Embedding Hardware Approximations in Discrete Genetic-based Training for Printed MLPs

Florentia Afentaki*, Michael Hefenbrock†, Georgios Zervakis*, Mehdi B. Tahoori‡
*University of Patras, Greece, †RevoAI GmbH, Germany, ‡Karlsruhe Institute of Technology, Germany,
*{afentaki, zervakis}@ceid.upatras.gr, †{michael.hefenbrock}@revoai.de, ‡{mehdi.tahoori}@kit.edu

***Abstract*—Printed Electronics (PE) stands out as a promising technology for widespread computing due to its distinct attributes, such as low costs and flexible manufacturing. Unlike traditional silicon-based technologies, PE enables stretchable, conformal, and non-toxic hardware. However, PE are constrained by larger feature sizes, making it challenging to implement complex circuits such as machine learning (ML) classifiers. Approximate computing has been proven to reduce the hardware cost of ML circuits such as Multilayer Perceptrons (MLPs). In this paper, we maximize the benefits of approximate computing by integrating hardware approximation into the MLP training process. Due to the discrete nature of hardware approximation, we propose and implement a genetic-based, approximate, hardware-aware training approach specifically designed for printed MLPs. For a 5% accuracy loss, our MLPs achieve over 5× area and power reduction compared to the baseline while outperforming state-of-the-art approximate and stochastic printed MLPs.**

***Index Terms*—Approximate computing, Electrolyte-gated FET, Multilayer Perceptron, Printed Electronics**

I. Introduction

Printed electronics (PE) offer a promising direction for integrating computing and intelligence across domains like smart bandages, disposable items, packaged goods, and smart packaging. These applications, including in-situ monitoring and the Fast-Moving Consumer Goods market, have specific fabrication cost, conformity, and time-to-market needs that traditional silicon-based electronics struggle to meet [1].

PE [2] enables these applications through additive manufacturing processes, producing conformal, low-cost hardware on demand. However, PE cannot match silicon VLSI systems in density, area, or speed due to larger feature sizes from imprecise printing. PE circuits operate at frequencies ranging from a few Hz to a few kHz [3], with micrometer-sized features [4]. The large feature size and the capacitance in PE technology raise area and power consumption, deterring conventional digital architectures, like Machine Learning (ML) classifiers [2] which are the primary printed applications [5].

As an attempt to address the above limitations, the authors in [2] leverage the customization potential offered by low-fabrication and Non-Recurring Engineering costs associated with printed circuits through bespoke circuit designs. The term "bespoke" refers to fully-customized circuits tailored to specific ML model and dataset. While [2] successfully reduced area and power for simple ML algorithms, the hardware overheads remain prohibitive for more complex algorithms like Multilayer Perceptron (MLP). Table I presents the hardware cost of several printed MLP circuits that follow the bespoke design of [2]. As shown, all MLPs feature excessive power consumption and cannot be powered by any available printed battery [2], [6] while their area requirement is above 12cm^2, thus being unsuitable for the most printed applications [5].

Targeting printed MLPs, [5]–[8] employed Approximate Computing (AxC) exploiting the high error resilience of ML applications [9]. Leveraging that in bespoke ML circuits the model's coefficients are hardwired and thus determine the circuit's area, [6] and [5] replace the MLP's coefficients with more area-efficient values reducing the multipliers' area. Additionally, to reduce the cost of additions, [6] applied gate-level pruning while [5] used truncation in accumulations. Armeniakos et al. [7], extend [6] by applying voltage over-scaling. In all the above works, multipliers are still required and thus the obtained gains are limited. In contrast, [10] designed stochastic printed MLPs but resulted in poor accuracy.

Prior works focus on employing AxC post-training, often compromising the balance between area and accuracy, thereby yielding sub-optimal results. In this work, we address this limitation by incorporating hardware approximation during MLP training. Gradient-based learning (backpropagation) relies on the loss function differentiability. Unfortunately, hardware approximations often involve discrete variables and decisions, which do not permit computing the gradients [11]. Additionally, to achieve hardware-aware training, both accuracy and area overhead must be addressed as objectives, framing the training process as a multi-objective optimization problem. Traditional gradient-based learning methods are not directly applicable to multi-objective optimization and often necessitate to be transformed to a single-objective optimization problem [12]. Therefore, we employ evolutionary methods like a Genetic Algorithm (GA), which, due to their capability to work with discrete variables, enable us to fully leverage hardware AxC techniques during our training. AxC circuits' inherent attributes and evolutionary circuit design principles are expected to synergize positively [13].

We propose a GA-based approach for training hardware and approximation-aware bespoke printed MLPs. While evolutionary approaches can be time-consuming for training large neural networks, for the small MLPs typically used in printed electronics [2], [5], GA-based training offers a promising alternative. It allows highly optimized bespoke MLP circuits by incorporating discrete hardware-aware approximations during training, enabling simultaneous optimization of accuracy and hardware costs within a reasonable time frame.

TABLE I
EVALUATION OF THE BASELINE PRINTED MLPS

MLP	Baseline				
	Topology[1]	Parameters[2]	Acc[3]	Area (cm^2)	Power (mW)
Breast Cancer	(10,3,2)	38	0.980	12.0	40.0
Cardio	(21,3,3)	78	0.881	33.4	124
Pendigits	(16,5,10)	145	0.937	67.0	213
RedWine	(11,2,6)	42	0.564	17.6	73.5
WhiteWine	(11,4,7)	83	0.537	31.2	126

[1]MLP topology. [2]MLP parameters. [3]Accuracy.

For approximations, we adopt the state-of-the-art power-of-two (pow2) weight quantization, to eliminate multiplications, and a finer-grained unstructured pruning, implemented through Full-Adders (FAs) removal, to a approximate additions.

To the best of our knowledge, this is the first time that such a framework[1] is proposed for hardware-efficient printed MLP circuits. Our experiments across various MLPs show that our framework reduces both area and power by over $5\times$ compared to the exact baseline and outperforms the current state-of-the-art approximate works. Notably, it enables printed-energy-harvester operation for the majority of the examined MLPs.

II. BACKGROUND ON PRINTED ELECTRONICS

PE employs various printing methods such as jet, screen, or gravure printing [14]. These printing techniques are characterized by being mask-less, portable, and additive in nature, leading to significant reductions in manufacturing costs and production timelines [15]. PE technology is being categorized into two main manufacturing approaches: additive and subtractive processes. The additive manufacturing process consists of deposition steps, where the functional materials are directly deposited on the substrate, while the subtractive process integrates both additive and subtractive stages, similar to methodologies seen in silicon-based techniques [9].

The simplicity and low equipment costs of additive manufacturing process, enable the production of remarkably low-cost electronic circuits, even sub-cent levels. However, this process is characterized by low precision fabrication which results in increased device latency and low integration density compared to silicon VLSI systems [9]. Though, target applications pose relaxed frequency and computational precision requirements, making printed circuits viable [9]. We focus on the Electrolyte-Gated FET (EGFET) technology, which features low supply voltage ($\leq 1\mathrm{V}$) and good mobility characteristics, making it well-suited for battery-powered applications [1].

III. APPROXIMATE COMPUTING TECHNIQUES

AxC promises significant area and power gains at the cost of some accuracy loss. AxC has found widespread use in ML applications, particularly due to the increased computational demands and the inherent approximate nature in ML models [9]. Considering that in printed ML circuits feasibility is the fundamental concern, preceding the need for high accuracy, in our work we design approximate printed MLPs targeting to minimize the associated hardware overheads. Hereafter, we briefly discuss the selection and hardware impact of the approximation techniques adopted in our work.

[1]https://github.com/floAfentaki/Approximation-Techniques-Targeting-Printed-MLPs

A. Mutliplier Approximation

The primary contributor to area consumption within a neuron is the multiplier, closely followed by the accumulator [11]. Consequently, the current state-of-the-art approximate printed ML approaches [5]–[7] primarily target to diminish the multipliers area. Nevertheless, despite the efforts made, multipliers are still required in [5]–[7], consuming considerable area. To address this, we propose to take a step further and implement multiplier-less neurons by adopting the well-known approximation of quantizing the weights as powers-of-two. As a result, each weight $w_{i,j}^{(l)}$, of the i-th input inside the j-th neuron of the l-th layer, is written as:

$$w_{i,j}^{(l)} = s_{i,j}^{(l)} \cdot 2^{k_{i,j}^{(l)}} \quad \text{with} \quad k_{i,j}^{(l)} \in [0, n-1), \tag{1}$$

where $s_{i,j}^{(l)} \in \{-1, +1\}$ is the sign of the weight, and n determines the number of bits used for weight representation. Assuming a bespoke implementation, as the design standard in PE [2], [5], multiplication by a constant pow2 weight requires only wiring. Hence, the cost of multipliers is effectively nullified. The sign s determines whether the product (input by pow2 weight) will be added or subtracted. Still, since the inputs are always positive (QReLU activation is used) and the weight sign is also fixed, the product will always be added or always subtracted. Therefore, if $s = -1$ only wiring and a few `NOT` gates are required since the '1' from all two's complement negations may be accumulated in the constant bias term before design, leading to no additional overhead. As typical, we also use low-bitwidth quantized biases denoted $b_j^{(l)}$.

B. Adder approximation

After eliminating the multipliers, the neuron's area is mainly determined by the area of the multi-operand adder. Unstructured pruning is a widely used technique to compress a model and reduce its complexity. Unstructured pruning removes connections and thus, in our printed MLPs where multipliers are already removed, unstructured pruning would directly remove a summand from the addition circuit. However, in small MLPs, like the targeted ones, where only a few weights are considered, pruning can lead to unacceptable accuracy loss while yielding only moderate hardware gains [8].

To address this, we adopt a more fine-grained approach to unstructured pruning, to strike a better balance between accuracy and the reduction of adder area. Instead of completely removing an entire connection, we selectively eliminate only certain bits. For example, if an input activation is a 6-bit binary signal $A = a_5a_4a_3a_2a_1a_{0_2}$, instead of nullifying the entire signal A, we nullify some of its bits, e.g., $A' = a_50a_3a_20a_{0_2}$. Again, since we design bespoke MLPs, accumulating multiple predefined '0' values in the same column of the adder tree will decrease the number of full-adders (FAs) required and potentially reduce its height. For instance, for every three constant '0' in a column, one FA is eliminated from that column and one carry connection to the right column is also

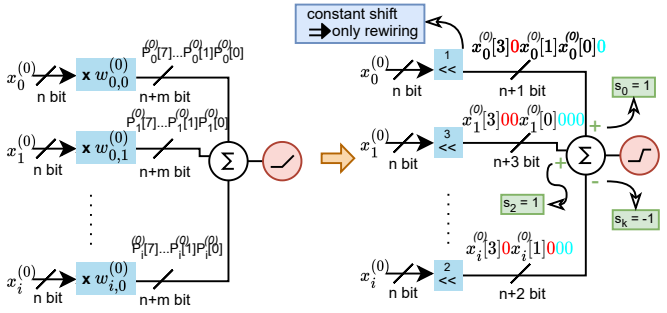

Fig. 1. Showcase of the approximate neuron. The figure on the left and right present the bespoke neuron before and after the hardware approximations.

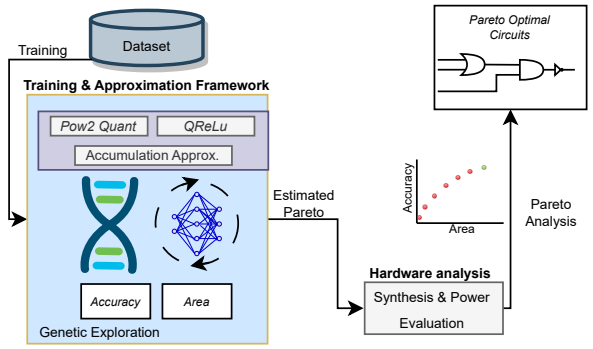

Fig. 2. Abstract high-level overview of our proposed framework.

removed. Similarly to unstructured pruning our approach can also be implemented using masking. For each weight $w_{i,j}^{(l)}$ we need to identify a mask $m_{i,j}^{(l)}$. For each '1' bit in $m_{i,j}^{(l)}$, the respective bit of the input activation is retained for the addition, while for each '0' bit in $m_{i,j}^{(l)}$, the respective bit of the input activation is removed. Assuming that $x_i^{(l)}$ is the input activation of the l-th layer, which is multiplied with $w_{i,j}^{(l)}$, then the mask is applied to the input $x_i^{(l)}$ as $x_i^{(l)} \odot m_{i,j}^{(l)}$, where $\odot$ is the the bitwise AND operation. Thus, the mask of the previous example is $m = 101101_2$ and $A' = A \odot m$. If a mask is zero, the entire summand is removed. Therefore, in (1), there's no need to define a zero weight, as it is hardware-equivalent to a zero mask. The size of the masks depends on the size of the input features of each layer. Moreover, we use the QReLU activation. Unlike ReLU activation, which yields unbounded outputs, QReLU effectively limits the size of its output, resulting in smaller required bitwidths. For our analysis, we use 4 bits for the inputs and 8 bits for the QReLU output. These values are small enough and result in almost no accuracy degradation compared to larger bitwidths. Hence, only small integer values are required to represent each mask $m_{i,j}^{(l)}$. Notably, the masks are used solely for the high-level representation of the applied approximation, e.g., training. In terms of hardware, there is no need for an AND gate for masking; the masked bits are directly removed from the adder.

C. Approximate Neuron

An illustrative example of applying the aforementioned approximation techniques are depicted in Fig. 1. For readability, this example assumes 4-bit activations. As shown in Fig. 1, multiplication is achieved through direct wiring of inputs, while addition approximation involves hardcoding zeros in the summand description, and the sign is also hardcoded.

As aforementioned, the primary factor influencing an approximate MLP's area is the accumulations involving multi-operand adder trees. Thus, a straightforward estimate for the MLP's area is the summation of the areas of these adder trees:

$$\text{Area}(\boldsymbol{\theta}) = \sum_{\forall l,j} \text{AdderArea}(\boldsymbol{\theta}_j^{(l)}), \tag{2}$$

where, $\boldsymbol{\theta}$ represents the approximate MLP and comprises all the aforementioned parameters, i.e., $m_{i,j}^{(l)}, s_{i,j}^{(l)}, k_{i,j}^{(l)}, b_j^{(l)}, \forall i, j, l$, and $\boldsymbol{\theta}_j^{(l)}$ represents the approximate neuron j of layer l and includes all the relevant parameters for that specific neuron. For each approximate MLP, $\boldsymbol{\theta} = \bigcup_{\forall l,j} \boldsymbol{\theta}_j^{(l)}$.

A simple but effective way to estimate the area of a multi-operand adder is counting the number of Full-Adders (FAs) it instantiates [16]. For simplicity, we assume only FAs for the reduction. Each FA performs a 3-to-2 reduction, meaning that for every three bits in a column, one bit remains, and one goes to the column to the right. Reduction is repeated until only two bits remain in each column. We implement a Python function to estimate $\text{AdderArea}(\boldsymbol{\theta}_j^{(l)})$ that takes as input the weights, masks, and bias of an approximate neuron $\boldsymbol{\theta}_j^{(l)}$, i.e., the parameters $m_{i,j}^{(l)}, s_{i,j}^{(l)}, k_{i,j}^{(l)}, b_j^{(l)}, \forall i$, calculates the non-zero bits in each column, and then recursively computes the number of required FAs.

IV. Proposed Framework

This section describes our framework, an abstract overview of which is depicted in Fig. 2. Our framework employs a discrete genetic-based training on a specified MLP topology and dataset to derive an estimated Pareto front of area-accuracy circuits, while applying the approximations discussed in Section III. Subsequently, a hardware evaluation of the evolved MLP circuits is conducted using EDA tools to identify the true Pareto-optimal circuits in terms of area and accuracy.

A. Hardware-Aware Training Flow

As depicted in Fig. 2, our framework implements a hardware approximation-aware training targeting efficient, in terms of accuracy and area, MLP classifiers. The training process essentially becomes a multi-objective optimization problem, taking into account both area and accuracy as objectives. As result, our optimization problem is a multi-objective minimization problem between area and classification error rate defined as:

$$\min_{\boldsymbol{\theta}} \; [\text{1-Accuracy}(\boldsymbol{\theta}, \mathcal{D}), \; \text{Area}(\boldsymbol{\theta})] \,, \tag{3}$$

where, $\mathcal{D}$ denotes the training data and, as defined above, $\boldsymbol{\theta}$ includes all learnable parameters $m_{i,j}^{(l)}, s_{i,j}^{(l)}, k_{i,j}^{(l)}, b_j^{(l)}, \; \forall i, j, l$. These parameters are represented in the discrete domain, which makes it infeasible, especially for the masks, to compute gradients and use traditional backpropagation training. Therefore, to handle the discrete space we implement a genetic-based training. The GA will not only explore weights

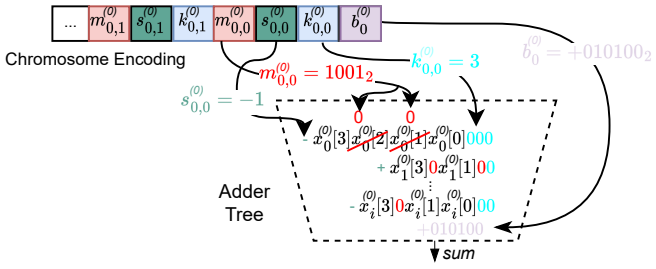

Fig. 3. Chromosome encoding and neuron construction.

$(s_{i,j}^{(l)}, k_{i,j}^{(l)})$ and biases $(b_j^{(l)})$ values but will also search the respective masks $(m_{i,j}^{(l)})$ to satisfy (3). Due to simplicity, low computational complexity, and enhanced convergence, we employ the Non-dominated Sorting Genetic Algorithm II (NSGA-II [17]) to address this multi-objective problem.

In the context of employing GA-driven training for MLPs, the weights' update process is contingent upon the genetic operations (i.e., mutation and crossover). The progress of training is influenced by the evolution of the optimization problem through natural selection, which involves a fitness function. In MLP training, the mutation operator introduces random alterations to neuron weights, while crossover combines winning weights. These genetic operations are applied randomly during the training process.

The area-accuracy Pareto-optimal points that the GA searches for, are consisted of chromosomes with the best combination of MLP masks, weights, and biases (learnable parameters in $\boldsymbol{\theta}$). When the genetic exploration ends, an estimated area-accuracy Pareto-optimal set is obtained. The trained coefficients and masks of the estimated Pareto front, are then automatically translated into an HDL description as discussed above and illustrated for example in Fig. 1. Next, hardware analysis is employed and the true Pareto front among the evaluated designs is obtained.

In order to facilitate the convergence of the evolutionary algorithm to a Pareto-optimal set, we create an initial population of semi-random chromosomes. This population is randomly selected and further doped with a small percentage (~10%) of nearly non-approximate solutions, exploring solutions of high accuracy at the early stages of evolution. Additionally, we impose a 10% upper bound on the acceptable accuracy loss, compared to the baseline accuracy [2], to consider solutions during the training that exhibit small accuracy degradation.

B. Formulation and Encoding of the Approximate Neuron

The approximations embedded during the training include pow2 weight quantization and fine-grain unstructured pruning. Integrating pow2 quantization into training simply restricts the solution space of the MLP's coefficients within pow2 values. As for the pruning method, we introduce an additional training parameter, i.e., the masks.

Treating the masks as a training parameter means that the corresponding decisions are incorporated into the exploration during training. Therefore, the output of the j-th neuron in the l-th layer of the hardware-approximated MLP is given by:

$$\text{QReLU}\left(\sum_i s_{i,j}^{(l)}(m_{i,j}^{(l)} \odot x_i^{(l)}) \ll k_{i,j}^{(l)} + b_j^{(l)}\right), \quad (4)$$

where, $\ll$ is the bit-shift operator, and $x_i^{(l)}$, $\forall i$, are input activations of the l-th layer. The rest are our learnable parameters.

Our genetic-based training must identify all the learnable parameters in $\boldsymbol{\theta}$ that minimize our objective function (3). The evaluated fitness function involves calculating the inference accuracy using (4) and estimating the hardware overhead using (2) and our high-level FA-count Python function.

Each gene in the GA chromosome represents a $m_{i,j}^{(l)}$, $s_{i,j}^{(l)}$, $k_{i,j}^{(l)}$, or $b_j^{(l)}$ parameter. An example of a chromosome's encoding is illustrated in Fig. 3 where the genes are grouped by weight $(m_{i,j}^{(l)}, s_{i,j}^{(l)}, k_{i,j}^{(l)})$ then by neuron, and finally by layer. Hence each gene is represented by an integer value (with the corresponding limits, e.g., weight size for $k_{i,j}^{(l)}$). Furthermore, Fig. 3 shows the direct relationship between the genes, the applied approximations, and the eventual hardware implementation, which highlights the hardware-awareness of our approach.

V. Results and Evaluation

In this section, we conduct a comprehensive evaluation of our framework. We evaluate the area and accuracy of the printed MLPs trained with our framework and compare them against the state-of-the-art exact baseline [2] as well as the proposed state-of-the-art printed MLPs [5], [7], [10]. Finally, we assess the effectiveness of our framework in enabling printed-battery-powered MLP classifiers.

A. Experimental Setup

We examine five datasets, namely Breast Cancer (BC), Cardiotocography (Ca), Pendigits (PD), Red Wine (RW), and White Wine (WW), from [18]. These datasets could form realistic printed applications as they feature inputs suitable for printed circuits and demand low precision, duty cycle, and sample rate requirements [1]. They have also been previously utilized in the related works [2], [5], [7], [10], ensuring a fair comparison for our analysis.

The inputs are normalized to $[0, 1]$ as in [2], [5] and are randomly stratified split into $70\%/30\%$ train/test sets, ensuring a balanced distribution of each target class in each of these sets. Mutation and crossover operators of the GA are set to 0.2% and 0.7% respectively. All circuits are synthesized using Synopsys Design Compiler S-2021.06 and mapped to the printed EGFET library [1], while VCS T-2022.06 and PrimeTime T-2022.03 are used for simulation and power analysis. The accuracy is reported on the test dataset, and all designs are synthesized at a relaxed clock period to improve area efficiency. Clock period of 200ms are applied to all MLPs, except for Pendigits, which requires 250ms. Such delay values align with typical PE performance [3]. The architecture of the MLPs is the same as the authors have reported in [2] and [5]. Our exact baseline are the bespoke printed MLPs circuits, designed following the approach outlined in [2], using

TABLE II
EVALUATION OF OUR PRINTED MLP FOR UP TO 5% ACCURACY LOSS.

MLP	Our Approximate MLPs				
	Accuracy	Area (cm^2)	Power (mW)	Area Reduction[1]	Power Reduction[1]
Breast Cancer	0.947	0.04	0.15	288×	274×
Cardio	0.873	1.73	6.5	19.3×	19.0×
Pendigits	0.893	12.7	40.2	5.3×	5.3×
RedWine	0.519	0.04	0.13	470×	579×
WhiteWine	0.508	0.20	0.74	122×	137×

[1] With respect to the corresponding bespoke exact baseline [2].

8-bit fixed point weights and 4-bit inputs. Their hardware characteristics, accuracy, and topology are reported in Table I. Given the large hardware overheads in printed circuits and the feasibility constraints of printed MLPs, we consider a 5% accuracy loss compared to the exact baseline [2] (see Table I) as an acceptable level of accuracy for our experiments.

B. Comparison Against the Baseline and State of the Art

First, we evaluate our framework and the state-of-the-art exact baseline MLPs [2]. Table II shows the area, power, and accuracy of our printed MLP that feature up to 5% accuracy loss. As shown, compared to the baseline, our MLP circuits achieve 181× and 203× area and power reduction on average, respectively. Notably the area gains range from 5.3× to 470× while the power gains range from 5.3× to 578×.

In Fig. 4, we present a comparison of the area and power gains of our printed MLPs compared with the state-of-the-art approximate [5], [7] and stochastic [10] ones. All MLPs in Fig. 4 feature the same inference latency. Despite a small clock period in [10], each inference takes 220-230ms due to the use of a stochastic bitstream of length 1024. For our circuits and [5], [10] a 1V operation is considered. In [7], Voltage Over-Scaling is used and the MLPs are operated below 0.8V. In Fig. 4, all values are normalized over the corresponding exact bespoke design [2]. Although, we and the authors in [5], [7] consider up to 5% accuracy loss compared to the baseline [2], it's worth noting that [10] cannot achieve such high accuracy. In fact, the average accuracy loss for the MLPs they consider in [10] is 35%.

As shown in Fig. 4, our framework significantly outperforms [5], [7] and [10]. Specifically, compared to [5], our MLPs achieve on average 13× area reduction and 14× power reduction. The area and power gains range from 1.8× to 36× and from 1.8× to 39×, respectively. Similarly, compared to [7], our MLPs achieve 25× lower area and 14.5× lower power on average. In [7] Pendigits is not considered, possible due to its high complexity. Finally, our MLPs deliver 19× and 26× area and power saving, respectively, compared to [10]. Only for Pendigits, the stochastic MLP of [10] attains slightly lower power and area than our approximate MLP. Though, [10] achieves only 22% accuracy while we achieve 89.3%.

C. Printed-Battery Power Operation

Next, we evaluate the effectiveness of our framework, in generating printed-battery powered MLP classifiers. As shown in Table II, Breast Cancer, RedWine, and WhiteWine classifiers are compatible with a Blue Spark 5mW battery, while Cardio can be powered by a Zinergy 15mW battery. Pendigits, with its larger topology requiring 145 parameters, cannot be powered even with a Molex 30mW battery.

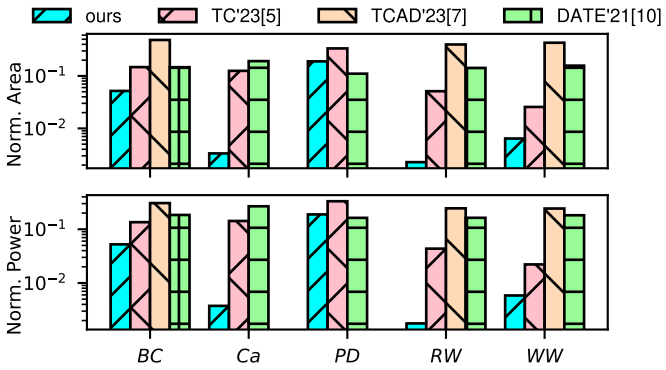

Fig. 4. (a) Area and (b) power reduction delivered by our printed MLPs and the state-of-the-art approximate [5], [7] and stochastic [10] ones. Area and power are normalized w.r.t. the baseline exact MLPs [2]. Y-axis is in logarithmic scale.

Our applied approximations result in faster MLPs compared to their exact baseline equivalents. This enables further power reduction by scaling the supply voltage without sacrificing performance, i.e., maintaining the same latency as the baseline circuit [2]. Hence, we can now power previously unpowered MLPs like Pendigits or use smaller printed energy sources. This is very advantageous for compact, self-powered designs like wearables. Implantable and wearable medical devices prioritize functionality, user comfort, and device longevity, thus harvesting energy from the body is highly efficient. Integrating energy harvesting ensures continuous operation [19].

Considering that EGFET printed circuits can operate down to 0.6V [20] and that printed batteries are customizable in terms of polarity, voltage, shape, etc. [21], we set the voltage supply of our approximate MLPs (MLPs in Table II) to the minimum supported value, i.e., 0.6V, and re-synthesize our designs. In Fig. 5, our MLPs, along with the baseline [2] and [5], are categorized based on their area and suitable power source. Again, the 5% accuracy constraint is considered. As shown in Fig. 4, among the state-of-the-art works, [5] outperformed [7], [10] in area-power gains and accuracy. In Fig. 5, the red zone denotes an unsustainable area where the circuit's size is considered excessively large for most printed applications or where there isn't an adequate power supply. The other zones use a different color based on the printed power source considered, e.g., printed battery Molex 30mW.

Fig. 5 demonstrates how our framework benefits the area-power trade-off by shifting the MLPs to a lower area-power space. As shown, all the baseline MLPs lie in the red zone while the approximate MLPs [5] mainly require large batteries. On the other hand, all our circuits, except for Pendigits lie within the green zone, indicating that they can be powered by only a printed energy harvester. Although our Pendigits MLP can be now powered by a Molex 30mW battery, its area might be impractical for most printed applications. Note

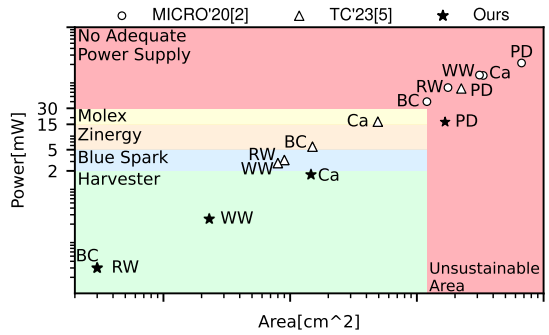

Fig. 5. Feasibility evaluation. Categorizing our printed MLPs, the baseline [2] and the approximate [5] ones based on their sustainability concerning area overhead and the availability of printed power sources.

that the Pendigits MLPs of [5] and [2] cannot be powered by any existing printed power source. On average, our MLPs at 0.6V achieve $912\times$ lower power compared to the baseline [2]. Similarly, compared to [5], our MLPs at 0.6V achieve $65\times$ lower power on average.

D. Execution Time Evaluation

Table III presents training execution time on an AMD EPYC 7552 with 256GB RAM. As expected, gradient-based training is faster than the conventional for the same number of evaluations, i.e., without approximation and hardware-awareness, GA-based training. Our approximate hardware-aware GA-based training averages only 100 minutes, even with over 26 million chromosome evaluations. This time is close to hardware-unaware conventional GA-based training. The runtime impact is minimal, albeit integrating the addition approximation into the training process, doubles the trainable parameters, necessitating a mask to be trained for each neuron's input. Overall, our approach features reasonable execution time, especially when considering the addressed limitations and complexity

VI. Conclusion

Printed electronics offer cost-effective, flexible, and conformal hardware, making it ideal for ultra-low-cost applications. However, the associated large feature size limits the feasibility of complex ML classifiers like MLPs. To overcome this, we integrate hardware approximation into the training and introduce a GA-based hardware-aware training method to design approximate bespoke printed MLP circuits. Our approach efficiently explores the discrete hardware approximation space, striking a balance between accuracy and hardware efficiency. Our evaluation shows that our printed MLPs outperform the state of the art in terms of hardware efficiency while maintaining similar accuracy. This advancement enables printed-battery-powered operation for all examined MLPs, with most of them being self-powered using a printed energy harvester.

Acknowledgment

This work is partially supported by the European Research Council (ERC) and co-funded 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).

TABLE III
Training Execution Times Evaluation in Minutes

MLP	Exec.Time Grad. (min)[1]	Exec.Time GA (min)[2]	Exec.Time GA-AxC (min)[3]
Breast Cancer	0.5	8	9
Cardio	2	42	45
Pendigits	14	298	344
RedWine	2	21	22
WhiteWine	7	77	79
Average	5	89	100

[1] Gradient with only accuracy as objective. [2] GA-based with only accuracy as objective. [3] GA-based with AxC techniques and both accuracy and area as objectives.

References

[1] N. Bleier, M. Mubarik, F. Rasheed, J. Aghassi-Hagmann, M. B. Tahoori, and R. Kumar, "Printed microprocessors," in *Annu. Int. Symp. Computer Architecture (ISCA)*, jun 2020, pp. 213–226.

[2] M. H. Mubarik *et al.*, "Printed machine learning classifiers," in *Annu. Int. Symp. Microarchitecture (MICRO)*, 2020, pp. 73–87.

[3] G. Cadilha Marques *et al.*, "Digital power and performance analysis of inkjet printed ring oscillators based on electrolyte-gated oxide electronics," *Applied Physics Letters*, vol. 111, no. 10, p. 102103, 2017.

[4] T. Lei *et al.*, "Low-voltage high-performance flexible digital and analog circuits based on ultrahigh-purity semiconducting carbon nanotubes," *Nature communications*, vol. 10, no. 1, p. 2161, 2019.

[5] G. Armeniakos, G. Zervakis, D. Soudris, M. B. Tahoori, and J. Henkel, "Co-design of approximate multilayer perceptron for ultra-resource constrained printed circuits," *IEEE Trans. Comput.*, pp. 1–8, 2023.

[6] G. Armeniakos, G. Zervakis, D. Soudris, M. B. Tahoori, and J. Henkel, "Cross-layer approximation for printed machine learning circuits," in *Design Automation and Test in Europe (DATE)*, 2022, pp. 190–195.

[7] G. Armeniakos, G. Zervakis, D. Soudris, M. B. Tahoori, and J. Henkel, "Model-to-circuit cross-approximation for printed machine learning classifiers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, pp. 1–1, 2023.

[8] A. Kokkinis *et al.*, "Hardware-aware automated neural minimization for printed multilayer perceptrons," in *Design Automation and Test in Europe (DATE)*, 2023.

[9] J. Henkel *et al.*, "Approximate computing and the efficient machine learning expedition," in *Int. Conf. on Computer-Aided Design (ICCAD)*, 2022, pp. 1–9.

[10] D. D. Weller *et al.*, "Printed stochastic computing neural networks," in *Design Automation and Test in Europe (DATE)*, 2021, pp. 914–919.

[11] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel, "Hardware approximate techniques for deep neural network accelerators: A survey," *ACM Comput. Surv.*, vol. 55, no. 4, nov 2022.

[12] H. Benmeziane, K. El Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, *A Comprehensive Survey on Hardware-Aware Neural Architecture Search*, Jan. 2021. [Online]. Available: https://uphf.hal.science/hal-03269441

[13] Z. Vasicek and L. Sekanina, "Evolutionary approach to approximate digital circuits design," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 432–444, 2015.

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

[15] J. S. Chang, A. F. Facchetti, and R. Reuss, "A circuits and systems perspective of organic/printed electronics: review, challenges, and contemporary and emerging design approaches," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 7, no. 1, pp. 7–26, 2017.

[16] N. H. Weste and D. Harris, *CMOS VLSI design: a circuits and systems perspective*. Pearson Education India, 2015.

[17] D. Kalyanmoy, P. Amrit, A. Sameer, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[18] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[19] M. M. H. Shuvo, T. Titirsha, N. Amin, and S. K. Islam, "Energy harvesting in implantable and wearable medical devices for enduring precision healthcare," *Energies*, vol. 15, no. 20, p. 7495, 2022.

[20] C. Marques *et al.*, "Progress Report on "From Printed Electrolyte-Gated Metal-Oxide Devices to Circuits"," *Advanced Materials*, vol. 31, 2019.

[21] S. Lanceros-Méndez and C. M. Costa, *Printed Batteries: Materials, Technologies and Applications*. Wiley, 2018.