

Brain Phase

Jordan Bartee

During the fall of 2008 I began an independent study with Martijn Zwartjes, one of the original authors/programmers of Reaktor. We wanted to see if we could implement an artificial neural network in Reaktor Core, the low-level DSP underbelly to the Reaktor Primary Level. Specifically we wanted to create a modular set of neural building blocks consisting of non-linear hidden layer and output layer perceptrons, and backpropagation learning algorithms to train them.

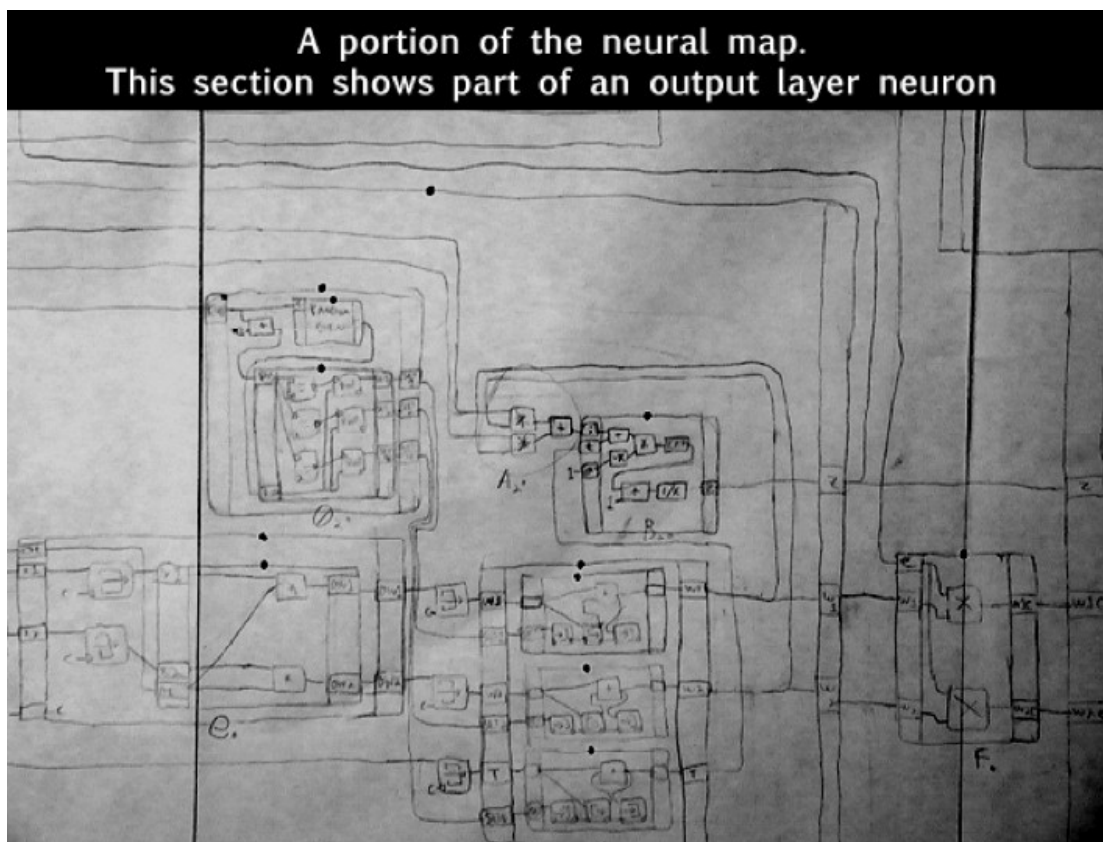
Of course the project was fraught with difficulties from the outset. Neither Martijn nor myself knew much about neural networks, and it was evident that Reaktor, while a better choice than Max/MSP due to its superior feedback resolution, was not going to be easy to work with compared to something like C or C++. I was determined to use Reaktor however because I could easily integrate an audio engine with the neural net, and have full midi/osc control over the learning algorithms (two things that would have been considerably more difficult to code in C).

After a long period of research we began building our neurons, followed by an even longer debugging process. During our debugging sessions we relied on a large physical map of the entire artificial brain, drawn in pencil (pictured on the following page). After we had a functional version of our network we set about collapsing the structure into the most modular, efficient, and optimized system we could.

The final version of the network consists of two inputs, two hidden layer neurons and one output layer neuron. On the Primary Level of Reaktor, the neurons appear as simple modules with a number of inputs and outputs. All the complexity of the neural structure is contained in the Core Level.

To create *Brain Phase I* I built a special version of the network hooked up to a series of audio modules that sonify various parameters of the network as it learns (weights, thresholds, deltaweights, deltathresholds, activation output, etc.). The same data is then routed into Max/MSP/Jitter where it is drawn into a matrix, thus generating the visual content. All the content is generated directly from the neural network— there is no human steering.

The neural network is initialized using a series of random seeds, and the network is trained on an XOR pattern set. Depending on the seed and pattern set, a nearly infinite string of variations can be produced. In this manner every performance of the piece represents only a single collapsed state from the global super-state of the piece.



Some of the algorithms used
in the network

①. RANDOMIZE

$$A. d = \sum_{i=1}^n x_i w_i$$

$$B. z = s(d+a) \quad \text{---} \quad s(x) = \frac{1}{1+e^{-ax}}$$

$$C. e = z(1-z)(y-z)$$

$$D. \Delta a = \lambda e$$

$$E. \Delta w_i = \Delta a x_i$$

$$F. g = \sum_{i=1}^r m_i e_i$$

$$G. e = z(1-z)g$$

$$H. \Delta a = \lambda e$$