NEUROMORPHIC COMPUTING

Unlocking Glass-like Dynamics in Vanadium Dioxide

THE COMPUTING BOTTLENECK

POST-VON NEUMANN ERA

Traditional computing architectures separate processing and memory, leading to the "Von Neumann bottleneck." As data volume explodes, the energy cost of shuttling data back and forth becomes unsustainable.

THE STRUCTURAL SOLUTION

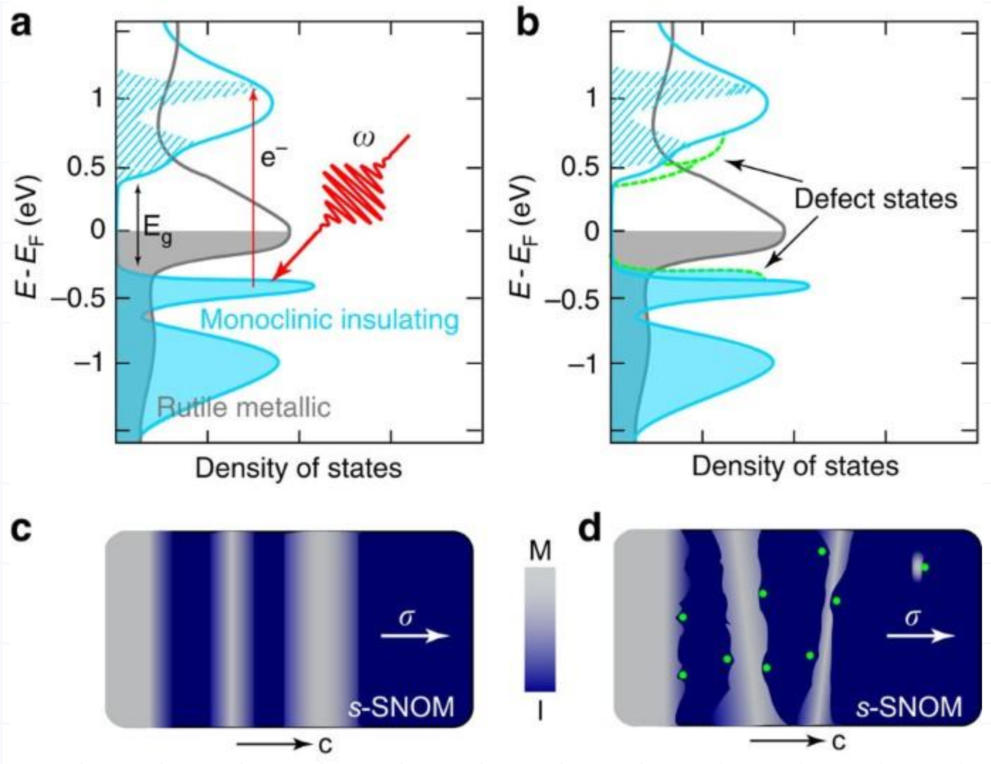
We need devices that can store and process information simultaneously, mimicking the biological brain. Manipulation of **structural states**, rather than just electronic states, offers a path to ultra-scaled, low-power functional devices.

VANADIUM DIOXIDE (VO2)

THE INSULATOR-METAL TRANSITION

VO2 is a strongly correlated material that undergoes a first-order insulator-metal transition (IMT) near room temperature.

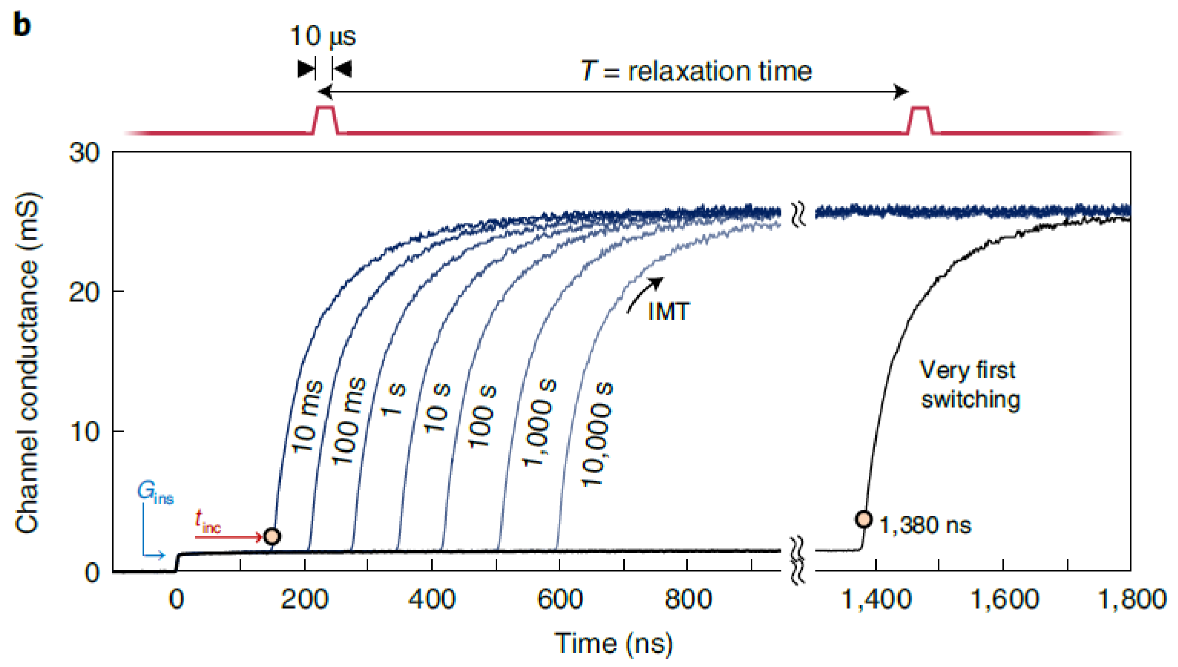
- **Drastic Change:** Conductivity changes by orders of magnitude.
- **Trigger Mechanisms:** Can be triggered by temperature, electric field, or doping.
- **The Challenge:** Controlling this transition precisely for computation has historically been difficult due to its abrupt nature.



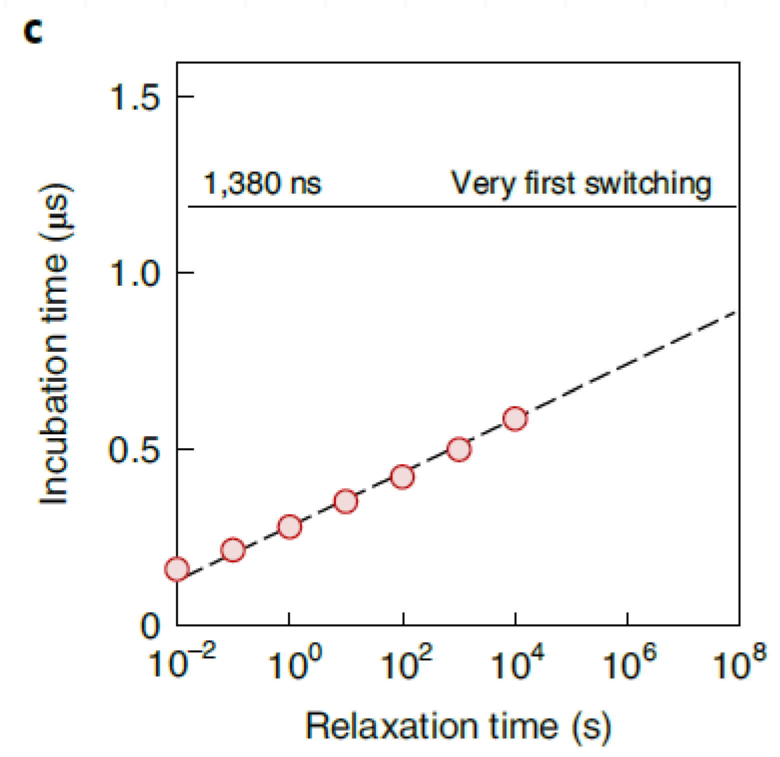
GLASS-LIKE DYNAMICS

Recent research reveals electronically accessible "glass-like" structural states in VO2. Unlike simple binary switching, these states can be arbitrarily manipulated and tracked for hours.

This allows the material to "remember" its history of excitation, similar to how glass retains a memory of its thermal history. This property is key to neuromorphic applications.



MEMORY MECHANISM: INCUBATION TIME

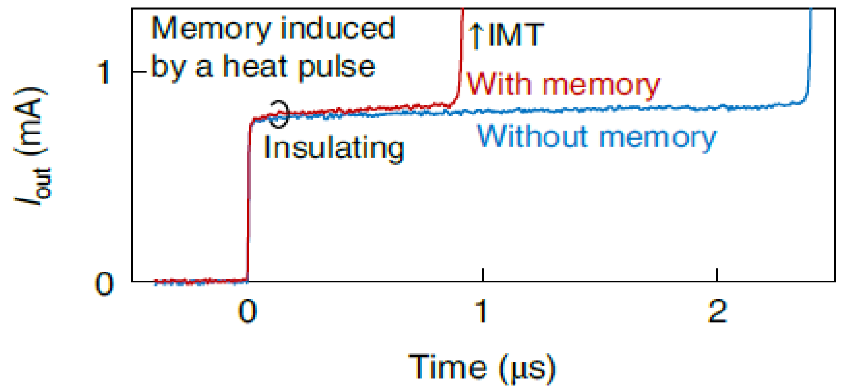


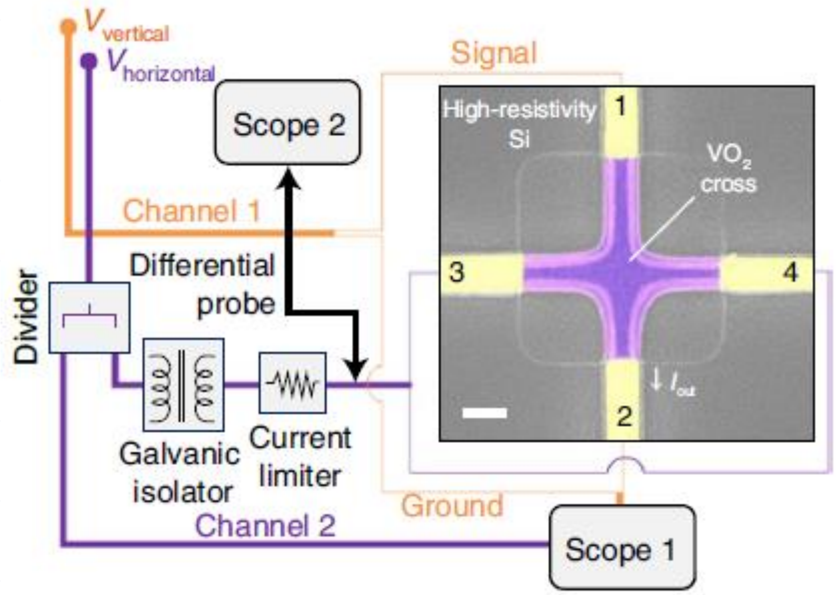
$$t_{\mathrm{inc}} = (78\,\mathrm{ns})\log[T/(160\,\mu\mathrm{s})]$$

The **incubation time** t_{inc} depends logarithmically on the relaxation time (T). This "memory" persists for over 10,000 seconds.

Possible mechanisms

- **Thermal Effects:** Thermal relaxation time is fast, hence heat accumulation plays no role, independent of excitation amplitude
- **Electric Current/Ion movement:** Purely thermally-driven IMT memory, no current in thermally-driven IMT.
- **Metal-VO2 junction:** Independent of the metal type, reproduced in cross structure.
- **Long-lived metallic domains:** No dependence to gap length, below 50nm

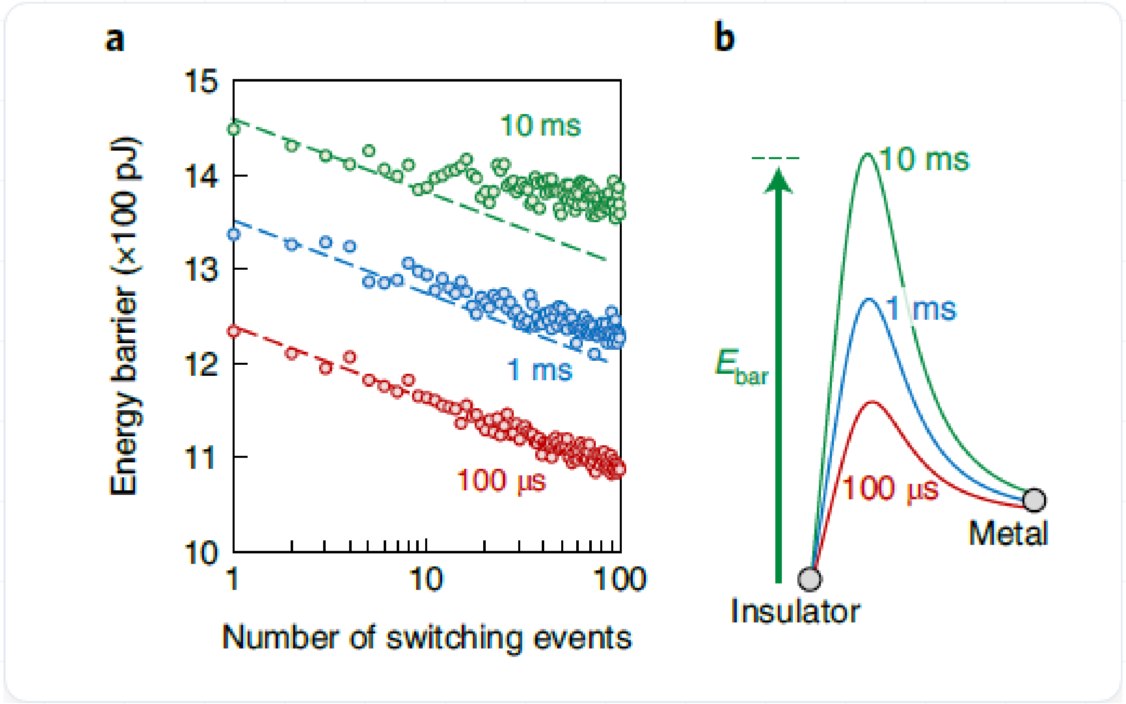




GLASSY NEURAL NETWORKS

Leveraging structural memory for hardware intelligence

HARDWARE-BASED WEIGHTS



ENERGY BARRIER MODULATION

The history of the device determines the switching energy barrier (E_{bar}) for future events. This acts as a physical "weight" in a neural network.

$$E_{\text{bar}} = E_0 - E_1 \ln\left[\sum_{k=1}^{n} \frac{T_0}{t_0 - t_k}\right],$$

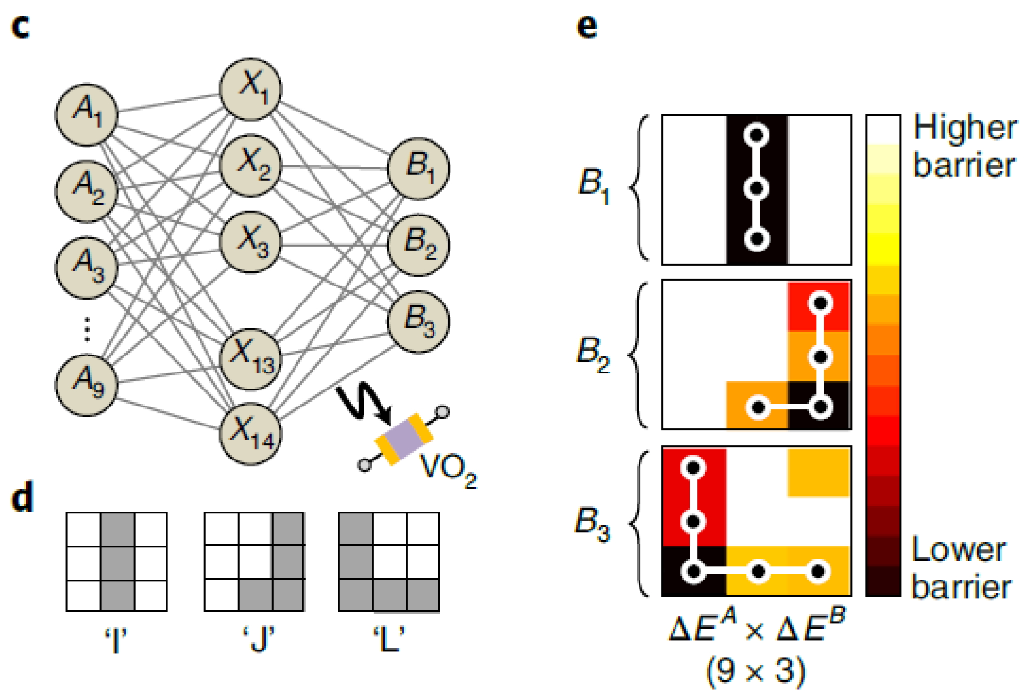
Frequent switching lowers the barrier, effectively "strengthening" the connection between nodes, just like biological synapses.

COMPUTATION-FREE TRAINING

NO WEIGHT CALCULATION NEEDED

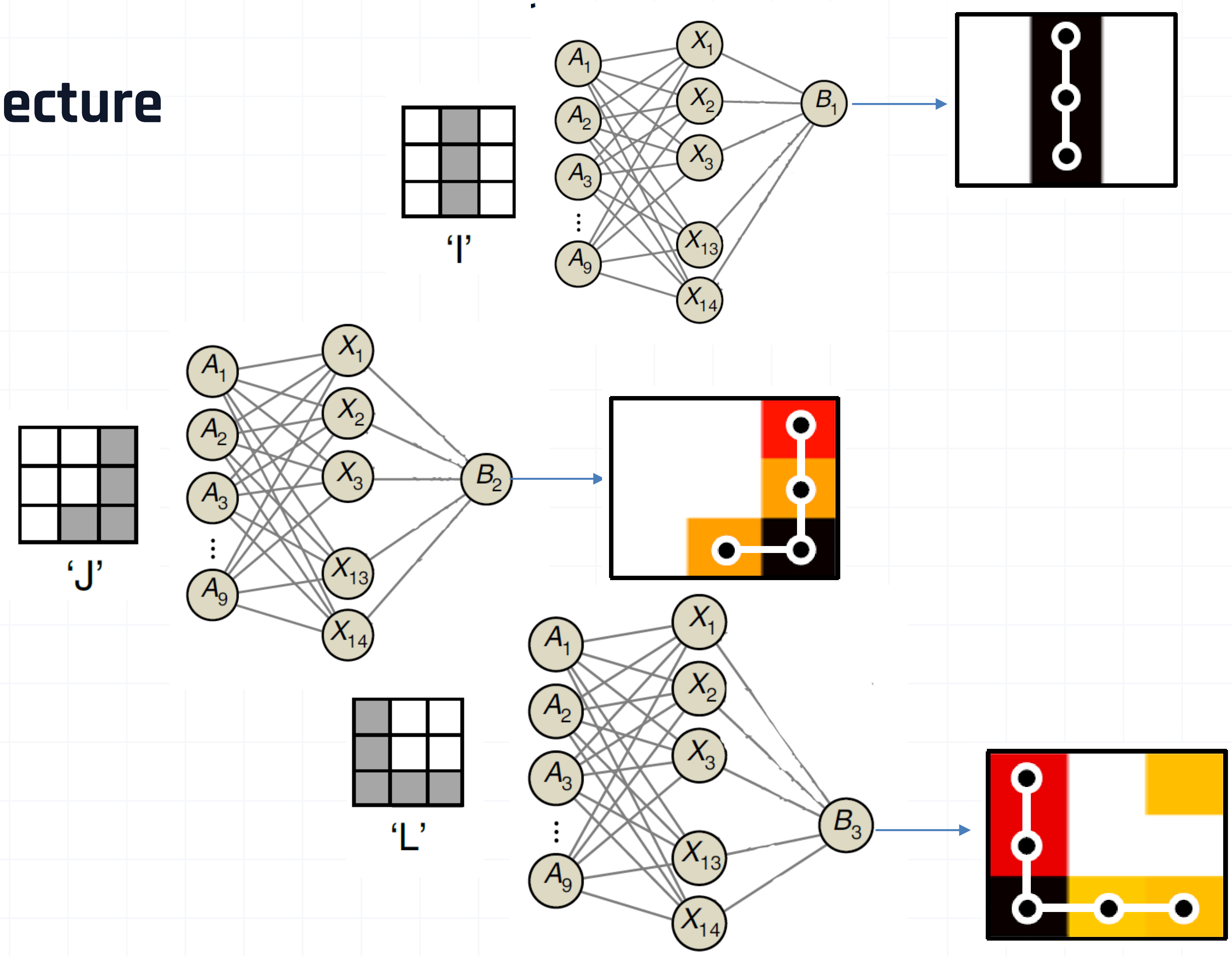
Traditional networks require complex algorithms (backpropagation) to calculate weights. With VO2 Glassy Neural Networks:

- Training is done purely by **hardware physics**.
- Applying currents to input nodes corresponding to a label (e.g., image 'J') automatically reduces the energy barrier for the correct pathway.
- The network self-organizes its conductive paths based on the input data.



Regenerated results

Proposed Architecture



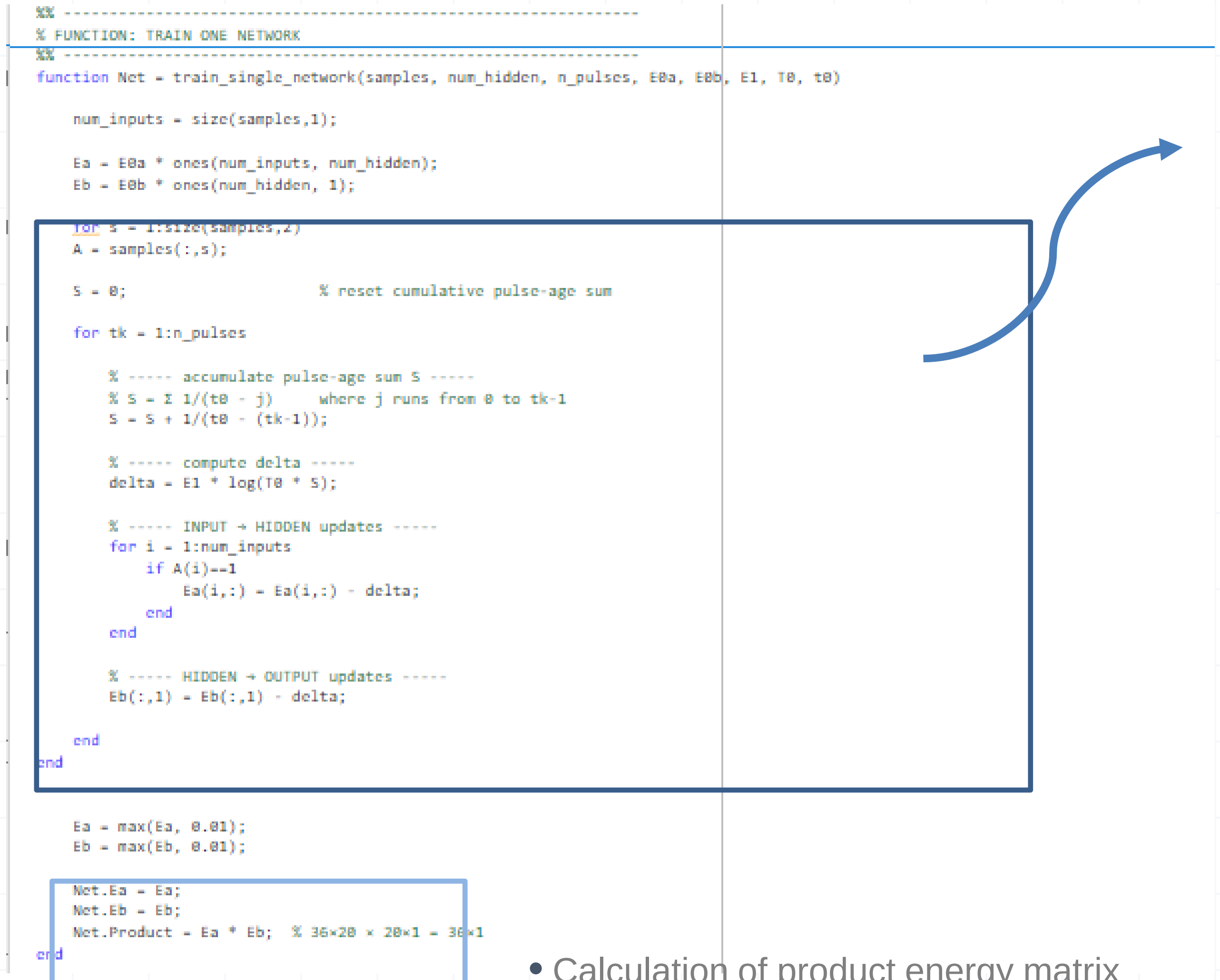
Coding parameters

```
num_inputs  = 9;
num_hidden  = 14;
num_outputs = 1;
n_pulses    = 5;

E0a = 2;
E0b = 2;
E1  = 0.02;

T0  = 1000;
t0  = 200;
```

# pulses per sample

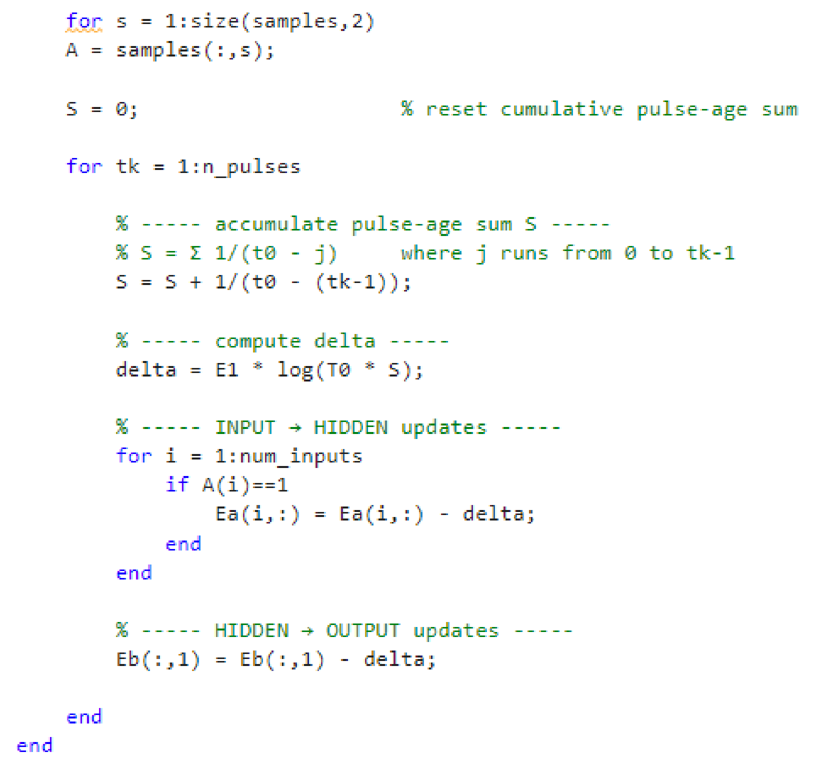
```
I_base = [0 1 0; 0 1 0; 0 1 0];
J_base = [0 0 1; 0 0 1; 0 1 1];
L_base = [1 0 0; 1 0 0; 1 1 1];

Bases = {I_base(:), J_base(:), L_base(:)};
Names = {'I','J','L'};
```

Training function



```
%% ---------------------------------------------------------------
% FUNCTION: TRAIN ONE NETWORK
%% ---------------------------------------------------------------
function Net = train_single_network(samples, num_hidden, n_pulses, E0a, E0b, E1, T0, t0)

    num_inputs = size(samples,1);

    Ea = E0a * ones(num_inputs, num_hidden);
    Eb = E0b * ones(num_hidden, 1);

    for s = 1:size(samples,2)
    A = samples(:,s);

    S = 0;                          % reset cumulative pulse-age sum

    for tk = 1:n_pulses

        % ----- accumulate pulse-age sum S -----
        % S = Σ 1/(t0 - j)      where j runs from 0 to tk-1
        S = S + 1/(t0 - (tk-1));

        % ----- compute delta -----
        delta = E1 * log(T0 * S);

        % ----- INPUT → HIDDEN updates -----
        for i = 1:num_inputs
            if A(i)==1
                Ea(i,:) = Ea(i,:) - delta;
            end
        end

        % ----- HIDDEN → OUTPUT updates -----
        Eb(:,1) = Eb(:,1) - delta;

    end
end

    Ea = max(Ea, 0.01);
    Eb = max(Eb, 0.01);

    Net.Ea = Ea;
    Net.Eb = Eb;
    Net.Product = Ea * Eb;  % 36×20 × 20×1 = 36×1
end
```

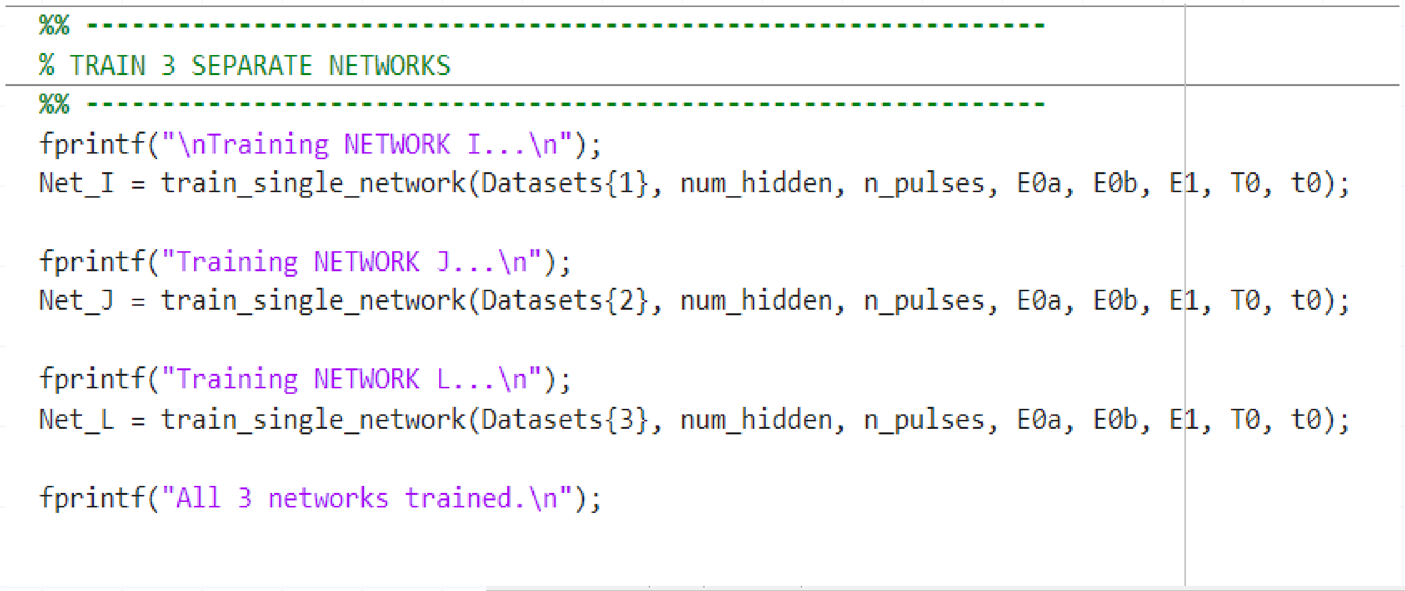
- Calculation of product energy matrix



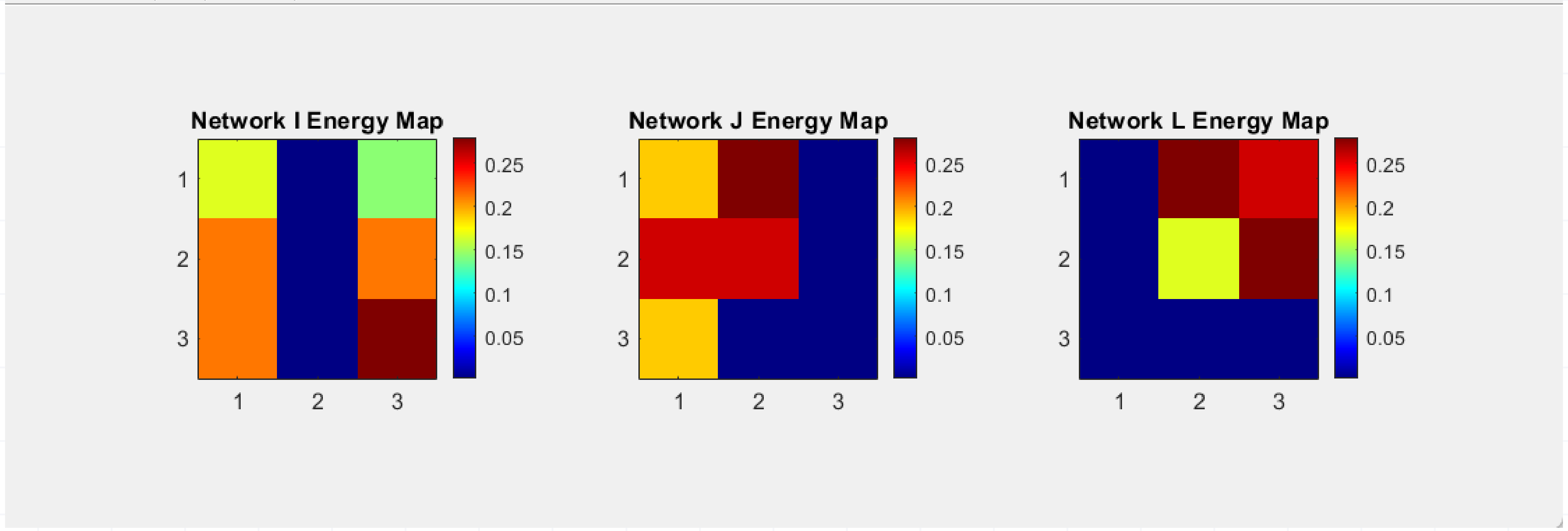
```
    for s = 1:size(samples,2)
    A = samples(:,s);

    S = 0;                          % reset cumulative pulse-age sum

    for tk = 1:n_pulses

        % ----- accumulate pulse-age sum S -----
        % S = Σ 1/(t0 - j)      where j runs from 0 to tk-1
        S = S + 1/(t0 - (tk-1));

        % ----- compute delta -----
        delta = E1 * log(T0 * S);

        % ----- INPUT → HIDDEN updates -----
        for i = 1:num_inputs
            if A(i)==1
                Ea(i,:) = Ea(i,:) - delta;
            end
        end

        % ----- HIDDEN → OUTPUT updates -----
        Eb(:,1) = Eb(:,1) - delta;

    end
end
```

- Lowering the energy of the corresponding path from Ea matrix and Eb matrix

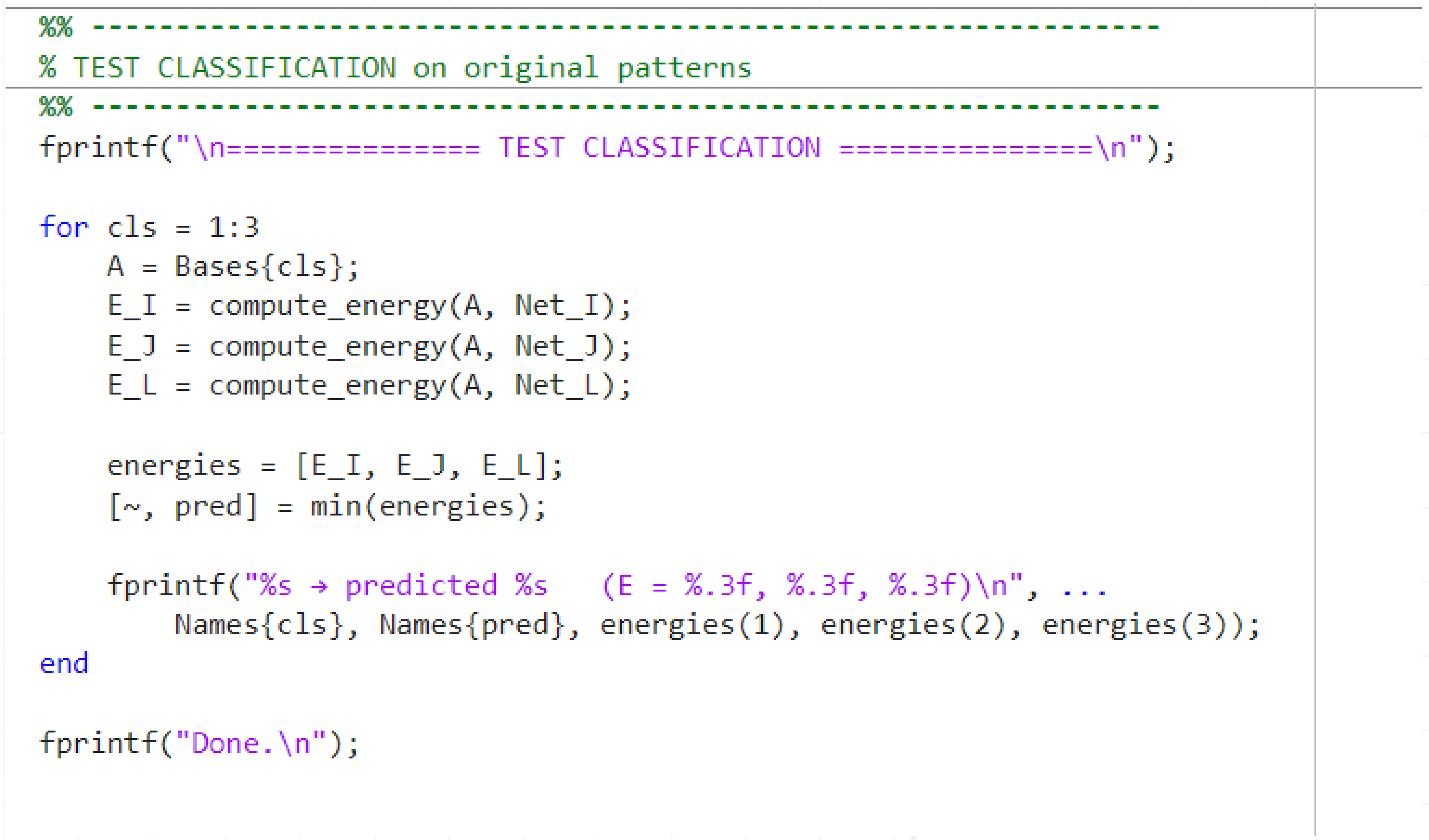
Training



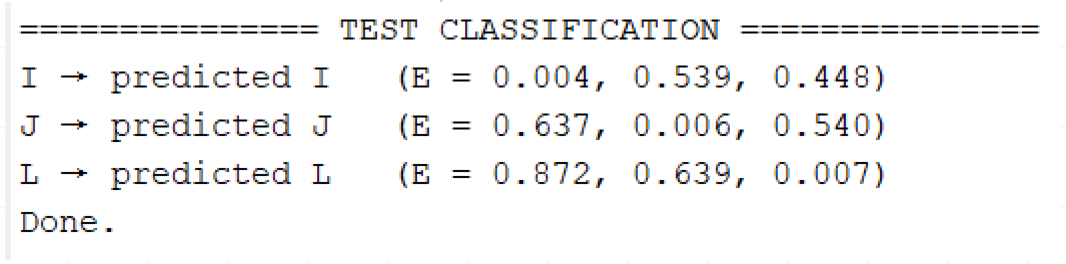
```
%% --------------------------------------------------------------
% TRAIN 3 SEPARATE NETWORKS
%% --------------------------------------------------------------
fprintf("\nTraining NETWORK I...\n");
Net_I = train_single_network(Datasets{1}, num_hidden, n_pulses, E0a, E0b, E1, T0, t0);

fprintf("Training NETWORK J...\n");
Net_J = train_single_network(Datasets{2}, num_hidden, n_pulses, E0a, E0b, E1, T0, t0);

fprintf("Training NETWORK L...\n");
Net_L = train_single_network(Datasets{3}, num_hidden, n_pulses, E0a, E0b, E1, T0, t0);

fprintf("All 3 networks trained.\n");
```



Test



```matlab
%% ----------------------------------------------------------------
% TEST CLASSIFICATION on original patterns
%% ----------------------------------------------------------------
fprintf("\n=============== TEST CLASSIFICATION ===============\n");

for cls = 1:3
    A = Bases{cls};
    E_I = compute_energy(A, Net_I);
    E_J = compute_energy(A, Net_J);
    E_L = compute_energy(A, Net_L);

    energies = [E_I, E_J, E_L];
    [~, pred] = min(energies);

    fprintf("%s → predicted %s   (E = %.3f, %.3f, %.3f)\n", ...
        Names{cls}, Names{pred}, energies(1), energies(2), energies(3));
end

fprintf("Done.\n");
```



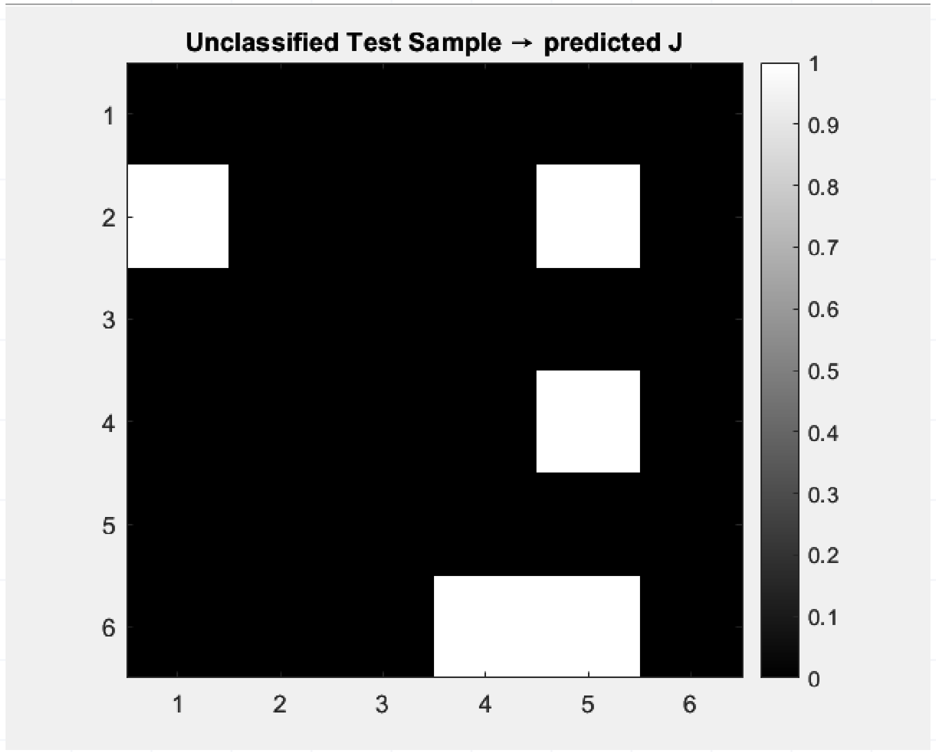
```
=============== TEST CLASSIFICATION ===============
I → predicted I   (E = 0.004, 0.539, 0.448)
J → predicted J   (E = 0.637, 0.006, 0.540)
L → predicted L   (E = 0.872, 0.639, 0.007)
Done.
```

"The system fine-tunes itself as it is exposed to unclassified samples."

— Self-Optimizing Hardware

The network continues to learn and optimize during operation, reducing energy barriers for correct classifications dynamically.

Performance against unclassified sample



```
UNCLASSIFIED SAMPLE → predicted class J
Energy vector = [I: 1.773, J: 0.343, L: 0.774]
```

THE FUTURE OF DATA PROCESSING

These glass-like functional devices could outperform conventional MOS electronics in speed, energy, and size, providing a robust platform for the next generation of neuromorphic computers.

QUESTIONS?

Thank you for your attention.