Fig. 7.9 Original 20 x 20 pixels of black-and-white MNIST handwritten digits: (a) ‘3’ and (b) ‘4’.

A Side Note: What Can One Do with 1 cm³?
Reference case: the human brain
- $P_{m}^{(brain)}$: 20 W
- (20% of the total dissipation, 2% of the weight)
- Power density: 15 mW/cm³
- Nerve cells only 4% of brain volume
- Average neuron density: 70 million/cm³

Project Guideline

Presentation
- Dec. 3 (Th)
- Dec. 5 (Fth)

Final (written) Report due by Dec. 9 (M) 5 pm

Evaluation criteria
- Relevance to Low Power High Speed Design
- Scope
- Achievement
- Quality of Presentation

written Report
- Title
- Brief Summary
- Problem Description
- Description of work done including Achievements
- Summary
- Reference List

Fig. 11.1 In the Von Neumann architecture (a), data (both operations and operands) must move to and from the dedicated central processing unit (CPU) along a bus. In contrast, in a non-Von Neumann architecture (b), distributed computing takes place at the location of the data, reducing the time and energy spent moving data around (Adapted from Barr et al. [1]).

Fig. 1.1 A revolutionary shift of the computing paradigm from the computation-centric (von Neumann architecture) to the data-centric (neuro-inspired architecture)
Fig. 11.2: Neuro-inspired non-Von Neumann computing [7–11], in which neurons activate each other through dense networks of programmable synaptic weights, can be implemented using dense crossbar arrays of nonvolatile memory (NVMe) and selector device pairs (Adapted from Bue et al. [10]).

Table 1.2: Summary of the device characteristics for synaptic devices

<table>
<thead>
<tr>
<th>Performance metrics</th>
<th>Desired criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device integration</td>
<td>High</td>
</tr>
<tr>
<td>Memristor-based device</td>
<td>High</td>
</tr>
<tr>
<td>Energy consumption</td>
<td>Low</td>
</tr>
<tr>
<td>Synaptic weight</td>
<td>High</td>
</tr>
<tr>
<td>Resistance</td>
<td>High</td>
</tr>
<tr>
<td>Endurance</td>
<td>High</td>
</tr>
</tbody>
</table>

Note: These numbers are application-dependent.

Fig. 6.2: AlGd/TiOx resistive switching devices. (a) Typical quasi-static V-I curves of memristor forming, set and reset, with the inset showing the material stack. (b) Analog properties of crossbar-integrated devices: tuning of the resistance measured at a non-disturbing voltage of 0.2 V to various values within the dynamic range. (c) Micrograph of a 12 x 12 crossbar circuit built with DUV lithography.

Fig. 6.7: (a) Graph diagram of the feed-forward network implemented using a memristive crossbar array. (b) Linearly separable training data set used to benchmark the fabricated neural network.

Fig. 14. Examples from different image data sets: MNIST [2618], CIFAR10 [2619], and SVHN [2620], to which neuromorphic systems have been applied for classification purposes.
Fig. 1.4 (a) The weight matrix between neuron layers in the network. (b) The crossbar array consists of perpendicular rows and columns with the resistive synaptic devices switched at each cross-point. The weights in the neural network are mapped to the conductance of the resistive synaptic devices.

\[ V_{out} = R_0 \left( I_1 - I_2 \right) - R_0 \sum_{j=1}^{3} V_j (G_{j}^p - G_{j}^n), \quad \text{for} \quad 1 \leq j \leq 3 \]
In situ training of the hardware starts by calculating the current classification error of the network in order to determine how to update synaptic weights and consequently the neurons. To do so, all 30 patterns were applied to the network, and outputs were measured as shown in Fig. 6.8 and explained in Eq. (6.5). Based on the measured outputs and known targets, the amount of update for each weight at the end of each epoch can be calculated using a regular delta rule [16] as:

$$\Delta w_{ij} = \alpha \sum_{\text{training data}} (y_i - y'_i) \frac{d}{dx} \left( \sum_{j=1}^{N} w_{ij} x_j \right) x_j$$  \hspace{1cm} (6.6)$$

where $\alpha$ is the learning rate, $y_i$ is the desired target output, and $f'(x)$ is the derivative of the utilized activation function. This equation shows that in order to shrink the gap between the actual and target outputs, at the end of each epoch in which direction and for how much each synaptic weight should be fine-adjusted. Although

![Image](https://example.com/image1.png)

**Fig. 6.11** (a) The recursive MLP fabricated on two printed circuit boards, one including two 20 x 20 crossbars with peripheral circulators while the other one implements neurons, the High-level diagram of the implemented MLP. Each set of weights is implemented with one crossbar.

![Image](https://example.com/image2.png)

**Fig. 6.12** (a) Linearly inseparable training data set used to benchmark the fabricated recursive MLP. (b) Classification performance of the network when it is trained using the ex situ training algorithm. As can be seen in this figure, 30 training patterns have been classified correctly. (c) Training the recursive MLP in situ. Due to the existing device-to-device variations, the network couldn’t classify all the patterns (mainly those patterns which have so many pixels in common).

![Image](https://example.com/image3.png)

**Fig. 11.4** In forward evaluation of a multilayer perceptron, each layer's neurons drive the next layer through weights $w_{ij}$ and a nonlinearity $f$. Input neurons are driven by input (for instance, pixels from successive MNIST images cropped to 25x25); the ten output neurons classify which digit was presented (Adapted from Barr et al. [1]).
Fig. 7.7 (a) Input stage for CNN architecture. (b) 2D matrix convolution. (c) Implementation of convolution for multiple feature maps into a cross-point array architecture by reduction of 2D kernel matrix into 1D column vector.

Fig. 7.7 (a) Input stage for CNN architecture. (b) 2D matrix convolution. (c) Implementation of convolution for multiple feature maps into a cross-point array architecture by reduction of 2D kernel matrix into 1D column vector.

2D matrix convolution:

\[ y_{m,n} = \sum_{i} \sum_{j} x_{i,j} f_{m,n} = \sum_{i,j} x_{i,j} f_{m,n,i,j} \]

weights

2D matrix convolution:

\[ y_{m,n} = \sum_{i} \sum_{j} x_{i,j} f_{m,n} = \sum_{i,j} x_{i,j} f_{m,n,i,j} \]

\[ = \sum_{j=0}^{n} (x_{i,j} f_{m,n,j} + x_{i,j} f_{m,n,j+1} + x_{i,j} f_{m,n,j-1} + x_{i,j} f_{m,n,j+2} + \ldots) \]

Fig. 7.8. Prewitt edge detector: (a) Horizontal kernel (ξ), (b) Vertical kernel (η).

Fig. 7.9. Original 20 × 20 pixels of black-and-white MNIST handwritten digits (a) “3” and (b) “4.”

5
Artificial Neural Network (ANN) vs. Spiking Neural Network (SNN)

\[ y_j = \sum \left( w_{ij} x_i \right) \]

Synaptic weights change as a function of relative timing of pre- and post-synaptic spikes.

Fig. 8.1: 2T-1R PCM synaptic cell array with axon drivers and neuron circuits on periphery.

Fig. 14: Schematic of BCM. The output signal is fed through 32 frequency bands which are combined for the BCM. Synapses are based on 80% yield DRAM devices. Output neurons become selective to different input spike shapes (adapted from Wu et al. (153)).

Fig. 5: A breakdown of what makes an ion channel membrane, grouped by overall type and used to reflect the number of distinct membranes.

Ionic Channels in Hodgkin-Huxley Model
A Memristive System (Chua and Kang, Proc. of the IEEE, 1976)

- Parallel-conductance model of the membrane
- Hodgkin-Huxley equations

Fig. 8: An overview of on-chip training algorithms. The size of the box corresponds to the number of papers in that category.
Fig. 2.3  Synaptic plasticity. (a) Relative firing of neuronal spikes from the presynaptic neuron and the postsynaptic neuron determines the weight change in synapse. The synaptic weight change is plotted as a function of relative timing delays, and post-spike. Reproduced with permission from R. and Thanesh. (b) Drawing of an ITIR neuron. (c) ITIR memory array. (d) True crossbar array without selector transistors. (e) Arrangement of peripheral circuits.