On the Sensitivity of Analog Artificial Neural Network Models to Process Variation

N. Afroz*, A. Sayem*, G. Volanis*, D. Maliuk†, H. Stratigopoulos‡ and Y. Makris*
*Department of Electrical and Computer Engineering, The University of Texas at Dallas, Richardson, TX 75080, USA
† EE Department, Yale University, New Haven, CT 06511, USA
‡ Sorbonne Université, CNRS, LIP6, Paris, France

2024 IEEE 42nd VLSI Test Symposium (VTS) | 979-8-3503-6378-4/24/$31.00 ©2024 IEEE | DOI: 10.1109/VTS60656.2024.10538718

***Abstract*—We investigate the impact of semiconductor manufacturing process variation on the accuracy of machine learning models implemented as analog Artificial Neural Networks (ANNs). Unlike their digital counterparts, where binary operations and weight representation ensure the robustness of a trained model across software and hardware, the continuous nature of weights and operations in analog ANNs makes the accuracy of a trained model inevitably sensitive to the exact parameters of each fabricated chip. As a result, expensive chip-in-the-loop training is necessitated to ensure high accuracy. Herein, we elucidate the nature and extent of the problem using actual measurements from multiple identically fabricated copies of an analog ANN chip and a variety of trained models. We quantify the accuracy loss when models are ported across chips, as well as the effort required for individually training each chip, and we discuss strategies for containing this effort.**

I. Introduction

Artificial intelligence, in general, and Artificial Neural Networks (ANNs), in particular, are used in myriads of contemporary applications [1]–[3], ranging from consumer electronics to autonomous systems and from medical instrumentation to critical infrastructure. While software and/or digital hardware implementations of ANNs currently enjoy the lion's share of the market, a number of emerging realities are necessitating the development and deployment of *analog* ANNs. Specifically, the exponential growth of sensory data from world-machine interfaces, known as the analog data deluge, along with the power, area and response-time constraints of distributed edge computing systems, call for the ability to autonomously sense, perceive, reason and rapidly act. This ability, which avoids overwhelming communication, storage and computational infrastructure of contemporary systems, is promised by silicon implementations of analog ANNs [4]–[7].

Such implementations, however, present new challenges that are unique to the analog domain and have yet to be thoroughly studied. Among them, we focus on the *impact of semiconductor process variation on the accuracy* of analog ANN models [8]–[16]. Indeed, due to the continuous domain in which computation is performed by analog ANNs, their operation is particularly sensitive to any differences in the circuit parameters, which are bound to occur due to process variation. As a result, unlike in digital neural networks, it is not possible to train a software model and subsequently upload the selected synaptic weights onto an analog neural network. Instead, a "chip-in-the-loop" approach is required, wherein the loss function of every candidate set of synaptic weights visited during training has to be evaluated on the hardware itself. To make matters worse, even if a model is learned on one chip, it will be different when uploaded on another (identical) chip.

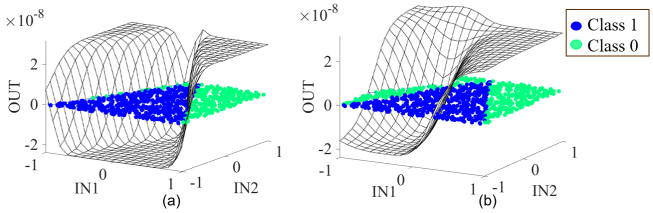

Figure 1: (a) Model Learned via Chip-in-the-Loop Training, (b) Model Ported to Same HW Resources on Identical Chip.

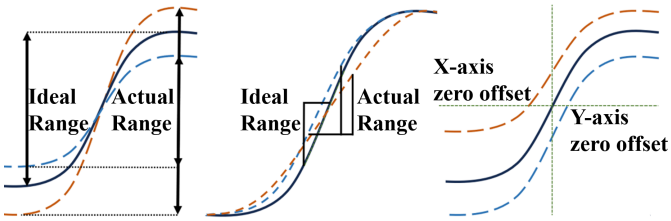

Figure 2: Impact of Process Variation on Analog Neuron's (a) Peak-to-Peak Range, (b) Slope, (c) X-Y Offset.

To illustrate this problem, consider Fig. 1(a), which shows the classification boundary learned by a fabricated analog ANN chip (see Section II for details) that was trained with samples from a dataset containing two classes, blue and green, respectively. Fig. 1(b) shows the classification boundary implemented when the weights of the model learned on the first chip are ported to an identical chip, fabricated on the same wafer. Evidently, the two boundaries are considerably different, with the first one achieving over 95% classification accuracy and the second achieving only 50%. This substantial discrepancy in accuracy of the model across two identical chips is caused by differences incurred by semiconductor process variation in the parameters of the analog circuits, which in turn introduce differences in their functionality. Consider, for example, one of the key components of neural networks, the neuron circuit, which implements a threshold activation function, such as a sigmoid, as shown in Fig. 2. In an ideal manufacturing process, where all circuit parameters have their nominal values, the neuron is designed to have a specific range and slope, and to have its mid-point centered in the origin of the X-Y plane (*i.e.*, solid sigmoids). In reality, however, circuit parameters of actual chips will deviate from their nominal values, resulting

979-8-3503-6378-4/24/$31.00 ©2024 IEEE

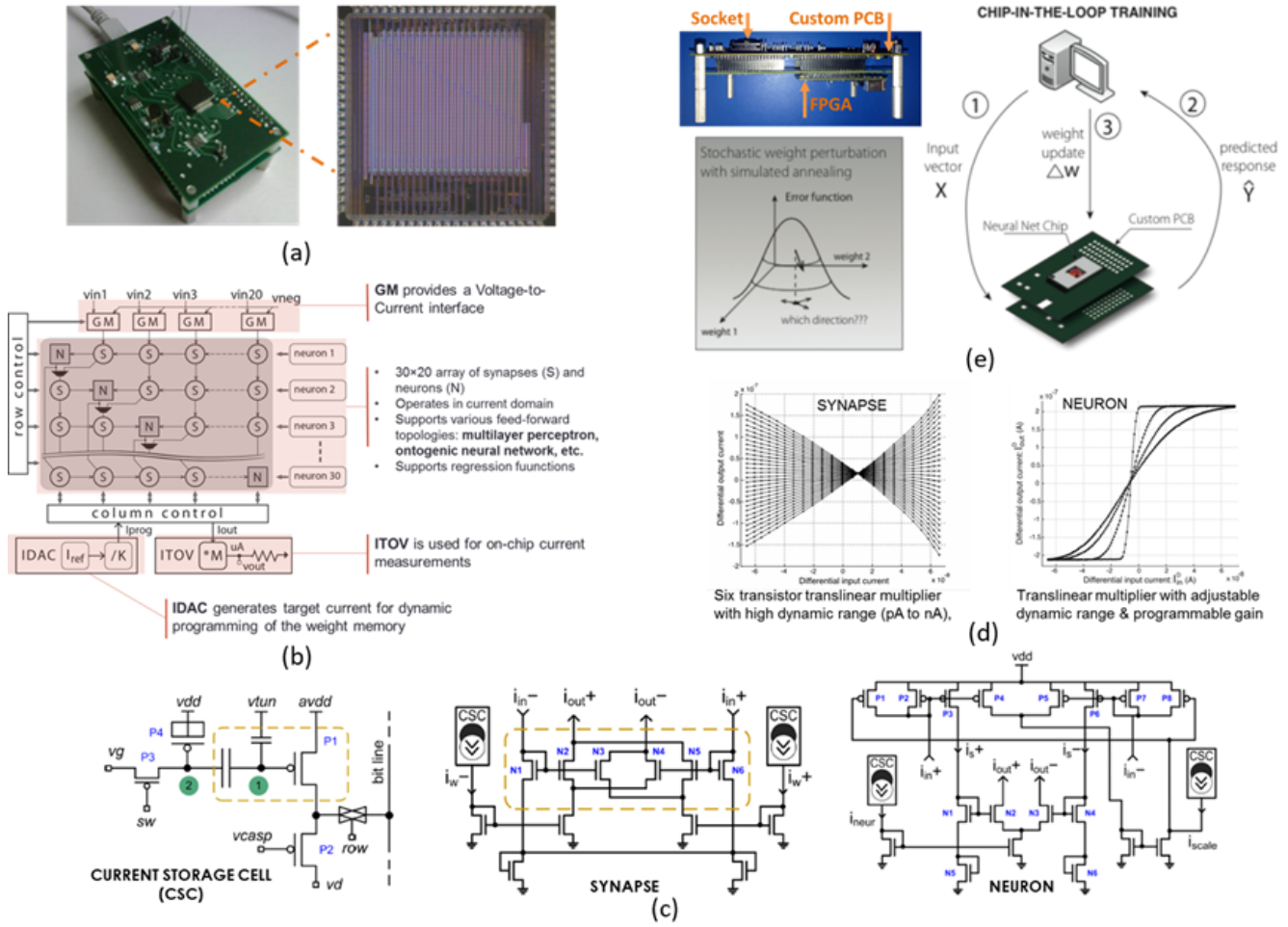

Figure 3: (a) Experimentation Platform, (b) Chip Architecture, (c) Circuit Details, (d) Synapse and Neuron Transfer Functions, (e) Chip-in-the-Loop Training.

in shifts in range and slope, as well as non-zero offsets in the X and Y dimensions of the sigmoid function (*i.e.*, dashed sigmoids). Accumulated across multiple neurons and synapses, these differences ultimately result in considerable discrepancies when a model is ported across chips.

Motivated by these observations, in this paper we seek to elucidate and quantify the impact of parametric differences in the silicon implementation of analog ANNs, which are caused by process variation, on the accuracy of the models that these networks implement. Furthermore, we seek to develop solutions, involving calibration of both the training strategy and the analog ANN circuitry, to ensure fast and accurate adaptation of trained models ported across chips.The remainder of the paper is structured as follows: In Section II, we describe the experimental platform used in this work. In Section III, we propose an array of solutions for ameliorating the impact of process variations on the accuracy of models implemented on hardware. In Section IV, we experimentally evaluate the effectiveness of the proposed solutions. Lastly, in Section V, we draw conclusions and we discuss future directions.

II. Experimentation Platform

For the purposes of this study, we use an analog ANN experimentation platform which is shown in Fig. 3(a). The core of this platform is a 3mm x 3mm die fabricated in TSMC's 0.35um technology, which can be interfaced through a custom-designed PCB. Fig. 3(b) shows the architecture of the programmable analog ANN [17], which consists of a reconfigurable 30x20 array of synapses (S) and neurons (N) operating in the subthreshold region and featuring sub-μW power consumption. The main circuits used in the core are shown in Fig. 3(c). Specifically, a novel current storage cell (CSC), shown on the left and comprising an analog floating gate transistor (FG), a capacitor, and supporting control transistors and terminals, is used for two purposes. First, it offers a compact non-volatile analog weight storage solution after training is completed, providing up to 10 bits of precision for each stored weight. Second, it provides a reconfiguration mechanism for supporting various neural network topologies and for adjusting various circuit parameters. The synapse circuit, shown in the middle, implements a four-quadrant multiplication function using two CSC cells for differential weight component storage and a six-transistor core. Its transfer function is shown on the left side of Fig. 3(d). The neuron circuit, shown on the right side of Fig. 3(c), performs a *tanh*-like nonlinear activation function on the outputs of the synapses connected to it, through normalization, gain con-

trol, and non-linear transformation, which is also performed through a translinear operation using a six-transistor core. Two CSC cells are used to control the slope of the threshold function, as shown on the right side of Fig. 3(d). Apart from the core, three peripheral circuits provide support for the training and operation of the neural network. The differential transconductors (GM) accept differential voltage signals as inputs and convert them into differential currents, as required by the core. A current-to-voltage converter (ITOV) facilitates the reading of internal currents such as weights and neural network responses. A current Digital-to-Analog Converter (IDAC) generates target currents for dynamic programming of the weight memory.

This platform supports various feed-forward classifier architectures, such as Multi-Layer Perceptions (MLPs) [18] and Ontogenic Neural Networks (ONNs) [19], as well as various non-linear regression models. As shown in Fig. 3(e), the chip is housed in a socket on a custom PCB and is interfaced through an FPGA board to a PC, where it can be controlled via Matlab. This configuration supports chip-in-the-loop training with various algorithms (e.g., resilient back-propagation (RPROP) [20]) implemented in software, whereby training samples are presented to the chip and the loss function is computed and used to drive weight perturbation in the next training iteration.

We note that this platform includes certain **Resource Calibration (RC)** capabilities which can be used for the purpose of counteracting process variation. Specifically, the two CSC cells inside the neuron block, which are programmable as they contain analog floating gate transistors, control the *range* and *slope* of the neuron activation function, respectively. Other non-idealities, however, such as the offset of the neuron activation function along its X and Y axis caused by process variation, cannot be calibrated in this platform. Nevertheless, the programmable nature of the platform allows for judicious selection of neurons with matching offsets across different chips to address this limitation. Similarly, no calibration capability is offered at the synapse circuit, relying instead on training to select weights that can counteract process variation.

III. Proposed Analog ANN Training Solutions

A straightforward approach to train an analog ANN, which we will refer to as **Loading Software Model**, would be to conduct training in software using the training dataset, and then port the learned weights to the physical hardware. However, this approach is hindered by the difference between ideal software model and actual chip parameters, caused by the variation of the fabrication process. An alternative solution to ameliorate the software/hardware differences, which we will refer to as **Chip-to-Chip Weight Transfer**, could involve chip-in-the-loop training using an actual chip and then porting the learned weights to other chips. While training directly in hardware takes into account the circuitry non-idealities that software model training is oblivious to, it still does not account for the chip-to-chip variations. Hence, an improvement upon this solution, which we will refer to as

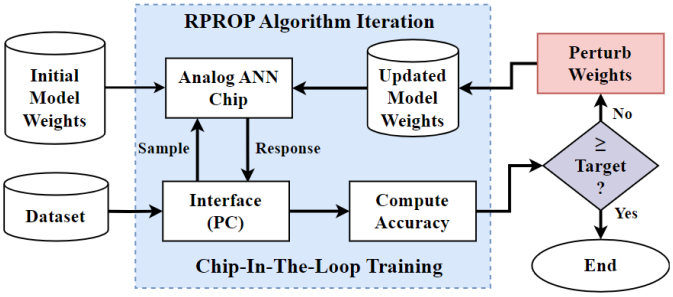

Figure 4: Baseline (BL) Training Method.

Resource Calibration, could involve a pre-processing phase wherein the neurons and synapses of the chip to which the learned weights will be ported are calibrated to match the transfer functions of the corresponding resources on the chip where training was conducted. However, while this is certainly a step in the right direction, perfectly matching the source and target chip resources is unlikely due to the continuous nature of analog signals and the limitations of the calibration circuitry. Therefore, additional chip-in-the-loop training is likely to be required for the target chip to reach a desired accuracy level. Three variants of this approach are introduced herein.

A. Baseline (BL) Training

Our starting point, which we will refer to as **Baseline (BL)** training, is based on the foundational chip-in-the-loop training solution. As, shown in Fig. 4, training starts with a randomly chosen initial set of weights for the analog ANN model. An iteration of the RPROP algorithm is then conducted, wherein the dataset is presented through a PC to the chip implementing the analog ANN with the current set of weights, one sample at a time, and responses are collected and evaluated to assess the accuracy of the current weights. If the target accuracy is not reached, the weights are perturbed through RPROP, reloaded onto the chip, and the process is repeated. Otherwise, the process terminates and the current weights in the chip represent the final model.

B. Model Calibration (MC)

While the BL method customizes the learned weights and the overall model to the exact circuit parameters of the target chip, it is time-consuming and has to be repeated from scratch for every chip. Therefore, to reduce the training time overhead, our next solution, which we will refer to as **Model Calibration (MC)** and which is shown in Fig. 5 inside the MC block, involves two strategies:

Incremental Training: Instead of using a set of randomly chosen initial weights, we initialize the target chip with the weights learned through chip-in-the-loop training on another chip. The rationale behind this strategy is that the source and target chips are nominally identical and any differences are only caused by process variation so their magnitude is likely to be small. Hence, porting the trained weights of the source chip to the target chip may give the RPROP algorithm a better starting point, bringing it closer to the region of weights where a solution for the target chip can be found and, thereby, accelerating its convergence. We point out, however, that this

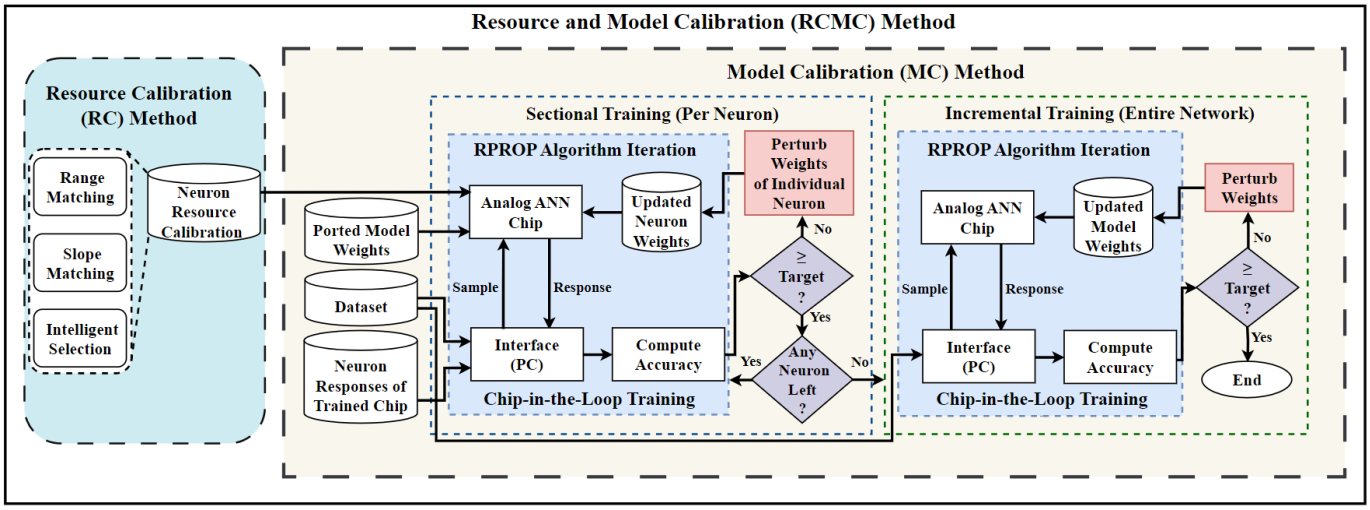

Figure 5: Model Calibration (MC) and Resource and Model Calibration (RCMC) Methods.

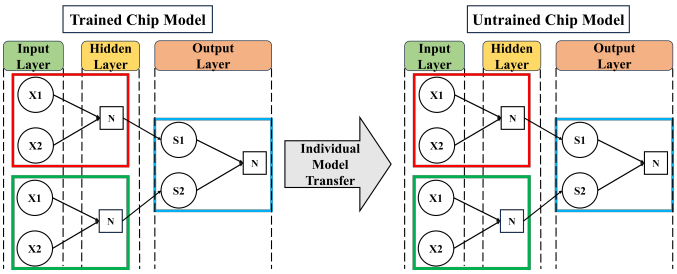

Figure 6: Sectional Training Method.

conjecture may not always hold true, both due to the stochastic nature of the RPROP training algorithm and due to the non-linearity of analog circuits.

Sectional Training: Instead of training the entire model all at once, this strategy focuses on one section (*i.e.*, neuron) of the analog ANN architecture at a time, starting from the input layer and progressing layer-by-layer toward the output layer, as shown in Fig. 6. By training one neuron of the target chip to match the model learned by the corresponding neuron of the source chip at a time, we seek to contain the impact of the stochasticity of the RPROP training algorithm, which is more pronounced during simultaneous updates of all neurons. While multiple training sessions are now required (*i.e.*, one per neuron), each such session is significantly faster than training the entire network, hence the overall training time is expected to be lower. We acknowledge, however, that this strategy requires the ability to observe and log the output of each section of the source chip for each sample in the training dataset, as this enhanced dataset is needed in order to train the corresponding section of the target chip.

As shown in Fig. 5, the MC method combines the sectional and incremental training strategies. First, using weights learned from a trained source chip as the initial model for the target chip, as well as responses for the dataset samples for each section of the source chip, we train sectionally each neuron of the target chip until it reaches the target accuracy. Then, using as a starting point the adjusted weights produced by sectional training, we conduct incremental training of the entire network. Having matched the models that each section implements, we expect that the time required for incrementally adjusting the overall model of the target chip will be small.

C. Resource Calibration and Model Calibration (RCMC)

Our final solution, which we will refer to as **Resource Calibration and Model Calibration (RCMC)**, combines the MC solution with the previously mentioned RC approach, as depicted in Fig. 5. We remind that RC is a pre-processing phase wherein the hardware resources that are used to construct the analog ANN in the target chip are calibrated to match the characteristics of the corresponding resources on the source chip. Specifically, in our case, such resource calibration commences with an attempt to match the range and the slope of the threshold activation function (*i.e.*, sigmoid) of each neuron in the target chip to those of the corresponding neuron in the source chip. However, due to the uncontrolled nature of process variation, sufficient matching may not always be available. Hence, leveraging the programmable nature of analog ANNs, which allow selection of resources to construct a network architecture, we also employ an intelligent resource selection process. The objective of this process is to cherry-pick resources (*i.e.*, neurons) in the target chip which are the most similar and, when calibrated, have the highest chance of matching the corresponding resources in the source chip.

IV. Experimental Evaluation

In this section, we use the Analog ANN experimentation platform described in Section II to further elucidate the impact of process variation on the accuracy of trained models when ported across different chips and to evaluate the effectiveness of the proposed solutions in ameliorating this problem.

A. Experimental Setup

To evaluate the solutions (*i.e.*, BL, MC, RCMC) proposed in Section III, we use three 2D synthetic datasets, as shown in Fig. 7. Each dataset contains 1000 elements split equally into two classes, color-coded as yellow (class 0) and blue (class

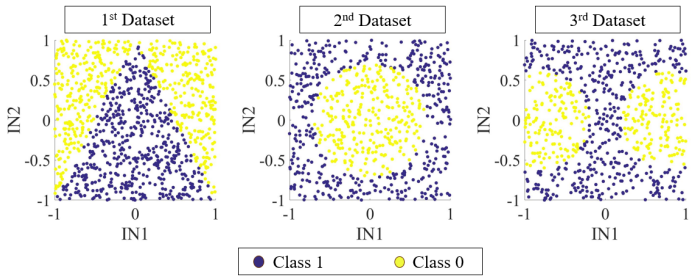

Figure 7: Three Synthetic Datasets.

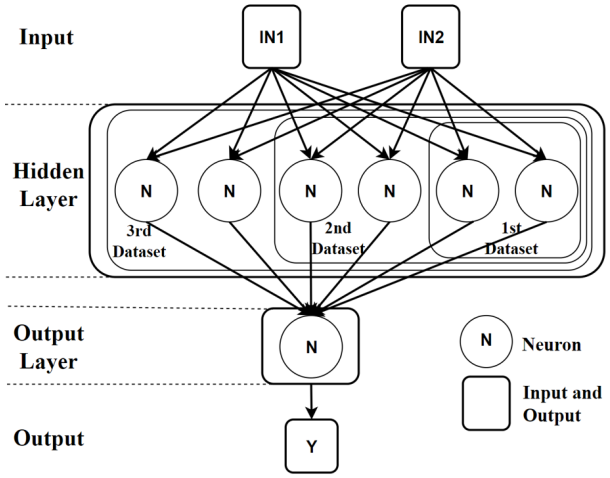

Figure 8: MLP Architecture With One Hidden Layer.

1), normalized in the range [-1, 1]. We then train a simple binary classifier architecture, consisting of an MLP network with one hidden layer and one output neuron, to separate the two populations. The hidden layer contains 2, 4, and 6 hidden neurons for the first, second, and third dataset, respectively. Fig. 8 illustrates the MLP architecture, where the two inputs correspond to the IN1 and IN2 coordinates of the dataset. For this experiment, we randomly selected 3 among the 40 die that were fabricated as part of a multi-project wafer (MPW) run, which we refer to as Chip 1, Chip 2, and Chip 3, respectively.

B. Comparison Metric

To evaluate the proposed methods, we set a specific target of 95% for the minimum accuracy that the trained models should achieve on each of the three chips and we focus on the time that it takes to reach that accuracy. Training time is expressed in *Number of Iterations* required for the training algorithm to reach the target level of accuracy on the training set. We note that, in each iteration, the entire training set is presented to the hardware implementation of the neural network with the latest set of synapse weights, so that the training algorithm can compute the error and adjust the weights accordingly. As a result, each iteration takes the same time for a given dataset.

C. Results and Comparison

Loading Software Model: Table I shows the accuracy for each of the three datasets when training is conducted in software and the weights of the trained model, whose accuracy exceeds 95% in software, are ported to the three chips in our

Table I: Accuracy when porting weights from software.

	Chip 1	Chip 2	Chip 3
1st Dataset	50%	49%	49%
2nd Dataset	59%	60%	61%
3rd Dataset	45%	46%	31%

Source / Destination	Chip 1			Chip 2			Chip 3		
	Dataset			Dataset			Dataset		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
Chip 1	N/A			69%	50%	65%	63%	47%	48%
Chip 2	50%	69%	67%	N/A			71%	59%	49%
Chip 3	55%	54%	65%	70%	61%	59%	N/A		

Figure 9: Accuracy when Porting Weights from Other Chips.

experiment. Evidently, due to the differences between ideal computation in software and actual computation in the analog ANN chips, the accuracy drops dramatically, leading to an average accuracy of approximately 50%, which for a 2-class classification is essentially a coin toss.

Chip-to-Chip Weight Transfer: The table in Fig. 9 summarizes the accuracy when a model that is trained in one chip is ported to another chip. Results are provided for all three datasets and for each of the three chips serving as the chip on which the model was trained and as the chip that the model was ported to. The results quantify and elucidate the extent of the sensitivity of analog ANN models to process variation: porting weights across identical chips using identical resources and on identical datasets results in an accuracy drop between 30% and 50%, rendering the ported model unusable.

Resource Calibration: The table in Fig. 10 reports the accuracy when the hardware resources of the target chip are calibrated to those of the source chip, prior to porting a model. Such calibration involves alignment of the range, slope and offset of the neurons used to construct the network. Evidently, compared to the accuracy of porting a model to the non-calibrated chip, the results are significantly improved, averaging an accuracy of approximately 75%. By itself, however, resource calibration is insufficient, as it fails to reach the target accuracy of 95%, mainly due to the inherent limitations of the calibration hardware which cannot achieve perfect matching of all resources across chips.

BL: The results for individually training each chip from scratch for each of the three datasets are summarized in the dotted white boxes of the main diagonal of the table shown in Fig. 11. Notably, there exists a significant difference in the number of training algorithm iterations required for each chip to reach the target 95% accuracy level for the same dataset. For example, the first dataset requires 970, 1873 and 630 iterations for the first, second and third chip, respectively. This difference stems in part from the stochastic nature of the RPROP algorithm [20] and in part from the variation in the actual hardware parameters which are caused by semiconductor manufacturing.

MC: The results for calibrating a model ported from one chip to another through sectional and incremental training are summarized in the first sub-row of each main row of the table

Source / Destination	Chip 1 Dataset 1st	Chip 1 Dataset 2nd	Chip 1 Dataset 3rd	Chip 2 Dataset 1st	Chip 2 Dataset 2nd	Chip 2 Dataset 3rd	Chip 3 Dataset 1st	Chip 3 Dataset 2nd	Chip 3 Dataset 3rd
Chip 1	N/A			75%	69%	76%	80%	73%	71%
Chip 2	93%	72%	78%	N/A			87%	71%	75%
Chip 3	72%	70%	66%	79%	67%	64%	N/A		

Figure 10: Accuracy Improvement via Resource Calibration.

Source / Destination	Method	Chip 1 Dataset 1st	Chip 1 Dataset 2nd	Chip 1 Dataset 3rd	Chip 2 Dataset 1st	Chip 2 Dataset 2nd	Chip 2 Dataset 3rd	Chip 3 Dataset 1st	Chip 3 Dataset 2nd	Chip 3 Dataset 3rd
Chip 1	MC	BL			220	925	338	580	282	294
	RCMC	970	1800	1900	90	455	208	38	201	119
Chip 2	MC	779	443	346	BL			400	1004	276
	RCMC	45	192	94	1873	1882	1840	36	318	84
Chip 3	MC	598	736	308	503	146	307	BL		
	RCMC	68	90	116	39	78	57	630	1236	560

Figure 11: Number of Iterations to Achieve 95% Accuracy.

shown in Fig. 11. For each target chip and for each dataset, results are provided for the model originating from each of the other two chips. As may be observed, MC reduces the number of training algorithm iterations, as compared to BL. For example, training Chip 2 from scratch for the first, second and third dataset requires 1873, 1882 and 1840 iterations, respectively, while calibrating the model of Chip 3 for use in Chip 2 reduces the number of iterations to 400, 1004 and 276, respectively. Reduction is observed in all cases, although its magnitude varies across chips and datasets.

RCMC: While the MC calibration solution provides significant training time reduction over BL, our results indicate that the savings are higher when combined with RC. Indeed, as shown in the second sub-row of each main row of the table shown in Fig. 11, first calibrating resources through intelligent neuron selection and sigmoid function matching, and then calibrating a ported model through sectional and incremental training, results in an order of magnitude better training time reduction than the MC method on its own. Overall, when using the proposed combined RCMC solution in our experiments, we achieved a training time reduction of up to 98% (*i.e.*, from 1873 to 36 iterations when porting to Chip 2 a model that was trained on Chip 3 to separate the first dataset).

To assist in contrasting the training time improvement achieved by the proposed solutions, the data from the table in Fig. 11 is also visualized in Fig. 12. The histograms corroborate our earlier observations, namely the significant training time reduction that the MC solution achieves over BL, as well as the drastically improved training time reduction that the combined RCMC solution accomplishes, making it the indisputable winner and method of choice.

V. Conclusion

Analog ANN implementations promise significant advantages in applications such as world-machine interfaces, where rapid, low-power sensing, computation and actuation is required. Such implementations, however, are susceptible to

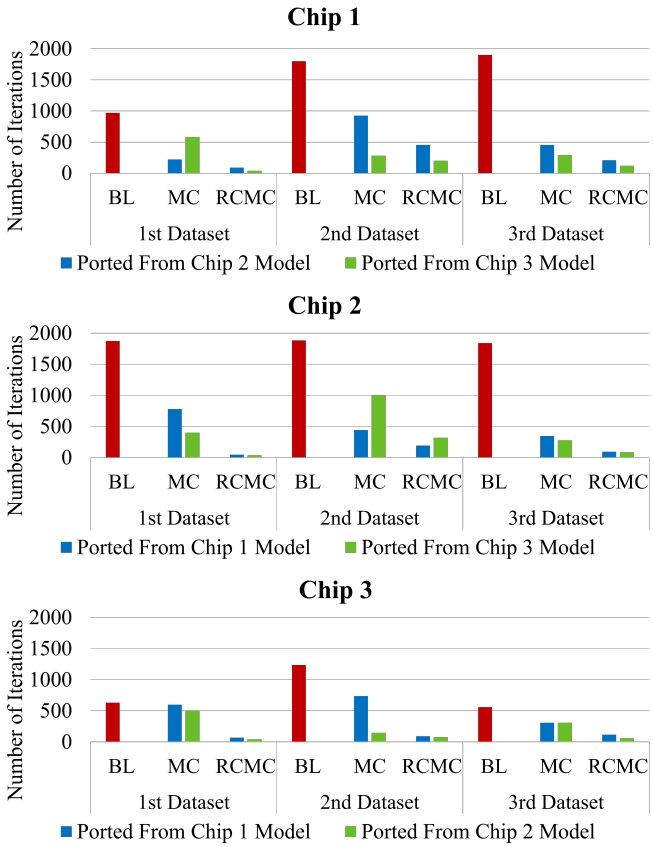

Figure 12: Comparison of Training Time for BL, MC and RCMC Solutions.

parametric differences introduced by manufacturing process variation, thereby introducing a new set of challenges for training an accurate model. Indeed, porting weights learned through training an equivalent software model to an analog ANN chip, or even porting weights learned through chip-in-the-loop training from one chip to another identically fabricated chip, results in significant accuracy loss. Moreover, while chip-in-the-loop training can address the problem, the training time required is a significant limitation. To alleviate this overhead, we developed an array of solutions that take into account the impact of process variation on both the source and the target chip and employ hardware resource calibration along with sectional and incremental training, while porting a model. Thereby, as we demonstrated in silicon using three copies of an analog ANN experimentation platform, the training time required is significantly reduced. Future efforts will focus on extending this study across a larger number of fabricated ICs, using more elaborate analog ANN topologies and more complex datasets, as well as on developing testing and error mitigation techniques for analog ANN chips, an emerging area that yet to be thoroughly investigated [21].

Acknowledgement

This research was partially supported by the NSF-ANR CHAMELEON project under Grants N° NSF-2214934 and ANR-22-CE94-0002-01.

REFERENCES

[1] S.-C. Liu, T. Delbruck, G. Indiveri, A. Whatley, and R. Douglas, *Event-based neuromorphic systems*. John Wiley & Sons, 2014.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[3] C. C. Aggarwal, *Neural networks and deep learning: A textbook*. Springer International Publishing, 2018.

[4] M. Valle, "Analog VLSI implementation of artificial neural networks with supervised on-chip learning," *Analog Integrated Circuits and Signal Processing*, vol. 33, pp. 263–287, 2002.

[5] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 µm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 270–281, 2015.

[6] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8 μ J/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, 2019.

[7] A. Rubino, C. Livanelioglu, N. Qiao, M. Payvand, and G. Indiveri, "Ultra-low-power FDSOI neural circuits for extreme-edge neuromorphic intelligence," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 45–56, 2021.

[8] A. Montalvo, R. Gyurcsik, and J. Paulos, "An analog VLSI neural network with on-chip perturbation learning," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 4, pp. 535–543, 1997.

[9] ——, "Toward a general-purpose analog VLSI neural network with on-chip learning," *IEEE Transactions on Neural Networks*, vol. 8, no. 2, pp. 413–423, 1997.

[10] P. Edwards and A. Murray, "Fault tolerance via weight noise in analog VLSI implementations of MLPs-a case study with EPSILON," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 9, pp. 1255–1262, 1998.

[11] A. Orgenci, G. Dundar, and S. Balkur, "Fault-tolerant training of neural networks in the presence of MOS transistor mismatches," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, no. 3, pp. 272–281, 2001.

[12] V. Koosh and R. Goodman, "Analog VLSI neural network with digital perturbative learning," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, no. 5, pp. 359–368, 2002.

[13] K. Jia, Z. Liu, Q. Wei, F. Qiao, X. Liu, Y. Yang, H. Fan, and H. Yang, "Calibrating process variation at system level with in-situ low-precision transfer learning for analog neural network processors," in *Proc. ACM/ESDA/IEEE Design Automation Conference (DAC)*, 2018.

[14] S. Moon, K. Shin, and D. Jeon, "Enhancing reliability of analog neural network processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 6, pp. 1455–1459, 2019.

[15] D. Janke and D. V. Anderson, "Analyzing the effects of noise and variation on the accuracy of analog neural networks," in *Proc. IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2020, pp. 150–153.

[16] S. A. El-Sayed, T. Spyrou, E. Afacan, L. A. Camuñas-Mesa, B. Linares-Barranco, and H.-G. Stratigopoulos, "Spiking neuron hardware-level fault modeling," in *Proc. IEEE IEEE International Symposium on On-Line Testing and Robust Systems (IOLTS)*, 2020.

[17] D. Maliuk and Y. Makris, "An experimentation platform for on-chip integration of analog neural networks: A pathway to trusted and robust analog/RF ICs," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 8, pp. 1721–1734, 2015.

[18] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[19] E. Fiesler, "Comparative bibliography of ontogenic neural networks," in *Proc. IEEE International Conference on Artificial Neural Networks*, 1994, pp. 793–796.

[20] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *Proc. IEEE International Conference on Neural Networks*, vol. 1, 1993, pp. 586–591.

[21] F. Su, C. Liu, and H.-G. Stratigopoulos, "Testability and dependability of AI hardware: Survey, trends, challenges, and perspectives," *IEEE Design & Test*, vol. 40, no. 2, pp. 8–58, 2023.