Sparse Coding with Memristor Networks

Samina Abdullah, Muyang Liu, Jinyi Gao

Overview

1. Memristors, Crossbar Arrays, and Network Operations
2. Sparse Coding
3. Lateral Inhibition
4. Locally Competitive Algorithm
5. Sparse Coding of Simple Inputs
6. Sparse Coding of Natural Images

Memristors - “Memory Resistors”

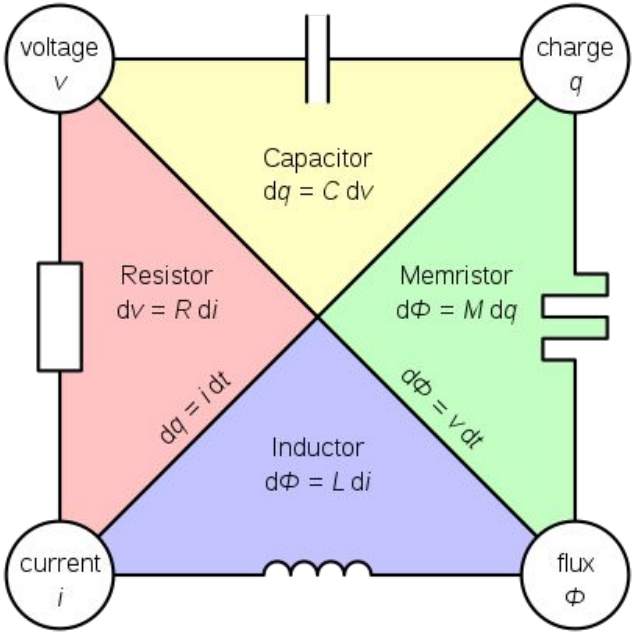
- Two-terminal device whose resistance value depends on internal state and history of external stimulation
- Internal state determined by the internal ion configuration which modulates local resistivity and device resistance

Memristors - “Memory Resistors”

- Require less energy to operate and are faster than present solid-state storage technologies
- Can store at least twice as much data in the same area
- “promising building block for next-generation non-volatile memory, artificial neural networks, and bio-inspired computing systems”
    - Pi et.al., *Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension*, Nature

Memristors - “Memory Resistors”

- Named “fourth fundamental nonlinear circuit element” by Leon Chua
- Nonlinear device that relates electric charge and magnetic flux
- Dynamic relationship between current and voltage including a memory of past voltages or currents

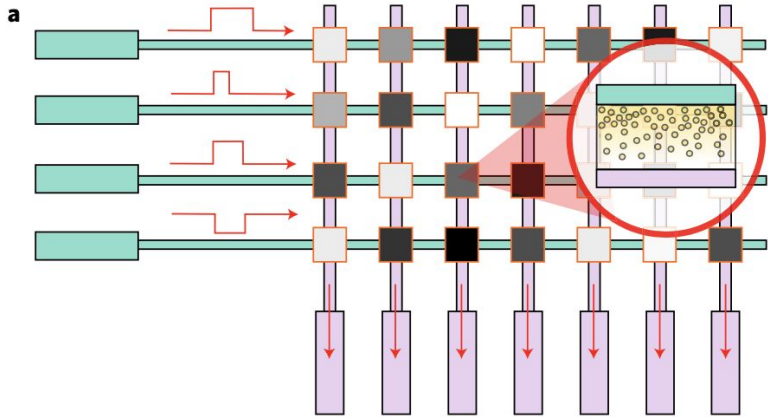


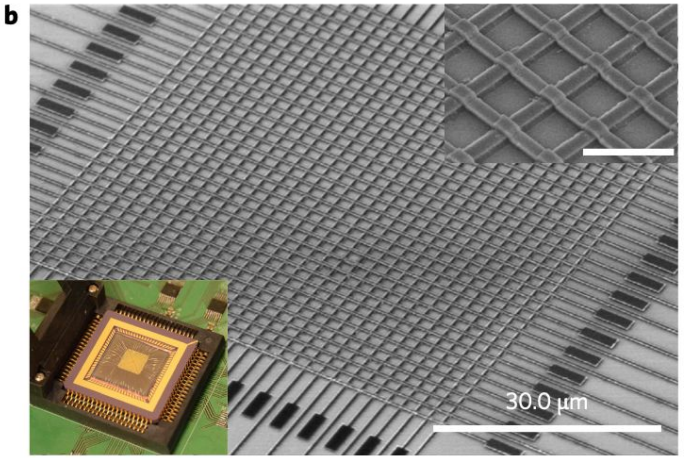
Memristors - Previous Work

- Extensively studied for both digital and analog circuit applications
- Arrays used as artificial neural networks to perform pattern classification tasks
- Proposed and analyzed for sparse coding and dictionary learning
- Desired density and connectivity for hardware implementation of neuromorphic computing systems

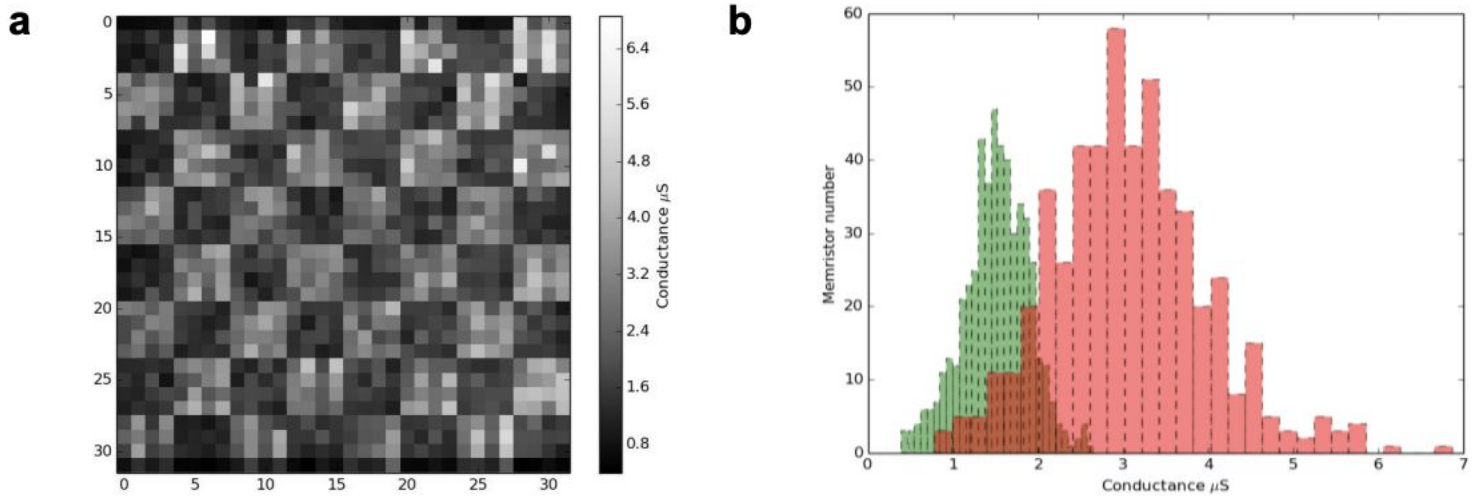
Memristor Crossbar Array and System Set-up

- 32 × 32 crossbar array of WO_X (tungsten oxide) - based analog memristors
- Variable conductance states
- Forward voltage (applied to rows), backward voltage (applied to columns)

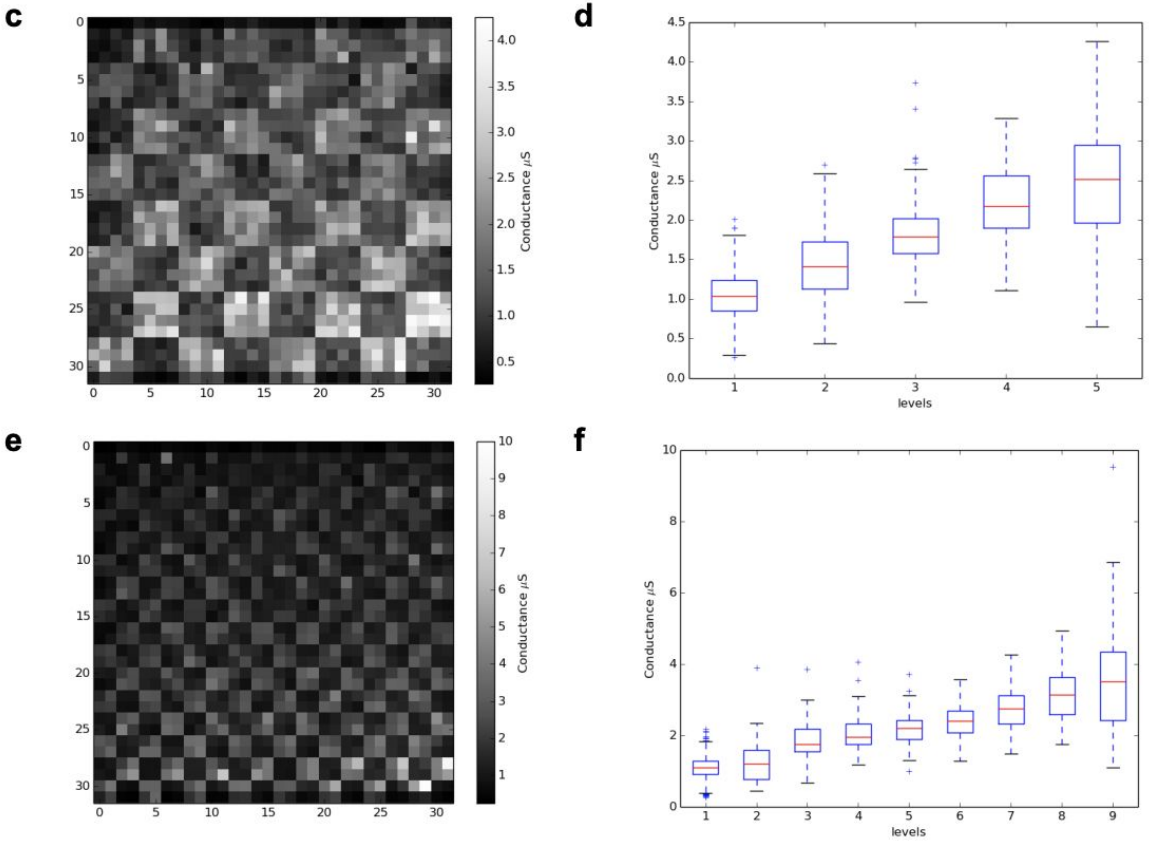




Memristor Array Conductance States



Memristor Array Conductance States



Memristor Network Operation

- Pattern matching and neuron inhibition operations to obtain a sparse, optimal representation of the input
- Weight updates through variable programming pulse widths
- Reconstructed image obtained from output neuron and features stored in crossbar array
- Verified operation with grey-scale images

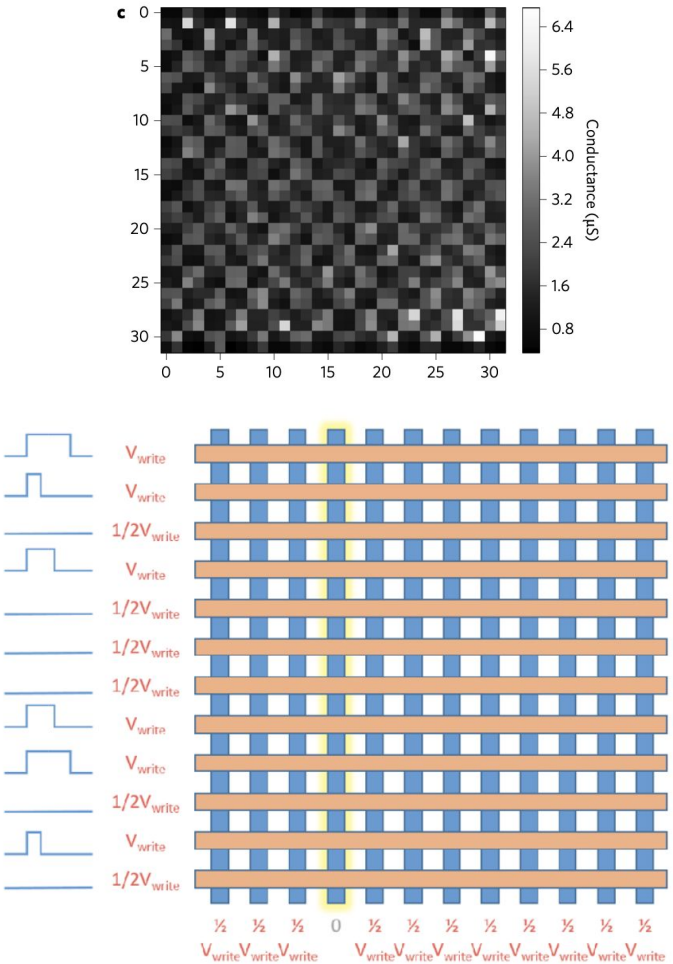


Figure S2 | Memristor array programming scheme.

Custom-built board → simultaneous random access

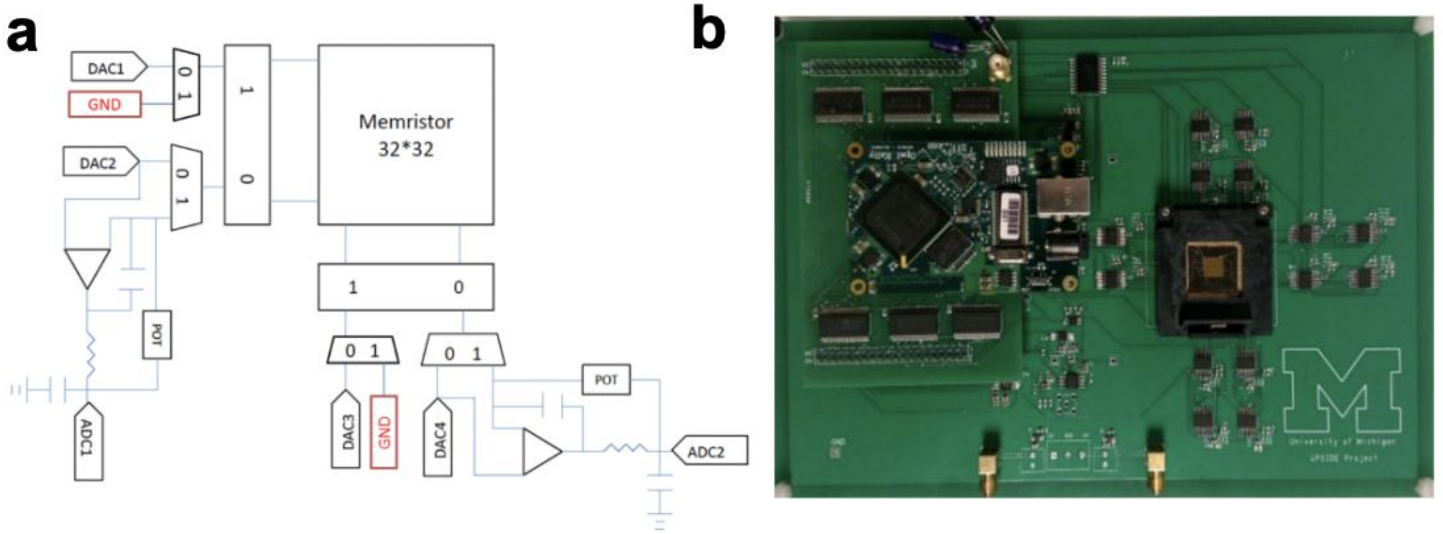


Figure S1 | a Block diagram of the test board. **b** Optical micrograph of the test board with an integrated memristor chip.

Sparse Coding

The objective of sparse coding can be summarized mathematically as minimizing an energy function, defined as:

$$\min_a(|x - Da^T|_2 + \lambda|a|_0)$$

where

- $|\cdot|_2$ and $|\cdot|_0$ are the L^2 - and the L^0 -norm, respectively
- x is the original input signal
- Da^T is the sparse representation of the input signal
- a is an n-element row vector representing the neuron activity coefficients
- λ is the threshold parameter

Sparse Coding

reconstruction error

$$\min_a \left(|x - Da^T|_2 + \lambda |a|_0 \right)$$

sparsity

Input signals → (Sparse coding) → Linear combination of a set of features, while **minimizing the number of features used.**

Locally Competitive Algorithm (LCA)

The neuron dynamics during LCA analysis can be summarized by:

$$\frac{du}{dt} = \frac{1}{\tau}\left(-u + x^T D - a(D^T D - I_n)\right) \qquad \text{(a)}$$

$$a_i = \begin{cases} u_i & \text{if } u_i > \lambda \\ 0 & \text{otherwise} \end{cases} \qquad \text{(b)}$$

where

- u_i is the membrane potential of neuron *i*
- τ is a time constant
- I_n is the n×n identity matrix
- λ is the threshold parameter

Threshold Parameter and Network Performance

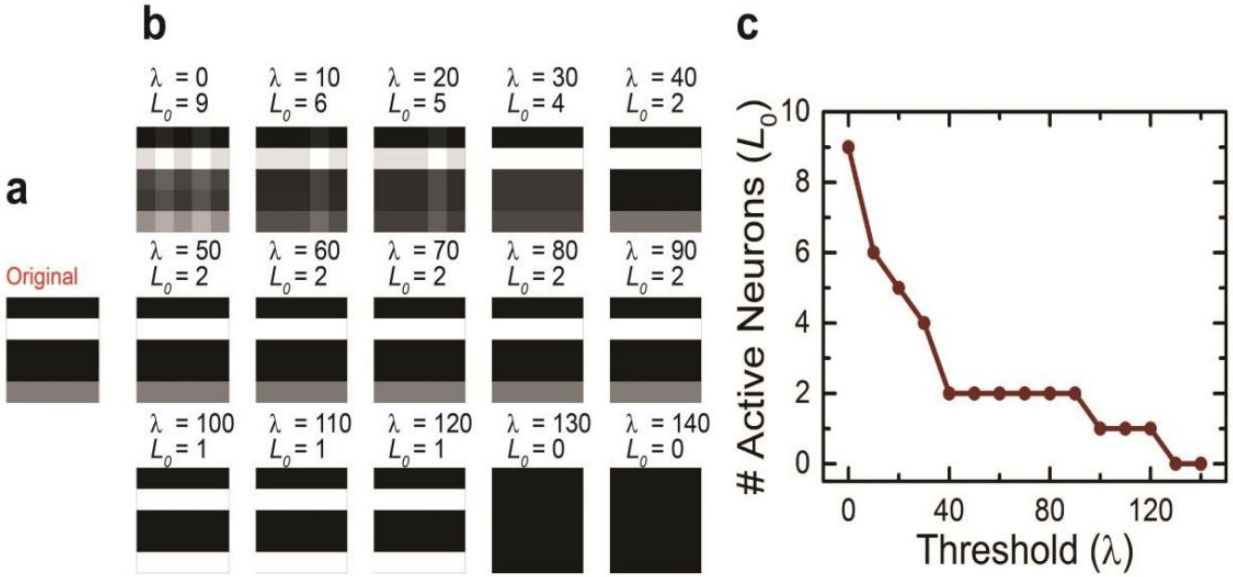


Figure S11 | Impact of the threshold parameter λ network performance. **a** Original image to be sparsely encoded. **b** Image reconstructions for $\lambda \in [0,140]$. The number of active neurons (L_0) is shown above each reconstruction. **c** L_0 as a function of λ. A higher threshold in general leads to more sparse representation.

Locally Competitive Algorithm (LCA)

Considering that implementing the inhibition effect D^TD can be very computationally intensive, the equation (a) can be rewritten as:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \frac{1}{\tau}\left(-u + (x - \hat{x})^T D + a \right) \qquad \text{(c)}$$

where $\hat{x} = Da^T$ is the reconstructed signal

Thus, matrix-matrix multiplication is reduced to two vector-matrix multiplication operations:

- $\hat{x} = Da^T$
- $(x - \hat{x})^T D$

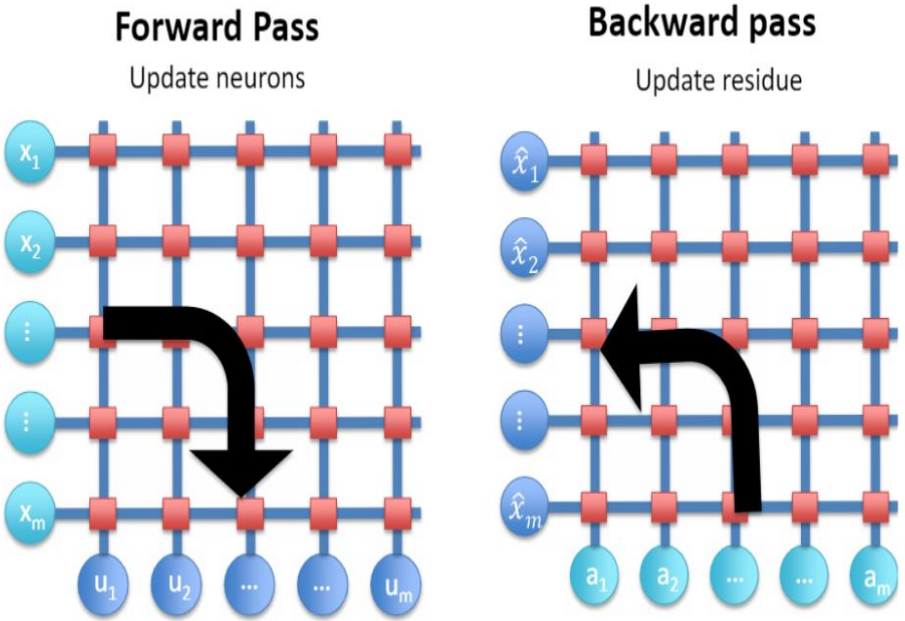
Lateral Inhibition

- First studied by Hartline in limulus eye.
- Lateral inhibition of visual cells enhances edge perception and increases contrast of brightness in visual images.
- In neurobiology, lateral inhibition is the capacity of an excited neuron to reduce the activity of its neighbors.

Mach bands
(seeing borders more sharply)

Lateral Inhibition

- Lateral inhibition is achieved using iterations of forward and backward passes in the same network in discrete time domain.
- The strength of the inhibition is proportional to the similarity of the neurons' receptive fields.
- Ensuring sparsity by preventing duplicate neurons from firing with the same/similar receptive fields.

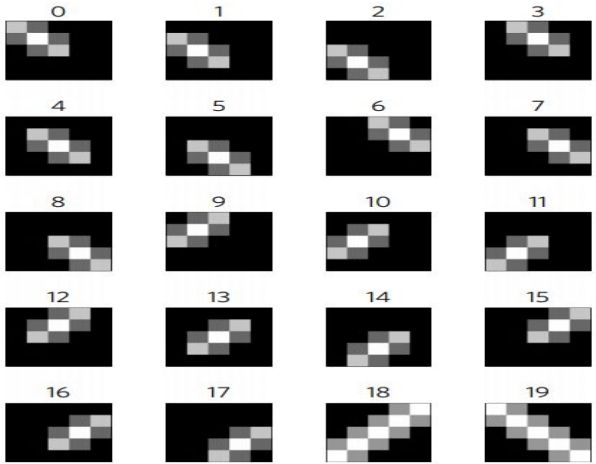


Sparse Coding of Simple Inputs

First Example: an image composed of diagonally oriented stripe features

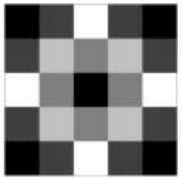
Dictionary: contains 20 stripe features

25 (inputs) × 20 (dictionary) sub-array from the 32 × 32 memristor array



Input image:

with four features

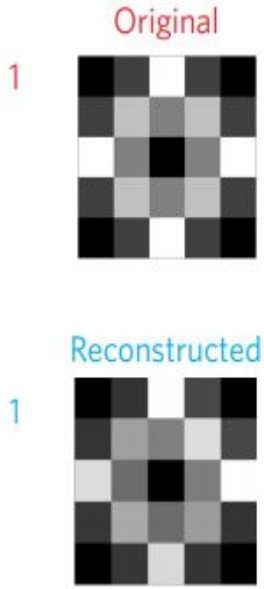


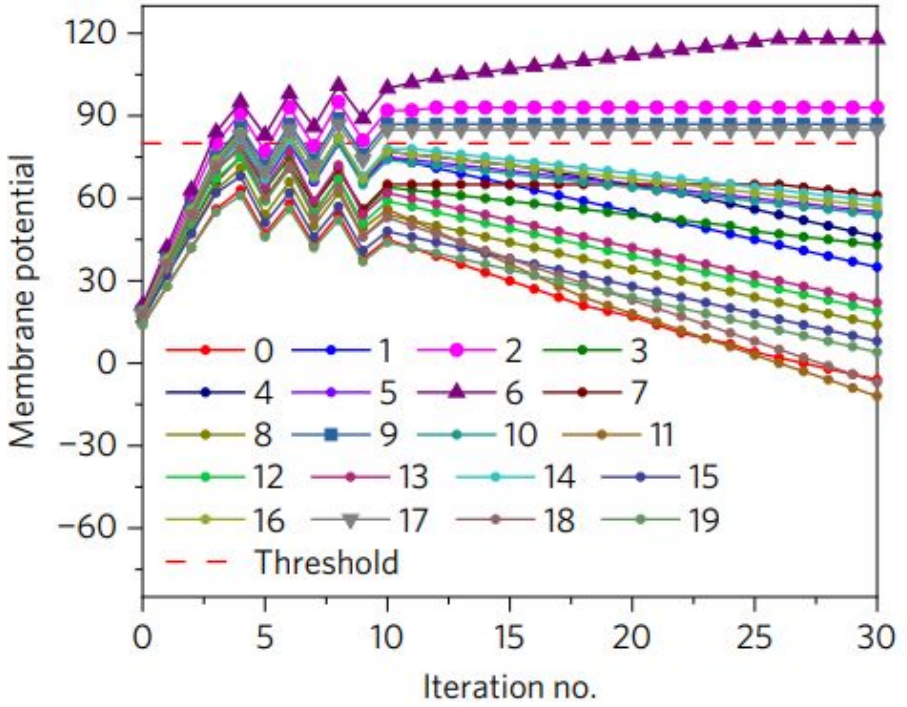
30 forward–backward iterations to make the network stable

Results

Input image was correctly reconstructed using neurons 2, 6, 9 and 17

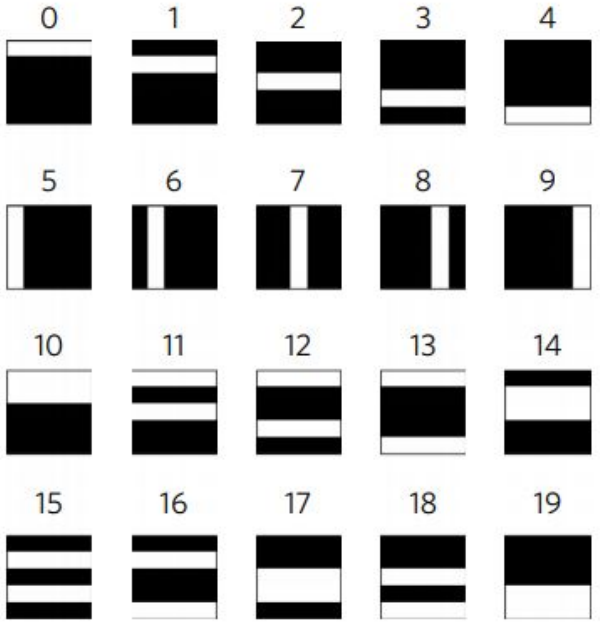
Algorithm drives the system to identify the native features of the input





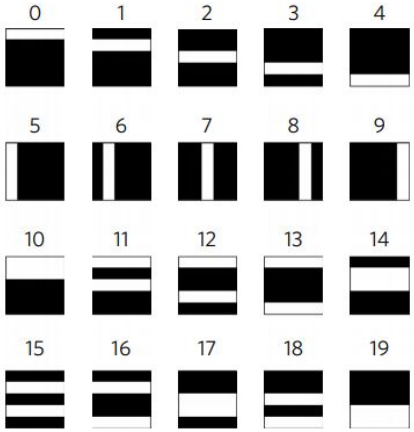
Sparse Coding of Simple Inputs (cont.)

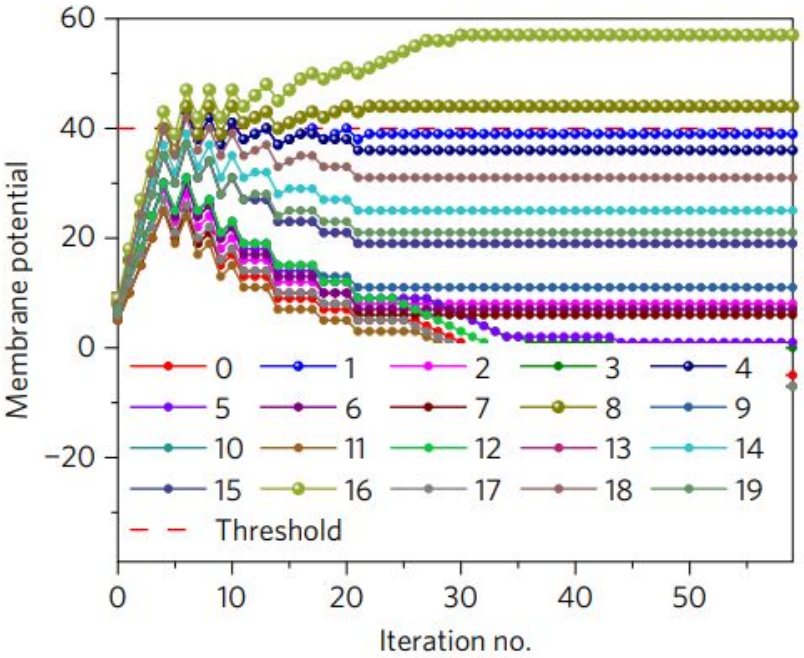
- Second example to demonstrate that the same memristor hardware system can process different types of inputs using a single general approach
- The paper reprogrammed a new dictionary of 20 elements composed of horizontally and vertically oriented bars
- Input dimensionality reduced, achieving greater than 2x over-completeness, highlighting selection of optimal solution



Results

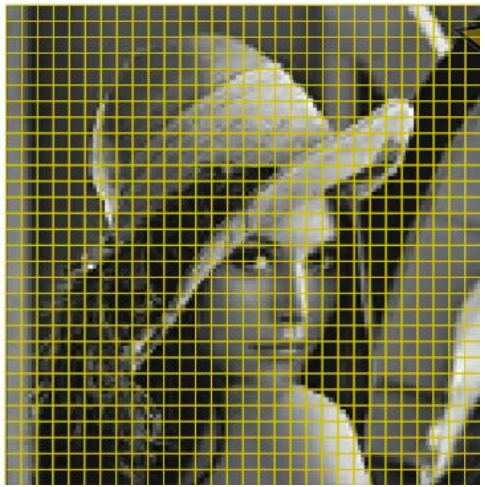
Correctly reconstructed the input image and picked the more efficient solution—a solution based on neurons 8 and 16





Sparse Coding of Natural Images

- Uses a 16 × 32 sub-array from the 32 × 32 memristor array
- Dictionary elements were learned offline using 4 × 4 patches randomly sampled from a training set consisting of nine natural images
- Input image was divided into 4 × 4 patches



Results

- Input image divided into 4x4 sections making each patch similar to the simple input

- Algorithm did not work as well as with simple inputs and produced unclear reconstructed image

More detailed simulation

- Dictionary trained via sparse coding instead of WTA and larger network receptive fields were used (8 × 8, corresponding to a 64 × 128 memristor array with 2× over-completeness)

- Online training is more appropriate as it can handle realistic device variations, since algorithm is self-adaptive during training, so it can still produce high-quality reconstruction

- Additional consideration: Experiments regarding the changes from algorithms and network should be set more detailedly

Conclusions

- Successfully demonstrated a sparse-coding hardware system in a memristor crossbar architecture
- Successful image reconstruction of simple inputs
- Improved image reconstruction of natural images using sparse-coding trained dictionary
- Online learning tolerates device variation
- Future studies integrating CMOS circuitry have significant predicted speed improvements enabling online learning implementation
- Help eliminate ‘von Neumann bottleneck’

References

1. Sheridan, P., Cai, F., Du, C. et al. Sparse coding with memristor networks. Nature Nanotech 12, 784–789 (2017). https://doi.org/10.1038/nnano.2017.83
2. Slide 3,4 image, Slide 5 info: nanowerk.com/memristor.php
3. Slide 5 image, info: https://en.wikipedia.org/wiki/Memristor
4. Slide 26: https://www.techopedia.com/definition/14630/von-neumann-bottleneck