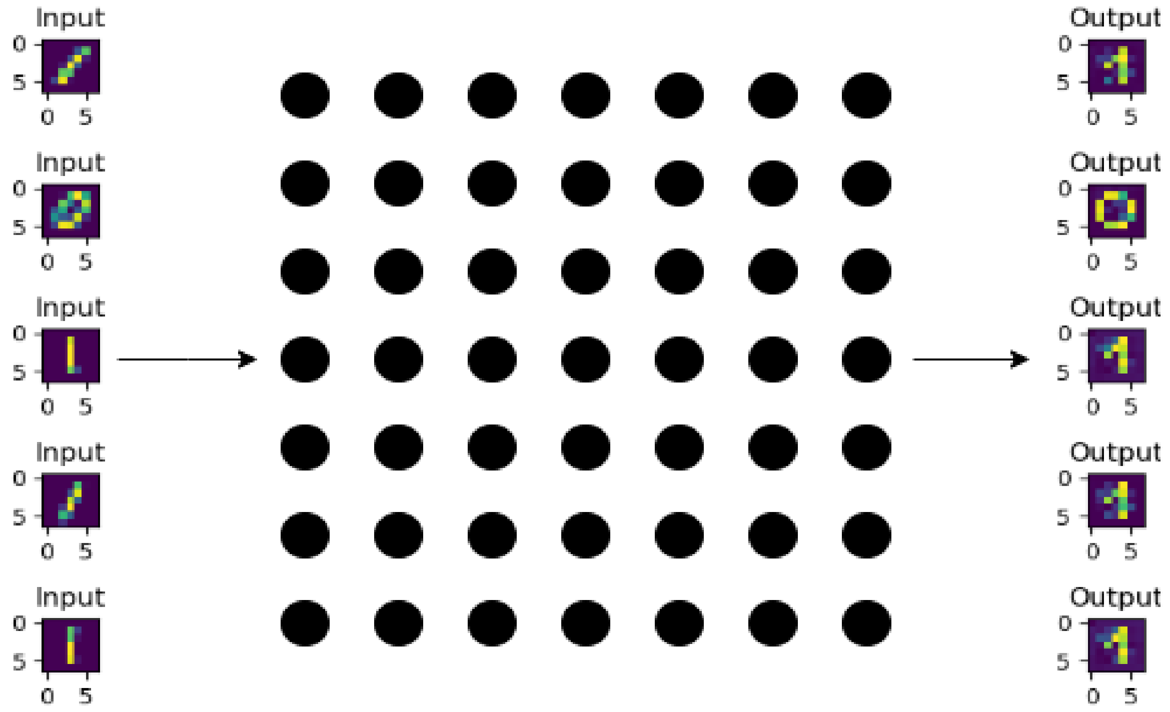
Circuit layout for image processing



For an N-pixel image, all N oscillators receive a phase that corresponds to the input greyscale image. The phase dynamics of the network converges to an image corresponding to one of learned patterns.

Associative memory

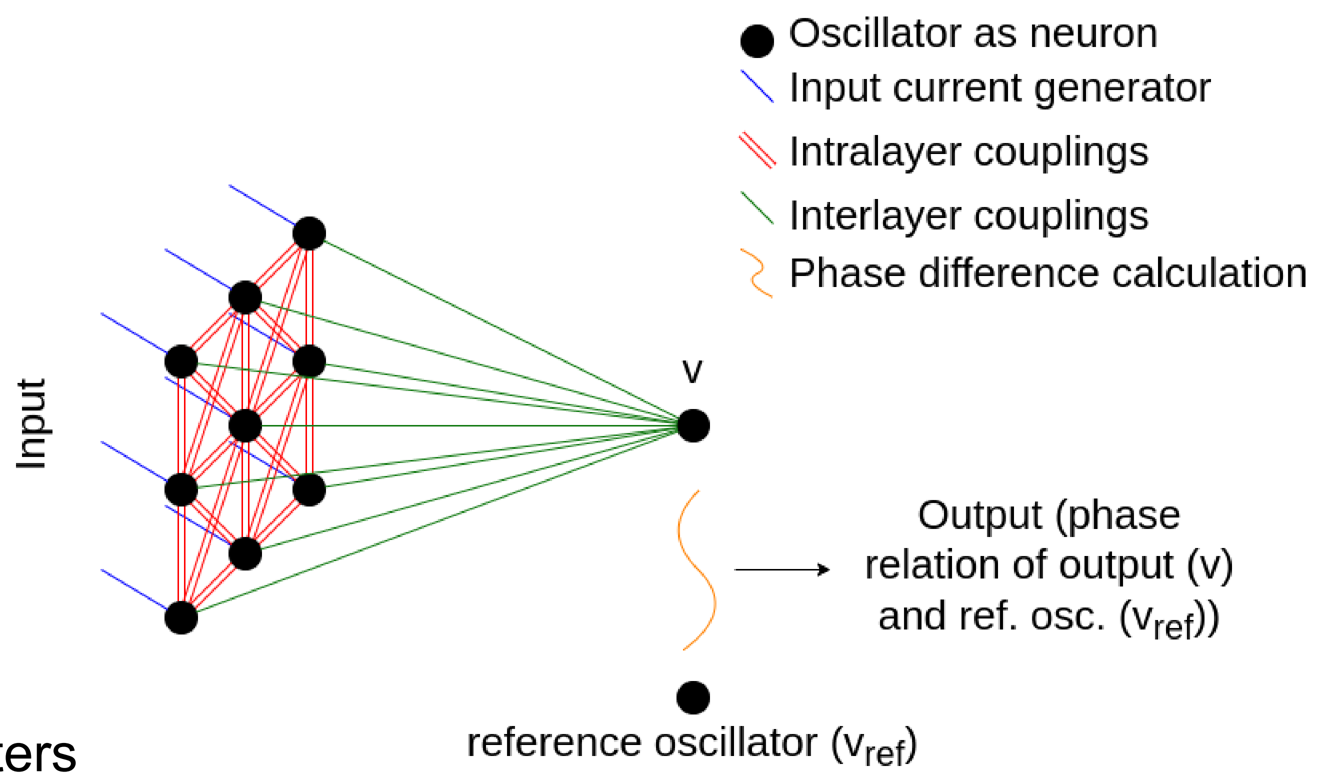
- *Goal: converge from handwritten digits to target images*
- *Machine learning works on various interconnect topologies (fully connected vs. nearest neighbor)*
- Test set: MNIST images (downsampled $28 \times 28 \longrightarrow 7 \times 7$)
- Fully connected single layer:
  - 49x48 = 2,352 parameters
  - *Hebbian Learning*
- Near-neighbor connected single layer:
  - 4x3 + 20x5 + 25x8 = 312 parameters
  - *Machine Learning*

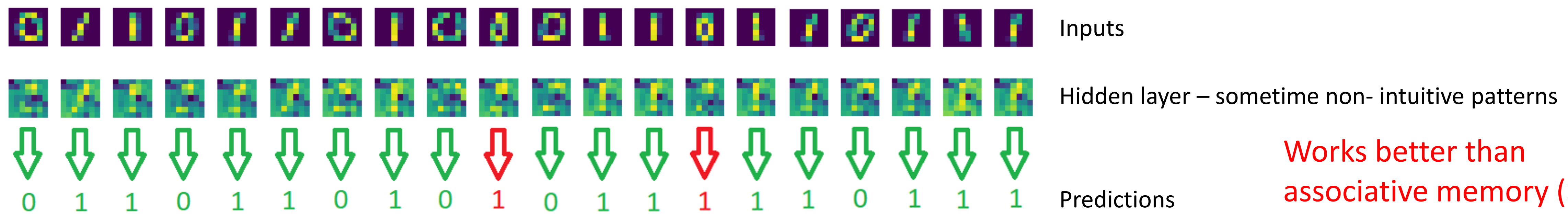


Over 95% accuracy on 2 digits (0, 1) of MNIST – Machine-learned near-neighbor-coupled network performs better than fully connected Hebbian network.

Beyond Associative Memory

- Loss function: maximize output for class 1, and minimize for class 0 (binary cross entropy)
- 14x14 downsampled MNIST images
- 2 layers (hidden - 196 ROs; output – 1 RO)
- Fully connected hidden layer:
  - 196x195 + 196 = 38,416 parameters
- Near-neighbor connected hidden layer:
  - 4x3 + 52*5 + 140*8 + 196 = 1,588 parameters


Oscillator as neuron
Input current generator
Intralayer couplings
Interlayer couplings
Phase difference calculation
Input
v
Output (phase relation of output (v) and ref. osc. (v_{ref}))
reference oscillator (v_{ref})

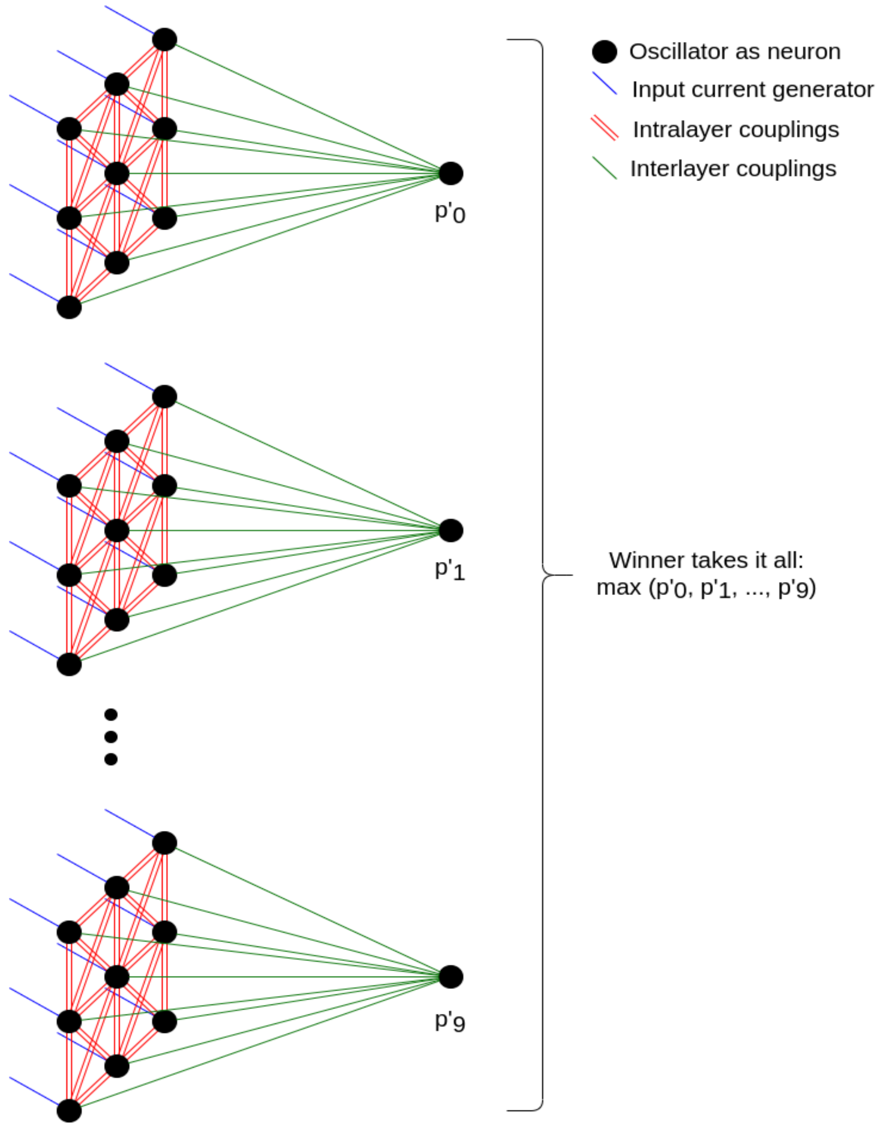


Works better than associative memory (>98%)

Oscillator network learns internal representations without an 'ideal' image

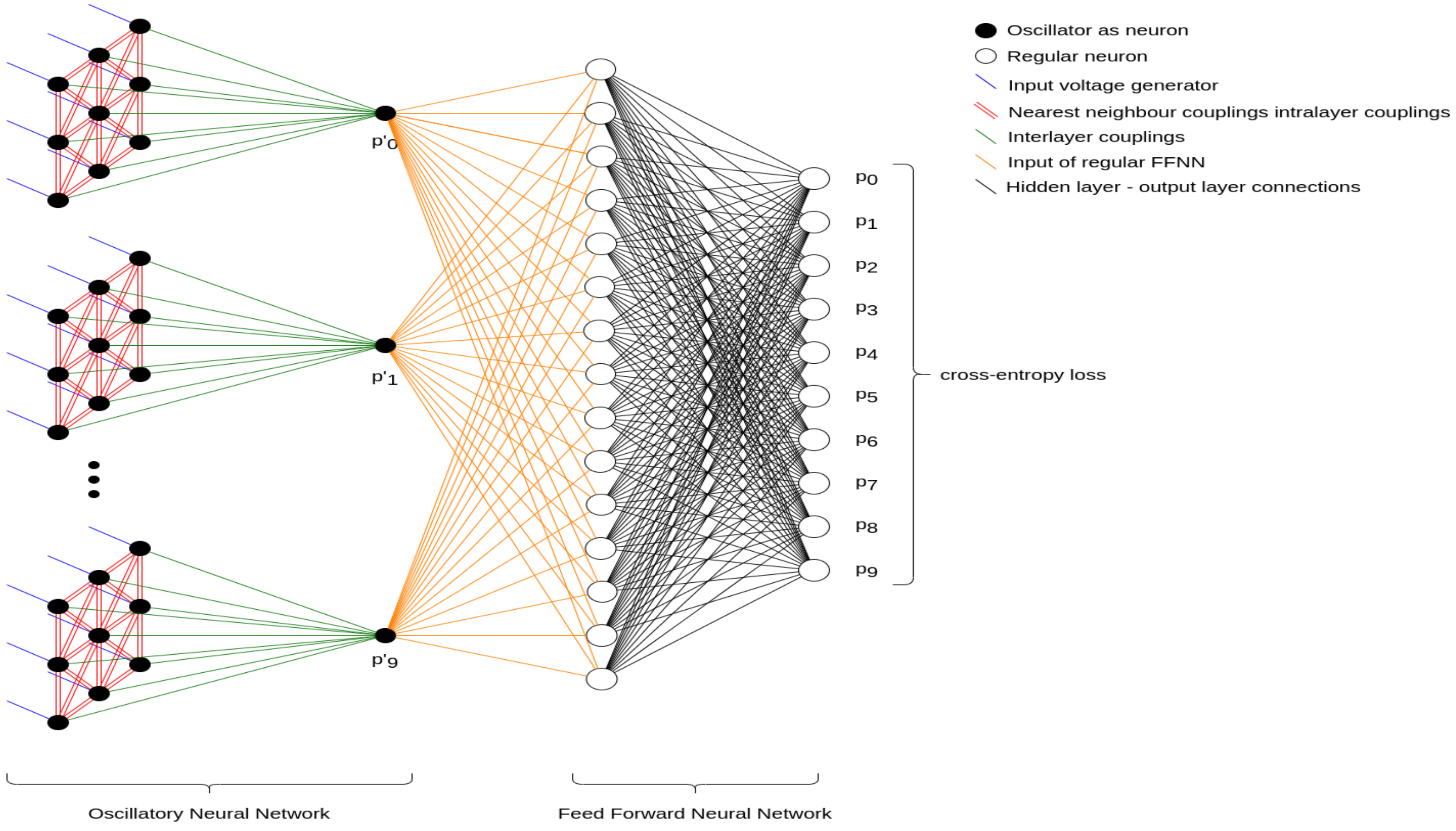
Recognizing ten digits is a much harder problem

- For ten classes: ten networks – each network recognizes one digit and they are trained separately
- Winner-takes-all algorithm for selecting the best
- 10*(4x3 + 52*5 + 140*8 + 196) = 15,880 internal parameters, still small network compared to a software Neural Net
- Accuracy is around 65-70% - much better than random guessing (10%), but far from good
- Ideally the networks should be trained together, not separately – but this is too computationally intensive



Adding a small ‘standard’ second layer achieves much-improved performance

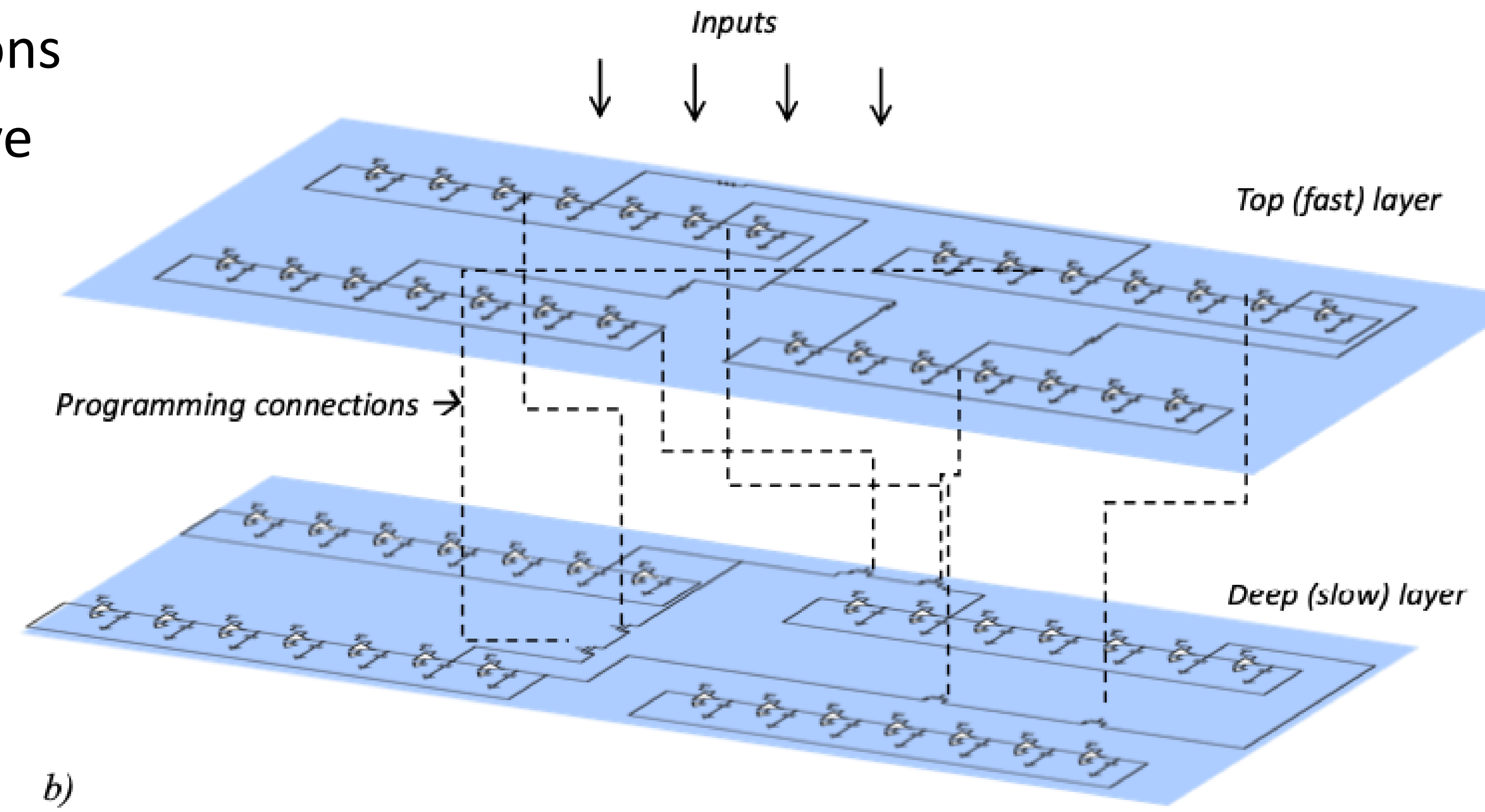
- A simple output layer with only 325 parameters achieves very good performance
- Accuracy **is 96 percent**, comparable to much larger (several hundred thousand parameter) neural nets
- Output layer is easy to train, so it can do the sophisticated function of deciding from the preprocessing layer space



Ring-Oscillator Networks are powerful analog processors – but they are hard to train. Combining them with easier-to-train standard neural nets is a promising approach.

Summary – a vision for a fully analog neural processor

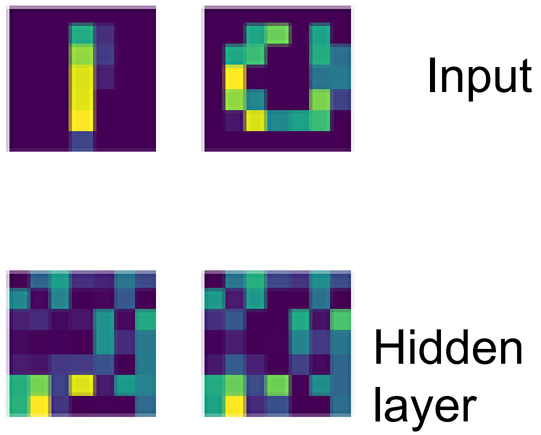
- Machine learning learning allows for realizing non-trivial neural operations
- Functions beyond simple associative memory can be learned
- Oscillator networks work best as preprocessors: input layers are much easier to train than deeper layers
- Multi-layer networks offer very good performance with very few internal parameters



Machine learning can be used for designing non-conventional hardware
Oscillator networks are one of the promising hardware candidates

Classification task

Example hidden layers for the different classes



- Binary classification using two layers
- $l(p) = -(1-y)\log(1-p) - ylog(p)$
  - p: “probability” of the input belonging to class 1 $\longrightarrow p \in [0,1]$
  - y: actual class of input $\longrightarrow y \in \{0,1\}$
- Training properties
  - 500-500 training samples, 50-50 test samples
  - 2 epochs, 10-element batches
  - 8400-8500s total simulation time
  - Integration time: ≈25s; learning step: ≈12-13s
  - Parameter count: 490 vs 2499
- 98-99% accuracy
- No obvious structure in hidden layer

Backup slide

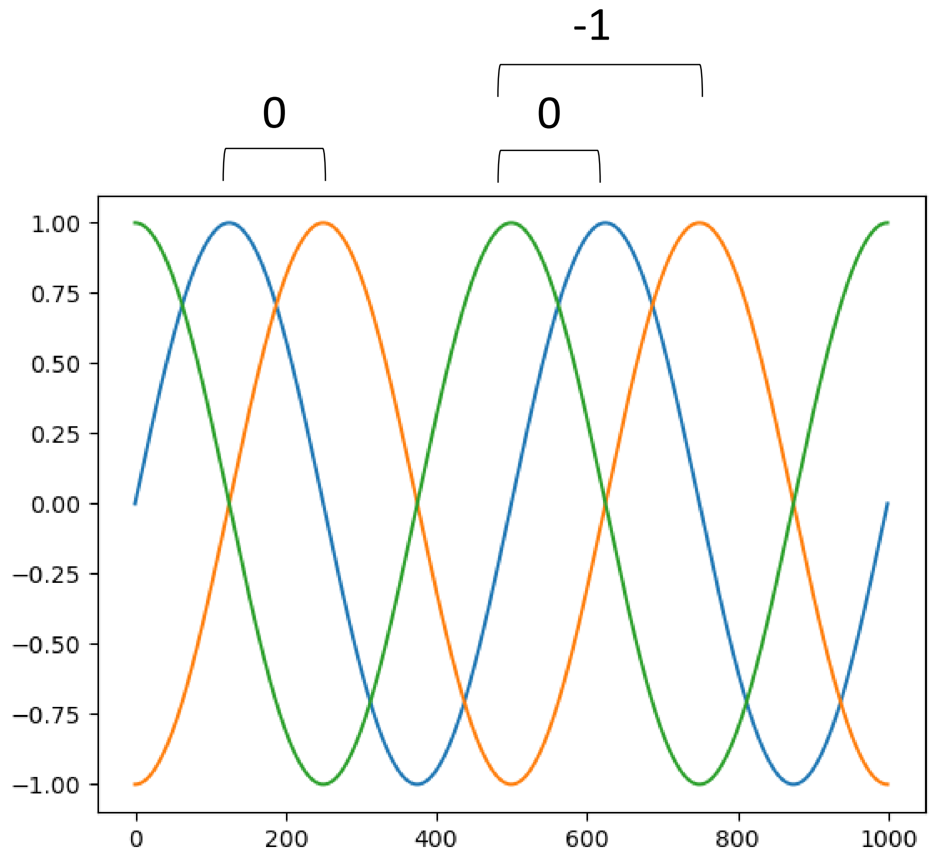
Cross-entropy loss: $e^{x_{n,y_n}} \frac{e^{x_{n,y_n}}}{\sum_{c=1}^{C} e^{x_{n,c}}}$

- $x_{n,m}$: relative phase of output oscillator corresponding to the class of m
- y_n: class of the n-th input
- C: number of classes for the classification

Reading the output (“phase difference”)

- Phase-like calculation: correlation
  - Dot product of a window of the output
  - Normalised by window length and std deviation
  - Symmetric operation
- Two oscillators with real phase difference from each other produces the same “phase-like” difference from designated oscillator if that has real phase difference from the two oscillators

Reference signal
Signal with diff.
Signal with diff


-1
0
0
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
200
400
600
800
1000