Evolutionary Approximation of Ternary Neurons for On-sensor Printed Neural Networks

Vojtech Mrazek
Brno University of Technology
Brno, Czech Republic
mrazek@fit.vutbr.cz

Argyris Kokkinis
Aristotle University of Thessaloniki
Thessaloniki, Greece
arkokkin@auth.gr

Panagiotis Papanikolaou
University of Michigan
Ann Arbor, USA
panagip@umich.edu

Zdenek Vasicek
Brno University of Technology
Brno, Czech Republic
vasicek@fit.vutbr.cz

Kostas Siozios
Aristotle University of Thessaloniki
Thessaloniki, Greece
ksiop@auth.gr

Georgios Tzimpragos
University of Michigan
Ann Arbor, USA
gtzimpra@umich.edu

Mehdi Tahoori
Karlsruhe Institute of Technology
Karlsruhe, Germany
mehdi.tahoori@kit.edu

Georgios Zervakis
University of Patras
Patras, Greece
zervakis@upatras.gr

ABSTRACT

Printed electronics offer ultra-low manufacturing costs and the potential for on-demand fabrication of flexible hardware. However, significant intrinsic constraints stemming from their large feature sizes and low integration density pose design challenges that hinder their practicality. In this work, we conduct a holistic exploration of printed neural network accelerators, starting from the analog-to-digital interface—a major area and power sink for sensor processing applications—and extending to networks of ternary neurons and their implementation. We propose bespoke ternary neural networks using approximate popcount and popcount-compare units, developed through a multi-phase evolutionary optimization approach and interfaced with sensors via customizable analog-to-binary converters. Our evaluation results show that the presented designs outperform the state of the art, achieving at least 6× improvement in area and 19× in power. To our knowledge, they represent the first open-source digital printed neural network classifiers capable of operating with existing printed energy harvesters.

CCS CONCEPTS

• **Hardware** → **Emerging technologies**; **Logic circuits**; • **Computing methodologies** → **Machine learning**.

KEYWORDS

Approximate Computing, Electrolyte-gated FET, Printed Electronics, Low-Power Classifiers, Ternary Neural Networks

ACM Reference Format:
Vojtech Mrazek, Argyris Kokkinis, Panagiotis Papanikolaou, Zdenek Vasicek, Kostas Siozios, Georgios Tzimpragos, Mehdi Tahoori, and Georgios Zervakis. 2024. Evolutionary Approximation of Ternary Neurons for On-sensor Printed Neural Networks. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '24), October 27–31, 2024, New York, NY, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3676536.3676728

ICCAD '24, October 27–31, 2024, New York, NY, USA

ACM ISBN 979-8-4007-1077-3/24/10
https://doi.org/10.1145/3676536.3676728

1 INTRODUCTION

Printed electronics hold significant promise for various applications, including smart packaging [**?**], in-situ monitoring [**?**], forensics [**?**], and accessible healthcare products and wearables [**? ? ? ? ?**]. A common denominator among these applications is the need for devices that are stretchable, porous, flexible, and conformal, all while adhering to strict area and power budgets. Another common feature is that processing typically involves classification, with recent research efforts directed towards the realization of printed bespoke machine learning (PBML) accelerators [**? ?**].

The term bespoke denotes fully customized circuit implementations (e.g., hardwired model parameters [**? ? ?**]) as well as approximations [**? ? ? ? ? ? ?**] per ML model and dataset. While this assumption may seem weak in other contexts, it is well-suited to printed electronics, where devices are considered disposable [**?**] due to their low fabrication cost and short turnaround times [**? ? ?**]. This specialization allows existing PBML designs, at least in theory, to operate with the power provided by available printed batteries [**?**]. However, the practicality of these solutions is often compromised by the overlooked cost of sensor-processor interfacing, a crucial aspect of the overall design since processing occurs directly on sensor data.

In this work, we address this problem holistically, starting our optimizations from the sensor boundary. Firstly, we replace costly analog-to-digital converters (ADCs) with efficient customized analog-to-binary converters (ABCs). Secondly, we explore ternary neural networks (TNNs) [**?**], where weights take only three values: $\{-1, 0, 1\}$, thus eliminating the need for multipliers. Thirdly, to further enhance hardware efficiency, we introduce approximate popcount and fused popcount-and-compare units. To facilitate their automated utilization, we implement a set of evolutionary multi-objective optimization techniques.

The result is approximate bespoke TNN designs that, on average, use 41% fewer resources and consume 42% less power than their exact counterparts, with no accuracy loss on common printed ML datasets [? ?]. Compared to state-of-the-art digital approximate printed classifiers, our evaluation indicates an average reduction of 32× in area and 34× in power consumption, when factoring in the gains from replacing ADCs with the proposed ABC design.

In summary, the main **contributions** of this research are:

- A custom analog-to-binary converter design that uses only two resistors and an analog comparator.
- Approximate designs for popcount and fused popcount-and-comparison units, accompanied by a detailed Pareto analysis.
- The development of an multi-phase evolutionary optimization framework for the automated deployment of our approximation techniques and the search of the formed design space.
- The first, to the best of our knowledge, open-source complete digital printed classifier solution[1] that can be fully powered by either available printed batteries [?] or energy harvesters [?].

2 RELATED WORK

Printed electronics frequently targets applications centered on classification tasks. This focus, combined with the expanding fast-moving consumer goods market, has spurred research into printed machine learning accelerators. Various efforts have explored neural network models in this context, as summarized in Table 1.

The design choices behind the proposed solutions are shaped by the technology's unique attributes and the requirements imposed by its application domains. Printed electronics employ additive and mask-less manufacturing techniques like jet, screen, or gravure printing [?]. These methods, coupled with low capital equipment costs, enable rapid and inexpensive circuit production—sometimes for less than a cent per unit. However, these same characteristics lead to large feature sizes, low integration density (far below silicon VLSI), and increased device latencies [? ?]. The latter is often not a concern due to the low sampling rates and relaxed performance requirements in typical applications [?]. As for the rest, to maximize functionality with minimum resources, bespoke designs are favored, especially since circuit programmability and reusability are less critical in this domain.

Bespoke designs are customized for specific classifier models trained on particular datasets, eliminating generality-related overheads [?]. This specialization extends to developing and implementing tailored approximation techniques [?]. For instance, Armeniakos et al. [?] proposed post-training weight replacement with more hardware-friendly approximates to reduce the cost of the underlying multipliers. This effort was further enhanced by introducing voltage over-scaling techniques where necessary [?]. In subsequent works, Armeniakos et al. [?] incorporated hardware-friendly weight approximation into the training process and implemented post-training addition approximation using simple truncation. Afentaki et al. [? ?] constrained weights to powers of 2—leveraging that, in bespoke circuits, a multiplication by a power of 2 is implemented simply by rewiring—approximated accumulations by pruning the adder trees, and employed bounded low-precision ReLu activation. Lastly, an alternative approximation method, stochastic computing [?] neural networks, was explored by Weller et al. [?].

[1]Available at: https://github.com/ehw-fit/approximate-popcount

Table 1: State-of-the-art printed neural networks.

Ref.	Digital	Ax. Mult.	Ax. Add.	Low Prec. Act. Func.	Inp. Size/ ADC Cost
[?]	✓	✗	✗	✗	high
[? ?]	✓	low	✗	✗	high
[?]	✓	moderate	low	✗	high
[? ?]	✓	high	moderate	✓	high
[?]	✗	✗	✗	✓	n/a
Proposed	✓	high	high	✓	low

The work presented here distinguishes itself from existing literature in several ways: it avoids the potential for significant accuracy losses associated with stochastic computing approaches; it employs a multiplier-free design and approximate adders for maximum efficiency; it is purely digital, avoiding the noise and variability issues that can affect low-resolution printed analog designs [? ? ? ?]; and it begins optimization at the sensor boundary, rather than post-ADC. To our knowledge, the only other printed classifier that considers sensor-processor interface optimizations is a unary decision tree accelerator [?]. However, this design relies on multiple analog comparators per analog-to-unary converter, indicating room for further improvement.

3 PROPOSED END-TO-END DESIGN

3.1 Sensor-Processor Interface

The sensor-processor interface typically consists of an array of ADCs. ADCs are often overlooked in existing optimization efforts despite their significant hardware costs. For example, a 4-bit printed flash ADC, as the one shown in Figure 1a, occupies 12 mm^2 and consumes 1 mW of power [?]. If incorporated into a recent printed multilayer perceptron design with 4-bit inputs [?], it increases its area by 1.11× to 44× and power consumption by 1.97× to 370×.

We propose to replace such ADCs with a custom analog-to-binary converter (ABC), presented in Figure 1b. Compared to the reference 4-bit ADC, our ABC design requires 8× fewer resistors, 15× less comparators, and does not require a priority encoder. This translates to an area of only 0.07 mm^2 and a power consumption of 0.03 mW—167× smaller and 34× more power efficient than the original ADC design. Additionally, we allow the use of resistors, with potentially different values R_1 and R_2. Having more than one voltage rails is inefficient in printed electronics, so we assume a common reference voltage V_{ref} for all ABCs. Adjusting the R_1/R_2 ratio enables to define the voltage threshold at which a specific input feature transitions between 1 and 0.

3.2 Bespoke Ternary Neural Networks

This works focuses on ternary neural networks. TNNs provide accuracy levels that fall between binary and standard neural networks, are easy to implement, and eliminate the need for hardware multipliers. Previous printed multilayer perceptron designs also avoided multipliers by constraining synaptic weight values to powers of 2 [? ? ?]. However, this wider range of weight values can result in larger accumulation circuits compared to TNNs, where

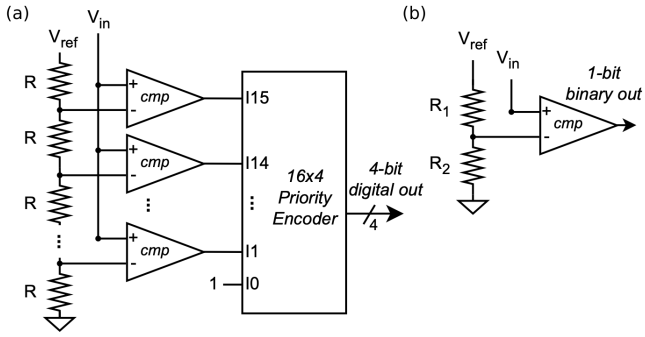

Figure 1: Panel a: A 4-bit flash ADC. Panel b: Proposed analog-to-binary converter. The ratio R_1/R_2 is used to regulate the voltage threshold at which an input feature becomes 1. V_{in} denotes the voltage output of the sensor, which feeds the two converters (a) or (b).

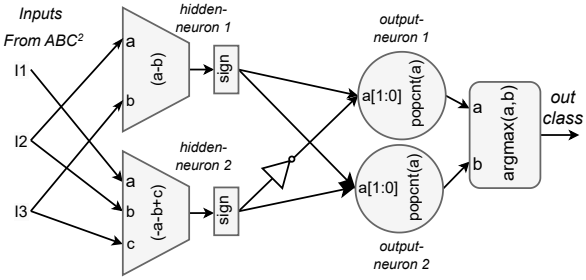

Figure 2: Bespoke $(3, 2, 2)$ TNN circuit with $[[[0, 1, -1], [-1, -1, 1]], [[1, -1], [1, 1]]]$ weights.

weights are restricted to only -1, 0, or 1. For example, assuming 4-bit inputs I_j, the sum $8\times I_1 - 2\times I_2 - 4\times I_3 - 2\times I_4$ requires 0.67 cm^2, while with ternary weights, the area of the respective sum is at worst 0.38 cm^2, i.e., 43% smaller. These improvements are enhanced under the assumption of binary inputs ($I_j \in \{0, 1\}$), such as those provided by the proposed ABCs. In this case, the corresponding area reduces to 0.2 cm^2, i.e., 3.4× smaller than the initial design with powers of 2 weights.

3.2.1 Inputs and Weights Quantization. We employ Qkeras [**?**] for quantization-aware training (QAT) of the TNN, allowing customization of quantizers for synaptic weights and inputs. For weights, we use the `ternary` quantizer with `alpha` set to 1, yielding coefficients in $\{-1, 0, 1\}$. Hidden-layer neuron inputs are quantized to $\{0, 1\}$ using the `quantized_bits` activation function. For first-layer neuron inputs, we extend the `quantized_bits` function to set specific quantization thresholds for each input feature. We normalize training set inputs to $[0, 1]$ and analyze their distribution to adjust the quantization threshold V_q based on whether the distribution is left-skewed, nearly normal, or right-skewed. This threshold determines whether an input is considered 1 or 0 in the binary representation.

In this work, we set V_q to the median of the normalized input distribution for each input, rather than treating it as a learnable parameter during training. However, process variations can affect the actual values of R_1 and R_2, potentially altering the R_1/R_2 ratio in the ABC design. This makes V_q susceptible to process variations, which could negatively impact classification accuracy. Optimizing for this is beyond the scope of this paper, but we propose incorporating variation-aware training techniques into QAT [**? ?**] to enhance the robustness of our TNNs.

3.2.2 Circuit Implementation. The trained TNN model is automatically translated into hardware by combining Icarus Verilog (iverilog) with a set of custom neuron templates designed for each layer. Adhering to bespoke design principles, the synaptic weights are hardcoded into the circuit description. For synaptic weights with a value of 0 between the input and hidden layer, the corresponding connection is removed from the hidden-layer neuron, effectively eliminating an addend from its accumulator. For non-zero weights, the sign determines the hardware operation—either addition or subtraction—over the respective input. Consequently, each hidden-layer neuron performs a multi-operand summation, with its output being the sign of the respective sum: 1 for a positive sum and 0 for a negative sum.

The output layer implementation is customized to accommodate the different encoding of hidden-layer neuron outputs ($\{-1, 1\}$) compared to ABC outputs ($\{0, 1\}$). We employ XNOR-based neurons [**?**], which use XNOR gates instead of multiplications and a popcount operation to determine the number of high inputs.

Given the fixed, known weights, XNOR gates are simplified: becoming NOT gates for weights of -1 and simple wires for weights of 1. The popcount operation is performed on $\{0, 1\}$ values, while our encoding assumes $\{-1, 1\}$. Thus, we apply a linear mapping that translates 0 to $\frac{1}{2}$. So, unlike the hidden layer connections, zero-valued weights are retained here as a $+0.5$ constant correction term. By ensuring, in training, the output neurons have the same number of zero-valued connections N, the correction term is consistent across all neurons and can be omitted without affecting the result of the classification, as $\text{argmax}(y + \frac{N}{2}, q + \frac{N}{2}) = \text{argmax}(y, q)$.

Lastly, to implement the argmax function, which identifies the output neuron with the highest score (i.e., highest popcount output), a set of comparators is employed. Figure 2 depicts a bespoke (3,2,2) TNN design. In this illustration, the hidden-layer neurons have weights $[0, 1, -1]$ and $[-1, -1, 1]$, while the output neurons have weights $[1, -1]$ and $[1, 1]$.

4 PROPOSED APPROXIMATION

The proposed approximation methodology, illustrated in Figure 3, consists of three main phases. In Phase 1, Cartesian genetic programming (CGP) [**?**] is used to evolve circuits that approximate the popcount function. In Phase 2, we identify Pareto-optimal combinations of these circuits to deliver the necessary approximate popcount-compare functionality for the hidden-layer neurons. Finally, in Phase 3, the NSGA-II evolutionary algorithm [**?**] is applied to integrate these approximate components into a bespoke TNN circuit, optimizing for maximum resource efficiency with minimal impact on accuracy.

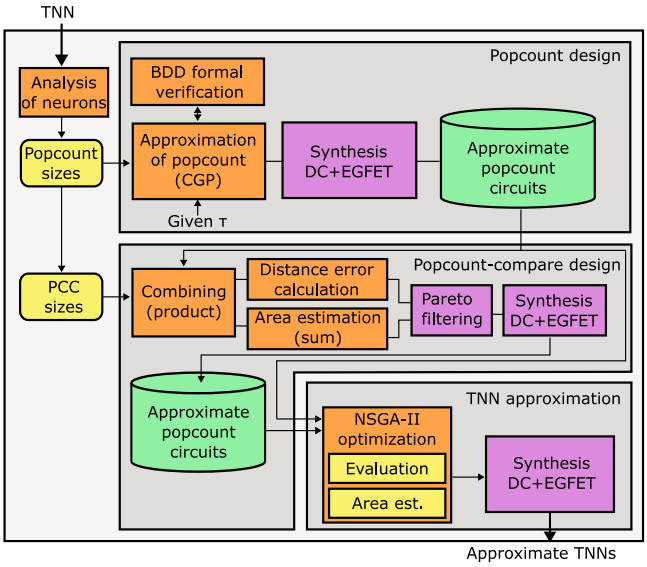

Figure 3: Overview of the proposed three-phase TNN approximation framework, including Cartesian genetic programming for popcount circuit approximations in Phase 1, a Pareto analysis for identifying optimal combinations of popcount circuits for the construction of approximate popcount-compare circuits in Phase 2, and their integration into bespoke TNN circuits using the NSGA-II evolutionary algorithm in Phase 3. The underlying technology assumed is EGFET, operating at voltages as low as 0.6-1 V.

4.1 Approximate Popcount & Popcount-compare Circuits

For hidden-layer neurons, a binary step activation function is used. The computation per neuron is:

$$\sum_{\forall i} w_i I_i \geq 0, \tag{1}$$

where w_i are synaptic weights and I_i are inputs. Given $w_i \in \{-1, 0, 1\}$, this transforms to:

$$\sum_{\forall i: w_i=1} I_i \; - \sum_{\forall i: w_i=-1} I_i \geq 0 \implies \sum_{\forall i: w_i=1} I_i \; \geq \sum_{\forall i: w_i=-1} I_i. \tag{2}$$

With $I_i \in \{0, 1\}$, each sum can be calculated by a popcount operation. Thus, each hidden-layer neuron requires two popcount circuits and a comparator. The comparator's complexity is minimal compared to this of the popcount operations. For example, with 8 inputs, the comparator uses only 16% of the hidden-layer neuron's total area; for 64 inputs, it's less than 3%. In the rest of this Section, we introduce approximate implementations of popcount-compare (for hidden-layer neurons) and popcount (for output-layer neurons) circuits to enhance area-efficiency.

4.1.1 Approximate Popcount Circuits. An n-bit popcount (PC) operation sums n binary inputs, typically implemented using a tree structure of adders. To approximate this exact circuit, we employ the evolutionary CGP algorithm [**?**]. In CGP, a simple search method is deployed over an integer address-based genetic representation of the circuit to create random mutations of an initial population P that contains the accurate PC circuit. To evaluate the candidate circuits, we use a fitness function. Each member c in population P receives a fitness score $F(c)$, with the highest-scoring individual becoming the parent of the next population. This parent then generates λ new candidate solutions through mutation. The process terminates when either the number of iterations exceeds a set maximum, or a predefined time limit is reached.

The searching algorithm's optimization criterion is the estimated area for the target technology (e.g., EGFET PDK [**?**]), while maintaining the error ε below a specific threshold, consistent with previous error-oriented approximation research [**?**]. This error constraint τ is incorporated into the fitness function as follows:

$$F(c) = \begin{cases} area(c), & \text{if} \varepsilon(c) \leq \tau \\ \infty, & otherwise. \end{cases} \tag{3}$$

PC circuits produce standard arithmetic outputs, allowing the use of conventional error metrics in the approximation process: mean arithmetic error ε_{mae} and worst-case arithmetic error ε_{wcae}. These metrics calculate the average or maximum error across all input combinations. However, for a large number of inputs n in PC circuits, evaluating all 2^n input vectors becomes impractical. To address this, a formal verification approach using binary decision diagrams (BDD) [**?**] is used.

4.1.2 Approximate Popcount-compare Circuits. Hidden-layer neurons are implemented using popcount-compare (PCC) circuits to realize Equation (2). Each PCC consists of two PC circuits with n_{pos} and n_{neg} input bits, respectively, and a j-bit comparator, where j is equal to $\lceil \log_2 \max(n_{pos}, n_{neg}) \rceil$. To ensure a comprehensive analysis and evaluation, PCCs are created for all n_{pos} and n_{neg} configurations found in the target TNNs.

Distance metric: In approximating non-arithmetic circuits, Hamming distance is often considered the sole suitable error metric. However, this approach falls short in the context of approximate computing, as it may not accurately represent the error levels [**?**]. For example, in 4-bit output arithmetic circuits, an error in one bit position ($0 \rightarrow 8$) can be more severe than errors in all four bits ($7 \rightarrow 8$). This limitation extends to relational operators as well. Consider two approximate $\geq_a$ and $\geq_b$ circuits: if $\geq_a$ erroneously determines that $0 \geq_a 1$ = TRUE (a 1-position deviation) and $\geq_b$ incorrectly asserts that $0 \geq_b 4$ = TRUE (a 4-position deviation), both errors result in the same Hamming distance of 1 (FALSE $\rightarrow$ TRUE), despite their clearly different magnitude of errors.

The Hamming distance's inability to capture error severity in the context of approximate computing is of particular importance to PCC approximation, which produces a one-bit output. To address this, we propose a new *distance metric* D, inspired by those used in approximate median filters and sorting networks [**?**]. This metric is defined as:

$$D(x, z) = \begin{cases} 0, & \text{if } rel(x, z) = rel'(x, z) \\ x - z, & \text{otherwise} \end{cases} \tag{4}$$

The distance D is a function of two inputs x and z, an accurate relational function rel with a Boolean output, and its approximate variant rel'. This provides a more nuanced measure of error magnitude, addressing the limitations of Hamming distance in this

context. Using D, we define the mean distance error (ε_{mde}) and worst-case distance error (ε_{wcde}) as follows:

$$\varepsilon_{mde} = \frac{1}{|G|} \sum_{\forall (x,z) \in G} |D(x,z)|; \ \varepsilon_{wcde} = \max_{\forall (x,z) \in G} |D(x,z)| \quad (5)$$

Here, G represents the input domain.

Pareto analysis: We create a library of approximate PCC circuits using the PC circuit approximation methodology described above. For each exact PCC circuit, we first generate all possible approximate PCC circuits by testing all combinations of n_{pos} and n_{neg} PC circuits from our approximate PC library, including exact PC circuits as zero-error designs. Subsequently, we identify Pareto-optimal PCC circuits using two criteria: (i) accuracy, evaluated through ε_{mde} for 10^6 random (x, z) pairs in Equations (4) and (5), and (ii) hardware cost, estimated by the combined area of the n_{pos} and n_{neg} PC circuits.

This search process is fully parallelizable and can be conducted at a high level (i.e., in Python) without the need for hardware evaluation. Consequently, the execution time remains well constrained, ensuring the scalability of our approximate PCC circuit search. Once the Pareto analysis is completed, the selected PCC circuits, including both comparators and approximate PC circuits, are synthesized to asses their actual hardware cost.

4.2 Approximate Circuit Integration

The last step is the integration of PC and PCC circuits from the developed libraries into a complete TNN circuit. The objective is to achieve the best area-accuracy trade-off. To this end, we deploy a multi-objective optimization based on the evolutionary NSGA-II algorithm [**?**]. As in prior optimizations, the two criteria are: accuracy and area. We encode the problem as an integer list, where each integer corresponds to one circuit from the PCC and PC library for the respective neuron. The sum of the areas of the selected PCC and PC circuits is used as a proxy for the overall area of the TNN. To evaluate the actual area and power results of the generated TNNs, we go again through a round of synthesis. The underlying technology assumed is EGFET, which allows operation at voltages as low as 0.6-1 V [**?**] and aligns well with low-power requirements of battery/self-powered IoT applications.

5 EVALUATION

The goal of the presented research is to develop bespoke TNN circuit designs that optimize hardware efficiency while minimizing any decrease in accuracy. Initially, we evaluate the proposed hardware implementations and the efficacy of our approximation methods. Subsequently, we conduct a comparative analysis against the current state-of-the-art printed neural networks.

Evaluation setup: For our evaluation, we utilize the five datasets from the UCI ML repository [**?**]. These datasets are chosen for two primary reasons: they enable direct comparisons with state-of-the-art designs, and they consist of sensor data suitable for printed electronics' applications, as discussed in studies [**? ?**]. To perform digital circuit synthesis and analysis, we rely on Synopsys Design Compiler and PrimeTime tools, used in conjunction with the EGFET standard cell library [**?**]. For the ABC design, we utilize Cadence Virtuoso with the EGFET PDK, running SPICE simulations to assess its area and power consumption. The voltage supply for all circuits is set to 0.6 V, in line with EGFET capabilities and consistent with prior work in the field [**? ?**]. In terms of operating speeds, all circuits are evaluated at a frequency of 5 Hz, although our designs can achieve frequencies up to 20 Hz. This decision is made to maintain consistency with existing solutions in the literature [**? ?**] and facilitate a fair comparison.

Table 2: TNN accuracy results.

Dataset	Exact MLP [?]			Our Exact TNN		
	A*	T†	I/W*	A*	T†	I/W*
Arrhythmia	0.62	(274,5,16)	4/8	0.60	(274,3,16)	1/2
Breast Cancer	0.98	(10,3,2)	4/8	0.98	(10,10,2)	1/2
Cardio	0.88	(21,3,3)	4/8	0.85	(21,3,3)	1/2
Redwine	0.56	(11,2,6)	4/8	0.56	(11,3,6)	1/2
Whitewine	0.54	(11,4,7)	4/8	0.50	(11,11,7)	1/2

*Inference accuracy. †Network Topology. *Bit-width Inputs/Weights.

TNN baseline: Exact TNN models are used as the baseline for accuracy comparisons. The selected datasets are split into 70% for training and 30% for testing. For training, the Adam optimizer is used with the number of epochs ranging from 10 to 20. The learning rate for each network is obtained through Bayesian optimization, aiming to maximize the model's inference accuracy with a maximum of 100 attempts. Learning rate parameters are selected from the range of 0.001 to 0.01. The final model is that with the highest accuracy among a set of 100 TNNs with various hyperparameters.

Regarding TNN hyperparameters, all models assume a single hidden layer. To determine the number of hidden-layer neurons, we perform an exhaustive search within the range of 1 to 40 neurons. For each iteration, we repeat the procedure described previously. Among the TNNs yielding the highest accuracy, we select the one with the fewest neurons. This exhaustive search takes less than one hour on an AMD EPYC 7552 server with 256 GB RAM.

Table 2 summarizes our findings across the examined datasets. For reference, we also include the accuracy numbers achieved by a baseline MLP network [**?**] utilizing 4-bit inputs and 8-bit weights. The comparison indicates 0-4% accuracy drop for our baseline TNNs. However, as will discussed in the rest of this section, this compromise enables a significant boost in hardware efficiency, ensuring that the operating requirements of our printed TNNs fall within the integration and power budget of the target technology.

5.1 Approximate Circuits Evaluation

5.1.1 Popcount Circuits. We generate approximate PC circuits with the following constraints and configurations. The error limits τ_{mae} and τ_{wcae} are logarithmically distributed from 0.1 to $0.5 \cdot 2^m$ and from 1 to $0.5 \cdot 2^m$, with $m = \lceil \log_2 n \rceil$, resulting in 2,090 approximate circuits. We also set CGP search termination criteria to 30, 60, and 300 minutes for PC sizes $n < 16$, $n < 32$, and $n < 60$. Using BDD evaluation, we achieve average speeds of 7,759 and 1,362 evaluations per second for 16-bit and 32-bit PCs. For 60-bit PC circuits, this number falls to 25 evaluations per second. To account for this, we extend the time limit, thereby increasing the chances of identifying solutions with better area-accuracy trade-offs.

A visualization of the obtained results for three PC circuit sizes is provided in Figure 4. The featured circuits are synthesized and

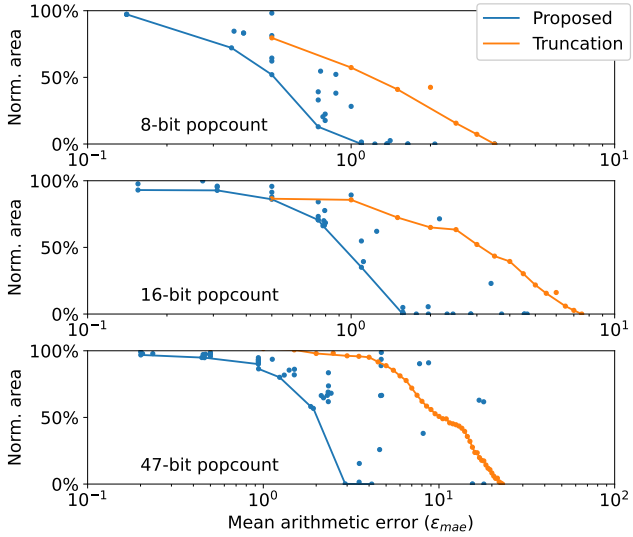

Figure 4: Comparison between the proposed approximation approach and the previously used truncation technique for PCC circuits of various sizes. The results displayed are based on post-synthesis area measurements.

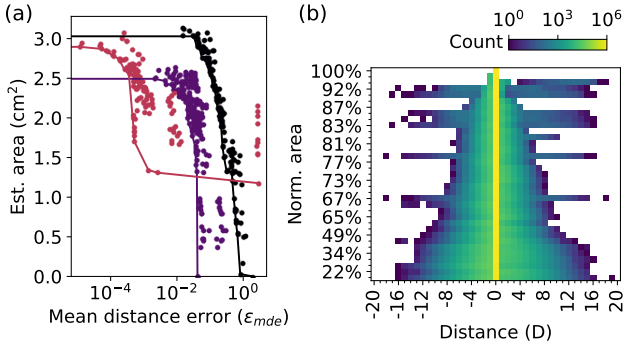

Figure 5: Panel a: Trade-off illustration for three PCC circuits applied to the Arrhythmia dataset. The circuits are characterized by their sizes (n_{pos}, n_{neg}): black (45, 39), purple (47, 30), and pink (60, 29). Panel b: Distribution of distance error for Pareto optimal PCC circuits of size $n_{pos} = 45, n_{neg} = 39$ (corresponding to the black line in Panel a). Area results are post-synthesis and normalized relative to the exact circuit.

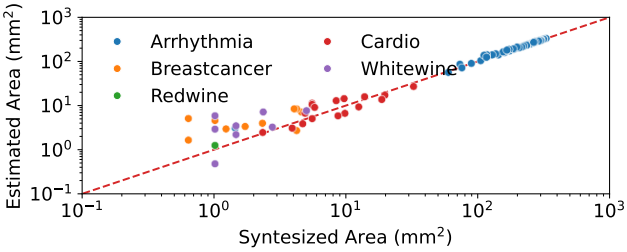

Figure 6: Comparison between area estimates and post-synthesis results for various PCC circuits. Design points with similar colors indicate the use of the same exact design as a starting point.

evaluated based the error metrics and methodology described above. For easier comparison, area results are normalized relative to exact PC circuit of the same size. Our results indicate that as the error limit increases, CGP optimization identifies increasingly area-efficient solutions. Additionally, we assess the efficacy of our approximation approach through a comparison of our PC circuits with variants derived using the truncation approximation approach, which has previously been applied in a wide range of approximate circuits, including multilayer perceptrons [**?**]. In this case, we achieve about 2× area reduction with ε_{mae} of only 0.5 for an 8-bit PC circuit, 1.1 for a 16-bit PC circuit, and 1.9 for a 47-bit PC circuit. Similar results to those shown in Figure 4 are observed for all PC circuit sizes.

5.1.2 Popcount-compare Circuits. Using the methods outlined in Section 4, we generate 5,841 approximate PCC circuits for our five target datasets. Figure 5a presents an area-versus-error analysis for three distinct PCC circuits customized for the Arrhythmia dataset. The results reveal significant overlap in designs, indicating that the approximate PCC circuit design space can be substantially pruned. This pruning reduces the size of the design space for TNN approximation. Consequently, we limit the approximate PCC library to 134 circuits across 22 different input sizes.

Figure 5b presents a detailed error analysis of the design points on the black Pareto frontier from Figure 5a. The logarithmic-scale histograms depict the distribution of distance error D for $|G| = 10^6$ random input samples. Hence, the x-axis represents the distance metric D introduced in Section 4, while the y-axis shows normalized area results, with area savings increasing as you move down the axis. The analyzed PCC circuit features $n_{pos} = 45$, $n_{neg} = 39$, and a 6-bit comparator, corresponding to the top hidden-layer neuron of our bespoke Arrhythmia TNN model.

Each row in the plot represents a Pareto-optimal approximation of the PCC circuit. At the top, the fully accurate design's histogram is entirely concentrated in the bin for $D = 0$, indicating no errors. As we move down the rows, accuracy is traded for reduced area, causing the probability mass to gradually spread to bins with distances further from 0. If our exploration allows differences between positive and negative PC circuits up to $\varepsilon_{wcde} = 4$, it results in: 12.6% area reduction (6th row from the top in Figure 5b), 95.57% error-free PCC operations, and a mean distance ε_{mde} of only 0.06. Moving down the y-axis, for $\varepsilon_{wcde} \leq 8$, area savings increase to approximately 30%. For approximations at the bottom of the plot, area can be reduced to as little as 25% of the accurate design while still maintaining over 80% correct results. These results are consistent across all PCC circuit configurations examined.

Given that the results presented in the first leg of Pareto analysis are based on area estimates, the final step is to evaluate how closely these estimates match the actual synthesis results. Figure 6 provides this comparison, illustrating the relationship between estimated and synthesized area values for the PCC circuits. While our estimation tends to underestimate smaller PCCs due to not accounting for comparator cost, occasional overestimations occur due to synthesis optimizations. However, overall, there is a good enough correlation between the actual and estimated area.

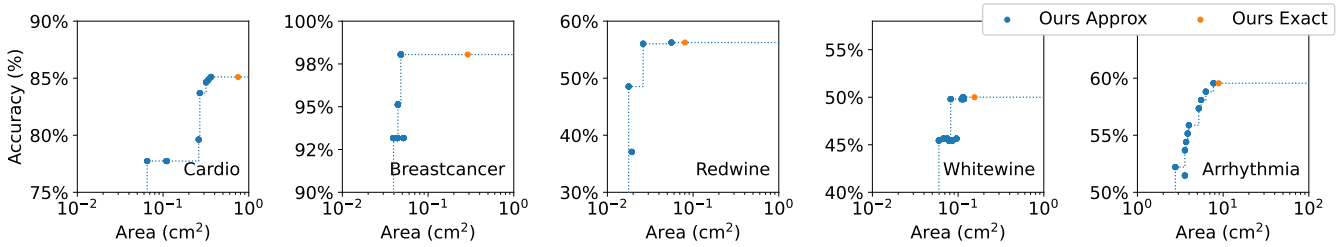

Figure 7: Post-synthesis area-versus-accuracy evaluation for Pareto-optimal approximate TNNs.

5.2 Approximate TNN Evaluation

The final stage of our evolutionary approximation approach utilizes the NSGA-II algorithm to construct a TNN design. This design incorporates components from our PCC (hidden neurons) and PC (output neurons) circuit libraries. We implement NSGA-II using the `pymoo` library, employing an integer-based representation. Each integer in this representation corresponds to the index of an approximate component for a given neuron. Depending on the dataset under examination, chromosome lengths range from 6 to 19 integers.

To navigate the design search space, which ranges from $2.4 \cdot 10^3$ approximate candidates for Cardio to $6.8 \cdot 10^{11}$ for Arrhythmia, we employ binary crossover with polynomial mutation. If we were to consider PC approximation for every popcount operation in the hidden-layer neurons instead of combining them in Pareto-optimal PCC circuits, these values would increase to $5.2 \cdot 10^5$ and $1.7 \cdot 10^{14}$, respectively, complicating the required exploration. The results of the design process for the Arrhythmia TNN (our largest TNN) are shown in Figure 8. The algorithm was run for 200 generations, taking less than 5 minutes to complete, with substantial progress achieved within the first 50 generations.

Figure 7 presents the accuracy-versus-area results of our explorations, featuring designs found on the Pareto optimal curve. Our findings reveal two key points: first, the generated approximate TNN designs can match the accuracy of their exact TNN counterparts while achieving an average area reduction of 41%; second, if we allow for up to a 5% decrease in accuracy compared to the exact TNN, the area savings increase to an average of 67%.

Comparison with State-of-the-art:

Table 3 presents a comparison of our results against the current state-of-the-art in printed neural networks. Our approximate TNN

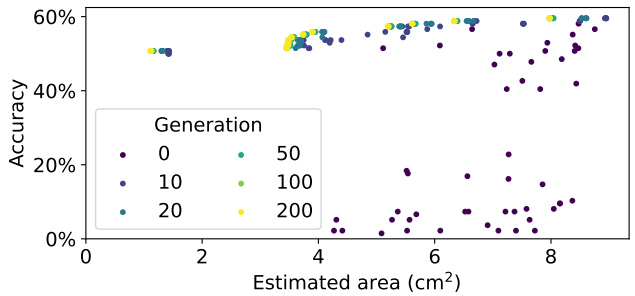

Figure 8: Evolutionary optimization of the Arrhythmia TNN over 200 generations.

Table 3: Comparison against the State of the Art. All circuits target the EGFET technology.

Dataset	Ref.	Accuracy (%)	w/o ADC cost		w/ ADC cost	
			Area (cm^2)	Power (mW)	Area (cm^2)	Power (mW)
Arrhythmia	Exact MLP [?]	62	266.00	998.00	298.06	1273.92
Arrhythmia	Ax. MLP [?]	60	13.51	12.80	45.57	288.72
Arrhythmia	Our Exact TNN	60	8.87	8.09	9.06	16.31
Arrhythmia	Our Ax. TNN	60	7.73	7.12	7.92	15.34
Arrhythmia	Our Ax. TNN	57	5.20	4.95	5.39	13.17
BreastCancer	Exact MLP [?]	98	12.00	40.00	13.17	50.07
BreastCancer	Ax. MLP [?]	97	5.10	18.00	6.27	28.07
BreastCancer	Ax. MLP [?]	93	1.50	5.69	2.67	15.76
BreastCancer	Ax. MLP [?]	94	0.08	0.08	1.25	10.15
BreastCancer	Ax. MLP [?]	94	0.03	0.03	1.20	10.10
BreastCancer	Our Exact TNN	98	0.29	0.31	0.30	0.61
BreastCancer	Our Ax. TNN	98	0.05	0.04	0.06	0.34
BreastCancer	Our Ax. TNN	93	0.04	0.04	0.05	0.34
Cardio	Exact MLP [?]	88	33.40	124.20	35.86	145.35
Cardio	Ax. MLP [?]	87	6.08	20.80	8.54	41.95
Cardio	Ax. MLP [?]	83	4.92	16.80	7.38	37.95
Cardio	Ax. MLP [?]	85	1.35	1.57	3.81	22.72
Cardio	Ax. MLP [?]	87	1.46	1.70	3.92	22.85
Cardio	Our Exact TNN	85	0.75	0.91	0.76	1.54
Cardio	Our Ax. TNN	85	0.36	0.42	0.37	1.05
Cardio	Our Ax. TNN	84	0.27	0.30	0.28	0.93
RedWine	Exact MLP [?]	56	17.60	73.50	18.89	84.58
RedWine	Ax. MLP [?]	55	1.06	3.95	2.35	15.02
RedWine	Ax. MLP [?]	51	0.87	3.25	2.16	14.33
RedWine	Ax. MLP [?]	55	0.03	0.02	1.32	11.10
RedWine	Ax. MLP [?]	52	0.03	0.03	1.32	11.11
RedWine	Our Exact TNN	56	0.08	0.09	0.09	0.42
RedWine	Our Ax. TNN	56	0.03	0.03	0.04	0.36
WhiteWine	Exact MLP [?]	54	31.20	126.40	32.49	137.48
WhiteWine	Ax. MLP [?]	53	6.47	21.30	7.76	32.38
WhiteWine	Ax. MLP [?]	49	0.79	2.77	2.07	13.84
WhiteWine	Ax. MLP [?]	50	0.25	0.25	1.54	11.33
WhiteWine	Ax. MLP [?]	51	0.23	0.25	1.52	11.33
WhiteWine	Our Exact TNN	50	0.16	0.18	0.17	0.51
WhiteWine	Our Ax. TNN	50	0.11	0.12	0.12	0.45
WhiteWine	Our Ax. TNN	45	0.06	0.07	0.07	0.40

designs demonstrate a superior accuracy-area trade-off compared to approximate multilayer perceptrons, even without considering the sensor-processor interface cost. Within a 1% accuracy variation, our TNNs achieve, on average, 42×, 2.17×, and 1.99× lower area compared to previous works on approximate multilayer perceptrons [**? ? ?**]. Power consumption is similarly reduced by 162×, 2.05×, and 1.97×, on average, respectively. When sensor-processor interface costs (ADCs and ABCs) are factored in, these gains grow further, with area reductions exceeding 6× and power savings of over 19× compared to the most efficient reported multilayer perceptron design [**?**]. When compared to an exact multilayer perceptron implementation [**?**], our approximate TNNs achieve 55× lower area and 97× less power at the expense of a 5% accuracy drop.

Notice that all our TNN circuits, except those targeting the Arrhythmia dataset, consume well below 2 mW, even when considering the power consumed by the ABCs and the power overhead of the sensors (approximately 5 μW based on previous studies [**?**]). Thus, they can be powered by a printed energy harvester [**?**]. To the best of our knowledge, this is the first digital printed classifier to achieve this. As for our Arrhythmia TNN designs, they can be powered by either a Zinergy (15 mW) or Molex (30 mW) printed battery [**?**]. This represents a significant advancement over previous solutions [**? ?**] targeting this application that exceeding the power budget of printed batteries.

6 CONCLUSION

Printed electronics, particularly classifier circuits, offer transformative potential in applications requiring flexible substrates and ultra-low costs. Despite recent advancements, practical implementation faces challenges due to low integration density and limited power from printed batteries and harvesters. Our research approaches these issues holistically, focusing on optimizations from the sensor-processor interface to the processor itself. To our knowledge, we present the first open-source end-to-end digital printed classifier that meets both resource and power constraints Key contributions include: replacing ADCs with more efficient ABCs, customizable to specific input features; implementing bespoke ternary neurons, without costly multipliers and with approximate popcount and popcount-compare units; developing and deploying a multi-stage evolutionary optimization method that assists both in the approximation of the popcount and popcount-compare circuits and in their integration into a complete TNN accelerators. When sensor-processor interface costs are factored in, our results show a 32× improvement in area and 34× reduction in power compared to current state-of-the-art approximate digital printed neural network implementations, while maintaining comparable accuracy across diverse datasets.

ACKNOWLEDGMENTS

This work was supported by the Czech Science Foundation project 22-02067S, by the European Research Council (ERC), and by the H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" (H.F.R.I. Project Number: 17048). The authors thank Ampere for server support.

REFERENCES