Cross-Layer Approximation For Printed Machine Learning Circuits

Giorgos Armeniakos*, Georgios Zervakis†, Dimitrios Soudris*, Mehdi B. Tahoori†, and Jörg Henkel†
*National Technical University of Athens, Greece, †Karlsruhe Institute of Technology, Germany
*{armeniakos, dsoudris}@microlab.ntua.gr, †{georgios.zervakis, mehdi.tahoori, henkel}@kit.edu,

***Abstract*—Printed electronics (PE) feature low non-recurring engineering costs and low per unit-area fabrication costs, enabling thus extremely low-cost and on-demand hardware. Such low-cost fabrication allows for high customization that would be infeasible in silicon, and bespoke architectures prevail to improve the efficiency of emerging PE machine learning (ML) applications. However, even with bespoke architectures, the large feature sizes in PE constraint the complexity of the ML models that can be implemented. In this work, we bring together, for the first time, approximate computing and PE design targeting to enable complex ML models, such as Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs), in PE. To this end, we propose and implement a cross-layer approximation, tailored for bespoke ML architectures. At the algorithmic level we apply a hardware-driven coefficient approximation of the ML model and at the circuit level we apply a netlist pruning through a full search exploration. In our extensive experimental evaluation we consider 14 MLPs and SVMs and evaluate more than 4300 approximate and exact designs. Our results demonstrate that our cross approximation delivers Pareto optimal designs that, compared to the state-of-the-art exact designs, feature 47% and 44% average area and power reduction, respectively, and less than 1% accuracy loss.**

***Index Terms*—Approximate Computing, Machine Learning, Printed Electronics**

I. Introduction

Printed electronics emerge as a promising solution for application domains such as smart packaging, disposables (e.g., packaged foods, beverages), fast moving consumer goods (FMCG), in-situ monitoring, low-end healthcare products (e.g., as smart bandages) etc [1]. Such domains have requirements for ultra-low cost and conformality that silicon-based systems cannot satisfy. For example, increased manufacturing, packaging, and assembly costs of silicon prevent sub-cent costs while silicon systems cannot meet either stretchability, porosity, and flexibility requirements [2].

Although, printed circuits promise to satisfy these demands through their ultra-low cost additive manufacturing that enables conformal hardware on-demand, the large feature sizes in printed electronics mean that the associated hardware overheads will be prohibitive for complex circuits. The latter include implementations of machine learning (ML) classification algorithms that are required by a large number of printed applications in aforementioned domains [1]. Hence, works on printed ML classifiers are limited.

In [1], the authors exploit i) the hardware efficiency of bespoke architectures [3] and ii) the fact that the non-recurring engineering and low fabrication costs of printed electronics enable on-demand printing of ML circuits customized to a specific model and implement bespoke printed classifiers, paving the way for ML classification even on printed electronics. Such degree of customization is mainly infeasible in silicon systems. However, due to area and power constraints, [1] implemented only simple ML algorithms such as Decision Trees and Support Vector Machine Regression (SVM-Rs).

In this paper, we investigate the possibility of implementing more complex printed ML circuits and specifically Multi-Layer Perceptrons classifiers (MLP-Cs), Multi-layer Perceptron regressors (MLP-Rs), SVM classifiers (SVM-C), as well as SVM-Rs in printed electronics. To this end, to further improve the area efficiency compared to the state-of-the-art bespoke ML implementations [1], we adopt, for the first time, Approximate Computing (AC) principles in printed electronics. Approximate computing exploits the intrinsic error resilience of a large number of application domains, such as ML, and trades computational accuracy for gains in other metrics, such as area and power [4]. Designing approximate arithmetic circuits has gained a vast research interest [5], [6]. Moreover, to mitigate the increased complexity of approximate design, several works focus on approximate design automation [7]–[10]. Notably, significant research interest is shown in gate-level pruning approaches [11], [12] due to its inherent efficiency in reducing a circuit's area complexity. Gate-level pruning is a circuit agnostic technique that operates on an already optimized netlist and further reduces its area by removing selected gates. Though, netlist pruning state of the art mainly targets arithmetic circuits only. In addition, research activities on approximate bespoke circuits are very limited.

In this work, we leverage cross-layer approximation, that is proven to outperform single layer (either logic or algorithmic) techniques [4], [10], and we propose and implement an automated cross-approximation framework customized for approximate bespoke ML circuits. At the algorithmic level, our framework applies coefficient approximation in which the coefficients of a given trained model are replaced with more (bespoke) hardware-friendly values. At the logic level, we implement a netlist pruning approach that is customized for printed bespoke architectures and through a full search design space exploration it extracts Pareto-optimal solutions. Using our framework, we elucidate the impact of approximate computing on designing complex printed ML circuits. We demonstrate that, compared to the state-of-the-art exact printed ML circuits, our framework decreases the area (power) by 47%

TABLE I
Evaluation of Bespoke Printed ML Circuits.

	MLP-C					MLP-R					SVM-C					SVM-R				
	Acc[1]	T[2]	#C[3]	Area (cm^2)	Power (mW)	Acc[1]	T[2]	#C[3]	Area (cm^2)	Power (mW)	Acc[1]	T[2]	#C[3]	Area (cm^2)	Power (mW)	Acc[1]	T[2]	#C[3]	Area (cm^2)	Power (mW)
Cardio	0.88	(21,3,3)	72	33.4	97.3	0.83	(21,3,1)	66	21.6	65.9	0.90	3	63	15.1	46.8	0.84	1	21	6.8	22.9
Pendigits	0.94	(16,5,10)	130	67.0	213.0	0.37	(16,5,1)	85	-[4]	-	0.98	45	160	123.8	364.4	0.23	1	16	-[4]	-
RedWine	0.56	(11,2,6)	34	17.6	53.3	0.56	(11,2,1)	24	7.1	24.0	0.57	15	66	23.5	72.9	0.56	1	11	4.0	15.1
WhiteWine	0.54	(11,4,7)	72	31.2	98.4	0.53	(11,4,1)	48	13.1	40.7	0.53	21	77	28.3	87.4	0.53	1	11	4.2	15.5

[1] Accuracy using 8-bit coefficients and 4-bit inputs. [2] Model's topology (for SVMs: the number of classifiers). [3] Number of coefficients of the model.
[4] These models achieve low accuracy and are not evaluated.

(44%) on average. In many cases, these gains are sufficient to enable complex printed ML circuits. It is noteworthy that our framework requires 12min on average. The latter is critical for printed electronics due to their on-demand and point-of-use–even at low to moderate volumes–fabrication process.

Our novel contributions within this work are as follows:

1) This is the first work that evaluates the impact of approximate computing on printed electronics specifically when targeting printed ML circuits.
2) We propose, for the first time, an automated, cross-layer approximation framework for bespoke ML circuits.
3) Using our framework, we demonstrate that, in many cases, approximate computing can be used to integrate complex ML models in printed circuits[1].

II. Background

Printed electronics denotes a set of printing methods which can realize ultra low-cost [13], large area [14] and flexible [15] computing systems in combination with functional materials to realize transistors and passive components on various substrates. There are different processes for the fabrication of printed circuits, such as screen printing, jet-printing or roll-to-roll processes [16]. All these printing techniques refer to an additive manufacturing process, where functional materials are directly deposited on the substrate, which simplifies the production chain compared to subtractive silicon-based processes substantially [17]. This leads to savings in per unit-area costs and enables flexible hardware form factors.

Printed electronics do not compete with silicon-based electronics in terms of integration density, area and performance. Typical frequencies achieved by printed circuits are from a few Hz to a few kHz [18]. Similarly, the feature size tends to be several microns [19]. However, due to its form-factor, conformity and most importantly, significantly reduced fabrication costs – even for low quantities – it can target application domains, unreachable by conventional silicon-based VLSI.

Despite these attractive features, there are several limitations of printed electronics compared to traditional silicon technologies. Due to large feature sizes, the integration density of printed circuits is orders of magnitude lower than silicon VLSI. Additionally, due to large intrinsic transistor gate capacitances, the performance of printed circuits is orders of magnitude lower compared to nanometer technologies.

III. Cross-Layer Approximation For Printed ML Bespoke Architectures

In this section we present our automated cross-layer approximation framework for generating approximate printed ML circuits. Briefly, our framework receives as input a trained model (e.g., dumped from scikit-learn) and performs a hardware-driven coefficient approximation (algorithmic level). Then, it automatically generates the bespoke RTL description of the approximated model and synthesizes the circuit using the EDA tool and printed PDK. Finally, on the synthesized netlist, it applies logic approximation through gate-level pruning.

A. Bespoke ML Architectures

As aforementioned, printed electronics enable bespoke circuits customized per ML model [1]. In such customized implementations, the coefficients are hardwired in the circuit leading to high area efficiency. Hence, as a baseline for our work, we consider an area-optimized fully-parallel bespoke circuit for each ML model examined, following the design methodology of [1]. Specifically, we examine MLP-C, MLP-R, SVM-C, and SVM-R models trained on the Cardiotocography, Pendigits, RedWine, and WhiteWine datasets of the UCI ML repository [20]. Training is performed using scikit-learn and the randomized parameter optimization (RandomizedSearchCV) with 5-fold cross validation. Inputs are normalized to $[0, 1]$ and training/testing uses a random 70%/30% split. For the MLPs one hiden layer with up to five neurons and Relu activiation function are used. SVMs use a linear kernel and SVM-Cs are implemented with 1-vs-1 classification. We set the topology of each MLP so that each MLP features the least number of hidden nodes and all MLPs achieve close to maximum accuracy. To generate each bespoke circuit, the coefficients and intercepts are hardwired in the RTL description and fixed-point arithmetic is used. The precision for the coefficients and inputs is set to 8 and 4 bits, respectively, since these values delivered close to floating-point accuracy for all the models. The RTL descriptions are synthesized using Synopsys Design Compiler and mapped to the open source Electrolyte Gated Transistor (EGT) library [2], which is an inkjet-printed technology. EGT is low-voltage and allows battery powered printed circuits. All the circuits are synthesized at a relaxed clock (i.e., 250ms for Pendigits MLP-C and 200ms for the rest circuits) targeting to further improve the area efficiency. These delay values are in compliance with typical performance

[1] Our implementations are available at https://github.com/XX/XX

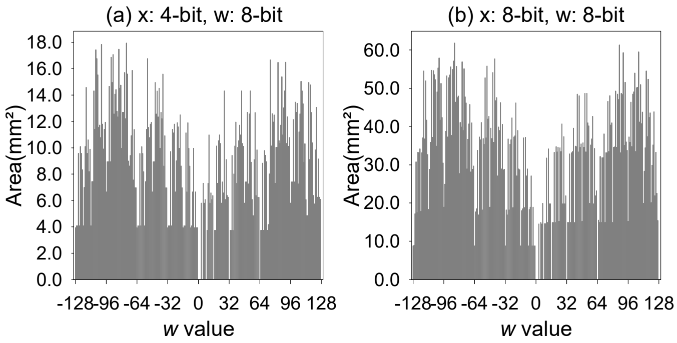

Fig. 1. The area of the bespoke multiplier w.r.t. the coefficient value w. Two architectures are considered: a) 4-bit inputs and 8-bit coefficients and b) 8-bit inputs and 8-bit coefficients. For reference the area of the conventional 4×8 and 8×8 multipliers is $83.61mm^2$ and $207.43mm^2$, respectively.

of printed electronics [18]. Circuit simulations are performed with Questasim and the test dataset to obtain the circuit's switching activity. Then, power estimation is performed using Synopsys PrimeTime and the obtained switching activity. The aforementioned EDA tool-flow is used throughout our paper.

Table I reports the characteristics (e.g., area, power) for our baseline bespoke implementations of all the considered models. As shown, the area of almost all the circuits ($> 20\text{cm}^2$) is prohibitive for most printed applications. Similarly, their power consumption is so high ($>$ 30mW) that they cannot be powered by a single existing printed battery. Only SVM-Rs and the RedWine MLP-R exhibit acceptable area and power.

B. Hardware-Driven Coefficient Approximation

A weighted sum (as for example example in the case of MLPs and SVMs) is expressed as:

$$S = \sum_{1 \leq i \leq N} x_i \cdot w_i, \tag{1}$$

where w_i are the predefined coefficients (or weights) obtained after training, x_i are the inputs, and N is the number of coefficients. In the case of bespoke ML architectures, these coefficients are hardwired within the circuit [1]. As a result, the area (and power) of each bespoke multiplier BM_w required to compute the product $x \cdot w$, $\forall x$, is determined by the value of the coefficient w and the width of the input x. For example, Fig. 1 presents the area of BM_w, $\forall w \in [-128, 127]$ (i.e., 8-bit coefficients), for 4-bit and 8-bit input values. For completeness, in the caption of Fig. 1 we also report the area of the conventional 4×8 and 8×8 multipliers. In both cases, the bespoke multipliers BM_w feature significantly lower area than the conventional multiplier for all the w values. Moreover, it is evident that the area of BM_w highly depends on w and the input bitwidth. However, similar trend is observed in Fig 1a and 1b, i.e., neighbouring w values may feature significantly different area. Importantly, in many cases the area may be nullified, e.g., when w is a power of two. This motivates us to investigate and propose a hardware-driven coefficient approximation, tailored for bespoke architectures, that replaces a coefficient value w with a neighbouring value $\tilde{w}$ so that $\texttt{AREA}(\text{BM}_w) < \texttt{AREA}(\text{BM}_{\tilde{w}})$.

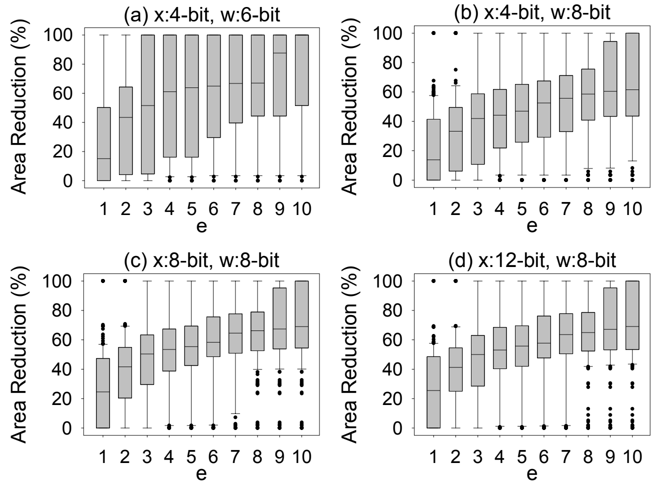

Fig. 2. The area reduction delivered by our coefficient approximation when $(w - \tilde{w}) \leq e$. Several bespoke multipliers are considered (a-d).

Fig. 2 presents the area reduction that is achieved by our coefficient approximation with respect to several bespoke multipliers sizes (a-d). To generate each boxplot in Fig. 2, for all w, we select $\tilde{w}$ so that $\tilde{w}$ features the lowest $\texttt{AREA}(\text{BM}_{\tilde{w}})$ and $\tilde{w} \in [w - e, w + e]$, where e is a given threshold (x-axis). Clipping is applied at the borders. As shown in Fig. 2, the obtained $\text{BM}_{\tilde{w}}$ feature significantly lower area than the BM_w. Our coefficient approximation delivers a median area reduction of more than 19% when $e = 1$ while this value increases to 53% when $e = 4$. Nevertheless, in most cases, for $e \geq 4$ the area reduction becomes less significant. For example, in Fig. 2b, the median area reduction is 44% for $e = 4$ and increases to only 61% for $e = 10$. In many cases, in Fig. 2, the area reduction goes up to 100% or it is 0%. The former is explained by the fact that w was replaced by $\tilde{w}$ that was a power of two and thus the area reduction is 100% since $\tilde{w}$ features zero area. On the other hand, in the cases that w features the lowest area in the segment $[w - e, w + e]$, w is not replaced and the area reduction is zero.

When replacing w by $\tilde{w}$, the multiplication error is equal to $x \cdot (w - \tilde{w})$. Thus, the error ϵ_S of the weighted sum is:

$$\epsilon_S = \sum_{1 \leq i \leq N} x_i \cdot (w_i - \tilde{w_i}) \tag{2}$$

Considering positive inputs (see Section III-A), by systematically selecting $\tilde{w_i}$ to balance the positive and negative errors (i.e., $w_i - \tilde{w_i}$), we can minimize (2).

Given an MLP or SVM, we implement our proposed hardware-driven coefficient approximation as follows:

1) Given the coefficients w_i and the bitwidth of the inputs, we evaluate the area of all the bespoke multipliers ($\texttt{AREA}(\text{BM}_{\tilde{w}})$), $\forall i$ and $\forall \tilde{w} \in [w_i - e, w_i + e]$. This step uses Synopsys Design Compiler and the EGT PDK [2] for circuit synthesis and area analysis.

2) For all the coefficients w_i, create a set $R_i = \{\tilde{w}_i^-, \tilde{w}_i^+\}$ s.t. $\tilde{w}_i^- \in [w, w+e]$ and $\tilde{w}_i^-$ features the smallest area in that segment, i.e., $\texttt{AREA}(\mathrm{BM}_{\tilde{w}_i^-})$ = min($\texttt{AREA}(\mathrm{BM}_{\tilde{w}_i})$), $\forall \tilde{w} \in [w, w+e]$. Similarly, we select $\tilde{w}_i^+ \in [w-e, w]$. By definition replacing w with $\tilde{w}_i^-$ generates a negative error while replacing w with $\tilde{w}_i^+$ generates a positive error.
3) We perform a brute-force search to select the configuration $\{\tilde{w}_i : \tilde{w}_i \in R_i, \forall i\}$ so that $\sum_{\forall i}(w_i - \tilde{w}_i)$ is minimized. In case of a tie, we select the one that minimizes $\sum_{\forall i}\texttt{AREA}(\mathrm{BM}_{\tilde{w}_i})$.

Note that steps 1-3 are executed for each weighted sum, i.e., neuron in MLPs and 1-vs-1 classifier in SVMs. In addition, we set e=4 in our analysis since for $e > 4$ the area gains quite saturate (see Fig. 2). At the worst case, step 1 required less than 6s using 12 threads (i.e., limit of available licenses). In step 3 we implement an exhaustive search to extract the final configuration. Unlike conventional silicon VLSI, in printed electronics the examined ML models are rather small in size (in terms of number of parameters). Hence, each weighted sum (neuron or classifier) features only a limited number of coefficients, i.e., the size of the design space is well constrained. It is noteworthy, that in the worst case, step 3 required only 5s using 80 threads. The aforementioned execution times refer to a dual-CPU Intel Xeon Gold 6138 server. In our optimization (steps 1-3) the sum $\sum_{\forall i}\texttt{AREA}(\mathrm{BM}_{\tilde{w}_i})$ is used as a proxy of the area of the weighted sum. In other words, by minimizing the area (through our coefficient approximation) of the required bespoke multipliers, we aim in minimizing the area of the weighted sum. We evaluate our area proxy against 1000 randomly generated weighted sum circuits (i.e., random coefficients and input sizes). The Pearson correlation coefficient between the area of the weighted sum obtained by Design Compiler and the area estimation using $\sum_{\forall i}\texttt{AREA}(\mathrm{BM}_{\tilde{w}_i})$ is 0.91, i.e., perfect linear correlation. Hence, our proxy precisely captures the area trend and minimizing $\sum_{\forall i}\texttt{AREA}(\mathrm{BM}_{\tilde{w}_i})$ in our optimization, will result in a weighted sum circuit with minimal area. Finally, since our technique replaces the coefficient values with approximate more hardware-friendly ones, it does not require any specific/custom hardware implementation (e.g., as usually done in logic approximation). Hence, it can be seamlessly integrated in any design framework and exploits all the optimization and IPs (e.g., multipliers) of synthesis tools.

C. Netlist Pruning

To further increase the area efficiency, in addition to our coefficient approximation, we apply netlist pruning. Netlist pruning is based on the observation that the output of several gates in a netlist remains constant (‘0’ or ‘1’) for the majority of the execution time. Hence, removing such a gate from the netlist and replacing its output with a constant value, results in low error rate. Netlist pruning has been widely studied to enable circuit-agnostic approximation [11], [12]. In this section we provide a brief description of how we implemented and tailored netlist pruning for bespoke printed ML architectures. First we define two pruning parameters for a gate: τ is the maximum percentage of time that the gate's output is ‘0’ or ‘1’ and ϕ the most significant output bit (starting from 0) that the gate is connected to (through any path). Using τ and ϕ we can constraint the error frequency and the error magnitude, respectively. For example, assume that gate U1 is ‘1’ the τ=90% of the time and that ϕ=3. Replacing U1 by ‘1’ will result in an error rate of 10% and the maximum error will be less than 2^4. Netlist pruning is mainly implemented using heuristics [11] and thus optimality cannot be guaranteed. In our work, leveraging that i) bespoke architectures feature significantly fewer area/gates than conventional architectures and ii) that ML models for printed electronics are rather limited in size, we use τ and ϕ to constraint the pruning design space and we implement an exhaustive search to obtain Pareto-optimal solutions. Aiming for high area-efficiency, we prune all gates that feature τ and ϕ less or equal to given constraints τ_c and ϕ_c. Since all the pruned gates feature $\phi \leq \phi_c$, the maximum output error is less than 2^{ϕ_c+1} irrespective of the number of pruned gates. Overall, our coarse-grained approach ensures maximum area reduction while satisfying a maximum error threshold and enables fast design space exploration since only a gate's τ and ϕ need to be calculated.

Leveraging ϕ we filter all the gates that feature high τ and prune those that satisfy a given worst-case error. In the case of regressors (MLP-R and SVM-R) this works well and many gates are pruned for low ϕ_c values. However, classifiers require special consideration. MLP-C and SVM-C use an argmax function at the end to translate the numerical predictions (e.g., values of output neurons in MLP-C) to a class. As a result, the paths passing from all the gates are eventually congested in a few output bits, limiting the pruning granularity (possible ϕ_c). Moreover, argmax breaks the correlation between the introduced numerical error in predictions and the final output. For example, argmax([0.9, 0.1])=argmax([0.4, 0.1])=0. For this reason, for the classifiers, we calculate ϕ for each gate with respect to the inputs of the argmax function. For example, assume an MLP-C with k output neurons O_1,..., O_k. We define the value ϕ of a gate as $\max_{\forall i}\phi(O_i)$, where $\phi(O_i)$ is the most significant output bit of the neuron O_i that the gate is connected to. If such a path doesn't exist, we set $\phi(O_i) = -1$.

Given a netlist (either exact or coefficient approximated), our netlist pruning operates as follows:

1) Run RTL simulation of the synthesized netlist using the training dataset and Questasim to obtain the switching activity interchange format (SAIF) file.
2) Parse SAIF to calculate τ and the respective constant value (‘0’ or ‘1’) for each gate. For example, if the output of a gate is the 85% of the time “1’ and the 15% it is ‘0’, then τ=85% and the constant value is ‘1’.
3) Extract all the gates with $\tau \leq \tau_c$ and calculate their ϕ. ϕ is easily calculated with the synthesis tool by reporting paths from a gate to the outputs.
4) Prune all gates with $\tau \leq \tau_c$ and $\phi \leq \phi_c$, i.e., replace their output with the constant value extracted in step 2.
5) Synthesize the pruned netlist and evaluate its area and power as well as its accuracy on the test dataset.

The pruned netlist is synthesized to exploit all optimizations of the synthesis tool, e.g., constant propagation. Steps 1 and 2 are executed only once. Step 3 is executed $\forall\tau_c \in [80\%, 99\%]$. Then, for each τ_c, steps 4-5 are executed $\forall\phi_c \in \Phi_\tau$. Φ_τ is equal to the set of the unique ϕ values obtained in step 3. Φ_τ enables us to explore only the relevant ϕ_c values, accelerating our full search exploration. For example, if all the gates that feature $\tau \geq 99\%$ affect only the zero and first output bits, then $\phi_c > 1$ is meaningless since it will return the same solution as $\phi_c = 1$. Unlike the pruning state of the art that examines only very simple circuits [11], [12], our implementation is evaluated on complex ML circuits. Moreover, as explained above for the classifiers, conventional pruning [11], [12] cannot be used.

IV. Results and Analysis

In this section we evaluate the efficiency of our framework in reducing the area complexity and we investigate the impact of approximate computing on printed ML circuits. Fig. 3 presents the Accuracy-Area Pareto space for all the printed ML circuits examined (see Table I). Area is reported as a normalized value w.r.t. the area of the respective baseline bespoke circuit. Identical results are obtained for the power consumption. Due to space limitations, we present the area reduction since it is our primary optimization goal in printed circuits. In Fig. 3, the black triangles are the baseline bespoke designs. The red star is the design that applies only our proposed coefficient approximation. The green dots are the designs that are generated by our framework (i.e., Coefficient Approximation & Pruning). In addition, Fig. 3 also presents the designs that apply only pruning (gray 'x'). To generate these designs, we apply our pruning method directly on the baseline circuit. In other words, the green dots are pruned derivatives of the red start while the gray 'x' are pruned derivatives of the black triangle. At this point note that our pruning method performs a full search exploration. However, the number of the explored designs is circuit dependent. As aforementioned, our pruning framework uses $80\% \leq \tau_c \leq 99\%$ for all circuits but the explored ϕ_c values are circuit and τ_c dependent. In total, we evaluated more than 4300 designs.

Overall, it is observed that approximate computing can effectively be employed to decrease the area complexity of the printed ML circuits since all the approximate designs feature lower area than the exact one (i.e., black triangle). For example, for most circuits, more than 57% area reduction can be achieved for less than 5% accuracy loss. As shown in Fig. 3, our coefficient approximation achieves 28% average area reduction for almost identical accuracy with the exact baseline and in most cases outperforms significantly the standalone gate-level pruning approximation. In addition, the designs generated by our cross-layer approximation framework (green dots) constitute mainly the optimal solutions since they mainly form the Pareto front in all the subfigures of Fig. 3.

In Table II, we consider a conservative accuracy loss threshold (i.e., only 1%) and we report the area and power values of the area-optimal circuits w.r.t. each approximation technique. As shown, compared to the baseline bespoke [1]

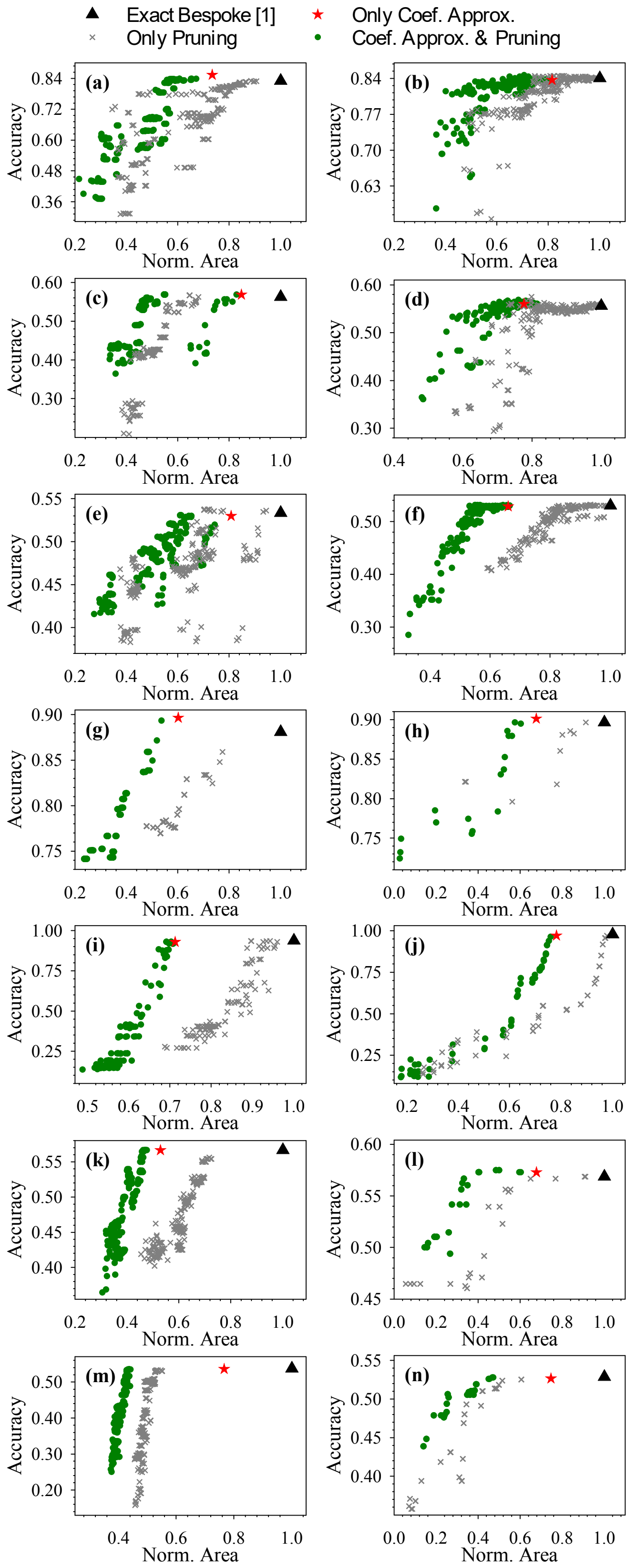

Fig. 3. Accuracy vs normalized area Pareto space. ML models examined: Cardio MLP-R (a), SVM-R(b), MLP-C (g), and SVM-C (h). Pendigits MLP-C (i) and SVM-C (j). RedWine MLP-R (c), SVM-R(d), MLP-C (k), and SVM-C (l). WhiteWine MLP-R (e), SVM-R(f), MLP-C (m), and SVM-C (n).

TABLE II
AREA AND POWER EVALUATION FOR LESS THAN 1% ACCURACY LOSS. HIGHLIGHTED DESIGNS CAN BE POWER BY A MOLEX 30MW BATTERY.

ML Circuit	Coeff. Approx. & Pruning				Only Coeff. Approx.				Only Pruning			
	A[1]	P[2]	AG[3]	PG[3]	A[1]	P[2]	AG[3]	PG[3]	A[1]	P[2]	AG[3]	PG[3]
Card MLP-R	12	37	45	44	16	49	27	26	18	56	16	15
Card SVM-R	3.5	13	49	42	5.5	19	19	15	5.0	18	26	22
RW MLP-R	3.3	12	53	49	6.0	21	15	14	4.6	17	35	30
RW SVM-R	2.6	10	35	33	3.1	12	22	22	2.9	11	27	25
WW MLP-R	8.0	27	39	35	11	34	20	17	9.2	29	30	28
WW SVM-R	2.2	8.5	47	45	2.8	11	34	32	3.4	13	19	19
Card MLP-C	17	54	48	44	20	62	40	36	33	97	0	0
Card SVM-C	8.7	29	43	38	10	33	33	29	14	43	8.7	8.3
Pend MLP-C	46	153	31	28	48	143	29	33	60	194	10	9.0
Pend SVM-C	97	287	22	21	97	287	22	21	121	357	2.2	1.8
RW MLP-C	8.0	27	55	50	9.3	30	47	43	18	53	0	0
RW SVM-C	7.6	26	68	65	16	50	32	31	15	49	35	33
WW MLP-C	13	42	57	57	24	73	23	26	16	52	47	48
WW SVM-C	11	36	61	59	21	65	26	25	15	46	49	47

[1] Area (cm^2). [2] Power (mW). [3] Area and Power Gain compared to the bespoke baseline [1] (in %).

implementations, our cross-layer approximation delivers on average 47% and 44% area and power reduction, respectively. The corresponding values of our coefficient approximation are 28% and 26%. Standalone pruning achieves only 22% and 20% average area and power reduction, respectively. Finally, to put these gains into perspective, we highlight in Table II the circuits that could be power by one printed Molex 30mW battery. Overall, in addition to the circuits that even their exact bespoke baseline [1] can be power by a printed battery, our cross-layer approximation enables, for the first time, also printed MLP-C with 2 hidden nodes and 34 coefficients in total, printed MLP-R with 4 hidden nodes and 48 coefficients, and SVM-C with 15 classifiers and 66 coefficients. On the other hand, compared with [1], single layer approximation (only pruning or coefficient approximation) can not enable any additional circuits to be powered by a printed battery.

Finally, in Table III we report the overall execution time of our framework. As aforementioned due to the on-demand and point-of-use fabrication process, it is critical that the design time overhead due to approximation is kept to minimum. As shown in Table III, the average execution time of our framework is only 12min while its median value is 8min. At the worst case, our framework required only 48min for the Pendigits MLP-C, that is however very large to be considered for a printed circuit. Note that the times in Table III refer to the full design exploration (i.e., all the green dots in each subfigure of Fig. 3). As a result, the "true" Pareto front is obtained and the user may select a Pareto-optimal point based on the requirements (e.g., area-accuracy) of the printed application.

V. CONCLUSION

TABLE III
EXECUTION TIME OF OUR FRAMEWORK IN MINUTES

	MLP-C	MLP-R	SVM-C	SVM-R
Cardio	26	7	1	9
Pendigits	48	-	14	-
RedWine	7	6	2	7
WhiteWine	23	8	2	8

Printed electronics prevail as a promising solution for a large number of application domains that require ultra low-cost, conformality, low time-to-market, nontoxicity, etc. Nevertheless, large feature sizes in printed electronics prohibit the implementation of complex circuits. In this work, we propose, for the first time, a novel cross-layer approximation framework that exploits peculiar printed electronics features and generates optimal approximate printed circuits for ML applications. Our work demonstrates that approximate computing can enable the realization of several complex printed ML circuits.

REFERENCES

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

[2] 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.

[3] H. Cherupalli, H. Duwe, W. Ye, R. Kumar, and J. Sartori, "Bespoke processors for applications with ultra-low area and power constraints," in *Annu. Int. Symp. Computer Architecture (ISCA)*, 2017, pp. 41–54.

[4] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, and J. Henkel, "Invited: Cross-layer approximate computing: From logic to architectures," in *Design Automation Conference (DAC)*, 2016, pp. 1–6.

[5] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[6] H. Waris, C. Wang, and W. Liu, "Hybrid low radix encoding-based approximate booth multipliers," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 67, no. 12, pp. 3367–3371, 2020.

[7] I. Scarabottolo, G. Ansaloni, G. A. Constantinides, L. Pozzi, and S. Reda, "Approximate logic synthesis: A survey," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2195–2213, 2020.

[8] S. Barone, M. Traiola, M. Barbareschi, and A. Bosio, "Multi-objective application-driven approximate design method," *IEEE Access*, vol. 9, pp. 86 975–86 993, 2021.

[9] V. Mrazek, M. A. Hanif, Z. Vasicek, L. Sekanina, and M. Shafique, "Autoax: An automatic design space exploration and circuit building methodology utilizing libraries of approximate components," in *Design Automation Conference (DAC)*, 2019.

[10] G. Zervakis, S. Xydis, D. Soudris, and K. Pekmestzi, "Multi-level approximate accelerator synthesis under voltage island constraints," *IEEE Trans. Circuits Syst., II, Exp. Briefs*, vol. 66, no. 4, pp. 607–611, 2019.

[11] J. Schlachter, V. Camus, K. V. Palem, and C. Enz, "Design and applications of approximate circuits by gate-level pruning," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 5, pp. 1694–1702, 2017.

[12] I. Scarabottolo, G. Ansaloni, G. A. Constantinides, and L. Pozzi, "Partition and propagate: an error derivation algorithm for the design of approximate circuits," in *Design Automation Conference (DAC)*, 2019, pp. 1–6.

[13] V. Subramanian *et al.*, "Printed electronics for low-cost electronic systems: Technology status and application development," in *European Solid-State Circuits Conference (ESSCIRC)*, 2008, pp. 17–24.

[14] P.-Y. Chen *et al.*, "30.1: Invited paper: 65-inch inkjet printed organic light-emitting display panel with high degree of pixel uniformity," in *SID Symposium Digest of Technical Papers*, vol. 45, no. 1, 2014, pp. 396–398.

[15] M. G. M. et al., "All-printed flexible and stretchable electronics," *Advanced Materials*, vol. 29, no. 19, p. 1604965, 2017.

[16] A. de la Fuente Vornbrock, D. Sung, H. Kang, R. Kitsomboonloha, and V. Subramanian, "Fully gravure and ink-jet printed high speed pbttt organic thin film transistors," *Organic Electronics*, vol. 11, no. 12, pp. 2037–2044, 2010.

[17] J. Chang, X. Zhang, T. Ge, and J. Zhou, "Fully printed electronics on flexible substrates: High gain amplifiers and dac," *Organic Electronics*, vol. 15, no. 3, pp. 701–710, 2014.

[18] 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.

[19] 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.

[20] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml