Chapter #

UNSUPERVISED PROBABILISTIC NEURAL COMPUTATION IN ANALOGUE VLSI

Subtitle

H. Chen, P. Fleury, and A.F. Murray
School of Engineering and Electronics, Edinburgh University

Abstract: This chapter introduces the *Continuous Restricted Boltzmann Machine*, a probabilistic neural algorithm which is both useful in modelling continuous data and amenable to VLSI implementation. The capabilities of the model are explored with both artificial and real data. The computing units (neurons) and the unsupervised training rule have been implemented in VLSI. These results demonstrate the feasibility of a full VLSI model that uses continuous probabilistic behaviour to model the noise associated with all real signals, and therefore acts as a robust classifier or novelty detector.

Key words: Probabilistic neural computation, Probabilistic VLSI, Boltzmann Machine

1. INTRODUCTION

As interests in implantable systems and hybrid bio-electrical systems grow, sensors and electronic circuits are exposed to noisy environments that introduce not only serious sensory drifts but also non-negligible electronic noise. An intelligent system able to preprocess such noisy and drifting data at the sensory or bio-electrical interface is thus of great importance. By "intelligence", we mean the ability to classify or detect novelty in noisy data. Deterministic neural architectures (e.g. the Multi-Layer-Perceptron), though proven to be useful as classifiers, requires supervised, on-line calibration to deal with noisy and drifting data. In addition, the accurate arithmetic required by deterministic neural computation becomes infeasible in hardware as the signal-to-noise ratio greatly degrades in noisy environments.

Probabilistic neural computation offers a more flexible route to improved data modelling, utilising stochasticity to model the natural variability of real

data. The principle of probabilistic neural computation is that the input of a computing unit (called *neuron* hereafter) merely decides the *probability* of the neuron's output. The probabilistic neurons are therefore less sensitive to the noise or computational errors at their inputs, rendering probabilistic neural computation a potential candidate for underpinning an embedded intelligent and real system.

However, probabilistic models have not yet been shown to be both useful and hardware-amenable. The continuous-valued nature of most real-world signals further restricts the choice of models. Though precise computation of conditional probability and reliable Bayesian rules have been demonstrated in VLSI implementation in [1][2][3], precise computation is vulnerable to both intrinsic electronic noise and environmental interferences. The VLSI implementation of Bayesian rules is thus not suitable for underpinning an intelligent system exposed to noisy environments. Progress in stochastic arithmetic computation [4][5][6][7] suggests that coding continuous values into probabilistic digital pulses may reduce vulnerability to noise. However, the probabilistic-pulse scheme is linked to a fully-digital VLSI implementation which is both power- and area- consuming. It is therefore not ideal for an intelligent embedded system, where power and area are critical.

This chapter introduces a continuous-valued probabilistic model, the *Continuous Restricted Boltzmann Machine* (CRBM) [8], capable of modelling analogue (continuous-valued) data with a simple and unsupervised training algorithm. The Boltzmann Machine [9] is a probabilistic model consisting of binary stochastic neurons. The CRBM, though a descendant of the Boltzmann Machine, comprises continuous stochastic neurons. They are analogous to the neurons in the Diffusion Network [10], but have limited interconnect [11]. As a result, the CRBM shows greater ability to model analogue data than does the Boltzmann Machine or the Restricted Boltzmann Machine [8], while the training of the CRBM can be accomplished with only one step of Gibbs sampling, unlike the Boltzmann Machine or the Diffusion Network [9][10]. With a suitable approximation, the training algorithm also requires only subtraction and multiplication. It is thus computationally inexpensive in both software and hardware.

The capabilities of the CRBM are demonstrated with both artificial and real analogue data [8], as detailed in the following sections. The analogue VLSI implementation of the CRBM [12] is presented and discussed. As the stochastic behaviour of the CRBM is driven by noise injection to its neurons, the CRBM learns to adapt its "internal noise" to model the variability in the external environments, i.e. the "external noise". The full CRBM in VLSI is therefore a potential candidate for intelligent embedded system.

2. CONTINUOUS RESTRICTED BOLTZMANN MACHINE

The Continuous Restricted Boltzmann Machine consists of continuous-valued stochastic neurons. Let s_i represent the state of neuron i, and w_{ij} the connection between neuron i and neuron j. The stochastic behaviour of the neuron i is described as

$$s_i = \varphi_i\left(\sum_j w_{ij}s_j + \sigma \cdot N_i(0,1)\right)$$
$$\text{with} \quad \varphi_i(x_i) = \theta_L + (\theta_H - \theta_L)\cdot\frac{1}{1+\exp(-a_i x_i)} \tag{1}$$

where $N_i(0,1)$ represents a sample of Gaussian noise with zero mean and unit variance, σ represents a noise-scaling constant, and $\varphi_i(\cdot)$ denotes the sigmoid function with asymptotes at θ_H and θ_L. Parameter a_i controls the slope of sigmoid function and consequently the output probability distribution of the neuron i [13], as shown in Fig.1. A large value for a_i leads to a sharp sigmoid function resulting in a *binary stochastic* neuron. A small a_i renders the neuron's input noise negligible and leads to a *near-deterministic* neuron. When a_i has an intermediate value such that the noise variation is comparable to the width of the linear region of the sigmoid function, the neuron i becomes *continuous stochastic*. Its output distribution then approximates a Gaussian with mean $\sum_j w_{ij}s_j$ and variance σ^2. Adapting a_i effectively adapts the influence of the input noise on the output distribution, so we refer to a_i as the "noise-control parameter".

The CRBM is obtained by connecting the continuous stochastic neurons into a restricted form, as shown in Fig.2. Each circle in Fig.2 represents one stochastic neuron, and the neurons are grouped into two layers, one visible and one hidden layers. The black circles represent the bias neurons, whose outputs are permanently "on". The inter-layer connections of the neurons are

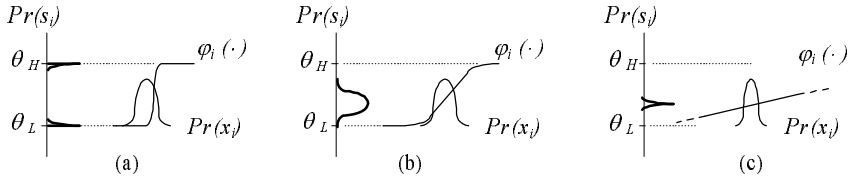

Figure #-1. The output distribution of a continuous stochastic neuron can vary from (a)binary stochastic, to (b) continuous stochastic, and to (c)near-deterministic.

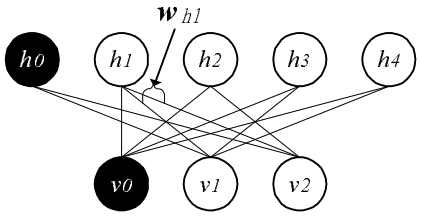

Figure #-2. The diagram of stochastic neurons connected in a restricted form.

symmetric ($w_{ij} = w_{ji}$). The visible neurons are the interface between the CRBM and its environment. The number of visible neurons thus normally corresponds to the number of dimensions of the environment or the data the CRBM must model. Hidden neurons function as individual "experts", each of which represents a particular part of the overall model defined by the connections between one hidden neuron and all visible neurons (ex. $\boldsymbol{w}_{h1}$ in Fig.2). Therefore, a particular hidden neuron responds more to specific inputs when their combination state aligns with part of the model associated with that hidden neuron. This advantageous feature is especially useful when the model is employed as a classifier. Classifying unknown data can then be achieved by observing the response of the hidden neurons, as demonstrated in [8].

The CRBM is a stochastic generative model that learns to reconstruct data points with the same probability distribution as the training data. The training rules of the CRBM are derived by minimising the "contrastive divergence" between the training data and the one-step Gibbs sampled data [14][8]. During training, each training datum $\boldsymbol{v}$ is clamped to the visible neurons such that the output probabilities of the hidden neurons are conditionally independent. The states of the hidden neurons $\mathbf{h}$ are then sampled. The same procedure is repeated so the visible and hidden neurons are sampled once more. The new visible states $\hat{\mathbf{v}}$ are derived from $\mathbf{h}$ and the corresponding new states of the hidden neurons $\hat{\mathbf{h}}$ re-sampled. $\hat{\mathbf{v}}$ and $\hat{\mathbf{h}}$ are called one-step Gibbs sampled states. The parameters of the CRBM are updated according to the following equation [8].

$$\begin{aligned} \Delta w_{ij} &= \eta_w \left(\langle v_i h_j \rangle - \langle \hat{v}_i \hat{h}_j \rangle \right) \\ \Delta a_i &= \frac{\eta_a}{a_i^2} \left(\langle s_i^2 \rangle - \langle \hat{s}_i^2 \rangle \right) \end{aligned} \tag{2}$$

where v_i and h_j refer to the states of visible neuron i and hidden neuron j, respectively, and s_i represents both v_i and h_i. The constants η_w and η_a define

the training rates of w_{ij} and a_i, respectively, and the bracket $\langle\cdot\rangle$ in Eq.(2) denotes the expectation value over all training data. Eq.(2) shows that the training rules of the CRBM mainly comprise multiplications and subtractions. Implementing the CRBM is therefore inexpensive in both software and hardware.

3. TRAINING ARTIFICIAL AND REAL DATA

Simple, but non-trivial, two-dimensional datasets, shown in Fig.3a, were generated to demonstrate the CRBM's ability to model analogue data. The training data in Fig.3a comprise two clusters of 200 data points, resembling the measurements of two sensors exposed to two different noisy environments. A CRBM with two visible neurons and four hidden neurons was trained to model this artificially-generated data, with $\eta_w = 1.5$, $\eta_a = 1$, $\theta_H = -\theta_L = 1$, and $\sigma = 0.2$ for all neurons. Note that the number of hidden neurons was chosen empirically to optimise modelling ability [8]. After 4000 training epochs, the CRBM reconstructed the data points as depicted in Fig.3b. This near-equilibrium reconstruction was obtained by initially setting the visible states to random values, and then Gibbs sampling hidden and visible neurons alternatively for 20 steps. The reconstructed points have approximately the same distribution as the training data, indicating that the CRBM modelled the training data successfully.

Fig.4 shows the evolution of the noise-control parameter $\{a_i\}$ during training, revealing that the CRBM adapts not only its weights but also its "internal noise" to model the variability in the training data. Very early in the training, the $\{a_i\}$ of the visible neurons rises abruptly to form *near-binary stochastic* neurons. At this stage, the CRBM effectively explores the gross structure of the training data. The $\{a_i\}$ of the visible neurons then gradually decreases to reduce the noise level of the visible neurons, such that the CRBM is able to capture the detailed distribution of the training data in

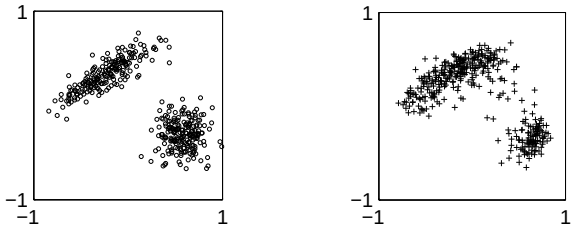

Figure #-3. (a) The artificially-generated two-dimensional training data (b) the reconstruction by the trained CRBM from 400 randomly-distributed data after 20 steps of Gibbs sampling

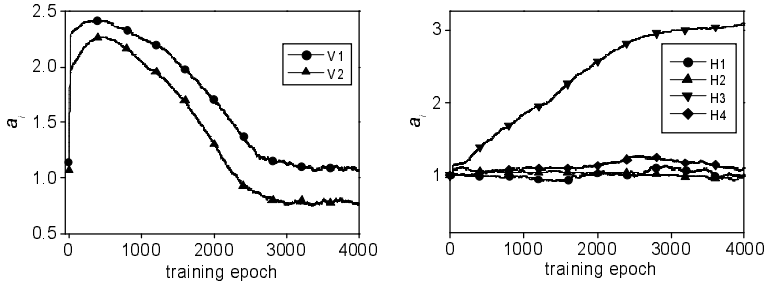

Figure #-4. The evolution of the "noise-control" parameter $\{a_i\}$ of (a)visible neurons (b)hidden neurons during training

its $\{w_{ij}\}$. The $\{a_i\}$ of the hidden neurons in Fig.4b, on the other hand, reveals the formation of a "decision maker". The a_i of the hidden neuron H3 rises to a large value (≈ 3) by the end of the training, while the $\{a_i\}$ of the other hidden neurons remained around *1*. The hidden neuron H3 has therefore become a near-binary stochastic neuron. Its output is likely to indicate which cluster the datum presented to visible neurons belongs to.

To highlight the modelling richness of the CRBM and to demonstrate its ability to classify real data [8], we have trained a CRBM with four hidden neurons to model the ECG data used in [15] and [16]. After training, the CRBM was able to reconstruct both normal heartbeats and ventricular ectopic beats (VEBs) (Fig.5), despite the fact that the VEBs represent merely 1% of training data. Furthermore, one hidden neuron became near-binary stochastic after training, as did the hidden neuron H3 in Fig.4b. This hidden neuron was thus used to indicate the presence of VEBs in the testing data, as depicted in Fig.6. The strong peaks in Fig.6 highlight the VEBs clearly. VL marks the minimum output corresponding to a VEB and QH marks the

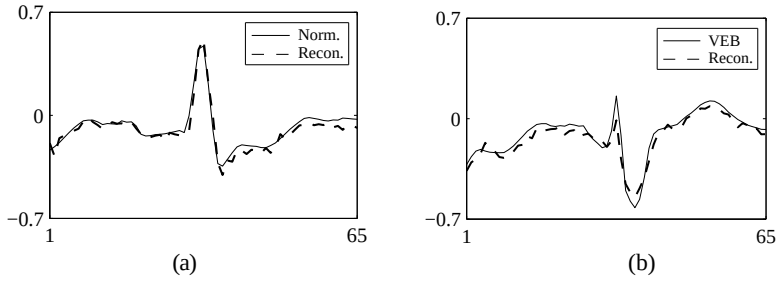

Figure #-5. The ECG traces, of (a) normal heartbeats (b) ectopic beats, sampled from training data (solid line) and reconstructed by the trained CRBM (dashed line) after 20 steps of Gibbs sampling from initially-random input.

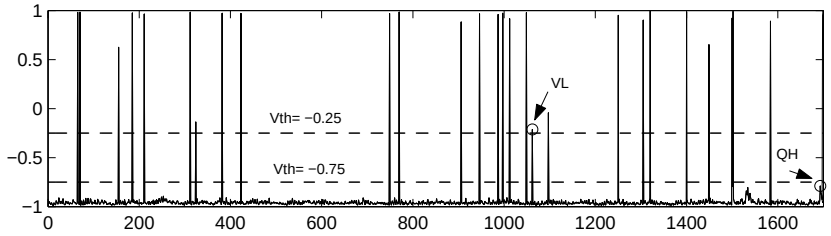

Figure #-6. The output of the decision-maker hidden neuron in response to 1700 testing data.

maximum output corresponding to a normal heartbeat. A simple linear classifier with threshold between the two dashed lines will detect VEBs with an accuracy of 100%. Compared to [16] where a 64-16-64 MLP was trained to model the same data, the best accuracy achieved was only 99.3%. The CRBM is therefore a reliable classifier, and it is expected that layering a supervised classifier on the CRBM, to "fuse" the output activities of hidden neurons, will lead to improved results.

4. SIMPLIFICATION FOR HARDWARE IMPLEMENTATION

The training rules in Eq.(2) can be simplified to further facilitate their hardware implementation. Updating the parameters by variable-size steps is more demanding than is a fixed-size step. We therefore chose to take only the sign of the contrastive divergence [15]. The bracket $\langle\cdot\rangle$ in Eq.(2) denotes the expectation value over all the training data. However, accumulating and retaining all these values is very impractical and area costly. We therefore approximate the expectation value by the average of only *four* training data as opposed to that of all training data. The simplified training rules are written as;

$$\begin{aligned}\Delta w_{ij} &= \eta_w sign\left(\left\langle s_i s_j \right\rangle_4 - \left\langle \hat{s}_i \hat{s}_j \right\rangle_4\right) \\ \Delta a_i &= \eta_a sign\left(\left\langle s_i^2 \right\rangle_4 - \left\langle \hat{s}_i^2 \right\rangle_4\right)\end{aligned} \tag{3}$$

where s_i and s_j represent the state of neuron i and j. Note that the denominator $1/a_i^2$ in Eq.(2) is also absorbed and $\langle\cdot\rangle_4$ indicates the average over four training data.

To validate the simplifications made, a CRBM with four hidden neurons was trained with Eq.(3) to model the artificially-generated training data previously used in Sec.3. The number of training data was reduced to 20, as

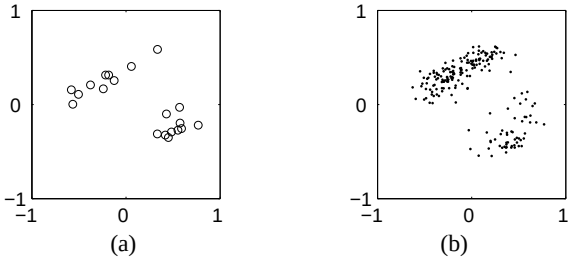

Figure #-7. (a)20 artificially-generated training data (b) 200 data points reconstructed by the CRBM after 20 steps of Gibbs sampling

shown in Fig.7a, in order to simplify the hardware testing. Fig.7b shows the data points reconstructed by the CRBM after 30,000 training epochs with η_w = 1.5, η_a = 15 for the visible neurons and η_a = 1 for hidden neurons. Fig.7b reveals that the CRBM was still able to capture the correct distribution of the training data. Therefore, the simplifications of the training rules made in Eq.(3) reduce the hardware complexity at the only cost of a slightly slower convergence time, with this set of data.

5. A NOISY NEURON IN VLSI

Fig.8 shows the circuit diagram of a noisy neuron. The outputs of the wide-range four-quadrant multipliers [12] are summed up into a current representing $\Sigma_j w_{ij} s_j$, while the differential pair, Mna and Mnb, transforms the noise voltage v_{ni} into a noise current $i_{ni} = g_m (v_{ni}-V_{nr})$. The voltage V_{sigma} then controls the transconductance g_m, scaling the noise current as σ in Eq.(1). The I-V converter, composed of an operational amplifier and a voltage-controlled active resistor, subsequently feeds its output to the sigmoid function. The exponential nonlinearity of the sigmoid function is achieved

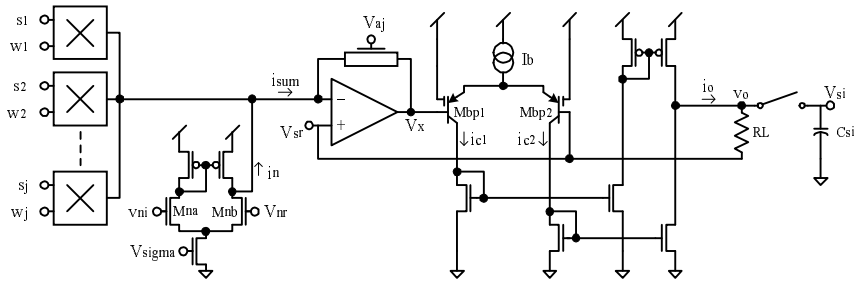

Figure #-8. The circuit diagram of a noisy neuron

by operating the PMOS differential pair, Mbp1 and Mbp2, in the lateral bipolar mode [17], resulting in a differential output current as following.

$$i_o = i_{c1} - i_{c2} = I_b \cdot \phi \left(\frac{I_{sum} \cdot R(V_{ai})}{V_t} \right) \tag{4}$$

where ϕ (·) denotes the sigmoid function φ (·) with $\theta_H = -\theta_L = 1$, and $V_t=kT/q$ is the thermal voltage. The resistor R_L finally converts i_o into the voltage $v_o=i_oR_L+V_{sr}$. Eq.(4) indicates that V_{ai} controls the feedback resistance of the I-V converter, and consequently adapts the nonlinearity of the sigmoid function as a_i in Eq.(1). The measured DC characteristics of the fourquadrant multiplier and the sigmoid function, corresponding to various V_{ai}, are shown in Fig.9. To ensure the normal operation of the voltage-controlled resistor, the voltage V_{ai} are limited to [1, 3]V [12].

The noise generator is implemented by Linear Feedback Shift Register (LFSR) [18].Fig.10a shows the measured amplitude distribution of one

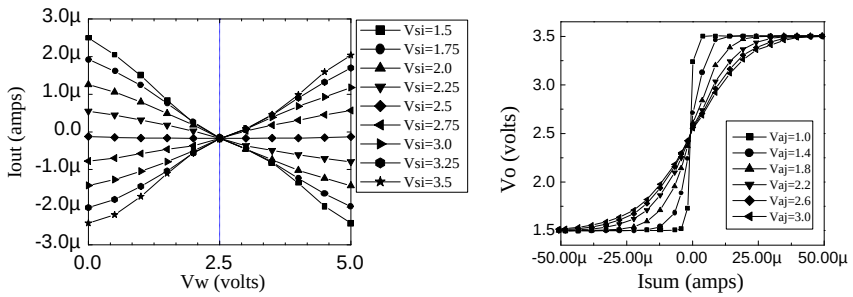

Figure #-9. The measured DC characteristics of (a) a wide-range four-quadrant multiplier and (b) a sigmoid function with variable nonlinearity.

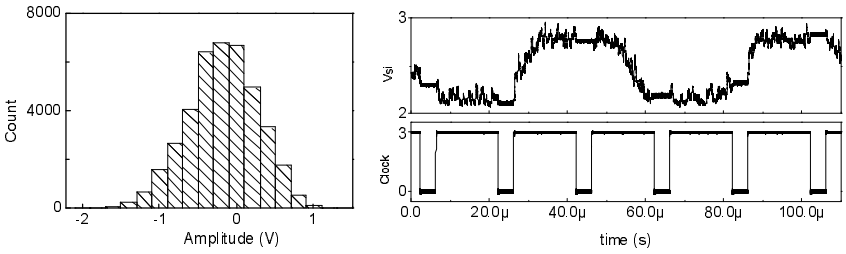

Figure #-10. The measured (a) amplitude distribution of noise signal and (b) the output of a noisy neuron with the injection of the signal in (a).

channel of the LFSR noise generator. The distribution approximates a Gaussian distribution. The maximum amplitude of the noise input v_{ni} is scaled to 0.4V and injected into the noisy neuron. Fig.10b shows the measured output of the noisy neuron with $\{s_i\}$ sweeping between 1.5 and 3.5V, $\{w_{ij}\}$ = 4V, and V_{ai}*=1.8V*. Such $\{s_i\}$ and $\{w_{ij}\}$ forces the neuron's output to sweep a sigmoid-shaped curve as Fig.9b, while the noise injection disturbs the curve to give a continuous-valued probabilistic output. A neuron state V_{si} was sampled periodically and hold with negligible clockfeedthrough whenever the clock went low. So our noisy neuron has a continuous-valued probabilistic output rather than the binary-valued output, which the stochastic neuron of the Boltzmann Machine in [19] has.

6. MINIMISING-CONTRASTIVE-DIVERGENCE TRAINING IN VLSI

The block diagram of the MCD training circuit in VLSI, along with the digital control signals, is depicted in Fig.11a. During the training mode ($\overline{\text{LER}}/\text{REF}$ =1), the first two clocks, *CKsi* and *CKsj*, sample s_i and s_j and the four-quadrant multiplier produces a current I_+ proportional to $s_i \cdot s_j$. Then *CK+* samples and memorises I_+. The one-step Gibbs sampled states $\hat{s}_i$ and $\hat{s}_j$ are sampled by *CKsip* and *CKsjp* to produce another current I_-. ***C****Kq* then sample and holds the output of current subtractor, I_{sub} , which represents the difference between the initial data and its one-step reconstruction. It should be noted that using the same multiplier to compute $s_i \cdot s_j$ and $\hat{s}_i \cdot \hat{s}_j$ enhances the training quality, as the subtractor cancels the offsets and other non-idealities of the multiplier. The previous clocking sequence is repeated for *four* times, such that *four* I_{sub} are accumulated and averaged to produce I_{ave}, which

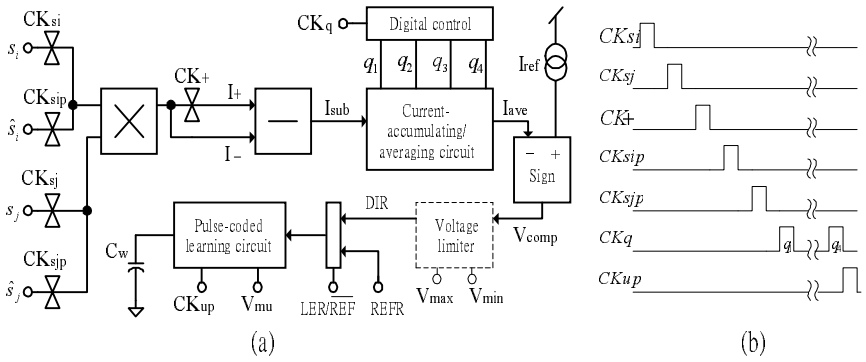

Figure #-11. (a)The diagram of the MCD training circuit and (b)its digital control signals

represents $\langle s_i s_j \rangle_4 - \langle \hat{s}_i \hat{s}_j \rangle_4$ in Eq.(3). The sign circuit [20] then compares I_{ave} to a reference current to determine the training direction (DIR). Finally, the training circuit, triggered by *CKup*, "nudges" the voltage stored on the C_w up or down accordingly. The dash-lined box represents the voltage-limiting circuit [12] used only for the parameters $\{a_j\}$, whose range is limited to [1,3] V. In the refresh mode (LER/$\overline{\text{REF}}$=0), the signal REFR, rather than DIR, determines the updating direction, maintaining the parameter to a learnt value for the reconstruction phase of multi-step Gibbs sampling.

Fig.12a shows the circuit of our current subtractor. Its accuracy is enhanced by using improved Wilson current mirrors to minimise the effect of channel length modulation. The bias current I_{bias} defines the reference-zero current, such that the output current I_{sub} is always non-negtive. The sum of the current inputs I_{in+} + I_{bias} is then accurately mirrored to M8 & M6, while the current I_{in-} is mirrored to M13 & M15. As a consequence, the difference current, $I_{in+} - I_{in-} + I_{bias}$, is mirrored to the output transistors M20 and M18. The measured DC characteristics of the current subtractor are depicted in Fig.12b, revealing that the response is very linear. The bottom graph of Fig.12b expresses the error, i.e. the non-linearity, as a percentage. The error remains extremely low (< 5%) in the critical range where $I_{in+} \approx I_{in-}$,

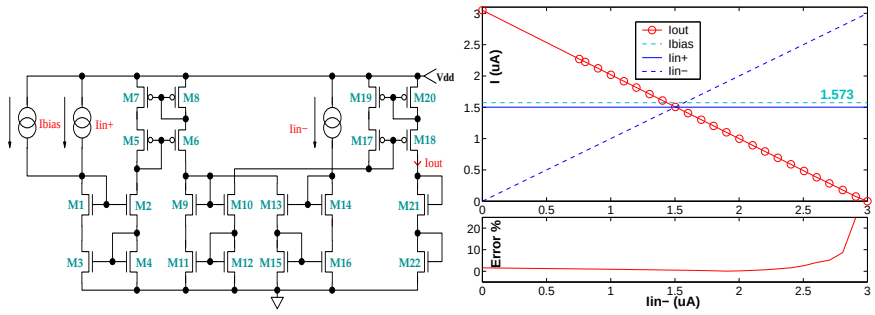

Figure #-12. (a) The circuit and (b) the measured DC characteristic of the current comparator

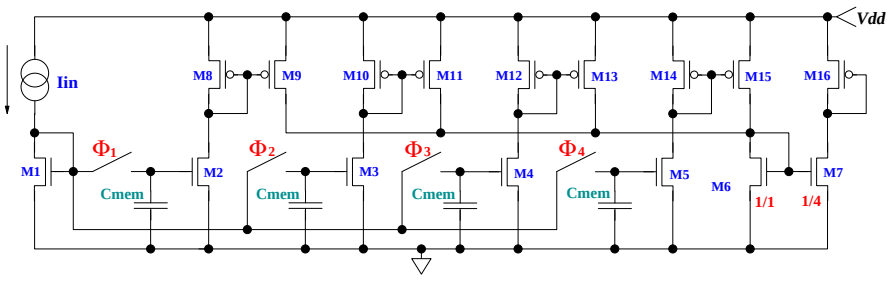

Figure #-13. Current accumulator/averaging circuit

and only increases when the output current approaches zero.

To accumulate four I_{sub} and calculate its average value, the current accumulator circuit [20] (Fig.13) juxtaposes four dynamic current mirrors [21]. Each current mirror is active on a different clock cycle (q_1~q_4 of Fig.11). The dynamic current mirror relies on the fact that the current flowing through a MOS transistor depends primarily on the voltage applied to its gate. So for two matched MOSFETs, any V_{GS} will generate an identical current in each transistor. As the current I_{in} produces a voltage at the gate of the active load M1, each clock cycle passes this voltage onto one of the short-term memory capacitors, C_{mem}, biasing another identical transistor. After four clock cycles, four I_{sub} are memorised on M2, M3, M4 and M5, and are subsequently mirrored (by M8-M15) and summed at M6. Finally, the 4-to-1 size ratio between M6 and M7 computes the average of the four currents. Although not represented in Fig.13, dummy switches are used to compensate for charge injection [22] caused by the switching of q_1-q_4. The test results presented in Table 2 show that the accumulation and averaging of currents occurs with 7% accuracy. However, the consistency of the results over 6 chips suggests that the output currents are actually mostly affected by small negative offsets. These offsets are acceptable as they will be trained out during training, reducing the margin of error of our circuit.

The pulse-coded training circuit consists of a pulse generator (Fig.14a) and the training cell proposed in [23] (Fig.14b). When the training cell is enabled (EN=1), the training cell charges or discharges the voltage stored on C_w according to the directional input $\overline{\mathrm{INC}}/\mathrm{DEC}$. Though the training step-size is adjustable via V_P and V_N in Fig.14b, we prefer to adjust it by controlling a width-variable pulse to the input (EN) of the training cell. This enables us to control the training step precisely by monitoring the pulse width, controlled by the voltage V_{mu} [20]. $\boldsymbol{V}_{mu}$ fixes the training parameter η of Eq.(3). Since the input capacitance of each training cell is less than 0.1pF, one pulse generator can control all the training cells, giving rise to a uniform training. Each of the CRBM training parameters η_w, η_{av}, and η_{ah} requires only one pulse generator. Fig.14c shows the chip results. Pulses as short as 5ns (V_{mu}=2.5V) to as wide as 100ms (V_{mu}=4.5V) can be obtained. With V_N=0.75V, V_P=4.3V and C_w=1pF changes as small as 1mV are easily achieved, therefore allowing accurate control of the training parameters.

Table #-1. Test results of the current accumulator/averaging circuit

I_{in}	Chip1	Chip2	Chip3	Chip4	Chip5	Chip6	Error(%)
0.5	0.464	0.473	0.463	0.451	0.467	0.474	6.94
0.999	0.937	0.948	0.931	0.878	0.939	0.936	7.09
1.499	1.409	1.419	1.397	1.351	1.405	1.411	6.69
1.999	1.875	1.891	1.857	1.819	1.864	1.881	6.72
2.498	2.335	2.333	2.321	2.263	2.334	2.329	7.15
2.998	2.794	2.816	2.776	2.729	2.792	2.796	7.14

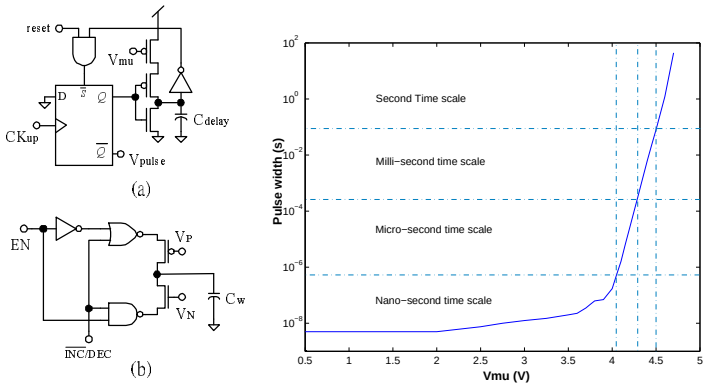

Figure #-14. The pulse-coded training circuit consists of (a) a pulse generator (b) the training cell proposed in [23]. (c)The measured variable pulse width versus V_{mu}

The MCD training circuitry controls every single training parameter on-chip. To ease the testing, s_i and $\hat{s}_i$ are fixed at 3.5V, while s_j and $\hat{s}_j$ alternate between 1.5V and 3.5V, as shown by the traces *sj* and *sjp* in Fig.15 and Fig.16. With the reference zero being defined at 2.5V, the parameters should learn down when s_j = 3.5V and $\hat{s}_j$ = 1.5V, and learn up when s_j = 1.5V and $\hat{s}_j$ = 3.5V. Fig.15 shows the measured on-chip training of two parameters, w_{ij} and $a_{i,}$ with different training rates. Both parameters were initially refreshed to 2.5V when LER/REF=0, and subsequently started to train up and down in response to the changing *sj* and *sjp* when LER/REF=1. As controlled by different pulse widths (Pulse1 and Pulse2), the two parameters were updated with different step sizes (10mV and 34mV), but in the same direction. The

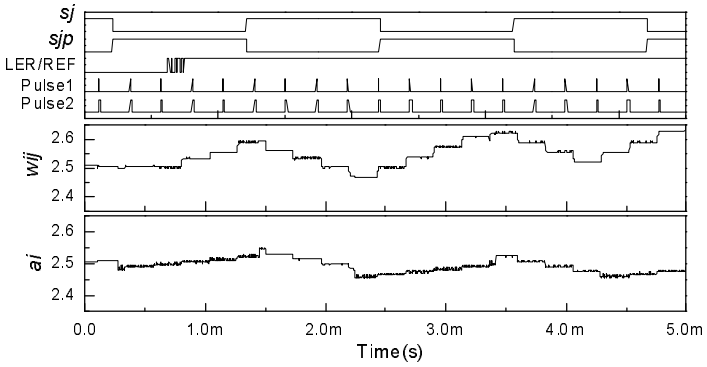

Figure #-15. Measurement of parameter ai and wij training in different training rates

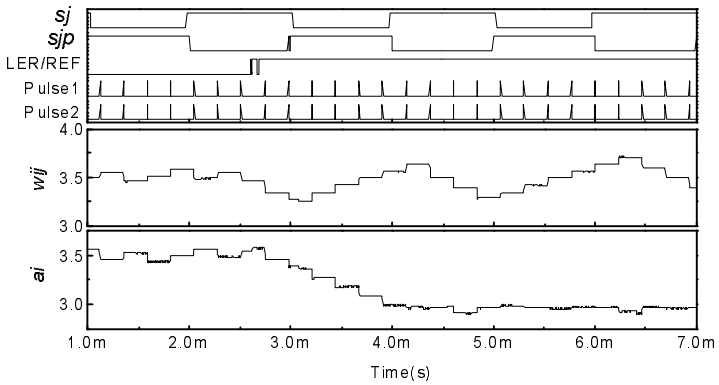

Figure #-16. Measurement of parameter ai and wij training in different direction

trace of parameter a_i shows digital noise attributable to sub-optimal layout and has been improved in subsequent design. Fig.15 shows the measured on-chip training of two parameters, w_{ij} and $a_{i,}$ in different directions. Both parameters were refreshed to 3.5V and the voltage V_{max} for a_i set to 3V. Therefore, the voltage-limiting circuit forces a_i to decrease toward V_{max}, while w_{ij} remains training up and down when LER/REF=1.

7. CONCLUSION

The Continuous Restricted Boltzmann Machine is a probabilistic neural algorithm amenable to hardware implementation. Simulation results show that the CRBM models the variability of data by adapting not only its weights but also its "internal noise" to guarantee a robust output for classification. The measured results, on the other hand, demonstrate on-chip unsupervised training for the CRBM neurons. The full CRBM system will thus has continuous probabilistic behaviour that enhances not only the system's "intelligence" for classification, but also the system's immunity to noise and local computational errors. As the continuous stochastic behaviour relies on the noise injection into the system, the full CRBM system further opens a possibility of utilising VLSI intrinsic noise for computation. This issue is especially important when the fabrication process enters the deep-sub-micron era, where VLSI intrinsic noise becomes non-negligible.

References

[1] Specht, D. F., "Probabilistic Neural Networks," *Neural Networks*, vol. 3 pp. 109-118, 1990.

[2] Aibe, N., Yasunaga, M., Yoshihara, I., and Kim, J. H. A Probabilistic Neural Network Hardware System Using a Training-parameter Parallel Architecture. 3, 2270-2275. 12-5-2002. Proceedings of the 2002 International Joint Conference on Neural Networks.

[3] Hsu, D., Bridges, S., Figueroa, M., and Diorio, C. Adaptive Quantization and Density Estimation in Silicon. 15. 2003. Advances in Neural Information Processing Systems (NIPS02). 2002.

[4] Bout, D. E. V. D. and MillerIII, T. K., "A Digital Architecture Employing Stochasticism for the Simulation of Hopfield Neural Nets," *IEEE Transactions on Circuits and Systems*, vol. 36, no. 5, pp. 732-738, May1989.

[5] Clarkson, T. G., Gorse, D., Taylor, J. G., and C.K.Ng, " Training Probabilistic RAM Nets Using VLSI Structures," *IEEE Transactions on Computers*, vol. 41, no. 12, pp. 1552-1561, Dec.1992.

[6] Clarkson, T. G., Ng, C. K., and Guan, Y., "The pRAM : An Adaptive VLSI Chip," *IEEE Transactions on Neural Networks*, vol. 4, no. 3, pp. 408-412, May1993.

[7] Brown, B. D. and Card, H. C., "Stochastic Neural Compuation I : Computational Elements," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 891-905, Sept.2001.

[8] Chen, H. and Murray, A. F., "A Continuous Restricted Boltzmann Machine with an Implementable Training Algorithm," *IEE Proceedings of Vision, Image and Signal Processing*, vol. 150, no. 3, pp. 153-158, 2003.

[9] Hinton, G. E. and Sejnowski, T. J., "Training and Retraining in Boltzmann Machine," in Rumelhart, D., McClelland, J. L., and the PDP Research Group (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* Cambridge, Massachusetts: MIT, 1986, pp. 283-317.

[10] Movellan, J. R., "A training theorem for networks at detailed stochastic equilibrium," *Neural Computation*, vol. 10, no. 5, pp. 1157-1178, 1998.

[11] Smolensky, P., "Information Processing in Dynamical Systems: Foundations of Harmony Theory," in Rumelhart, D., McClelland, J. L., and the PDP Research Group (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* Cambridge, Massachusetts: MIT, 1986, pp. 195-281.

[12] Chen, H., Fleury, P., and Murray, A. F. Minimising Contrastive Divergence in Noisy, Mixed-mode VLSI Neurons. Advances in Neural Information Processing Systems (NIPS2003) . 2004.

[13] Frey, B. J. Continuous Sigmoidal Belief Networks Trained Using Slice Sampling. 9, 452-458. 1997. Advances in Neural Information Processing Systems.

[14] Hinton, G. E., "Training Products of Experts by Minimizing Contrastive Divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771-1800, Aug.2002.

[15] Murray, A. F., "Novelty detection using products of simple experts - a potential architecture for embedded systems," *Neural Networks*, vol. 14, no. 9, pp. 1257-1264, 2001.

[16] Tarassenko, L., Clifford, G., and Townsend, N., "Detection of ectopic beats in the electrocardiogram using an auto-associative neural network," *Neural Processing Letters*, vol. 14, no. 1, pp. 15-25, 2001.

[17] Vittoz, E., "MOS Transistor Operated in the Lateral Bipolar Mode and Their Application in CMOS Technology," *IEEE Jounals of Solid-State Circuits*, vol. SC-18, no. 3, pp. 273-279, June1983.

[18] Alspector, J., Gannett, J. W., Haber, S., Parker, M. B., and Chu, R., "A VLSI-efficient Technique for Generating Multiple Uncorrelated Noise Sources and Its Application to Stochastic Neural Networks," *IEEE Transactions on Circuits and Systems*, vol. 38, no. 1, pp. 109-123, 1991.

[19] Alspector, J., Jayakumar, A., and Luma, S. Experimental Evaluation of Training in a Neural Microsystem. 4, 871-878. 1992. Advances in Neural Information Processing Systems (NIPS91).

[20] Fleury, P. and Murray, A. F. Mixed-Signal VLSI Implementation of the Products of Experts' Contrastive Divergence Training Scheme. 5, 653-656. 2003. Proceedings of the IEEE International Symposium on Circuits And Systems (ISCAS' 2003).

[21] Wegmann, G. and Vittoz, E. A., "Basic Principles of Accurate Dynamic Current Mirrors," *IEE Proceedings on Circuits, Devices and Systems*, vol. 137, no. 2, pp. 95-100, 1990.

[22] Wegmann, G., Vittoz, E. A., and Rahali, F., "Charge Injection in Analog {MOS} Switches," *IEEE Journal of Solid-State Circuits (JSSC)*, vol. 22, no. 6, pp. 1091-1097, Dec.1987.

[23] Cauwenberghs, G., "An Analog VLSI Recurrent Neural Network Training a Continuous-Time Trajectory," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, pp. 346-361, Mar.2003.