In-situ Tuning of Printed Neural Networks for Variation Tolerance

Michael Hefenbrock, Dennis D. Weller, Jasmin Aghassi-Hagmann, Michael Beigl and Mehdi B. Tahoori
Karlsruhe Institute of Technology, Karlsruhe, Germany
{michael.hefenbrock, dennis.weller, jasmin.aghassi, michael.beigl, mehdi.tahoori}@kit.edu

***Abstract*—Printed electronic (PE) can meet the requirements of many application domains with requirements on cost, conformity, and non-toxicity which silicon-based computing systems cannot achieve. A typical computational task to be performed in many of such applications is classification. Therefore, printed Neural Networks (pNNs) have been proposed to meet these requirements. However, PE suffers from high process variations due to low resolution printing in low-cost additive manufacturing. This can severely impact the inference accuracy of pNNs. In this work, we show how a unique feature of PE, namely additive printing can be leveraged to perform in-situ tuning of pNNs to compensate accuracy losses induced by device variations. The experiments show that, even under 30 % variation of the conductances, up to 90 % of the initial accuracy can be recovered.**

I. Introduction

Printed electronics (PE) is an emerging technology to fabricate ultra-low-cost, on-demand, and flexible electronic systems. These characteristics are especially suitable for application domains with requirements that cannot be met by conventional silicon-based electronics, such as smart packaging for fast moving consumer goods (FMCG) [1]. Among various fabrication technologies in PE, inkjet-printing with, e.g. electrolyte-gated transistors (EGTs) [2], has shown to be a promising solution to facilitate point-of-use and on-demand fabrication, due to its low-cost mask-less additive manufacturing process [1].

Nevertheless, PE suffers from high intrinsic variations due to the difficulty in proper alignment of printed patterns arising from low resolution of low-cost additive manufacturing [1], [3]. Variations in inkjet printing arise from the dispersion of the ink on the substrate, droplet jetting oddness, satellite drops, wetting or missing droplets [4]. Moreover, due to large feature sizes (3 orders of magnitude larger compared to CMOS), the latency is high and the device count is rather low. Therefore, the implementation of conventional digital architectures in PE have exorbitantly high hardware overheads [5].

In this regard, printed Neural Networks (pNNs) are promising, as they allow for low-complex designs and can moreover reduce the high device latencies of PE by the parallel computing capability of NNs. Due to these benefits, NNs in PE have already been proposed as a solution for classification problems [6]–[9]. In these schemes, the synaptic weights at the crossbar interconnects were realized by printed resistors, which allow for customization and implementation of circuitry resembling NNs.

However, variations in the printing process manifest in inaccuracies of the synaptic weights. Therefore, it is imperative to mitigate the resulting inference losses during the training process to regain the neural network accuracy. To increase the NN accuracy, the authors of [10] included a variation penalty term during the training method as well as variation-aware mapping of the NN to the circuit in a post-processing step after testing of all devices. The method presented in [11] relies on measurements of device variations per-chip to adapt the training scheme. In both works, no emphasis is given on the manifestation of process variations from physical devices up to the model-level of the NN. Although these approaches work well for memristor-based crossbar architectures, they cannot directly be applied to inkjet-printed pNN where the synaptic weights are represented by printed resistors, whose resistance can only be decreased (through adding conductive layers).

In this paper, we propose an in-situ (post-fabrication) tuning method for printed pNN to increase their accuracy under device variations. We take advantage of the additive feature of inkjet printing, where additional layers of resistors can be printed after the initial fabrication, which is not possible in traditional silicon-based VLSI. The initially designed circuit is fabricated and the resistances of the weights are measured. Afterwards, the adjustment to obtain the correct weights is calculated using a proposed algorithm. Finally, the correction of the weights (resistances) is performed by additively printing resistors on top of the originally-printed ones. The experiments show that, even under 30 % variation of the conductances, up to 90 % of the initial accuracy can be recovered, by often only treating 20 - 30 % of neurons with the proposed approach.

The proposed method complements the existing approaches for variation-aware training, allowing to regain the remaining lost accuracy in a per-device manner. The method makes no assumption about the type of variation the device was subjected to before treatment. Hence, it should also be able to counteract runtime degradations such as aging-induced conductance degradation.

II. Printed Neural Networks

A. Realization of pNNs

The fundamental mathematical operations required in most of the NN types are the multiply-add computation between an input vector with a weight vector and a one-dimensional non-linear activation function. Both operations are the building blocks to form a synaptic neuron, and by their interconnection to a network, a neural network architecture is realized. In Fig. 1, the proposed design of such a neuron implementation in PE is depicted. In PE, the synaptic weights $w_1, w_2, \cdots$ are implemented as printed resistors with conductances $g_1, g_2, \cdots$. By applying the input voltages $V_1, V_2, \cdots$ to the resistors, the

978-3-9819263-6-1/DATE22/©2022 EDAA

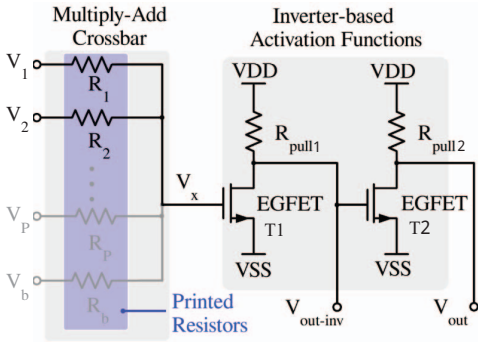

Fig. 1. Schematic of inverter-based neuron design in PE with input signals V_1, V_2 and bias input voltage V_b.

currents sum up according to Kirchoff's rule and the voltage V_x can be computed as (Fig. 1):

$$V_x = \frac{\sum_i V_i \cdot g_i}{\sum_i g_i} = \sum_i V_i \; w_i \implies w_i = \frac{g_i}{\sum_j g_j}, \qquad (1)$$

where w_i are the resulting weights and i is used to index the weights/resistances. Hence, by changing the conductances g_i via adjustment of the geometries (width, length or thickness) of the printed resistor, desired weights w_i may be achieved (see Fig. 2). The result of the multiply-add operation V_x is then passed to the inverter-based activation functions [11], implemented by a printed transistor (T_1, T_2) and a pull-up resistor $(R_{\text{pull1}}, R_{\text{pull2}})$. The outputs of each inverter stage $(V_{\text{out-inv}}, V_{\text{out}})$ are complementary and by connecting the inverted outputs, negative weights can be expressed [12].

B. Training of pNN

Training pNN refers to finding a set of printable conductance values, that allow the printed pNN to satisfies the desired functionality. To find these conductance values, the training algorithm proposed in [9] can be used which was specifically designed for training pNNs. In this approach the conductance values are learned alongside the connectivity, i.e., whether a neuron shall be connected to an inverted input or not, as only either connection is required for pNNs. Models for activation and inverter functions can be obtained by fitting parameterized tanh function through measurement or simulation data of printed devices [9].

C. Variation in Printed Resistances

Low-cost additive manufacturing results in low resolution droplet printing which impacts circuit behaviour and leads to a variation in conductances of printed resistors. The variations in the printing layers lead to deviations of the effective resistor width and length. To model the variations of the conductances, a random multiplicative variation (as in [9]) is assumed, i.e.,

$$g = \bar{g} \cdot (1 + r) \quad \text{with} \quad p(r) = \mathcal{U}(-\epsilon, \epsilon), \qquad (2)$$

where $\bar{g}$ denotes the intended conductance and r is a random variable drawn from a uniform distribution with bounds ϵ. By selecting different values for ϵ, e.g. 0.1, a variation of 10% from the intended conductance $\bar{g}$ is obtained.

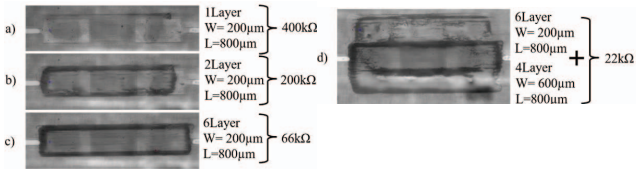

Fig. 2. Microscope photos: reprinting of a resistor by adding layers: a), b), c) or increasing the width: d).

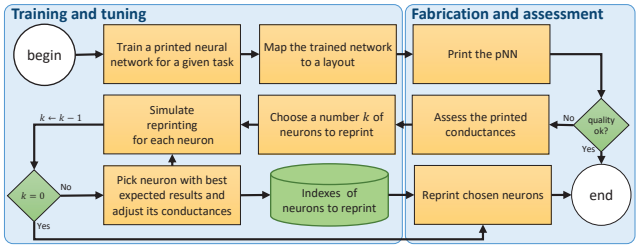

Fig. 3. A Flowchart of the full post-fabrication procedure. The left part is done in software and based on a pNN model, while the right part relates to measurement and fabrication of the pNN.

III. In-Situ Tuning Methodology

A. Resistor tuning in printed additive manufacturing

One of the key features of printed electronics is additive manufacturing. In the context of post-fabrication tuning, this allows for additive refinements to tune the conductances of printed resistors after their initial fabrication. This capability not only allows a finer tuning of the desired values as in [13], but also offers an opportunity to combat process variation as targeted in this work.

It is important to note that, in the context of printed electronics, the exhaustive electrical circuit testing may not be required for every device. Instead, most of the imperfectly printed components should be detectable from optical inspections — enabled by transparent layers and no rigid packaging — on the droplet patterns and their associated printing quality. Once an imperfectly printed resistor is detected, additional layers or even geometries can be printed over the initially fabricated resistor to mitigate the yield degradation. The additional resistor variations in this step can be neglected as the printing effort in this additional printing step is increased to reach the specified resistances. In Fig. 2, such a re-printing process is illustrated.

The starting point of the procedure is a pNN model that was learned according to the training procedure from [9] and mapped to a respective pNN. After fabrication, no satisfying results were obtained due to variations in the fabrication process. This triggers our post-fabrication procedure (see Fig. 3).

B. Modelling assumptions

Before the details of the post-fabrication procedure are outlined, we clarify the assumptions underlying the procedure. First of all, it is assumed that only the crossbar conductances exhibit variation. The other circuit components, like the activation and inverter function, exhibit no variation. This could for example be justified in a split-manufacturing scenario, where

the activation and inverter circuits were prefabricated through an industrial replication printing process with comparably little variations. Then, only the crossbar conductances are left to be fabricated to configure the circuit on-demand via inkjet printing.

In the following, we denote the collection of crossbar conductances g_i of a given printed neuron by $\boldsymbol{g}$. Furthermore, $\boldsymbol{g}^{old}$ denotes the initially fabricated conductances. Consequently, we denote the new target conductances found through the post-fabrication procedure by $\boldsymbol{g}^{new}$. Given this notation, we assume the following:

1) The printed conductances $\boldsymbol{g}^{old}$ of the pNN can be accurately assessed.
2) A limited range of conductance $g_i \in [g_{min}, g_{max}]$ can be realized.
3) Reprinting increases the conductance, hence $g_i^{new} \geq g_i^{old}$.
4) Reprinted conductances exhibit a lower variation $\epsilon_{reprint}$.

Based on these assumptions, the next section describes a procedure to find a new, feasible set of conductances $\boldsymbol{g}^{new}$ to restore the initially intended weights.

C. Restoring the weights of a printed neuron

For the training of pNNs, (surrogate) conductances are used as learnable parameters (see [9]). They are preferred over directly finding a set of weights $\boldsymbol{w}$, as they allow a simpler treatment of the circuit and technology constraints. Nevertheless, the weights w_i, which are formed as a function of the conductances $\boldsymbol{g}$, determine the output of the neuron. As long as the conductances form the correct weights, the pNN model, and hence the pNN, yield the correct output. While the conductances $\boldsymbol{g}$ uniquely determine the weights $\boldsymbol{w}(\boldsymbol{g})$, the reverse is not true. In fact, the weights $\boldsymbol{w}(\boldsymbol{g})$ are scale-invariant with respect to the conductances $\boldsymbol{g}$, as

$$w_i(s \cdot \boldsymbol{g}) = \frac{(s \cdot g_i)}{\sum_j (s \cdot g_j)} = \frac{\cancel{s} \cdot g_i}{\cancel{s} \cdot \sum_j g_j} = \frac{g_i}{\sum_j g_j} = w_i(\boldsymbol{g}). \quad (3)$$

Hence, there is a whole (half-)space of possible vectors $\boldsymbol{g}$ yielding the same weights $\boldsymbol{w}(\boldsymbol{g})$. This key observation forms the basis of the post-fabrication procedure.

In the following, for a given neuron, let $\boldsymbol{g}^\star$ be the conductances obtained from the training algorithm, and let $\boldsymbol{g}^{old}$ denote the actually printed conductances obtained after fabrication. Due to (3), $\boldsymbol{w}(\boldsymbol{g}^\star) = \boldsymbol{w}(s \cdot \boldsymbol{g}^\star)$. Thus, by finding a scaling factor $s \in \mathbb{R}^+$ for which $\boldsymbol{g}^{new} = s \cdot \boldsymbol{g}^\star$ is feasible, the same, originally intended weights $\boldsymbol{w}(\boldsymbol{g}^\star)$ are realized.

In principle, any scaling factor s can be chosen for which the specified constraints hold. However, to save time and material, the optimal scaling factor $s^\star$ should be chosen such that an overall minimal adjustment for all conductances $\boldsymbol{g}$ is required. This can be expressed as

$$\begin{aligned} s^\star = \underset{s \in \mathbb{R}^+}{\operatorname{argmin}} \sum_i |s \cdot g_i^\star - g_i^{old}| \text{ s.t } \forall i \text{ with } g_i^{old} \neq 0 : \\ s \cdot g_i^\star \geq g_i^{old} + T \\ s \cdot g_i^\star \leq g_{max}. \end{aligned} \quad (4)$$

The formulation represents a one dimensional optimization problem with two times as many constraints as there are nonzero conductances in the neuron. After solving (4) and obtaining $s^\star$, the initially fabricated conductances $\boldsymbol{g}^{old}$ of the neuron can be adjusted to $\boldsymbol{g}^{new} = s^\star \cdot \boldsymbol{g}^\star$ which realizes the intended weights according to (3). For brevity, the process of finding and applying $\boldsymbol{g}^{new}$ to the pNN is simply referred to as reprinting in the following.

TABLE I
THE BENCHMARK RESULTS OF SEVERAL TRAINED PRINTED NEURAL NETWORKS. FOR EACH DATASET, A GRID-SEARCH WAS PERFORMED OVER DIFFERENT INITIAL LEARNING RATES α, PENALTY FUNCTION COEFFICIENTS λ AND MARGIN m. THE NETWORKS WITH THE BEST TRAINING MAA OF TEN RANDOM INITIALISATIONS (SEEDS) IS REPORTED.

Dataset	Architecture	Hyperparameters	Measuring-aware Accuracy (MaA)		
	neurons/layer	(α, m, λ)	Train	Test	random guess
Acute Inflammations	6-4-3-2	(1.0, 0.7, 0.0)	1.0000	1.0000	0.4750
Balance Scale	4-4-3-3	(0.1, 0.4, 0.01)	0.9354	0.9034	0.4396
Breast Cancer Wisconsin	9-4-3-2	(0.1, 0.5, 0.001)	0.9808	0.9697	0.6667
Energy efficiency (y1)	8-4-3-3	(0.1, 0.5, 0.1)	0.8813	0.8740	0.4331
Energy efficiency (y2)	8-4-3-3	(0.01, 0.3, 0.01)	0.9163	0.9094	0.4646
Iris	4-4-3-3	(0.01, 0.9, 0.001)	0.9800	0.9400	0.2800
Mammographic Mass	5-4-3-2	(0.1, 0.9, 0.1)	0.8351	0.8365	0.5503
Seeds	7-4-3-3	(0.01, 0.5, 0.1)	0.9500	0.9429	0.2714
Tic-Tac-Toe Endgame	9-4-3-2	(0.1, 0.7, 0.1)	0.9938	0.9716	0.6404
Vertebral Column (2 classes)	6-4-3-2	(0.1, 0.7, 0.1)	0.8599	0.8738	0.6893
Vertebral Column (3 classes)	6-4-3-3	(0.01, 0.1, 0.001)	0.7874	0.8350	0.5146

D. Post-fabrication tuning for printed neural networks

Through the problem formulation in the last subsection, a new conductance configuration $\boldsymbol{g}^{new}$ restoring the intended weights of a given neuron can be found. Trivially, the conductances of all neurons in the network could be reprinted to benefit from the smaller variation. However, since reprinting is time consuming, only a limited number neurons, say k, should be chosen for reprinting. This leads to the question of finding the best k neurons to reprint.

For a given neuron, the expected accuracy improvement for reprinting can be assessed through simulating and evaluating the reprinting procedure. Based on this estimate, promising neurons to reprint can be identified. To avoid having to enumerate all possible combinations of neurons to reprint, a sequential, greedy strategy is pursued. Here, for reprinting k neurons, k steps are taken. In each step, the expected accuracy improvement for reprinting each neuron is evaluated separately. Then, the most promising of those reprinting steps is executed.

IV. EXPERIMENTS

A. Experimental setup

For our evaluation, we adopt the pNN design proposed in [9] and train pNNs accordingly (without variation-aware training) on the same datasets. To find the hyperparameters (α, m, λ) (learning rate, margin, penalty coefficient) of the training routine, we perform a grid-search. Additionally, the evaluation metric "measuring-aware accuracy" (MaA) [9], which represents a stricter version of the classic accuracy, is adopted for evaluation. The results for the trained pNNs along with the chosen hyperparameters can be found in Table I.

After training, we simulate conductance variation according to (2) for $\epsilon \in \{10, 20, 30\}\%$, and apply our post-fabrication procedure reprinting $k = 0, \cdots, 60\%$ of the neurons of each network. The neurons to reprint are chosen successively based

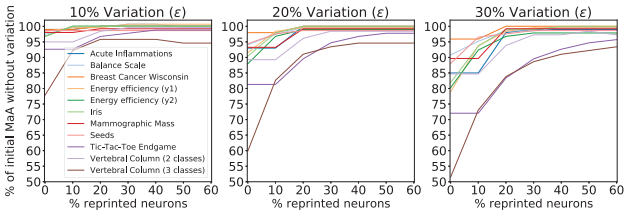

Fig. 4. The post-fabrication procedure applied to printed neural networks trained on different datasets and subjected to conductance variation of $\epsilon = 10, 20, 30$ % (see (2)). The x-axis displays the % of the total neurons that are reprinted (k). The y-axis denotes the % of the MaA achieved compared to the variationless pNN (see Table I).

on the training MaA after a simulated (variationless) reprinting step. After a total of k neurons have been identified for reprinting, the conductances of the chosen neurons are adjusted jointly and the reprinting variation $\epsilon_{reprint}$ is applied. Then, the MaA of the pNN is calculated on the test set. This procedure is repeated 100 times and the average test MaAs are calculated for the final evaluation.

Naturally, the post-fabrication procedure requires an extended range of printable conductance values to increase conductances that were set to g_{max} in training. For the experiments, a conductance range of 1:600 is selected for reprinting (compared to 1:100 for training). When reprinting the conductances of a neuron, the previously printed conductances g^{old} are given. Hence, only the adjustment $\boldsymbol{g}^{new} - \boldsymbol{g}^{old}$ is considered to vary with the reprinting variation of $\epsilon_{reprint} = 5\%$. The minimal adjustment distance T is set to $T = g_{min}$ in our experiments. This choice can be justified by the assumption that g_{min} may represent the conductance of the smallest printable geometry. To allow for a comparison of different datasets/networks, we report the test set results for different percentages of reprinted neurons. Furthermore, as the achievable results on the datasets differ, i.e., reaching 100 % is often not possible, the results for each network are normalized with respect to their MaA at $\epsilon = 0\%$, i.e., without any conductance variation (see Table I). The results from the post-fabrication procedure can be seen in Fig. 4.

B. Results

Evidently, the more neurons are reprinted, the better the achieved MaA. However, the effectiveness of reprinting saturates when reprinting more than 30% of the neurons. This is likely due to the reprinting variation $\epsilon_{reprint}$, which does not permit to recover the variationless result (and may sometimes even decrease the results slightly). Moreover, the higher the variation, the more neurons need to be reprinted to recover MaA levels close to the initial accuracy. Finally, reprinting about $20 - 30\%$ of the neurons allows to restore $90 - 95\%$ of the initial $\epsilon = 0\%$ MaA for most datasets/networks. Two exceptions are *Vertebral Column* with 3 classes and *Tic-Tac-Toe Endgame*, which require more neurons to be reprinted to achieve comparable results.

V. Conclusion

In this work, we proposed a post-fabrication tuning procedure for printed neural networks. The method leverages a unique feature of printed electronics, namely additive manufacturing capabilities, to refine the crossbar-conductances after their initial fabrication. Through this, erroneous weights stemming from variations in the conductances of the crossbar resistors can be corrected to restore the initially intended weights. Our experiments showed that reprinting less than a third of the neurons is often sufficient to recover more than 90% of the model accuracy.

Acknowledgements

This work was supported by the Ministry of Science, Research and Arts of the state of Baden-Württemberg in form of the MERAGEM doctoral program.

References

[1] Z. Cui, *Printed electronics: materials, technologies and applications*. John Wiley & Sons, 2016.

[2] G. Cadilha Marques, D. Weller, A. T. Erozan, X. Feng, M. Tahoori, and J. Aghassi-Hagmann, "Progress report on "from printed electrolyte-gated metal-oxide devices to circuits"," *Advanced Materials*, p. 1806483, 2019.

[3] F. Rasheed, M. Hefenbrock, M. Beigl, M. B. Tahoori, and J. Aghassi-Hagmann, "Variability modeling for printed inorganic electrolyte-gated transistors and circuits," *IEEE Transactions on Electron Devices*, vol. 66, no. 1, pp. 146–152, 2018.

[4] S. A. Mohassieb, K. Kirah, E. Dorsam, A. S. G. Khalil, and H. M. S. El-Hennawy, "Inkjet printing of a 20 ghz coplanar waveguide monopole antenna using copper oxide nanoparticles on flexible substrates: effect of drop spacing on antenna performance," *Progress In Electromagnetics Research*, vol. 73, pp. 87–95, 2017.

[5] N. Bleier, M. Mubarik, F. Rasheed, J. Aghassi-Hagmann, M. Tahoori, and R. Kumar, "Printed microprocessors," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

[6] R. A. Nawrocki, R. M. Voyles, and S. E. Shaheen, "Neurons in polymer: Hardware neural units based on polymer memristive devices and polymer transistors," *IEEE Transactions on Electron Devices*, vol. 61, no. 10, pp. 3513–3519, 2014.

[7] Y. van De Burgt, A. Melianas, S. T. Keene, G. Malliaras, and A. Salleo, "Organic electronics for neuromorphic computing," *Nature Electronics*, p. 1, 2018.

[8] D. D. Weller, M. Hefenbrock, M. B. Tahoori, J. Aghassi-Hagmann, and M. Beigl, "Programmable neuromorphic circuit based on printed electrolyte-gated transistors," in *Proceedings of the Asia South Pacific design automation conference (ASP-DAC)*, 2020.

[9] D. D. Weller, M. Hefenbrock, M. Beigl, J. Aghassi-Hagmann, and M. B. Tahoori, "Realization and training of an inverter-based printed neuromorphic computing system," *Scientific Reports*, vol. 11, no. 1, 2021. [Online]. Available: https://doi.org/10.1038/s41598-021-88396-0

[10] B. Liu, H. Li, Y. Chen, X. Li, Q. Wu, and T. Huang, "Vortex: variation-aware training for memristor x-bar," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 15.

[11] A. BanaGozar, M. A. Maleki, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Robust neuromorphic computing in the presence of process variation," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017, pp. 440–445.

[12] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of bsb recall function using memristor crossbar arrays," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 498–503.

[13] A. T. Erozan, G. Y. Wang, R. Bishnoi, J. Aghassi-Hagmann, and M. B. Tahoori, "A compact low-voltage true random number generator based on inkjet printing technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1485–1495, 2020.