Highly-Bespoke Robust Printed Neuromorphic Circuits

Haibin Zhao[1], Brojogopal Sapui[1], Michael Hefenbrock[2], Zhidong Yang[1]
Michael Beigl[1], and Mehdi B. Tahoori[1]
[1]*Karlsruhe Institute of Technology,* [2]*RevoAI GmbH*
[1]{haibin.zhao, brojogopal.sapui, zhidong.yang, michael.beigl, mehdi.tahoori}@kit.edu
[2]michael.hefenbrock@revoai.de

***Abstract*—With the rapid growth of the Internet of Things, smart fast-moving consumer products, and wearable devices, requirements such as flexibility, non-toxicity, and low cost are desperately required. However, these requirements are usually beyond the reach of conventional rigid silicon technologies. In this regard, printed electronics offers a promising alternative. Combined with neuromorphic computing, printed neuromorphic circuits offer not only the aforementioned properties, but also compensate for some of the weaknesses of printed electronics, such as manufacturing variations, low device count, and high latency. Generally, (printed) neuromorphic circuits express their functionality through printed resistor crossbars to emulate matrix multiplication, and nonlinear circuitry to express activation functions. The values of the former are usually learned, while the latter is designed beforehand and considered fixed in training for all tasks. The additive manufacturing feature of printed electronics allows the design of highly-bespoke designs. In the case of printed neuromorphic circuits, the circuit is optimized to a particular dataset. Moreover, we explore an approach to learn not only the values of the crossbar resistances, but also the parameterization of the nonlinear components for a bespoke implementation. While providing additional flexibility of the functionality to be expressed, this will also allow an increased robustness against printing variation. The experiments show that the accuracy and robustness of printed neuromorphic circuits can be improved by 26% and 75% respectively under 10% variation of circuit components.**

I. Introduction

Next-generation electronics, such as soft-robots, fast-moving consumer goods, wearables, and the Internet of Things (IoT) infrastructures, shows a trend towards lightweight, flexibility and cheapness [1]. However, classical silicon-based technologies cannot meet these requirements due to their bulky substrates and high production costs from the complicated subtractive manufacturing processes. In this regard, printed electronics (PE) becomes a promising candidate, as the additive manufacturing used for their fabrication offers ultra-low production costs, as it requires fewer fabrication steps and equipment compared to conventional lithography processes used in silicon fabrication. Additionally, depending on the choice of printing materials and substrates, printed circuits can be non-toxic and flexible [2], [3].

With the rapid development in data science and machine learning, neuromorphic computing receives increasing attention for its expressiveness and ability to approximate desired functionalities [4]. Moreover, the fundamental operations in neuromorphic computing are simple (generally weighted-sum operations and activation functions), which facilitates the implementation of neuromorphic computing by simple circuit primitives. By printing and connecting these primitives, printed neuromorphic circuits are realized. More specifically, the circuits are the interconnection of multiple printed neurons that mainly consist of resistor crossbars for weighted-sum operations and nonlinear circuits as activation functions.

Multiple works have already proposed adaptations of the designs and/or training algorithms for printed neuromorphic circuits. For example, [1] proposed an inverter-based printed neuromorphic circuit, consisting of resistor crossbars, tanh-like nonlinear circuits, and negative weight circuits. They also proposed a design algorithm for the circuits, which considers the variation of resistance values in the crossbar. Similarly, [5] and [6] investigated the aging of crossbars in neuromorphic circuits and proposed an adapted training objective to mitigate aging effects.

However, these works focus solely on the resistors in the crossbars as learnable parameters, which correspond to the weights in neural networks (NNs). For the nonlinear components, the same predefined and fixed parameterization was used. In this regard, the unique feature of PE, i.e., high customizability, is not fully leveraged, because the design and optimization of nonlinear subcircuits can also be adapted to the target tasks. In this work, we shift the attention to the nonlinear subcircuits, namely the negative weights circuits and activation circuits with the intention to make their characteristics learnable. This means, different tasks may have different activation circuits with different characteristic curves. To assess the behavior for different configurations of these components, we learn small, parameterized artificial NNs based on SPICE simulation data. These NN models then serve as surrogate models of the nonlinear circuits, which transform the component values to the characteristic curves. As these models are fully differentiable, their circuit parameterization can be learned alongside the values for the crossbar resistances (i.e., weights) in training. This added flexibility not only allows to learn more expressible functions with the same circuitry, but also increases the robustness of the resulting circuits against manufacturing variations, which is also a major issue in additive printing.

In summary, the contributions of this work are:

- We approximate the behavior of the nonlinear circuit components through regression NNs. They then serve as surrogate models for their behaviors and allow us to learn

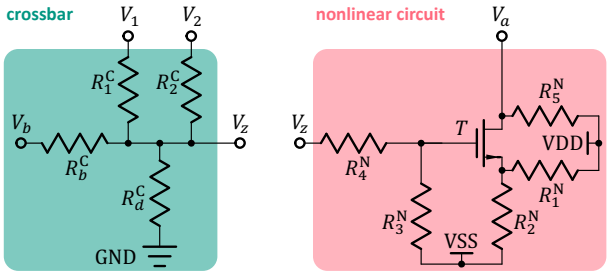

Fig. 1. Primitive subcircuits of printed neuromorphic circuits. The left part indicates a resistor crossbar, while the right part shows the schematic of a circuit for nonlinear transformation.

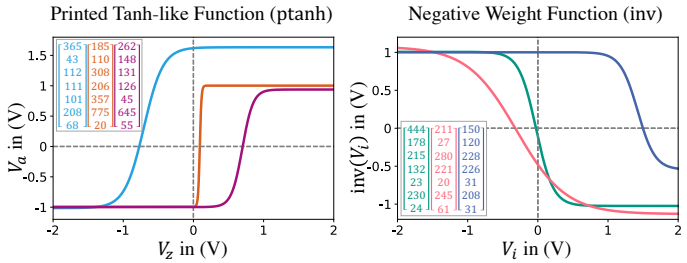

Fig. 2. Exemplary characteristic curves of a ptanh circuit (left) and a negative weight circuit (right) with certain physical parameter values. The legends with corresponding colors indicate the values of physical parameters ω.

individual component values in the main training routine.
- We extend variation-aware training of printed neural networks for the consideration of learnable nonlinear subcircuits. This means, not only the variance of resistors in the crossbar is taken into account, but also the components in nonlinear circuits.
- We conduct experiments to prove the feasibility of learning nonlinear subcircuits, and show that it can additionally increase the robustness against manufacturing variations.

The rest of this paper is structured as follows: Sec. II briefly introduces the background material. Sec. III describes the proposed surrogate models of the nonlinear circuits, their integration into the printed neural networks, and an adapted variation-aware training approach. In Sec. IV, we evaluate the effectiveness of the proposed models on benchmark datasets. Sec. V concludes this work.

II. Preliminary

A. Printed Electronics

Similar to color printing, PE technology is an additive fabrication process for electronic components based on various printing techniques, such as roll-to-roll printing and jet printing [7].

Depending on the printing materials and substrates, printed circuits can offer several advantages, such as non-toxicity, lightweight, and flexibility. Additionally, as the additive manufacturing process for PE requires less and cheaper equipment compared to subtractive manufacturing of the conventional silicon-based VLSI, printed circuits can be fabricated at an ultra-low cost. This advantage further facilitates the development of new products by enabling the cost-effective production of highly-customized devices in small batches. Another formidable benefit of additive manufacturing is that, the printed components can be modified at almost any stage in the production, which enables the post-processing and even the repair of printed devices. Furthermore, the simplicity of the manufacturing process and the low non-recurring engineering costs of additive manufacturing provide the ability to fabricate and customize highly personalized or task-specified electronic devices.

Despite these benefits, additive manufacturing also presents drawbacks to printed electronics, such as large feature sizes and low integration density. Moreover, printed electronics may also suffer from higher fabrication variabilities. To this end, rather than replacing the conventional silicon-based technology, printed electronics should be leveraged as a complement by offering its unique advantages in various applications.

Considering both advantages and disadvantages, several fundamental components of printed computing systems have been successfully realized, including but not limited to Boolean logic [8], storage elements [9], and amplifiers [10].

In this work, we leverage the high customizability of PE to tailor the nonlinear subcircuits in printed analog neuromorphic circuits to specific tasks. These nonlinear subcircuits were previously considered as fixed for all tasks. Moreover, we respect the high variation of PE by employing variation-aware training for the nonlinear circuit components. As a result, the printed neuromorphic circuits are customized and optimized to the given tasks, leading to significant enhancements in circuit performance (e.g., accuracy in classification tasks) and improved robustness to printing variations.

B. Printed Analog Neuromorphic Circuit

Neuromorphic circuits implement the functionalities that are equivalent to the operations in artificial NNs, specifically, weighted-sums and nonlinear activations. There have been multiple hardware-level implementations for digital neuromorphic systems [11], but these digital schemes are not particularly well-suited to PE, which is characterized by large feature sizes and low integration density. Consequently, the analog neuromorphic computing approach has become increasingly attractive within the PE community, as analog solutions necessitate significantly lower device counts. For example, a 3-input 1-output 4-bit digital neural neuron requires hundreds of transistors, whereas an analog approach requires only less than ten [1].

In printed analog neuromorphic circuits, the weighted-sums are realized through printed resistor crossbars and nonlinear activations are implemented via inverter-based circuitry. These components are explained next.

a) Resistor crossbar: The left part in Fig. 1 shows a resistor crossbar for weighted-sum operation. According to Kirchhoff's law,

$$\sum_i \frac{V_i - V_z}{R_i^{\mathrm{C}}} + \frac{V_b - V_z}{R_b^{\mathrm{C}}} - \frac{V_z}{R_d^{\mathrm{C}}} = 0.$$

V_i and V_z denote the input and output voltages of the crossbar, V_b denotes a constant bias voltage (e.g., 1V) Moreover, R_i^{C}, R_b^{C}, and R_d^{C} refer to the corresponding resistances of the voltages. The superscript $(\cdot)^{\mathrm{C}}$ distinguishes the resistors in the

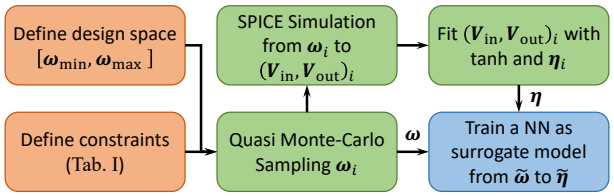

Fig. 3. Pipeline for modelling of the nonlinear circuits. The orange boxes indicate the feasible design space, the green boxes denote the process of building the dataset, and the blue box refers to the model approximation.

crossbar from that in the nonlinear circuits (which will be introduced later). After expressing the resistances R by the conductances, $g = 1/R$, we obtain

$$V_z = \sum_i \frac{g_i^{\mathrm{C}}}{G} V_i + \frac{g_b^{\mathrm{C}}}{G}, \tag{1}$$

where $G = \sum_i g_i^{\mathrm{C}} + g_b^{\mathrm{C}} + g_d^{\mathrm{C}}$. Consequently, the output voltage V_z is produced by a weighted-sum (including a bias) of the input voltages V_i. It thus resembles the weighted-sum operation in artificial NNs.

b) Printed tanh-like (ptanh) circuit: Following the resistor crossbar, a circuit for nonlinear transformation is printed. The right part in Fig. 1 represents the schematic of an inverter-based nonlinear circuit. By cascading of two inverters, a tanh-like transformation function can be resembled, whose characteristic curve can be approximately described by a modified $\tanh$ function, $\mathrm{ptanh}(\cdot)$, i.e.,

$$V_a = \mathrm{ptanh}(V_z) = \eta_1 + \eta_2 \cdot \tanh\left((V_z - \eta_3) \cdot \eta_4\right), \tag{2}$$

where V_z and V_a are the input and output voltages of the tanh-like circuit. The parameter vector $\boldsymbol{\eta} = [\eta_1, \eta_2, \eta_3, \eta_4]$ modifies the shape and position of the $\tanh$ function. Note that $\boldsymbol{\eta}$ are only auxiliary parameters that do not directly relate to the values of the physical components in the nonlinear circuits, namely, the resistors R_i^{N} and the dimensions of the electrolyte-gated transistor T, with its width W and length L [12]. By modifying these physical parameters, summarized by $\boldsymbol{\omega} := [R_1^{\mathrm{N}}, R_2^{\mathrm{N}}, R_3^{\mathrm{N}}, R_4^{\mathrm{N}}, R_5^{\mathrm{N}}, W, L]$, we can achieve different curve characteristics, see Fig. 2 (left).

c) Negative weight circuit: According to Eq. 1, the weights are formed through the conductances in the crossbar. Consequently, they cannot be negative. To address this problem and express negative relationships, the printed *negative weight circuit* is proposed. Whenever a negative weight is needed, the respective input V_i is instead transformed with the negative weight circuit, $\mathrm{inv}(\cdot)$, before it is connected to the succeeding resistor crossbar. Multiple circuits are possible to implement the negative weight function. But as a shortcut, we use the same circuit as ptanh circuit in this work. The characteristic curve of the circuit can also be described by a modified $\tanh$ function, namely,

$$\mathrm{inv}(V_i) = -\left(\eta_1 + \eta_2 \cdot \tanh\left((V_i - \eta_3) \cdot \eta_4\right)\right). \tag{3}$$

Fig. 2 (right) shows some characteristic curves of the *negative weight circuit* with different physical quantities.

C. Printed Neural Network

To obtain the optimal component values in a neuromorphic circuit, a machine learning based model of the circuit, i.e., printed neural network (pNN), was proposed in [1]. Usually, the learnable parameters of the pNN are the surrogate conductances $\boldsymbol{\theta} = [\theta_1, \theta_2, ...]$. Each θ_i resembles the conductance values g_i^{C} and the corresponding *negative weight circuits*. Specifically, the absolute value of the surrogate conductance $|\theta_i|$ indicates the conductance to be printed, while the sign of θ_i denotes whether the input voltage V_i should be inverted. In this way, **training** a pNN can be understood as **designing** a printed neuromorphic circuit.

A pNN is also able to consider further technical constraints. For example, the range of printable conductances is not arbitrary: With a given printable range of conductances, i.e., $g_i \in \{0\} \cup [\mathrm{G}_{\min}, \mathrm{G}_{\max}]$ (zero refers to not printing), θ_i must be in $[-\mathrm{G}_{\max}, -\mathrm{G}_{\min}] \cup \{0\} \cup [\mathrm{G}_{\min}, \mathrm{G}_{\max}]$. This constraint can be ensured through projecting all infeasible values into feasible range in the forward pass, while ignoring the projection in the backward pass. This technique of obtaining gradients is commonly referred to as the straight-through (gradient) estimator [13].

Moreover, the variation of values caused by fabrication errors can also be considered in pNNs: By modifying the learnable parameters to stochastic variables, e.g., $\boldsymbol{\theta} \sim p_\theta(\boldsymbol{\theta})$, the variation of printed values can be simulated. In this case, the expected loss function w.r.t. variation is minimized. A more concrete example is introduced in Sec. III-C.

III. Methodology

Additive manufacturing uniquely allows PE to have highly customized circuit components. However, only the crossbars are designed task-specifically in previous work, whereas the nonlinear subcircuits are always predefined and fixed. To enable individual adjustment of the nonlinear circuits, we extend the learnable parameters in pNNs by the parameters for nonlinear components. Specifically, we introduce differentiable, NN-based surrogate models to describe the behaviors of the nonlinear circuits, and integrate them into the pNNs. Moreover, thanks to the surrogate models, the printing variation in nonlinear circuits can also be considered into the training of pNNs.

A. Modelling of Nonlinear Circuits

To enable the learning of physical quantities $\boldsymbol{\omega}$ rather than the auxiliary variables $\boldsymbol{\eta}$, a differentiable mapping $\boldsymbol{\omega} \mapsto \boldsymbol{\eta}$ is required. Therefore, we choose NNs as surrogate models to approximate this mapping. As shown in Fig. 3, to model the nonlinear circuit, the design space of circuit components is defined first. Subsequently, since the mapping is approximated by NNs, the dataset that describes the circuit behavior is generated (via SPICE simulation). Finally, NNs are trained as the surrogate models for nonlinear circuits, which are denoted by $\hat{\boldsymbol{\eta}}(\boldsymbol{\omega})$ in the following.

a) Design space: In accordance with printing technology and circuit design experience, we introduce the following constraints to the circuit components: The ranges of the parameters

$[\boldsymbol{\omega}_{\min}, \boldsymbol{\omega}_{\max}]$ are found by performing sweep analysis in the simulation tool, which leads to tanh-like characteristic curves. Moreover, the resistances R_1^{N} and R_3^{N} must be larger than R_2^{N} and R_4^{N}, respectively. Otherwise, the voltage divider cannot meet the assumption of a constant ratio due to the connections with surrounding circuit elements. Tab. I summarizes all the constraints of the design space.

TABLE I
FEASIBLE DESIGN SPACE OF NONLINEAR CIRCUIT

	R_1^{N} (Ω)	R_2^{N} (Ω)	R_3^{N} ($k\Omega$)	R_4^{N} ($k\Omega$)	R_5^{N} ($k\Omega$)	W (μm)	L (μm)
minimal	10	5	10	8	10	200	10
maximal	500	250	500	400	500	800	70
inequality	$R_1^{\mathrm{N}} > R_2^{\mathrm{N}}$		$R_3^{\mathrm{N}} > R_4^{\mathrm{N}}$		-	-	-

b) Dataset: The dataset consists of pairs of physical design parameters $\boldsymbol{\omega}$ and the corresponding auxiliary parameters $\boldsymbol{\eta}$ for tanh-like functions. To obtain a set of $\boldsymbol{\omega}$ that are representative enough for the whole feasible design space, we employ Quasi Monte-Carlo sampling [14] to draw 10 000 points in the feasible design space, which are denoted by $\boldsymbol{\omega}_i$, $i = 1, ..., 10\,000$. Afterwards, we use Cadence Virtuoso[1] for SPICE simulation based on a prior developed printed Process Design Kit (pPDK) [12] to simulate the input and output voltages $(\boldsymbol{V}_{\mathrm{in}}, \boldsymbol{V}_{\mathrm{out}})_i$ for each sampled circuit (parameterized by $\boldsymbol{\omega}_i$). The green points in Fig. 4 (left) exemplify a simulation result with a certain $\boldsymbol{\omega}_i$. Note that the number of points plotted in the figure has been reduced for clarity of visualization.

To extract $\boldsymbol{\eta}_i$, we fit the simulated data $(\boldsymbol{V}_{\mathrm{in}}, \boldsymbol{V}_{\mathrm{out}})_i$ by Eq. 2 or Eq. 3 (depending on the circuit) with minimal Euclidean distance, i.e.,

$$\boldsymbol{\eta}^* = \arg\min_{\boldsymbol{\eta}} \left\| \mathrm{ptanh}_{\boldsymbol{\eta}}(\boldsymbol{V}_{\mathrm{in}}) - \boldsymbol{V}_{\mathrm{out}} \right\|_2 .$$

By now, we have collected physical design parameters $\boldsymbol{\omega}_i$ and their corresponding auxiliary parameters $\boldsymbol{\eta}_i$. With this dataset, NN-based surrogate models $\hat{\boldsymbol{\eta}}(\boldsymbol{\omega})$ can be trained to describe the transformation from $\boldsymbol{\omega}_i$ to $\boldsymbol{\eta}_i$.

c) Surrogate nonlinear circuit models: Since the relationship between $\boldsymbol{\omega}_i$ and $\boldsymbol{\eta}_i$ is complicated, we propose to approximate it by surrogate models based on artificial NNs. Empirically, data normalization can significantly improve the performance of NNs. However, the ratios of voltage dividers, i.e., $R_2^{\mathrm{N}}/R_1^{\mathrm{N}}$ and $R_4^{\mathrm{N}}/R_3^{\mathrm{N}}$, and the ratio between W and L are critical features of the circuits. If each parameter is normalized independently, the ratio information is weakened. We therefore extended the design parameters manually with these three ratios, i.e.,

$$\boldsymbol{\omega} \mapsto [R_1^{\mathrm{N}}, R_2^{\mathrm{N}}, R_3^{\mathrm{N}}, R_4^{\mathrm{N}}, R_5^{\mathrm{N}}, W, L, k_1, k_2, k_3],$$

where k_1, k_2, and k_3 denote the aforementioned ratios. After the extension, we normalize the data $\boldsymbol{\omega}$ to $\tilde{\boldsymbol{\omega}}$ as the input of the NNs. Similarly, the target outputs of the NNs are also the normalized values of $\boldsymbol{\eta}$, which are denoted by $\tilde{\boldsymbol{\eta}}$. The maximal and minimal values $\boldsymbol{\omega}_{\min}$, $\boldsymbol{\omega}_{\max}$, $\boldsymbol{\eta}_{\min}$, and $\boldsymbol{\eta}_{\max}$ are saved to perform denormalization later. To find the best

[1]https://www.cadence.com

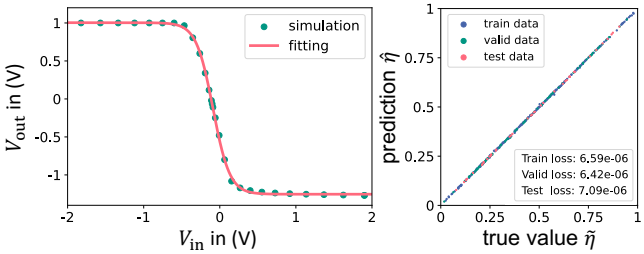

Fig. 4. Left: parameter fitting from $(\boldsymbol{V}_{\mathrm{in}}, \boldsymbol{V}_{\mathrm{out}})$ to $\boldsymbol{\eta}$. Green points show the simulated input/output voltages, and the red curve indicates the fitted tanh-like function parameterized by $\boldsymbol{\eta}$. Right: visualization of the results from the surrogate model. The x-axis and the y-axis refer to the true value $\tilde{\boldsymbol{\eta}}$ and predicted value $\hat{\boldsymbol{\eta}}(\boldsymbol{\omega})$. Blue, green, and red colors denotes the data from training, validation, and test sets.

NNs for surrogate models, we randomly split the dataset into training set (70%), validation set (20%), and test set (10%), train the NNs on the training set, and use the MSE loss on the validation set to find the best hyperparameters for NNs. After the hyperparameter tuning, a 13-layer NN (#neurons: 10-9-9-8-8-7-7-6-6-6-5-5-5-4) is obtained as the final surrogate nonlinear circuit model. The plot on the right side of Fig. 4 visualizes the results of all three sets from a surrogate model. We can thus conclude that, there is no overfitting on the training data and the surrogate model provides acceptable predictions from the component values $\boldsymbol{\omega}$ to the characteristic curves $\boldsymbol{\eta}$.

B. Constraints on the Design Parameters of the pNN

To integrate the surrogate models into pNNs, we introduce an additional learnable parameter $\mathfrak{w}$ to the pNNs. To respect the feasible range of the parameters, we consider the learnable parameter as the normalized one. Moreover, to include the inequality constraints, we do not directly learn R_2^{N} and R_4^{N}, but their corresponding ratios k_1 and k_2. Thus, the learnable parameter $\mathfrak{w}$ corresponds to $[R_1^{\mathrm{N}}, R_3^{\mathrm{N}}, R_5^{\mathrm{N}}, W, L, k_1, k_2]$, as shown in Fig. 5. Due to the normalization and inequality constraints, all values in $\mathfrak{w}$ should be in the range $(0, 1)$. Thus, we pass $\mathfrak{w}$ through a sigmoid function to ensure this constraint.

To convert the parameters further to printable values, we denormalize the first five elements and reassemble the vector with $R_2^{\mathrm{N}} = R_1^{\mathrm{N}} k_1$ and $R_4^{\mathrm{N}} = R_3^{\mathrm{N}} k_2$. Since the elements R_2^{N} and R_4^{N} are inferred by k_1 and k_2, they may not be in their feasible range (Tab. I). This issue can be addressed by simply clipping them to their feasible range. Now, $\mathfrak{w}$ is converted to the component values that should be printed in the neuromorphic circuits, i.e., $\boldsymbol{\omega}$ (**printable values** in Fig. 5).

To match the input of surrogate models, the parameters are extended and normalized as discussed before. The resulting $\tilde{\mathfrak{w}}$ are put into the surrogate model to estimate the normalized $\boldsymbol{\eta}$, i.e., $\tilde{\boldsymbol{\eta}} = \hat{\boldsymbol{\eta}}(\tilde{\mathfrak{w}})$. Finally, the tanh-like function can be built and integrated into pNNs after denormalization.

C. Variation-aware Training of pNN

Variation-aware training refers to taking the fabrication errors of printed components into account during the training for pNNs [1]. In variation-unaware training, the loss function L depends on the training data $(\mathbf{x}, \mathbf{y})$, the surrogate conductances $\boldsymbol{\theta}$, and the parameters for nonlinear circuits $\mathfrak{w}$. Therefore, the

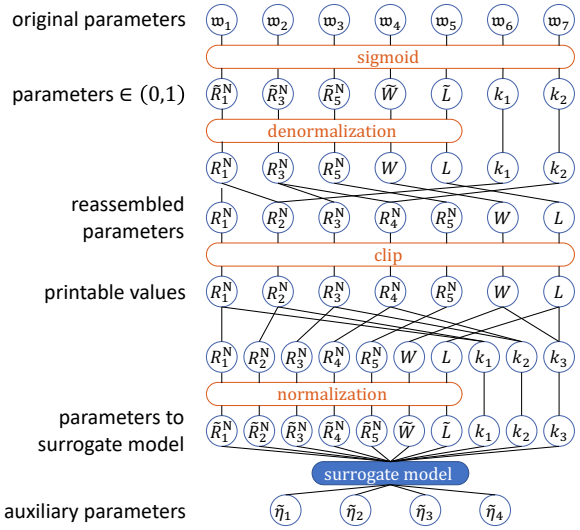

Fig. 5. Flowchart for the processing of the learanble parameter for a surrogate model of the nonlinear circuit.

loss function can be denoted as $L(\boldsymbol{\theta}, \mathfrak{w}, \mathbf{x}, \mathbf{y})$. In the following, we denote variation-unaware training as *nominal* training.

Benefiting from the differentiable surrogate models, the variation of the components in nonlinear circuits can also be considered. In variation-aware training, the parameters $\boldsymbol{\theta}$ and $\mathfrak{w}$ are no longer represented by deterministic values, but modeled as stochastic variables $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$ and $\mathfrak{w} \sim p(\mathfrak{w})$ to respect random fabrication errors. With reparameterization, they can be formed as $\varepsilon_\theta \boldsymbol{\theta}$ and $\varepsilon_\omega \mathfrak{w}$, where each element in ε_θ and ε_ω are i.i.d. variables following a uniform distribution $\mathcal{U}[1-\epsilon, 1+\epsilon]$, because the printing variation is mainly driven by printing limited resolution. Here, ϵ can be chosen to reflect the precision of printing resolution. Moreover, it is notable that, the stochastic variable ε_ω is not multiplied by the original learnable parameter, but the one refers to the component values to be printed (i.e., printable values in Fig. 5). Consequently, for variation-aware training, we minimize the expected loss value w.r.t. the random variables, i.e.,

$$\begin{aligned}\underset{\boldsymbol{\theta}, \mathfrak{w}}{\text{minimize}}\, \mathcal{L}(\boldsymbol{\theta}, \mathfrak{w}) &= \mathbb{E}_{\boldsymbol{\varepsilon}_\theta, \boldsymbol{\varepsilon}_\omega}\left\{L(\varepsilon_\theta \boldsymbol{\theta}, \varepsilon_\omega \mathfrak{w}, \mathbf{x}, \mathbf{y})\right\} \\ &= \iint L(\varepsilon_\theta \boldsymbol{\theta}, \varepsilon_\omega \mathfrak{w}, \mathbf{x}, \mathbf{y}) p(\varepsilon_\theta) p(\varepsilon_\omega) \mathrm{d}\varepsilon_\theta \mathrm{d}\varepsilon_\omega.\end{aligned}$$

Since, this integration can not be solved analytically, we estimate $\mathcal{L}$ by Monte-Carlo approximation, i.e.,

$$\mathcal{L} \approx \frac{1}{N} \sum_{\varepsilon'_\theta, \varepsilon'_\omega} L(\varepsilon'_\theta \boldsymbol{\theta}, \varepsilon'_\omega \mathfrak{w}, \mathbf{x}, \mathbf{y}), \quad \text{with} \quad \begin{aligned} \varepsilon'_\theta &\sim p(\varepsilon_\theta), \\ \varepsilon'_\omega &\sim p(\varepsilon_\omega), \end{aligned}$$

where N is the number of samples ε'_θ and ε'_ω drawn from their distributions in each epoch. Any gradient-based algorithm can be employed to solve this optimization problem, e.g., SGD [15] and Adam [16].

IV. Experiment

We implement[2] the pNNs with learnable nonlinear circuit and conduct experiments with both nominal and variation-aware training.

[2]https://github.com/Neuromophic/LearnableNonlinearCircuits

A. Experiment Setup

a) Datasets: In line with the related work [5], we utilize 13 benchmark classification datasets (see Tab. II), whose complexity matches the target application domains of PE, taking into account the large feature size and low device count of PE. The datasets are then randomly split into training (60%), validation (20%), and test (20%) sets.

b) Hyperparameters: Regarding the topology, we decide $\#input$-3-$\#output$ for pNNs, as in [5]. As optimizer, we utilize Adam with default settings. Since $\boldsymbol{\theta}$ and $\mathfrak{w}$ represent different underlying effects (i.e., weights and activation functions), we choose different learning rates for them, namely $\alpha_\theta = 0.1$ for $\boldsymbol{\theta}$ and $\alpha_\omega \in \{0, 0.005\}$ for $\mathfrak{w}$ ($\alpha_\omega = 0$ refers to a non-learnable nonlinear circuit). We use early-stopping with the patience of $5\,000$ epochs as the stop criterion for the training. Regarding the variation, we select $\epsilon \in \{0\%, 5\%, 10\%\}$ to reflect nominal training as well as low and high printing resolutions, because typical printing resolutions range from $20\,\mu\mathrm{m}$ to $100\,\mu\mathrm{m}$ [17], whereas the component feature sizes in printed neuromorphic circuits are on the order of $1\,\mathrm{mm}$ [1]. For Monte-Carlo approximation, we select $N_{\text{train}} = 20$. We run the experiment ten times with random seeds in $\{1, ..., 10\}$.

B. Ablation Study

To investigate the contribution of learnable nonlinear circuits and the variation-aware training, we conduct two ablation studies. In (a), we explore the influence of the learnable nonlinear circuit, i.e., we set $\alpha_\omega = 0$ and observe the change in results. In (b), we investigate the effect of variation-aware training, meaning that we perform both nominal training and variation-aware training with non-learnable nonlinear circuits. As baseline, neither variation-aware training nor learnable nonlinear circuits is adopted.

C. Result

After training all pNNs, we select the best pNNs in each setup w.r.t. the validation loss, as these circuits would be the ones to be printed, and evaluate them on the test sets. For pNNs from nominal training, we test them with $\epsilon = 5\%$ and $\epsilon = 10\%$ respectively. As for pNNs from variation-aware training, we test with variation according to the respective training ε. All pNNs are tested with $N_{\text{test}} = 100$ Monte-Carlo samples. The mean and standard deviation from N_{test} samples are calculated and reported in Tab. II.

To demonstrate the results from different experiment setups more intuitively, we average the accuracies and standard deviations over all datasets to a scalar for each experiment setup. The averaged values are reported in both Tab. II and Tab. III.

D. Discussion

By comparing the baseline with pNNs with learnable nonlinear circuit from variation-aware training, significant improvement becomes evident: The (mean) accuracy has been increased by 19% and 26% with respect to 5% and 10% variation, while the robustness (standard variation) has been improved (reduced) by 73% and 75%, respectively. By observing the learnable nonlinear circuit and the variation-aware training

TABLE II
RESULT OF THE EXPERIMENT ON 13 BENCHMARK DATASETS

Dataset	Non-learnable nonlinear circuit				Learnable nonlinear circuit			
	Nominal training		Variation-aware training		Nominal training		Variation-aware training	
	5%	10%	5%	10%	5%	10%	5%	10%
Acute Inflammation	0.821 ± 0.083	0.800 ± 0.085	0.891 ± 0.078	0.874 ± 0.044	0.956 ± 0.089	0.882 ± 0.111	1.000 ± 0.000	0.999 ± 0.012
Balance Scale	0.778 ± 0.095	0.682 ± 0.203	0.810 ± 0.040	0.705 ± 0.066	0.870 ± 0.050	0.824 ± 0.108	0.880 ± 0.004	0.877 ± 0.008
Breast Cancer Wisconsin	0.907 ± 0.031	0.864 ± 0.097	0.924 ± 0.018	0.878 ± 0.149	0.951 ± 0.035	0.880 ± 0.136	0.963 ± 0.008	0.931 ± 0.039
Cardiotocography	0.768 ± 0.017	0.748 ± 0.051	0.775 ± 0.014	0.768 ± 0.009	0.773 ± 0.088	0.756 ± 0.198	0.774 ± 0.004	0.763 ± 0.002
Energy Efficiency (y_1)	0.816 ± 0.080	0.709 ± 0.178	0.838 ± 0.000	0.835 ± 0.015	0.889 ± 0.044	0.835 ± 0.148	0.889 ± 0.032	0.847 ± 0.012
Energy Efficiency (y_2)	0.663 ± 0.090	0.615 ± 0.142	0.727 ± 0.054	0.714 ± 0.073	0.837 ± 0.079	0.772 ± 0.146	0.883 ± 0.023	0.867 ± 0.026
Iris	0.706 ± 0.092	0.659 ± 0.104	0.807 ± 0.103	0.733 ± 0.140	0.726 ± 0.141	0.657 ± 0.181	0.912 ± 0.034	0.843 ± 0.045
Mammographic Mass	0.613 ± 0.100	0.569 ± 0.116	0.689 ± 0.125	0.642 ± 0.136	0.630 ± 0.134	0.609 ± 0.120	0.782 ± 0.017	0.766 ± 0.053
Pendigits	0.313 ± 0.155	0.234 ± 0.145	0.365 ± 0.063	0.253 ± 0.085	0.391 ± 0.131	0.331 ± 0.137	0.554 ± 0.038	0.548 ± 0.047
Seeds	0.520 ± 0.158	0.413 ± 0.191	0.766 ± 0.043	0.717 ± 0.132	0.749 ± 0.039	0.656 ± 0.140	0.820 ± 0.034	0.820 ± 0.041
Tic-Tac-Toe Endgame	0.630 ± 0.000	0.630 ± 0.000	0.633 ± 0.009	0.634 ± 0.020	0.682 ± 0.209	0.630 ± 0.000	0.713 ± 0.012	0.660 ± 0.017
Vertebral Column (2 cl.)	0.681 ± 0.071	0.650 ± 0.089	0.682 ± 0.070	0.666 ± 0.049	0.706 ± 0.069	0.659 ± 0.107	0.716 ± 0.007	0.661 ± 0.000
Vertebral Column (3 cl.)	0.593 ± 0.137	0.570 ± 0.130	0.602 ± 0.071	0.570 ± 0.118	0.612 ± 0.132	0.574 ± 0.157	0.634 ± 0.086	0.634 ± 0.075
Average	**0.678 ± 0.085**	**0.626 ± 0.118**	**0.731 ± 0.053**	**0.691 ± 0.080**	**0.752 ± 0.095**	**0.697 ± 0.130**	**0.809 ± 0.023**	**0.786 ± 0.029**

separately in Tab. III, we conclude that learnable nonlinear circuit and variation-aware training contribute 58% and 42% to the improvement of the accuracy under 5% variation, whereas 52% and 48% in case of 10% variance. Regarding the robustness, almost all the contribution is provided by variation-aware training.

In short, depending on the variations (5%-10%), the proposed method in this work provides 19%-26% improvement of pNNs regarding mean accuracy and around 75% improvement regarding robustness. In this method, the learnable nonlinear circuit contributes more to the resulting improvement in accuracy, while the improvement of robustness is provided mainly by variation-aware training.

TABLE III
SUMMARIZED RESULTS FROM ABLATION STUDY

Learnable nonlinear circuit	Variation-aware training	ϵ_{test}	
		5%	10%
✓	✓	0.809 ± 0.023	0.786 ± 0.029
✓	✗	0.752 ± 0.095	0.697 ± 0.130
✗	✓	0.731 ± 0.053	0.691 ± 0.080
✗	✗	0.678 ± 0.085	0.626 ± 0.118

V. CONCLUSION

Printed analog neuromorphic circuits have become increasingly attractive, especially in emerging fields like IoT and wearable computing. This is due to their unique characteristics, such as flexibility, lightweight, and inexpensiveness, which are unmatchable by the traditional silicon-based chips.

To leverage another unique feature of PE, i.e., highly customized additive manufacturing, this work focuses on the bespoke nonlinear circuits in the printed neuromorphic circuits. Additionally, the printing variations of the components in both crossbars and nonlinear circuits are considered by introducing the variation-aware training for pNNs, which aims to increase the robustness of the printed neuromorphic circuits.

The preliminary experiment proved that, by introducing these two approaches, the accuracy and robustness of the pNNs were significantly improved. Ablation study reveals that, the learnable nonlinear circuit provides a unique contribution to the final improvement in accuracy, while the robustness is mainly provided by the variation-aware training.

REFERENCES

[1] D. D. Weller *et al.*, "Realization and Training of an Inverter-based Printed Neuromorphic Computing System," *Scientific reports*, vol. 11, no. 1, pp. 1–13, 2021.

[2] I. I. Labiano *et al.*, "Flexible Inkjet-printed Graphene Antenna on Kapton," *Flexible and Printed Electronics*, vol. 6, no. 2, p. 025010, 2021.

[3] J. Chang *et al.*, "Challenges of Printed Electronics on Flexible Substrates," in *2012 IEEE 55th international midwest symposium on circuits and systems (MWSCAS)*. IEEE, 2012, pp. 582–585.

[4] Z. Yu *et al.*, "An Overview of Neuromorphic Computing for Artificial Intelligence Enabled Hardware-based Hopfield Neural Network," *IEEE Access*, vol. 8, pp. 67 085–67 099, 2020.

[5] H. Zhao *et al.*, "Aging-Aware Training for Printed Neuromorphic Circuits," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22)*, 2022.

[6] S. Zhang *et al.*, "Aging-aware Lifetime Enhancement for Memristor-based Neuromorphic Computing," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1751–1756.

[7] Z. Cui, *Printed Electronics: Materials, Technologies and Applications*. John Wiley & Sons, 2016.

[8] S. Conti *et al.*, "Low-Voltage 2D Materials-Based Printed Field-Effect Transistors for Integrated Digital and Analog Electronics on Paper," *Nature communications*, vol. 11, p. 3566, 2020.

[9] B. Huber *et al.*, "Fully Inkjet Printed Flexible Resistive Memory," *Applied Physics Letters*, vol. 110, p. 143503, 2017.

[10] M. Kondo *et al.*, "Design of Ultraflexible Organic Differential Amplifier Circuits for Wearable Sensor Technologies," in *IEEE Int. Conf. on Microelectronic Test Structures (ICMTS)*, 2018, p. 79.

[11] C. D. Schuman *et al.*, "A Survey of Neuromorphic Computing and Neural Networks in Hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[12] F. Rasheed *et al.*, "Variability Modeling for Printed Inorganic Electrolyte-gated Transistors and Circuits," *IEEE transactions on electron devices*, vol. 66, no. 1, pp. 146–152, 2018.

[13] Y. Bengio *et al.*, "Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation," *arXiv preprint arXiv:1308.3432*, 2013.

[14] I. Soboĺ, "Quasi-Monte Carlo Methods," *Progress in Nuclear Energy*, vol. 24, no. 1-3, pp. 55–61, 1990.

[15] S.-i. Amari, "Backpropagation and Stochastic Gradient Descent Method," *Neurocomputing*, vol. 5, no. 4-5, pp. 185–196, 1993.

[16] D. P. Kingma *et al.*, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] S. Khan *et al.*, "Technologies for Printing Sensors and Electronics over Large Flexible Substrates: A Review," *IEEE Sensors Journal*, vol. 15, p. 3164, 2014.