11 Dedicated Neuro-Hardware

Recently, many neuro-chips have been designed and built. Although many techniques, such as digital and analogue electronics, optical computers, chemical implementation, and bio-chips, are investigated for implementing neuro-computers, only digital and analogue electronics, and in a lesser degree optical implementations, are at present feasible techniques. We will therefore concentrate on such implementations.

11.1 General issues

11.1.1 Connectivity constraints

Connectivity within a chip

A major problem with neuro-chips always is the connectivity. A single integrated circuit is, in current-day technology, planar with limited possibility for cross-over connections. This poses a problem. Whereas connectivity to nearest neighbour can be implemented without problems, connectivity to the second nearest neighbour results in a cross-over of four which is already problematic. On the other hand, full connectivity between a *set* of input and output units can be easily attained when the input and output neurons are situated near two edges of the chip (see figure 11.1). Note that the number of neurons in the chip grows linearly with the size of the chip, whereas in the earlier layout, the dependence is quadratic.

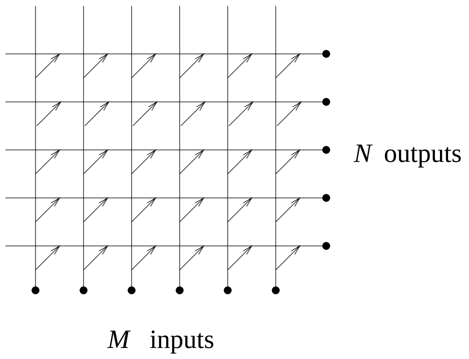

Figure 11.1: Connections between M input and N output neurons.

Connectivity between chips

To build large or layered ANN's, the neuro-chips have to be connected together. When only few neurons have to be connected together, or the chips can be placed in subsequent rows in feed-forward types of networks, this is no problem. But in other cases, when large numbers of neurons in one chip have to be connected to neurons in other chips, there are a number of problems:

- designing chip packages with a very large number of input or output leads;
- fan-out of chips: each chip can ordinarily only send signals two a small number of other chips. Amplifiers are needed, which are costly in power dissipation and chip area;
- wiring.

A possible solution would be using optical interconnections. In this case, an external light source would reflect light on one set of neurons, which would reflect part of this light using deformable mirror spatial light modulator technology on to another set of neurons. Also under development are three-dimensional integrated circuits.

11.1.2 Analogue vs. digital

Due to the similarity between artificial and biological neural networks, analogue hardware seems a good choice for implementing artificial neural networks, resulting in cheaper implementations which operate at higher speed. On the other hand, digital approaches offer far greater flexibility and, not to be neglected, arbitrarily high accuracy. Also, digital chips can be designed without the need of very advanced knowledge of the circuitry using CAD/CAM systems, whereas the design of analogue chips requires good theoretical knowledge of transistor physics as well as experience.

An advantage that analogue implementations have over digital neural networks is that they closely match the physical laws present in neural networks (table 9.1, point 1). First of all, weights in a neural network can be coded by one single analogue element (e.g., a resistor) where several digital elements are needed[1]. Secondly, very simple rules as Kirchoff's laws[2] can be used to carry out the addition of input signals. As another example, Boltzmann machines (section 5.3) can be easily implemented by amplifying the natural noise present in analogue devices.

11.1.3 Optics

As mentioned above, optics could be very well used to interconnect several (layers of) neurons. One can distinguish two approaches. One is to store weights in a planar transmissive or reflective device (e.g., a spatial light modulator) and use lenses and fixed holograms for interconnection. Figure 11.2 shows an implementation of optical matrix multiplication. When N is the linear size of the optical array divided by wavelength of the light used, the array has capacity for N^2 weights, so it can fully connect N neurons with N neurons (Fahrat, Psaltis, Prata, & Paek, 1985).

A second approach uses volume holographic correlators, offering connectivity between two areas of N^2 neurons for a total of N^4 connections[3]. A possible use of such volume holograms in an all-optical network would be to use the system for image completion (Abu-Mostafa & Psaltis, 1987). A number of images could be stored in the hologram. The input pattern is correlated with each of them, resulting in output patterns with a brightness varying with the

[1]On the other hand, the opposite can be found when considering the *size* of the element, especially when high accuracy is needed. However, once artificial neural networks have outgrown rules like back-propagation, high accuracy might not be needed.

[2]The Kirchoff laws state that for two resistors R_1 and R_2 (1) in series, the total resistance can be calculated using $R = R_1 + R_2$, and (2) in parallel, the total resistance can be found using $1/R = 1/R_1 + 1/R_2$ (Feynman, Leighton, & Sands, 1983).

[3]Well ... not exactly. Due to diffraction, the total number of independent connections that can be stored in an ideal medium is N^3, i.e., the volume of the hologram divided by the cube of the wavelength. So, in fact $N^{3/2}$ neurons can be connected with $N^{3/2}$ neurons.

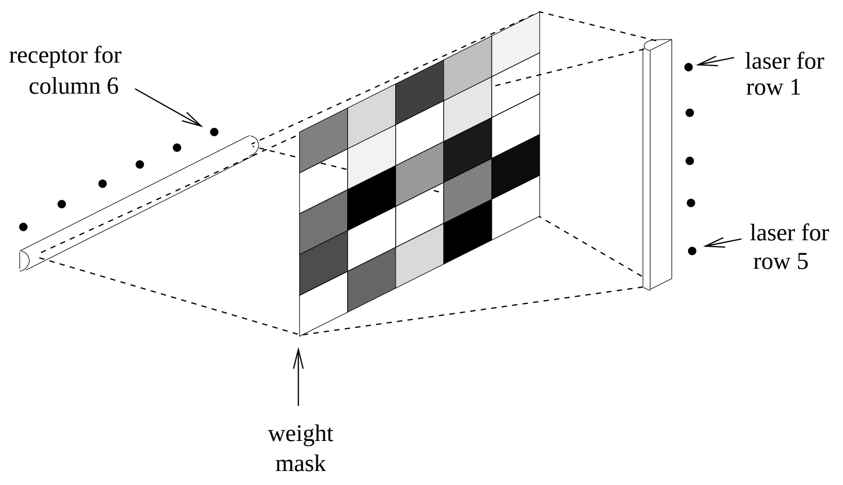

Figure 11.2: Optical implementation of matrix multiplication.

degree of correlation. The images are fed into a threshold device which will conduct the image with highest brightness better than others. This enhancement can be repeated for several loops.

11.1.4 Learning vs. non-learning

It is generally agreed that the major forte of neural networks is their ability to *learn*. Whereas a network with fixed, pre-computed, weight values could have its merit in industrial applications, on-line adaptivity remains a design goal for most neural systems.

With respect to learning, we can distinguish between the following levels:

1. **fixed weights**: the design of the network determines the weights. Examples are the retina and cochlea chips of Carver Mead's group discussed below (cf. a ROM (Read-Only Memory) in computer design);

2. **pre-programmed weights**: the weights in the network can be set only once, when the chip is installed. Many optical implementations fall in this category (cf. PROM (Programmable ROM));

3. **programmable weights**: the weights can be set more than once by an external device (cf. EPROM (Erasable PROM) or EEPROM (Electrically Erasable PROM));

4. **on-site adapting weights**: the learning mechanism is incorporated in the network (cf. RAM (Random Access Memory)).

11.2 Implementation examples

11.2.1 Carver Mead's silicon retina

The chips devised by Carver Mead's group at Caltech (Mead, 1989) are heavily inspired by biological neural networks. Mead attempts to build analogue neural chips which match biological neurons as closely as possible, including extremely low power consumption, fully analogue hardware, and operation in continuous time (table 9.1, point 3). One example of such a chip is the *Silicon Retina* (Mead & Mahowald, 1988).

Retinal structure

The off-center retinal structure can be described as follows. Light is transduced to electrical signals by photo-receptors which have a primary pathway through the *triad synapses* to the *bipolar cells*. The bipolar cells are connected to the *retinal ganglion* cells which are the output cells of the retina. The *horizontal cells*, which are also connected via the triad synapses to the photo-receptors, are situated directly below the photo-receptors and have synapses connected to the axons leading to the bipolar cells.

The system can be described in terms of the triad synapse's three elements:

1. the photo-receptor outputs the logarithm of the intensity of the light;
2. the horizontal cells form a network which averages the photo-receptor over space and time;
3. the output of the bipolar cell is proportional to the difference between the photo-receptor output and the horizontal cell output.

The photo-receptor

The photo-receptor circuit outputs a voltage which is proportional to the logarithm of the intensity of the incoming light. There are two important consequences:

1. several orders of magnitude of intensity can be handled in a moderate signal level range;
2. the voltage difference between two points is proportional to the contrast ratio of their illuminance.

The photo-receptor can be implemented using a photo-detector, two FET's[4] connected in series[5] and one transistor (see figure 11.3). The lowest photo-current is about $10^{-14}A$ or 10^5 photons

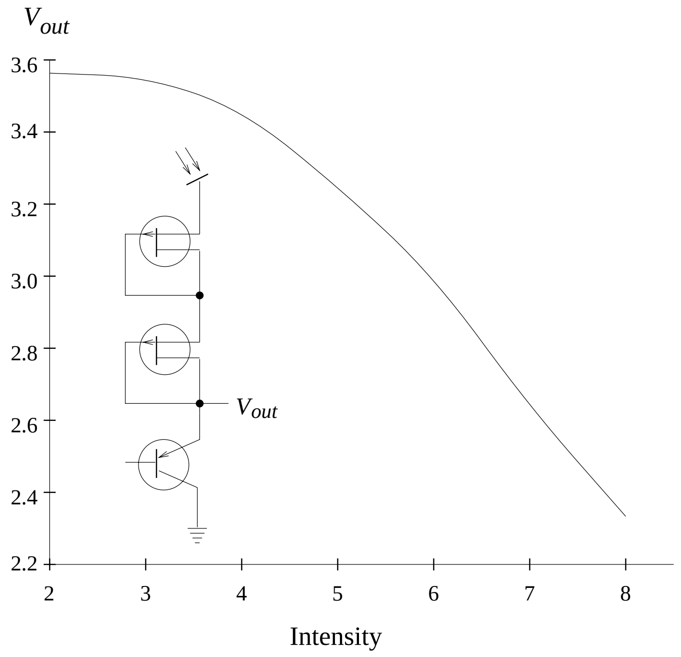

Figure 11.3: The photo-receptor used by Mead. To prevent current being drawn from the photo-receptor, the output is only connected to the gate of the transistor.

per second, corresponding with a moonlit scene.

[4]Field Effect Transistor

[5]A detailed description of the electronics involved is out of place here. However, we will provide figures where useful. See (Mead, 1989) for an in-depth study.

Horizontal resistive layer

Each photo-receptor is connected to its six neighbours via resistors forming a hexagonal array. The voltage at every node in the network is a spatially weighted average of the photo-receptor inputs, such that farther away inputs have less influence (see figure 11.4(a)).

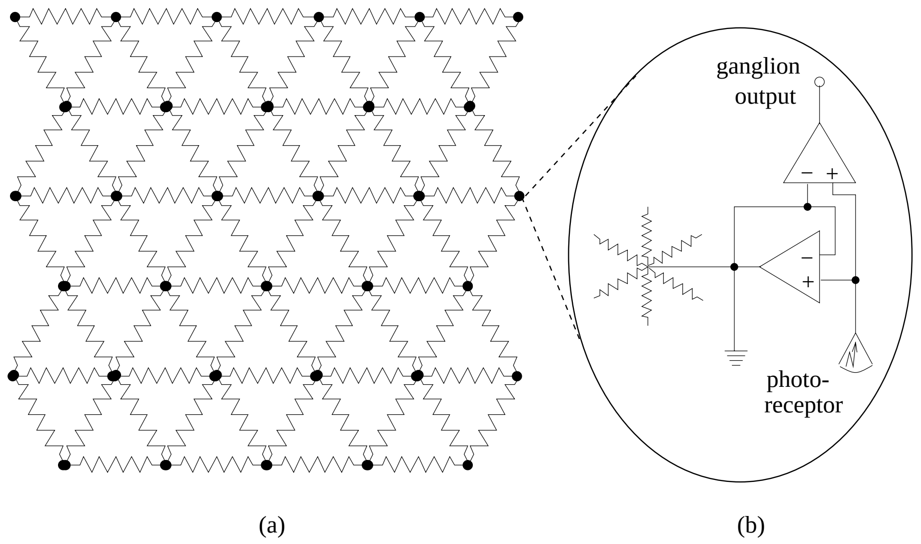

Figure 11.4: The resistive layer (a) and, enlarged, a single node (b).

Bipolar cell

The output of the bipolar cell is proportional to the difference between the photo-receptor output and the voltage of the horizontal resistive layer. The architecture is shown in figure 11.4(b). It consists of two elements: a wide-range amplifier which drives the resistive network towards the photo-receptor output, and an amplifier sensing the voltage difference between the photo-receptor output and the network potential.

Implementation

A chip was built containing 48×48 pixels. The output of every pixel can be accessed independently by providing the chip with the horizontal and vertical address of the pixel. The selectors can be run in two modes: static probe or serial access. In the first mode, a single row and column are addressed and the output of a single pixel is observed as a function of time. In the second mode, both vertical and horizontal shift registers are clocked to provide a serial scan of the processed image for display on a television display.

Performance

Several experiments show that the silicon retina performs similarly as biological retina (Mead & Mahowald, 1988). Similarities are shown between sensitivity for intensities; time responses for a single output when flashes of light are input; response to contrast edges.

11.2.2 LEP's LNeuro chip

A radically different approach is the LNeuro chip developed at the Laboratoires d'Electronique Philips (LEP) in France (Theeten, Duranton, Mauduit, & Sirat, 1990; Duranton & Sirat, 1989). Whereas most neuro-chips implement Hopfield networks (section 5.2) or, in some cases, Kohonen networks (section 6.2) (due to the fact that these networks have *local* learning rules), these digital neuro-chips can be configured to incorporate any learning rule and network topology.

Architecture

The LNeuro chip, depicted in figure 11.5, consists of an multiply-and-add or *relaxation* part, and a learning part. The LNeuro 1.0 has a parallelism of 16. The weights w_{ij} are 8 bits long in the relaxation phase. and 16 bit in the learning phase.

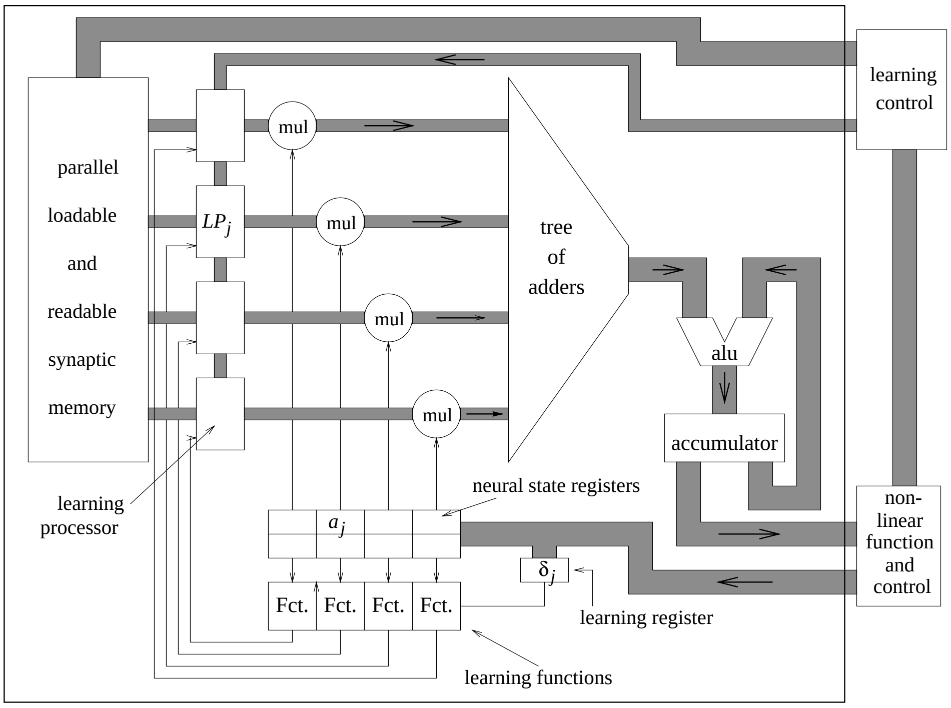

Figure 11.5: The LNeuro chip. For clarity, only four neurons are drawn.

Multiply-and-add

The multiply-and-add in fact performs a matrix multiplication

$$y_k(t+1) = \mathcal{F}\left(\sum_j w_{jk} y_j(t)\right). \qquad (11.1)$$

The input activations y_k are kept in the *neural state registers*. For each neural state there are two registers. These can be used to implement synchronous or asynchronous update. In the former mode, the computed state of neurons wait in registers until all states are known; then the whole register is written into the register used for the calculations. In asynchronous mode, however, every new state is directly written into the register used for the next calculation.

The arithmetical logical unit (ALU) has an external input to allow for accumulation of external partial products. This can be used to construct larger, structured, or higher-precision networks.

The neural states (y_k) are coded in one to eight bits, whereas either eight or sixteen bits can be used for the weights which are kept in a RAM. In order to save silicon area, the multiplications $w_{jk}y_j$ are serialised over the bits of y_j, replacing N eight by eight bit parallel multipliers by N eight bit AND gates. The partial products are saved and added in the *tree of adders*.

The computation thus increases linearly with the number of neurons (instead of quadratic in simulation on serial machines).

The activation function is, for reasons of flexibility, kept off-chip. The results of the weighted sum calculation go off-chip serially (i.e., bit by bit), and the result must be written back to the neural state registers.

Finally, a column of latches is included to temporarily store memory values, such that during a multiply of the weight with several bits the memory can be freely accessed. These latches in fact take part in the learning mechanism described below.

Learning

The remaining parts in the chip are dedicated to the learning mechanism. The learning mechanism is designed to implement the Hebbian learning rule (Hebb, 1949)

$$w_{jk} \leftarrow w_{jk} + \delta_k y_j \tag{11.2}$$

where δ_k is a scalar which only depends on the output neuron k. To simplify the circuitry, eq. (11.2) is simplified to

$$w_{jk} \leftarrow w_{jk} + g(y_k, y_j)\delta_k \tag{11.3}$$

where $g(y_k, y_j)$ can have value -1, 0, or $+1$. In effect, eq. (11.3) either increments or decrements the w_{jk} with δ_k, or keeps w_{jk} unchanged. Thus eq. (11.2) can be simulated by executing eq. (11.3) several times over the same set of weights.

The weights $\mathbf{w}_k$ related to the output neuron k are all modified in parallel. A learning step proceeds as follows. Every *learning processor* (see figure 11.5) LP$_j$ loads the weight w_{jk} from the synaptic memory, the δ_k from the *learning register*, and the neural state y_j. Next, they all modify their weights in parallel using eq. (11.3) and write the adapted weights back to the synaptic memory, also in parallel.