Device Mismatch in a Neuromorphic System Implements Random Features for Regression

Ole Richter*, René Felix Reinhart†*, Stephen Nease*, Jochen Steil†*, Elisabetta Chicca*

*Cluster of Excellence Center in Cognitive Interactive Technology - CITEC
†Research Institute for Cognition and Robotics - CoR-Lab
Bielefeld University, Bielefeld, Germany
Email: orichter@techfak.uni-bielefeld.de, freinhart@cor-lab.uni-bielefeld.de, snease@techfak.uni-bielefeld.de
jsteil@cor-lab.uni-bielefeld.de, chicca@cit-ec.uni-bielefeld.de

***Abstract*—We use a large-scale analog neuromorphic system to encode the hidden-layer activations of a single-layer feed forward network with random weights. The random activations of the network are implemented using the device mismatch inherent to analog circuits. We show that these activations produced by analog VLSI implementations of integrate and fire neurons are suited to solve multi dimensional, non linear regression tasks. Exploitation of the device mismatch eliminates the storage requirements for the random network weights.**

***Index Terms*—Neuromorphic, Feed Forward Neural Networks With Random Weights, Extreme Learning Machine, aVLSI, device mismatch.**

I. Introduction

The use of randomness in neural networks has been the subject of considerable work for many decades. The inventors of radial basis function networks (RBFs) [1] were among the first to propose random selection of network parameters, which in their case was the set of RBF centers. This idea was later applied to multilayer perceptrons (MLPs) [2], [3]. These networks contain a single hidden layer of neurons whose weights have been randomly initialized, and the output weights can be computed analytically by solving a set of linear equations. While random hidden layer (RHL) networks later gained popularity under the moniker Extreme Learning Machine (ELM) [4], we will refer to them as RHL-MLPs for clarity. In contrast to traditional neural networks, which are trained by gradient descent and require backpropagation of errors, learning in RHL-MLPs is restricted to a linear read-out layer and can be accomplished by fast and efficient linear regression. There are many benefits of using RHL-MLPs over traditional neural networks [5]: RHL-MLPs learn very quickly, tend to avoid problems of gradient descent such as local minima, and can be used to train networks with non differentiable activation functions. RHL-MLPs have been successfully used for non linear regression and classification tasks.

This paper presents results from multidimensional function approximation when the hidden layer activations of an RHL-MLP are implemented with neuromorphic hardware. Neuromorphic hardware can be broadly defined as any circuit or system optimized for the simulation or emulation of neurobiology. A subset of neuromorphic hardware focuses on creating analog circuits which implement spiking neural networks [6], [7]. Such systems are beneficial because of their low power consumption and utilization of spikes for computation, which results in lower-latency processing.

We create the random input weights and resulting hidden layer activity in one such system using the device mismatch inherent to all analog hardware. Rate-coded input spike trains are applied to differential pair integrator (DPI) synapses [8], which transmit excitatory postsynaptic currents to leaky integrate-and-fire neurons [9]. All of the circuits are identically biased, but since they are different physical devices, their responses differ due to device-to-device mismatch [10]. This mismatch causes different responses in the hidden-layer neurons to the same input spike train. We exploit the device mismatch to efficiently implement the hidden layer of an RHL-MLP on neuromorphic hardware. In fact, there is no additional storage required to represent the random network parameters on chip. The alternative approach would be to build a dedicated memory on chip large enough for storing all these random weights. We record the hidden-layer responses through the Address-Event Representation (AER) protocol [11], and then train the weights between the hidden layers and output neuron offline.

Previous work has shown that this approach yields promising results. The idea to use silicon spiking neurons for RHL-MLPs was originally proposed in [12], and architectural details and simulations were provided in [13]. The architecture of [13] consisted of current mirrors which feed the RHL-MLP inputs into spiking neurons. Mismatch of the current mirrors provided the random weights. Simulations of the architecture were completed using a single neuron on a Field Programmable Analog Array, and the randomness was simulated in Matlab. In contrast, our system realizes the entire hidden layer on chip, and we exploit the mismatch present in the chip.

Another recent result [14] demonstrated that a combination of systematic and random offset in an analog neuron's transfer function allow for function approximation. As above, the function approximation was based on readings from a single test circuit, rather than an array of devices on a fully-realized platform.

Function approximation has previously been demonstrated on an analog neuromorphic platform [15]. In that work, different activation functions were induced by explicitly adding different biases to the neurons. These biases were Gaussian spike trains with randomly chosen mean firing rates supplied as synaptic current. Rather, we demonstrate that sufficiently mismatched hardware generates activation functions which are varied enough to support function approximation without the need for another source of randomness. We also note that [15] implemented weighted connections from the hidden layer to the output neurons on-chip, while we do so on a standard desktop computer.

The paper is structured as follows. Section II discusses the device mismatch which enables the randomization of input weights. Section III contains a brief overview of the architecture of the RHL-MLP on chip. Section IV presents measured activation functions from the network, as well as results from offline learning tests for 1- and 2-dimensional functions. Section V discusses future work which will extend the capabilities of such hardware.

II. Network Architecture

The proposed system implements the following portion of an RHL-MLP on chip:

$$h_i = g\left(\mathbf{w}_i^T \mathbf{x} + \mathbf{b}_i^T \mathbf{b}\right), \quad i = 1, \ldots, H \tag{1}$$

where h_i is the activity of the ith hidden-layer neuron, $g(a)$ is the neuron's activation function, $\mathbf{w}_i$ and b_i are the randomized weights and bias from the input neurons to the ith hidden-layer neuron, $\mathbf{x}$ is the activity of the input neurons, and $\mathbf{b}$ is the bias activity. In traditional RHL-MLPs, $g(a) = 1/(1 + exp(-a))$, but we will replace this function with the measured activation functions of leaky integrate-and-fire neurons. The basic architecture of this implementation is shown in Fig. 1.

All of these terms are implemented on an aVLSI platform with 128 leaky integrate-and-fire neurons [16]. Each neuron is connected to 2 excitatory and 2 inhibitory DPI synapses, as well as 28 excitatory plastic synapses, which are not used in this work. Excitatory synapses source current onto the neuron's membrane capacitance, and inhibitory synapses sink current from this capacitance. The number of inhibitory synapses per neuron sets the limitation on the dimensionality of the input for the 1-chip setup.

In order to attain the widest possible range of $\mathbf{w}_i$ values, half of the synapses are inhibitory, and half are excitatory, assuming that we focus on an even distribution of weights. For a two-dimensional input, this yields four possible combinations of synapse types and synapse sources. We have therefore divided our hidden neurons into four groups, each group getting a different combination of inhibitory/excitatory synapses from input x_1 or x_2.

The biases for each group are set by a combination of two neurons firing at a constant rate. Neuron b_1 is connected to the hidden neurons via an excitatory synapse. Neuron b_2 is connected to the hidden neurons via an inhibitory synapse.

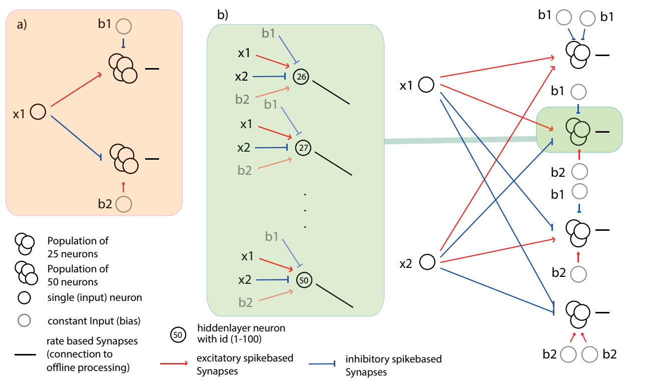

a)
b1
x1
b2
b)
b1
x1
x2
b2
26
27
50
x1
x2
b1
b2
Population of 25 neurons
Population of 50 neurons
single (input) neuron
constant Input (bias)
rate based Synapses (connection to offline processing)
hiddenlayer neuron with id (1-100)
excitatory spikebased Synapses
inhibitory spikebased Synapses

Fig. 1. (a) Architecture of an RHL-MLP/ELM with 1-dimensional input. The input x_1 is transmitted to all $H = 100$ neurons in the hidden layer, either through an excitatory (neuron 1 to 50) or inhibitory (neuron 51 to 100) synapse. Each neuron uses its own physical synapse with individual mismatch. To retain a certain bias activity a constant excitatory frequency is applied to the last 50 neurons, and a constant inhibitory frequency is applied to the first 50 neurons. The mean frequency is then computed from the AER output events of every neuron and stored for offline computation. (b) Architecture of an ELM with 2-dimensional input. The input $\mathbf{x}$ is broadcast to $H = 100$ neurons. Each neuron receives an input from x_1 and x_2 through either an inhibitory or excitatory synapse. Each neuron is also connected to a bias that is a combination of a constant excitatory input b_1 and a constant inhibitory input b_2. The neuron activations are recorded through an Address Event Representation (AER) Interface and transmitted to a computer, where offline processing is performed.

III. Device Mismatch Enables Random Hidden-Layer Weights

Variation in the fabrication of CMOS devices yields inevitable mismatch between the characteristics of identically-drawn transistors. In above-threshold operation, this mismatch is typically attributed to variations in the threshold voltage V_T and the current scaling factor β, which are usually modelled as varying independently [17]. The mapping of subthreshold current mismatch directly to V_T and β mismatch has been debated within the literature, but it is generally considered to be affected similarly [17].

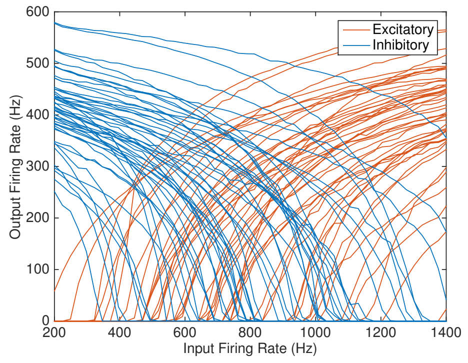
The fixed-weight synapses on this chip which implement the weight and bias terms are restricted such that all synaptic weights of a given type (excitatory or inhibitory) have the same nominal value. However, each synapse has some random offset ϵ which is a function of the mismatch. Thus, each w_i and b_i in Eq. 1 can be written as

$$w_i, b_i = \begin{cases} w_{exc} + \epsilon_i & : excitatory \\ -(w_{inh} + \epsilon_i) & : inhibitory \end{cases}, \tag{2}$$

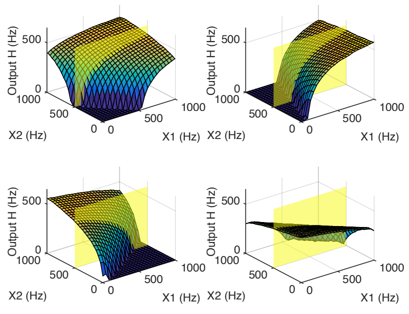
where w_{exc} and w_{inh} are constants across the chip and ϵ_i is unique to each synapse.

Additionally, the leaky integrate-and-fire neurons are subject to mismatch. We can model them very simply as linear activation functions over a certain frequency range. If we do so, both the slope and frequency range are subject to mismatch, as shown below:

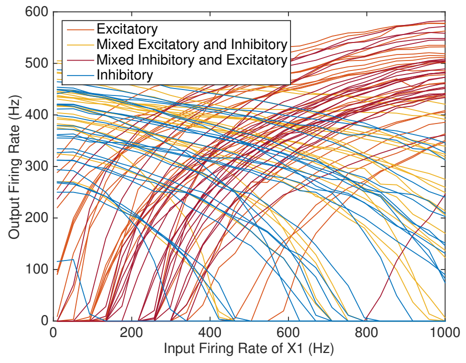
$$g(f) = \begin{cases} 0 & : f < f_1 + \epsilon_{f1,i} \\ (m + \epsilon_{m,i})f & : f_1 + \epsilon_{f1,i} < f < f_2 + \epsilon_{f2,i} \\ g_{max} + \epsilon_{g_{m,i}} & : f > f_2 + \epsilon_{f2,i} \end{cases}. \tag{3}$$



(a) 1-dimensional activations



(b) 2-dimensional activations



(c) 2-dimensional activations cut through

Fig. 2. Hidden layer activities as a function of input activity. (a) All 100 activities in the 1-dimensional case. Data taken for $V_w = 650mV$ [16], $V_{ref} = 200mV$ (b) The 2D measured activities for neurons 1, 26, 51, 76 representative for each sub population. Data is taken for $V_w = 650mV$ [16], $V_{ref} = 200mV$ [16] for all 100 neurons. (c) One representative cut though measurements in the 2-dimensional case. The yellow plane in 2b indicates the position of 2c. It shows x_1 swept and x_2 fixed at 505Hz.

Our system exploits this mismatch to automatically generate the randomized activations required for RHL-MLPs. The mismatch occurs at both the neuron and synapse level. To quantify the relative contributions of each circuit to the overall mismatch, we ran a series of experiments in which we measured neural activity with different circuits connected to the neuron. In each experiment, we computed the mean firing rate of each of the 100 neurons over one second of stimulation, normalized the results, and then computed the standard deviation of the resulting data distribution.

When the only source of input to the neuron was a DC current (I_{in}) [16] provided by a pFET biased with a gate voltage of 2.78V, the standard deviation of the distribution of normalized rates was 0.0716. When a regular spike train of 200 Hz was applied to excitatory synapses onto the neurons, the standard deviation was 0.1289. When a constant current (I_{in}) [16] was again injected by a pFET with a gate voltage of 2.78V and a regular inhibitory spike train was applied at 700Hz, the standard deviation was 0.1667. Finally, when the inhibitory (700Hz) and excitatory (260Hz) synapses were both active, the standard deviation is 0.2505. All data is normalised by the mean firing rate of the population for each measurement (which was approximately 340Hz in all cases). Thus all three circuits have a significant impact on the total mismatch.

For every excitatory input connection we provided a constant inhibitory bias b_1 ($1D$: 100Hz; $2D$: 250Hz) and for every inhibitory input synapse an excitatory bias b_2 ($1D$: 700Hz; $2D$: 600Hz) on a second physical synapse, as illustrated in Fig. 1. These two biases are constant regular spike trains and equal for all neurons. These biases are then mismatched by the synapses and behave like additional random offsets to the neuron. Unlike [15], we completely depend on the internal variability of the physical system, adding neither temporal nor quantile variability in the input or in the biases.

The mismatch in our system can most easily be seen by plotting the firing rate of our hidden layer neurons as a function of the input firing rates. Fig. 2a shows the variability in the 1-dimensional activities, and Fig. 2c shows the variability in the 2-dimensional activities.

The input to the network is a regular spike train for each input dimension in addition to the two constant bias frequencies. In the $1D$ case the input is mapped to a frequency between 200Hz and 1400Hz. In the $2D$ case it is mapped to an input frequency between 0Hz and 1000Hz. We stimulate the network for $5s$ for each input but only use $3s$ of data for our calculations starting from $t = 1s$. The time can be increased to gain robustness against environmental noise. For the $1D$ case we took 50 measurements in linear steps from 0 Hz to 1400 Hz, and for the $2D$ case 25 linear steps from 0Hz to 1000 Hz for each dimension, resulting in 625 measurements.

IV. Offline Learning for Nonlinear Regression

Once the activations of H hidden layer neurons have been recorded for a set of N example inputs, we can derive the output weights required to approximate the function $y = f(\mathbf{x})$. This is accomplished with a regularized least-squares solution:

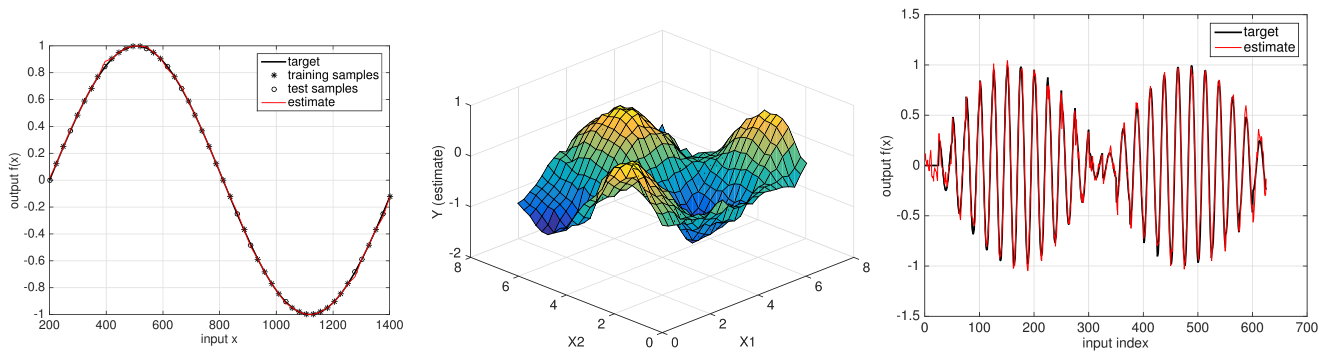
$$\mathbf{y} = f(\mathbf{x}) = \mathbf{W}^T\mathbf{h}(\mathbf{x}) \tag{4}$$

$$\mathbf{W} = \left(\mathbf{H^T H} + \lambda\mathbb{1}\right)^{\mathbf{-1}}\mathbf{H^T Y}, \tag{5}$$

where $\mathbf{H}$ is the $N \times H$ matrix of hidden-layer activations, λ is a regularization parameter, and $\mathbf{Y}$ is the $N \times 1$ vector of desired outputs. When the $\mathbf{W}$ have been calculated, we test the network's capability by applying new inputs to the system.

The results of learning the 1-dimensional function $y = \sin(2\pi x)$ are shown in Fig. 3a. The learning was evaluated over 10 trials, in which the data was randomly split into 38 training samples and 12 test samples. The average mean-squared training error over the 10 trials was 2.13E-5, while the average mean-squared test error was 0.017.

The results of learning the 2-dimensional function $y = \sin(2\pi x_1) * cos(2\pi x_2)$ are shown in Fig. 3b and 3c. The data was split randomly into 469 training samples and into 156 test samples. The average mean-squared error for training was 0.051, and the average error for testing was 0.0642.



(a) Visualization of 1D Results (b) Learned Reconstruction of 2D Function (c) Visualization of 2D Results

Fig. 3. (a) Regression results from a single trail for the 1-dimensional function $y = sin(2\pi x)$. The regression used activations measured when the excitatory weight was set to $V_w = 650mV$ [16] and the neuron's refractory period was set to $V_{ref} = 200mV$ [16]. We used a regularization parameter $\lambda = 1$. (b) Regression results from a single trail for the 2-dimensional function $y = sin(2\pi x_1) * cos(2\pi x_2)$. Activations were measured for a weight of $V_w = 650mV$ [16] and refractory period of $V_{ref} = 200mV$ [16]. The regularization parameter was $\lambda = 1$ (c) is a 1D visualization of the target and estimates (training and test) for the 2D function from Fig. 3b.

V. Conclusion

We have presented the implementation of the random weights and hidden layer neurons of random hidden layer networks (RHL) or an Extreme Learning Machine (ELM) using a neuromorphic platform. Offline learning based on the hidden layer activations allowed us to train the system to compute functions with 1- and 2-dimensional inputs. Extending to more input dimensions is possible if we use a multi-chip setup. The limitation of 2 dimensions on this chip only exists because the chip was not designed for this application, which can be easily solved by building a chip with the appropriate number of inhibitory synapses, or by using multiple existing chips.

Implementing the weights between the hidden layer and the output layer is the logical next step of this work. Given that the synapses are binary, this would require using multiple synapses to represent a single weighted connection, with each synapse encoding an incremental weight change. On-chip learning is another goal. The possibility to use learning synapses which follow a spike- and activity-based Hebbian learning rule is a promising avenue of inquiry.

VI. Acknowledgment

We want to thank Harshawardhan Ramachandran for the fruitful discussions and ideas especially on the design of the tuning curves. This work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG). R.F. Reinhart is funded by the German BMBF within the Intelligent Technical Systems Leading-Edge Cluster Competition.

References

[1] D. Broomhead and D. Lowe, "Multi-variable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[2] W. F. Schmidt, M. Kraaijveld, R. P. Duin *et al.*, "Feedforward neural networks with random weights," in *Pattern Recognition, 1992. Vol. II. Conference B: Pattern Recognition Methodology and Systems, Proceedings., 11th IAPR International Conference on*. IEEE, 1992, pp. 1–4.

[3] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[4] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*, vol. 2. IEEE, 2004, pp. 985–990.

[5] ——, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[6] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proceedings of the IEEE*, vol. 102, no. 9, pp. 1367–1388, Sept 2014.

[7] G. Indiveri, B. Linares-Barranco, T. Hamilton, A. van Schaik, R. Etienne-Cummings, T. Delbruck, S.-C. Liu, P. Dudek, P. Häfliger, S. Renaud, J. Schemmel, G. Cauwenberghs, J. Arthur, K. Hynna, F. Folowosele, S. Saighi, T. Serrano-Gotarredona, J. Wijekoon, Y. Wang, and K. Boahen, "Neuromorphic silicon neuron circuits," *Frontiers in Neuroscience*, vol. 5, pp. 1–23, 2011.

[8] C. Bartolozzi and G. Indiveri, "Synaptic dynamics in analog VLSI," *Neural Computation*, vol. 19, no. 10, pp. 2581–2603, Oct 2007.

[9] G. Indiveri, E. Chicca, and R. J. Douglas, "A VLSI array of low-power spiking neurons and bistable synapses with spike–timing dependent plasticity," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 211–221, Jan 2006.

[10] M. Pelgrom, A. Duinmaijer, and A. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 5, pp. 1433–1440, October 1989.

[11] K. Boahen, "Point-to-point connectivity between neuromorphic chips using address-events," *IEEE Transactions on Circuits and Systems II*, vol. 47, no. 5, pp. 416–34, 2000.

[12] A. Basu, S. Shuo, H. Zhou, M. H. Lim, and G.-B. Huang, "Silicon spiking neurons for hardware implementation of extreme learning machines," *Neurocomputing*, vol. 102, pp. 125–134, 2013.

[13] E. Yao, S. Hussain, A. Basu, and G.-B. Huang, "Computation using mismatch: Neuromorphic extreme learning machines," in *Biomedical Circuits and Systems Conference (BioCAS), 2013 IEEE*. IEEE, 2013, pp. 294–297.

[14] C. S. Thakur, T. J. Hamilton, R. Wang, J. Tapson, and A. van Schaik, "A neuromorphic hardware framework based on population coding," *arXiv preprint arXiv:1503.00505*, 2015.

[15] F. Corradi, C. Eliasmith, and G. Indiveri, "Mapping arbitrary mathematical functions and dynamical systems to neuromorphic vlsi circuits for spike-based neural computation," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*. IEEE, 2014, pp. 269–272.

[16] G. Indiveri and E. Chicca, "A VLSI neuromorphic device for implementing spike-based neural networks," in *Neural Nets WIRN11 - Proceedings of the 21st Italian Workshop on Neural Nets*, Jun 2011, pp. 305–316.

[17] P. R. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *Solid-State Circuits, IEEE Journal of*, vol. 40, no. 6, pp. 1212–1224, 2005.