Formally Verifying Analog Neural Networks Under Process Variations Using Polynomial Zonotopes

Yasmine Abu-Haeyeh[†1], Tobias Ladner[†2], Matthias Althoff[2], Lars Hedrich[1]
[1] Goethe University Frankfurt, Germany
{abu-haeyeh,hedrich}@em.uni-frankfurt.de
[2] Technical University of Munich, Germany
{tobias.ladner,althoff}@tum.de

***Abstract*—Analog neural networks are gaining attention due to their efficiency in terms of power consumption and processing speed. However, since analog neural networks are implemented as physical circuits, they are highly sensitive to manufacturing process variations, which can cause large deviations from the nominal model. We present a polynomial-based model that resembles the performance of the neuron circuit under process variations. This model is formally verified via reachability analysis using polynomial zonotopes, thus avoiding conventional, time-consuming Monte Carlo simulations. We evaluate our proposed verification approach on three different datasets and on fully-connected and convolutional analog neural networks. Our experimental results confirm the effectiveness of our verification approach by reducing the verification time from up to a day to seconds while enclosing up to 99% of the variation samples.**

***Index Terms*—Analog neural networks, formal verification, set-based computing, energy-efficient computing, AI hardware.**

I. Introduction

The growing demand for artificial neural networks has driven the need for low-power hardware solutions, such as FPGA accelerators, in-memory computing, and CMOS-based inference circuits [1, 2]. This facilitates the deployment of neural networks in Internet of Things (IoT) devices, such as sensors, mobile devices, and wearable technology, where power consumption is a critical concern [3–5]. Analog inference hardware demonstrates efficiency in terms of processing speed, energy, and chip area. However, the design and verification of analog devices are challenging due to their vulnerability to perturbations such as noise, mismatch, and process variations [6–8]. The functionality of analog neural networks must be validated under these variations to ensure safety, which is usually performed by running time-consuming Monte Carlo simulations [9]. In this work, we formally verify the characteristics of analog neural networks influenced by varying process parameters.

A. Related Work

Modeling and analysis of the influence of varying process parameters on analog circuits have been investigated extensively [10–13]. However, recent advances in the design and implementation of analog neural networks have opened up new challenges that need to be further addressed: During semiconductor device fabrication, small variations—such as gate width, threshold voltage, channel length, and oxide thickness—naturally occur [14]. It is well known that small variations can significantly impact the accuracy of conventionally implemented neural networks [15], and similar effects have also been observed for process variations on analog neural networks [7, 8, 16, 17]. Hence, modeling the behavior of analog inference circuits under these variations is crucial to ensure the safety of these circuits and improve their design and robustness [18]. The emerging modeling methodologies either rely on device physics or machine learning engines [19]. Since developing accurate physics-based models is a challenging and time-consuming task, many works utilize machine learning for modeling analog circuits under process variations [12]. While such models provide the capability of identifying the sources of variability in the circuit, they are usually single-transistor models, which are insufficient to reflect the full system behavior and therefore cannot be extended to a system-level modeling approach. Other available analog neural network implementations employ Monte Carlo simulations to analyze the behavior of specific circuits under process variations [9, 20]. However, the simulations have to be rerun for each network architecture, which can take days as up to $100,000$ transistors have to be simulated.

Recently, it was shown that system-level verification of analog neural networks under device mismatch can be realized using formal neural network verification [18]. Neural network verification can provide formal guarantees of the prediction of a neural network under perturbations, where usually conventionally implemented neural networks are considered [21, 22]. These guarantees are obtained by rigorous mathematical proofs that enable reasoning about the safety of a neural network under perturbation [23]. Unfortunately, these proofs are usually difficult to obtain for large neural networks as it has been shown that verifying certain properties is NP-hard [24]. Thus, modern verifiers introduce relaxations to remain computationally feasible, where both optimization-based approaches [25–28] and approaches based on reachability analysis [29–32] are used.

B. Contributions

To summarize, our contributions are:

- An optimized polynomial-based parametric model for the analog neuron performance under process variations.
- The system-level verification of the model using reachability analysis.
- The scalability is demonstrated on various network architectures, including fully-connected and convolutional

[†] Equal contribution, sorted alphabetically.

979-8-3195-3997-7/26/$31.00 ©2026 IEEE

networks, and on benchmark classification tasks such as MNIST.

II. Preliminaries

A. Notation

We denote scalars and vectors by lowercase letters, matrices by uppercase letters, and sets by calligraphic letters. The i-th element of a vector $v \in \mathbb{R}^n$ is written as $v_{(i)}$. The element in the i-th row and j-th column of a matrix $A \in \mathbb{R}^{n \times m}$ is written as $A_{(i,j)}$, the entire i-th row and j-th column are written as $A_{(i,\cdot)}$ and $A_{(\cdot,j)}$, respectively. This is analogously used for tensors. Given $n \in \mathbb{N}$, we use the shorthand notation $[n] = \{1, \dots, n\}$. Let $\mathcal{S} \subset \mathbb{R}^n$ be a set and $f \colon \mathbb{R}^n \to \mathbb{R}^m$ be a function, then $f(\mathcal{S}) = \{f(x) \mid x \in \mathcal{S}\}$. An interval with bounds $a, b \in \mathbb{R}^n$ is denoted by $[a, b]$, where $a \leq b$ holds element-wise.

B. Neural Networks

We initially consider fully-connected neural networks with rectified linear unit (ReLU) activation, and later extend our approach to convolutional neural networks [33, Sec. 5.5.6].

Definition 1 (ReLU Neural Network [33, Sec. 5.1])**.** *Given an input $x \in \mathbb{R}^{n_0}$ and an output $y \in \mathbb{R}^{n_\kappa}$, a neural network $y = \Phi(x)$ with $\kappa \in \mathbb{N}$ layers can be formulated as*

$$\begin{aligned} h_0 &= x, \\ h_k &= L_k\left(h_{k-1}\right) = \mathrm{ReLU}(W_k h_{k-1} + b_k), \quad k \in [\kappa], \\ y &= h_\kappa, \end{aligned}$$

with outputs of hidden layers $h_k \in \mathbb{R}^{n_k}$, layer $L_k \colon \mathbb{R}^{n_{k-1}} \to \mathbb{R}^{n_k}$, weights $W_k \in \mathbb{R}^{n_k \times n_{k-1}}$, and bias $b_k \in \mathbb{R}^{n_k}$.

C. Set-Based Computing

We bound variations using continuous sets. To efficiently represent them in high-dimensional spaces, we use (matrix) polynomial zonotopes [34, 35] as a set representation.

Definition 2 (Matrix Polynomial Zonotope [35, Sec. 3])**.** *Given an offset $C \in \mathbb{R}^{n \times m}$, dependent generators $G \in \mathbb{R}^{n \times m \times h}$, independent generators $G_I \in \mathbb{R}^{n \times m \times q}$, and an exponent matrix $E \in \mathbb{N}_0^{p \times h}$, a matrix polynomial zonotope $\mathcal{PZ} = \langle C, G, G_I, E \rangle_{PZ} \subset \mathbb{R}^{n \times m}$ is defined as*

$$\mathcal{PZ} := \left\{ C + \sum_{i=1}^{h} \left(\prod_{k=1}^{p} \alpha_k^{E_{(k,i)}} \right) G_{(\cdot,\cdot,i)} + \sum_{j=1}^{q} \beta_j G_{I(\cdot,\cdot,j)} \;\middle|\; \alpha_k, \beta_j \in [-1, 1] \right\}.$$

Many of the required operations to verify analog neural networks can be computed efficiently using polynomial zonotopes. We discuss them only briefly here due to space limitations, and refer interested readers to the example in [35, Appendix A]. Given a matrix polynomial zonotope $\mathcal{PZ}_1 = \langle C_1, G_1, G_{I,1}, E_1 \rangle_{PZ}$, a matrix $A \in \mathbb{R}^{k \times n}$, and a matrix $B \in \mathbb{R}^{k \times m}$, an affine map [35, Eq. 6a] is computed by

$$\begin{aligned} A\mathcal{PZ}_1 + B &= \{AX_1 + B \mid X_1 \in \mathcal{PZ}_1\} \\ &= \langle AC_1 + B, AG_1, AG_{I,1}, E_1 \rangle_{PZ} \,. \end{aligned} \tag{1}$$

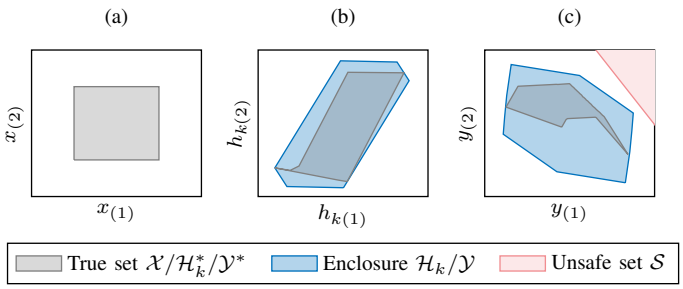

Fig. 1: Verification of neural networks: (a) input to neural network, (b) output of hidden layer, and (c) output of neural network. As the enclosure does not intersect with the unsafe set, we can also conclude that the unobserved true output set does not intersect with the unsafe set (5).

Given a second matrix polynomial zonotope $\mathcal{PZ}_2$ with matching dimensions, one can also efficiently compute their sum [35, Eq. 5]

$$\mathcal{PZ}_1 \oplus \mathcal{PZ}_2 = \{X_1 + X_2 \mid X_1 \in \mathcal{PZ}_1,\, X_2 \in \mathcal{PZ}_2\}\,, \tag{2}$$

and their multiplication [35, Lemma 1]

$$\mathcal{PZ}_1 \mathcal{PZ}_2 = \{X_1 X_2 \mid X_1 \in \mathcal{PZ}_1,\, X_2 \in \mathcal{PZ}_2\}\,. \tag{3}$$

D. Formal Verification of Neural Networks

We enclose all possible outputs of a neural network for a given input set $\mathcal{X} \subset \mathbb{R}^{n_0}$ by

$$\begin{aligned} \mathcal{H}_0^* &= \mathcal{X}, \\ \mathcal{H}_k^* &= L_k\left(\mathcal{H}_{k-1}^*\right), \quad k \in [\kappa], \\ \mathcal{Y}^* &= \mathcal{H}_\kappa^*. \end{aligned} \tag{4}$$

It is generally computationally infeasible to obtain these true sets (indicated by $\square^*$) [24] so that the output of each layer is enclosed:

Proposition 1 (Neural Network Enclosure [30, Sec. 3])**.** *Given an input set $\mathcal{X} \subset \mathbb{R}^{n_0}$ to a neural network Φ, we obtain an enclosure of the output set $\mathcal{Y} = \texttt{enclose}\left(\Phi, \mathcal{X}\right) \supseteq \mathcal{Y}^*$ by iteratively enclosing the output of each layer:*

$$\mathcal{H}_k = \texttt{enclose}\left(L_k, \mathcal{H}_{k-1}\right) \supseteq \mathcal{H}_k^*, \quad k \in [\kappa],$$

with $\mathcal{H}_0 = \mathcal{X}$ and $\mathcal{Y} = \mathcal{H}_\kappa$.

We utilize the enclosure $\mathcal{Y} \supseteq \mathcal{Y}^*$ of the output set to verify that unsafe outputs are impossible:

$$\mathcal{Y} \cap \mathcal{S} = \emptyset \implies \mathcal{Y}^* \cap \mathcal{S} = \emptyset, \tag{5}$$

where $\mathcal{S} \subset \mathbb{R}^{n_\kappa}$ defines a set of unsafe outputs. Many common verification tasks such as image classification can be formulated this way.

An example of this procedure is shown in Fig. 1. How tightly the obtained enclosures fit the true output sets heavily depends on the chosen set representation. It has been shown that enclosures using (matrix) polynomial zonotopes work very well in practice as they obtain tight enclosures and are fast to compute [30, 32, 36].

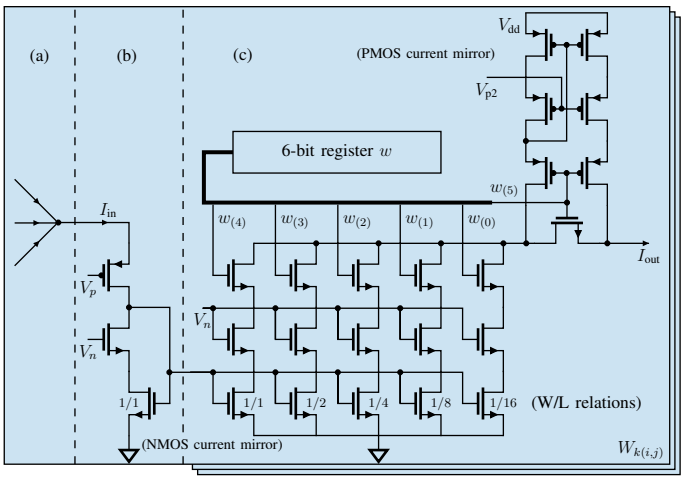

Fig. 2: Schematic of the analog neuron circuit: (a) current summation block, (b) ReLU activation function, and (c) a cell for each weight $W_{k(i,j)}$. The programmable weight bits $w_{(0:5)}$ are stored in a 6-bit register. The bit $w_{(5)}$ is used to realize negative weights by inverting the output current.

III. Analog Neural Networks Under Varying Process Parameters

A. Schematic of the Analog Neuron Circuit

The schematic view of the analog neuron used in this work is illustrated in Fig. 2, which represents the main building block of our 130 nm, 1.2 V CMOS inference circuit [37]. The neuron circuit functionality can be broken down into three main parts: (a) summation of input weighted currents with the bias current using Kirchhoff's laws, (b) a ReLU activation function implemented using the left side of the current mirror to rectify negative currents, and (c) the right side of the current mirror, which is utilized to scale the output current using a 5-bit programmable weight $w_{(0:4)}$ and an additional bit $w_{(5)}$ to store the sign of the weight using another upper PMOS current mirror. The neuron circuit accommodates an additional bias circuit that maps an 8-bit programmable bias to an analog output current. Neural network training is carried out off-chip using TensorFlow, followed by quantization to produce 6-bit quantized weights in the range of $[-2, 2]$.

Please note that in this circuit design, ReLU is applied before the weight multiplication. This design is chosen to minimize energy usage.

However, neural networks typically start with a linear layer and do not apply ReLU on the input. Thus, to maintain the inference circuit functionality for negative inputs, the circuit schematic in Fig. 2 is adapted to not apply a ReLU activation in the first layer and pass the negative input through instead. Additionally, an output layer with a weight of 1 is appended to apply the final ReLU activation at the network output if required by the nominal model.

B. Performance Analysis

Let us first analyze the performance of the circuit under varying process parameters to establish an accurate model of the circuit. In Fig. 3, we visualize Monte Carlo simulations of the neuron netlist under process variations. Fig. 3a shows an irregular variation distribution contrary to the normal distribution observed for device mismatch [18, Fig. 4]. We also observe that the distribution varies for each of the 64 quantized weight values. This is expected because different bits of the register activate different transistor groups. Additionally, the circuit behavior deviates depending on the sign of the input to the neuron. All the aforementioned challenging aspects lead to the need for deriving a unique parametric model for each of the 64 weights.

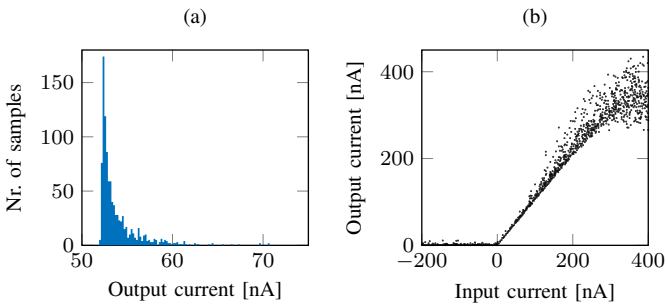

Fig. 3: Monte Carlo analysis of the neuron netlist under process variations: (a) parameter variations in a single neuron for an input current of 50 [nA], (b) output of the neuron impacted by all process parameters.

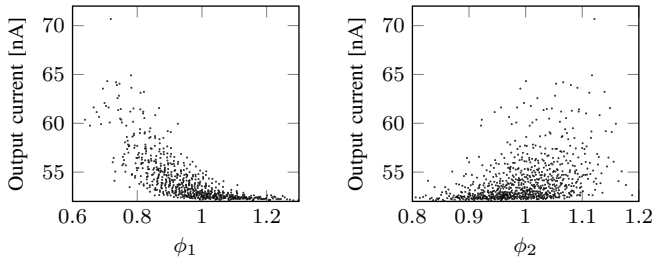

Fig. 4: Monte Carlo analysis of the output current of the neuron netlist versus the two dominant parameters: (a) ϕ_1 parameter variations, and (b) ϕ_2 parameter variations, where ϕ_1, ϕ_2 are sampled from a normal distribution.

C. Sensitivity Modeling

To identify the impact of varying process parameters on our analog neuron circuit, we run a sensitivity analysis with an underlying real process based on the PSP transistor model. As shown in Fig. 2, the neuron circuit contains both NMOS and PMOS transistors operating in the subthreshold region, which are subject to more than 50 model parameters.

Our sensitivity analysis runs Monte Carlo sampling with process variations on the neuron netlist and analyzes the variance contributions to the output current from all process parameters. Afterward, we choose the parameters with the highest contribution as the dominant parameters. This step has to be done only once per neuron netlist and is reused across all network architectures built from it. For our neuron netlist (Fig. 2), the behavior can be well described using two dominant parameters, ϕ_1 and ϕ_2, which represent two parameters influencing the surface potential of the NMOS transistor, affecting the threshold voltage and the overall transistor behavior. The strong influence of these parameters on the output current is shown in Fig. 4. While ϕ_1, ϕ_2 greatly influence NMOS transistors, they only have a negligible effect on PMOS transistors since the current mirror topology mitigates the impact of surface potential variations.

IV. Bounding Process Variations Using Set-Based Computing

In this section, we first establish a model describing the effect of process variations on the output of an analog neural network, followed by a description of how these variations can be bounded using set-based computing.

A. Modeling Process Variations

We model the uncertainty of the dominant parameters ϕ_1 and ϕ_2 around their nominal value of 1:

$$\begin{aligned} \phi_1 &\in \mathcal{D}_1 \subset \mathbb{R}, \\ \phi_2 &\in \mathcal{D}_2 \subset \mathbb{R}, \end{aligned} \tag{6}$$

where the domains $\mathcal{D}_1, \mathcal{D}_2$ are chosen to enclose a desired number of Monte Carlo simulations (Fig. 4). In particular, the process parameters ϕ_1, ϕ_2 are sampled from a normal distribution with standard deviations σ_1 and σ_2, respectively, and we choose, e.g., $\mathcal{D}_1 = [1-3\sigma_1, 1+3\sigma_1]$ to enclose $\approx 99\%$ of the simulations. The subtle variation produced by the rest of the varying process parameters is captured with a single third noise symbol $\phi_3 \in \mathcal{D}_3 \subset \mathbb{R}$. By choosing the domains large enough such that all measurements are contained, we say that the model is reachset conformant [38].

With these variations modeled, we can describe a neural network with respect to ϕ_1, ϕ_2 and account for all error terms. To distinguish between the analog implementation and the nominal neural network (Def. 1), we decorate the respective variables of the analog implementation with a tilde: $\widetilde{h}_k \in \mathbb{R}^{n_k}$, $\widetilde{W}_k \in \mathbb{R}^{n_k \times n_{k-1}}$, $\widetilde{b}_k \in \mathbb{R}^{n_k}$, $k \in [\kappa]$. The weights $\widetilde{W}_k$ are influenced polynomially by ϕ_1 and ϕ_2, and we obtain the respective coefficients $a_{k,l,m}, b_{k,l,m} \in \mathbb{R}$ of each of the 64 quantized weights of the nominal model individually using polynomial regression. Each entry $\widetilde{w}_k$ of $\widetilde{W}_k$ is modeled as:

$$\widetilde{w}_k = \sum_{l=0}^{3} \sum_{m=0}^{3} a_{k,l,m} {\phi_1}^l {\phi_2}^m + \phi_3, \quad l + m \leq 3. \tag{7}$$

The influence of ϕ_1, ϕ_2 on the bias $\widetilde{b}_k$ is negligible according to the polynomial regression. Moreover, for negative inputs to ReLU activations, the output of the respective neuron is set to:

$$\widetilde{h}_k = \sum_{l=0}^{3} \sum_{m=0}^{3} b_{k,l,m} {\phi_1}^l {\phi_2}^m, \quad l + m \leq 3, \tag{8}$$

to account for the rectified output. As described in Sec. III-A, we deploy a slightly different circuit for the first layer to deal with negative inputs; thus, the coefficients for the respective weights also differ slightly from the ones in all other layers. The polynomial model is both simple and scalable, as determining the coefficients for each weight only needs to be done once and can be reused for all architectures implemented using the same netlist (Fig. 2).

We also analyze the influence of process variations on convolutional neural networks [33, Sec. 5.5.6], and we find that our model also generalizes to convolutional neural networks and only requires updating the coefficients $a_{k,l,m}$, $b_{k,l,m}$. Please note that, as a convolution is applied—which has far fewer weights than a matrix multiplication—the overall variation is typically much smaller for convolutional neural networks. Thus, convolutional neural networks are more robust against process variations.

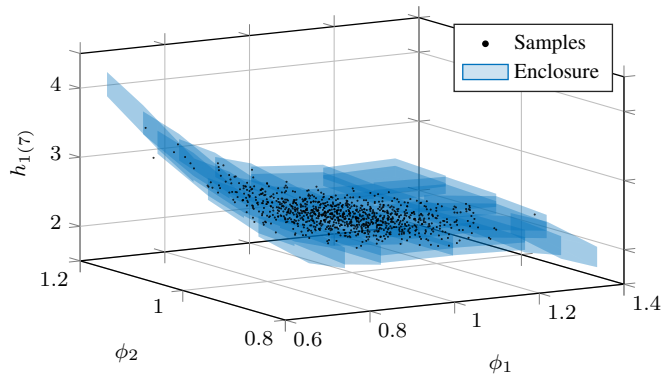

Fig. 5: Polynomial zonotope enclosure of neuron 7 of the first hidden layer—having the maximal value according to the nominal model—with respect to the main process parameters ϕ_1, ϕ_2.

B. Bounding Process Variations

Process variations can be bounded by following the general verification procedure described in Sec. II-D. We start the verification by propagating the input set $\widetilde{\mathcal{H}}_0 = \mathcal{X}$ through the first layer. In contrast to the general procedure, the weights $\widetilde{W}_1$ of linear layers (7) are no longer matrices but sets $\widetilde{\mathcal{W}}_1$ due to the uncertainties introduced in (6). Our polynomial model (7) enables us to represent $\widetilde{\mathcal{W}}_1$ as polynomial zonotopes, where the noise symbols ϕ_1, ϕ_2, ϕ_3 become (scaled) dependent factors α_1, α_2, α_3, as introduced in Def. 2.

Given $\widetilde{\mathcal{H}}_0$ and $\widetilde{\mathcal{W}}_1$, their multiplication is computed using (3). An example enclosure of the first linear layer is visualized in Fig. 5, where the enclosure is shown with respect to the main process parameters ϕ_1, ϕ_2. As these two parameters describe the behavior well, i.e., low variance for each (ϕ_1, ϕ_2)-pair, the enclosure barely has to be enlarged in the z-direction to enclose the sampled simulations. This confirms our analysis that the remaining process variations barely influence the final outcome. Please note that the enclosure is represented using a single set $\widetilde{\mathcal{H}}_1$ and is only split for visualization purposes.

As we proceed to propagate the set through the entire network, we repeatedly multiply $\widetilde{\mathcal{W}}_k$ with the current set $\widetilde{\mathcal{H}}_{k-1}$, $k \in [\kappa]$, and enclose the ReLU nonlinearities using standard techniques (Prop. 1), until we obtain an enclosure of the output set $\widetilde{\mathcal{Y}}$ to verify given specifications (5).

V. Experimental Results

To evaluate our modeling approach, we use three different classification datasets: Breast Cancer [39], Iris [40], and MNIST [41]. We simulate our analog inference circuit and run Monte Carlo simulations in Cadence Spectre [42] on a workstation with dual Intel Xeon E5-2683 v4 CPUs (16 cores each). We run the set-based modeling approach implemented in MATLAB using the CORA toolbox [43, 44] on a home computer with an Intel Core i7-1165G7 CPU. To evaluate the effectiveness of our modeling approach for each dataset, we compare it with $1,000$ Monte Carlo simulations with inputs taken from the test set. If not stated otherwise, all figures and tables are with respect to a 3σ confidence set (6).

A. Main Results

We report the required time to analyze the circuit under process variations in Tab. I. While the Monte Carlo simulations

TABLE I: Time comparison and average enclosure of Monte Carlo simulations of our approach per input pattern.

Dataset	#Classes	Approach	Time [s]	Enclosure
Breast Cancer	2	Cadence	5,479.72	–
		Ours	9.50	99.08%
Iris	3	Cadence	865.64	–
		Ours	1.15	98.01%
MNIST	10	Cadence	29,592.60	–
		Ours	1.77	97.77%
MNIST CNN	10	Cadence	84,409.00	–
		Ours	3.37	99.51%

TABLE II: Nominal and verified accuracy under process variations on the input patterns of all test datasets with varying desired enclosure of the Monte Carlo simulations via the standard deviations σ (6).

Dataset	#Patterns	Nominal	1σ	2σ	3σ
Breast Cancer	100	98%	96%	95%	27%
Iris	75	92%	88%	83%	48%
MNIST	100	87%	78%	71%	59%
MNIST CNN	100	93%	81%	64%	28%

require hours for the Breast Cancer examples and even up to a day for the MNIST CNN example, our approach runs in a few seconds. It is worth mentioning that we require a setup time of 12 minutes to extract the parametric model (7) for the netlist (Fig. 2) by running Monte Carlo simulations on a single neuron for 64 different weight values for positive and then negative input individually. However, this is only required once and can be reused for all architectures.

As shown in Tab. I, we are able to enclose on average around 99% of the output variation samples, as expected from the enclosed number of Monte Carlo samples using a 3σ confidence set (6). While the considered models are relatively small, we want to stress that this is not a limitation of our verification approach [21]. Formal verification scales to much larger networks [21]; however, the perturbations occurring in the analog implementation become too large to make useful predictions. For example, the Breast Cancer neural network only consists of three layers, and we have already observed large perturbations on the output for such sizes.

However, our verification approach can drastically reduce the design cycle of these larger models by avoiding the time-consuming Monte Carlo simulations. Using Tab. II, we can quickly assess the expected performance of the manufactured circuits. For example, circuits deviating up to 2σ retain a verified accuracy competitive with the accuracy of the nominal model, meaning that $\approx 95\%$ of the circuits can be used in production and only $\approx 5\%$ may be faulty. Moreover, the results highlight the need for a more robust circuit design to obtain fewer faults (3σ), and any new circuit design can quickly be validated as demonstrated in Tab. I.

B. Additional Experiments and Ablation Studies

1) Computing Enclosures and Verified Accuracies: In this experiment, we detail the training process and results for each dataset.

a) Diagnostic Breast Cancer Dataset: We trained the Diagnostic Breast Cancer Dataset in TensorFlow using a three-layer feedforward neural network of 30 inputs, a fully-connected layer with 8 hidden neurons, followed by another

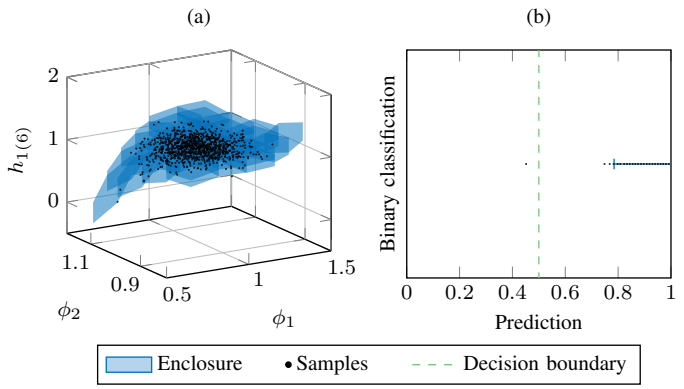

Fig. 6: Enclosure of breast cancer dataset: (a) enclosure of neuron 6 of the first hidden layer—having the maximal value according to the nominal model—with respect to the two dominant process parameters, and (b) enclosure of the output layer Monte Carlo samples.

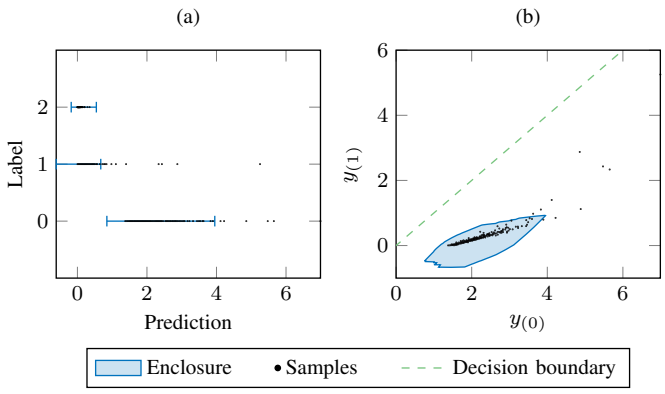

Fig. 7: Enclosure of Iris dataset: (a) projection of all dimensions, and (b) enclosure of the two most significant dimensions.

fully-connected layer with 16 hidden neurons, and a single output neuron for binary classification. An example enclosure with respect to the two dominant parameters ϕ_1 and ϕ_2 after the first hidden layer is depicted in Fig. 6a, and the enclosure of the entire network is depicted in Fig. 6b. We enclose 99.8% of the Monte Carlo samples in the output space and successfully verify the output simulations with label "1" as the lower bound of our obtained enclosure is above the classification threshold of 0.5. Doing so for all input patterns yields the verified accuracy.

b) Iris Dataset: The Iris classification network has 4 inputs, a fully-connected layer with 8 hidden neurons, and an output layer with 3 neurons representing the three classes. Fig. 7a shows the ranges of the individual outputs representing each class. The bounds of the computed output enclosure are shown as intervals for each class. As the bounds for the correct label "0" do not overlap with the other classes, we have verified the example [32, Prop. B.2]. Additionally, in Fig. 7b, we showcase the enclosure of the two largest dimensions, where we enclose 99% of the samples, again showing that our enclosure does not cross the decision boundary between the two classes.

c) MNIST Dataset: For the MNIST handwritten digit recognition benchmark, we implemented two different neural network architectures: a fully-connected network and a convolutional neural network. We reduced the network input size to 14×14 to decrease the circuit complexity and energy consumption. The fully-connected network consists

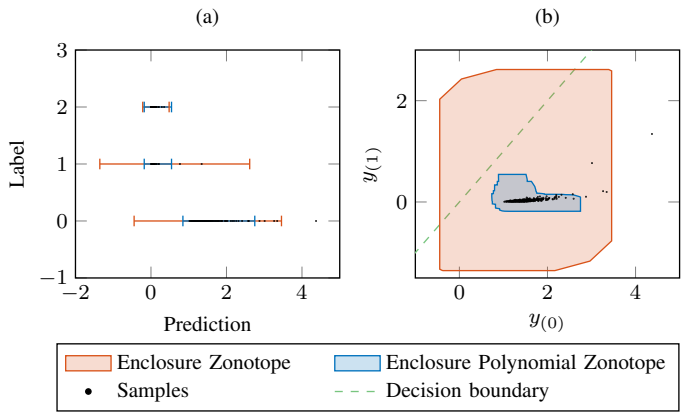

Fig. 10: Enclosure comparison using zonotopes (red) and polynomial zonotopes (ours, blue): (a) projection of all dimensions, and (b) enclosure of the two most significant dimensions.

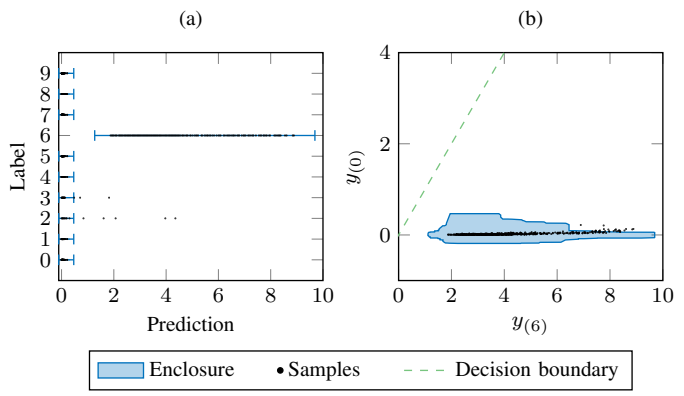

Fig. 8: Enclosure of the fully-connected network on MNIST: (a) projection of all dimensions, and (b) enclosure of the two most significant dimensions.

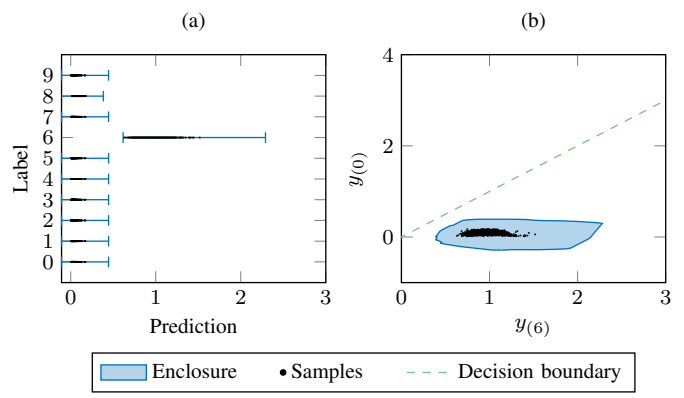

Fig. 9: Enclosure of the convolutional network on MNIST: (a) projection of all dimensions, and (b) enclosure of the two most significant dimensions.

of a fully-connected layer with 10 hidden neurons followed by an output layer with 10 neurons representing the digit classes. Furthermore, the convolutional network consists of a convolutional layer with a single filter of size 7×7, followed by a fully-connected layer of 64 neurons, and finally an output layer with 10 outputs. As shown in Fig. 8a, the output of the fully-connected network shows a high sensitivity to process variations. Nevertheless, the polynomial zonotope is able to verify the specification while enclosing 99.1% of the samples. The convolutional neural network in Fig. 9a exhibits much less sensitivity (note scaling of x-axis), where our model again verifies the specification and successfully encloses all the samples. In Fig. 8b and Fig. 9b, we again confirm the aforementioned results by showing the enclosure of the most significant outputs for each architecture.

2) Comparison to Non-Polynomial Modeling: Finally, we demonstrate that it is indeed necessary to have a polynomial model and that just using linear relaxations is not sufficient. To do so, we adapt the previous approach on mismatch variations [18] based on zonotopes [45] and apply it to process variations. In contrast to polynomial zonotopes (Def. 2), zonotopes can only represent convex sets as they only have independent generators G_I (see Def. 2), and thus cannot capture the higher-order polynomial dependencies we observed (Fig. 5). For this reason, we need to limit the highest degree in (8) to 1 for the enclosure based on zonotopes, resulting in much larger enclosures in each layer for zonotopes than for polynomial zonotopes. An example from the Iris dataset is shown in Fig. 10, where we are again able to verify the specifications using polynomial zonotopes, but are unable to do so using zonotopes, as the computed enclosure crosses the decision boundary. This reinforces the need for a polynomial model to effectively verify process variations.

VI. Conclusion

We have presented a scalable modeling approach for analog neural networks that incorporates process variations. Our polynomial-based model is obtained by analyzing the neuron circuit under varying process parameters on the transistor level. The model is then deployed in a system-level formal verification enclosing the variations in the output of each layer using reachability analysis. We evaluated our approach on three classification benchmarks with different network architectures, including convolutional neural networks. In contrast to Monte Carlo simulations, which are not feasible for deeper neural networks (already up to a day for MNIST CNN), our verification approach runs in a few seconds on a normal home computer. Thus, it can be utilized to optimize the inference circuit design and robustness against process variations, significantly speeding up the design cycle. In future work, we plan to integrate other sources of variation, such as device mismatch and noise, and extend our approach to larger inference circuits.

Acknowledgments

The authors gratefully acknowledge financial support by the project FAI (No. 286525601) funded by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG).

REFERENCES

[1] P. Dhilleswararao, S. Boppu, M. S. Manikandan, and L. R. Cenkeramaddi, "Efficient hardware architectures for accelerating deep neural networks: Survey," *IEEE Access*, 2022.

[2] A. Datar and P. Saha, "The promise of analog deep learning: Recent advances, challenges and opportunities," *arXiv preprint arXiv:2406.12911*, 2024.

[3] M. Yan, H. Wang, and S. Venkataraman, "PolyThrottle: Energy-efficient neural network inference on edge devices," *arXiv preprint arXiv:2310.19991*, 2023.

[4] P. Gao and M. Adnan, "Overview of emerging electronics technologies for artificial intelligence: A review," *Materials Today Electronics*, 2025.

[5] B. Tain, R. Millet, R. Lemaire, M. Szczepanski, L. Alacoque, E. Pluchart, S. Choisnet, R. Prasad, J. Chossat, P. Pierunek *et al.*, "J3DAI: A tiny DNN-based edge AI accelerator for 3D-stacked CMOS image sensor," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2025.

[6] X. Li, J. Le, L. T. Pileggi *et al.*, "Statistical performance modeling and optimization," *Foundations and Trends in Electronic Design Automation*, 2007.

[7] Z. Yan, X. S. Hu, and Y. Shi, "Computing-in-memory neural network accelerators for safety-critical systems: Can small device variations be disastrous?" in *IEEE/ACM International Conference on Computer-Aided Design*, 2022.

[8] T. P. Xiao, B. Feinberg, C. H. Bennett, V. Prabhakar, P. Saxena, V. Agrawal, S. Agarwal, and M. J. Marinella, "On the accuracy of analog neural network inference accelerators," *IEEE Circuits and Systems Magazine*, 2023.

[9] F. B. Gencer, X. Xhafa, B. B. İnam, and M. B. Yelten, "Design and validation of an artificial neural network based on analog circuits," *Analog Integrated Circuits and Signal Processing*, 2021.

[10] E.-D. Şandru, E. David, I. Kovacs, A. Buzo, C. Burileanu, and G. Pelz, "Modeling the dependency of analog circuit performance parameters on manufacturing process variations with applications in sensitivity analysis and yield prediction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[11] R. Narayanan, B. Akbarpour, M. H. Zaki, S. Tahar, and L. C. Paulson, "Formal verification of analog circuits in the presence of noise and process variation," in *IEEE Design, Automation & Test in Europe Conference & Exhibition*, 2010.

[12] Y. Lyu, W. Chen, M. Zheng, B. Yin, J. Li, and L. Cai, "Machine learning-assisted device modeling with process variations for advanced technology," *IEEE Journal of the Electron Devices Society*, 2023.

[13] D. Kim, J. Park, C. Shin, J. Jung, K. Shin, S. Baek, S. Heo, W. Kim, I. Jeong, J. Cho *et al.*, "GLOVA: Global and local variation-aware analog circuit design with risk-sensitive reinforcement learning," in *ACM/IEEE Design Automation Conference (DAC)*, 2025.

[14] M. Huff, "Important considerations regarding device parameter process variations in semiconductor-based manufacturing," *ECS Journal of Solid State Science and Technology*, 2021.

[15] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[16] N. Afroz, A. Sayem, G. Volanis, D. Maliuk, H. Stratigopoulos, and Y. Makris, "On the sensitivity of analog artificial neural network models to process variation," in *IEEE 42nd VLSI Test Symposium*, 2024.

[17] 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 *Annual Design Automation Conference*, 2018.

[18] Y. Abu-Haeyeh, T. Bartelsmeier, T. Ladner, M. Althoff, L. Hedrich, and M. Olbrich, "Formally verifying analog neural networks with device mismatch variations," in *IEEE Design, Automation and Test in Europe Conference*, 2025.

[19] X. Li, Z. Wu, G. Rzepa, M. Karner, H. Xu, Z. Wu, W. Wang, G. Yang, Q. Luo, L. Wang *et al.*, "Overview of emerging semiconductor device model methodologies: From device physics to machine learning engines," *Fundamental Research*, 2024.

[20] R. Nägele, J. Finkbeiner, V. Stadtlander, M. Grözing, and M. Berroth, "Analog multiply-accumulate cell with multi-bit resolution for all-analog ai inference accelerators," *IEEE Transactions on Circuits and Systems I: Regular Papers*, 2023.

[21] K. Kaulen, T. Ladner, S. Bak, C. Brix, H. Duong, T. Flinkow, T. T. Johnson, L. Koller, E. Manino, T. H. Nguyen *et al.*, "The 6th international verification of neural networks competition (VNN-COMP 2025): Summary and results," *arXiv preprint arXiv:2512.19007*, 2025.

[22] D. M. Lopez, M. Althoff, L. Benet, C. Blab, M. Forets, Y. Jia, T. T. Johnson, M. Kranzl, T. Ladner, and L. Linauer, "ARCH-COMP24 category report: Artificial intelligence and neural network control systems (AINNCS) for continuous and hybrid systems plants," in *International Workshop on Applied Verification for Continuous and Hybrid Systems*, 2024.

[23] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *International Conference on Computer Aided Verification*, 2017.

[24] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient SMT solver for verifying deep neural networks," in *International Conference on Computer Aided Verification*, 2017.

[25] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," in *Advances in Neural Information Processing Systems*, 2018.

[26] G. Katz, D. A. Huang, D. Ibeling, K. Julian, C. Lazarus, R. Lim, P. Shah, S. Thakoor, H. Wu, A. Zeljić, D. L. Dill, M. J. Kochenderfer, and C. Barrett, "The Marabou framework for verification and analysis of deep neural networks," in *International Conference on Computer Aided Verification*, 2019.

[27] P. Henriksen and A. Lomuscio, "Efficient neural network verification via adaptive refinement and adversarial search," in *European Conference on Artificial Intelligence*, 2020.

[28] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," in *ACM on Programming Languages*, 2019.

[29] T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, and M. Vechev, "AI2: Safety and robustness certification of neural networks with abstract interpretation," in *IEEE Symposium on Security and Privacy*, 2018.

[30] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," *Advances in Neural Information Processing Systems*, 2018.

[31] D. M. Lopez, S. W. Choi, H.-D. Tran, and T. T. Johnson, "NNV 2.0: The neural network verification tool," in *International Conference on Computer Aided Verification*, 2023.

[32] T. Ladner and M. Althoff, "Automatic abstraction refinement in neural network verification using sensitivity analysis," in *ACM International Conference on Hybrid Systems: Computation and Control*, 2023.

[33] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, 2006.

[34] N. Kochdumper and M. Althoff, "Sparse polynomial zonotopes: A novel set representation for reachability analysis," *IEEE Transactions on Automatic Control*, 2020.

[35] T. Ladner, M. Eichelbeck, and M. Althoff, "Formal verification of graph convolutional networks with uncertain node features and uncertain graph structure," *Transactions of Machine Learning Research*, 2025.

[36] L. Koller, T. Ladner, and M. Althoff, "Out of the shadows: Exploring a latent space for neural network verification," in *International Conference on Learning Representations*, 2026.

[37] F. Aul, N. Katsaouni, L. Krischker, S. Schmalhofer, M. H. Schulz, and L. Hedrich, "Schematic generation of programmable analog neural networks for signal processing," in *International Conference on SMACD and Conference on PRIME*, 2021.

[38] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff, "Model conformance for cyber-physical systems: A survey," *ACM Transactions on Cyber-Physical Systems*, 2019.

[39] W. Wolberg, O. Mangasarian, N. Street, and W. Street, "Breast Cancer Wisconsin (Diagnostic)," UCI Machine Learning Repository, 1995.

[40] R. A. Fisher, "Iris," UCI Machine Learning Repository, 1936.

[41] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist

[42] Cadence Design Systems, Inc., "Cadence virtuoso spectre simulator datasheet," 2024. [Online]. Available: www.cadence.com

[43] M. Althoff, "An introduction to CORA 2015," in *Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[44] M. Althoff, N. Kochdumper, T. Ladner, M. Perschl, and M. Wetzlinger, "CORA manual," *Technical University of Munich*, 2025. [Online]. Available: https://cora.in.tum.de/manual

[45] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *International Workshop on Hybrid Systems: Computation and Control*, 2005.