

Co-Design of Approximate Multilayer Perceptron for Ultra-Resource Constrained Printed Circuits

Giorgos Armeniakos, Georgios Zervakis,
Dimitrios Soudris, *Member, IEEE*,
Mehdi B. Tahoori, *Fellow, IEEE*, and
Jörg Henkel, *Fellow, IEEE*

Abstract—Printed Electronics (PE) exhibits on-demand, extremely low-cost hardware due to its additive manufacturing process, enabling machine learning (ML) applications for domains that feature ultra-low cost, conformity, and non-toxicity requirements that silicon-based systems cannot deliver. Nevertheless, large feature sizes in PE prohibit the realization of complex printed ML circuits. In this work, we present, for the first time, an automated printed-aware software/hardware co-design framework that exploits approximate computing principles to enable ultra-resource constrained printed multilayer perceptrons (MLPs). Our evaluation demonstrates that, compared to the state-of-the-art baseline, our circuits feature on average 6x (5.7x) lower area (power) and less than 1% accuracy loss.

Index Terms—Approximate Computing, Co-design, Multilayer Perceptron, Printed Electronics

1 INTRODUCTION

A trillion-dollar market of fast moving consumer goods (FMCG), disposables such as beverages, packaged foods, and toiletries, low-end healthcare products (e.g., smart bandages), etc., have seen limited impact from embedded computing [1]. Although for several of these domains, a typically required computational task is classification [2], such applications pose requirements for sub-cent costs, non-toxicity, porosity, stretchability, and conformity that silicon-based computing systems cannot satisfy. Printed Electronics establish as a key solution to enable such applications.

Printed electronics indicates a set of printing techniques that can realize ultra low-cost [3], large scale [4] and flexible hardware [5]. Printed electronics cannot compete with silicon-based electronics in integration density, area, or speed, mainly due to large feature sizes arising from low-cost and low-resolution printing. The typical operating frequency of printed circuits is from a few Hz to a few

kHz [6]. Similarly, the feature size tends to be several microns [7]. Nevertheless, due to its form-factor, conformity, and most importantly ultra-low fabrication costs of the additive manufacturing processes, that enable hardware on-demand even at very low quantities, printed electronics can target application domains untouchable by silicon VLSI. However, their large feature sizes and high intrinsic transistor gate capacitances result in elevated power and area compared to nanometer technologies. Additionally, the integration density of printed circuits is orders of magnitude lower compared to silicon systems, prohibiting thus the realization of complex circuits. For example, a printed MAC (multiply-accumulate) unit—core of most machine learning (ML) circuits—is six orders of magnitude larger than the CMOS one, while its power consumption is 8x higher [2].

Since low-cost embedded ML systems are usually task-specific and in order to address the aforementioned limitations, there is the opportunity of generating model-specific ML circuits that are more efficient compared to general-purpose ones. Leveraging the potential for high customization, offered by the low-fabrication costs of printed circuits, [2] designed bespoke ML circuits. The term bespoke refers to fully-customized, even per ML model and dataset, circuit implementations. Still, [2] examined only simple ML models (e.g., Decision Tree) rather than complex Multilayer Perceptrons (MLPs) that they pose infeasible hardware overheads for the ultra-resource constrained printed circuits. Thereby, works on printed ML designs are limited.

Fortunately, approximate computing (AC) can be employed to reduce the associated hardware overheads at the cost of some accuracy loss. Exploiting the inherent error resilience of a large number of application domains, e.g., ML, AC relaxes the accuracy of some computations and achieves improvements in hardware metrics such as area and power. The authors in [8], designed for the first time approximate printed ML classifiers. Through post-training algorithmic weight-approximation and hardware-level gate-pruning, [8] generates printed ML circuits with high area and power gains. However, as in [2], authors in [8] deduced that the hardware overheads of MLPs are prohibitive for printed circuits and additional optimizations are still required.

In this work, we embrace the bespoke design paradigm, enhance it with AC, and propose an automated co-design framework for approximate printed MLP circuits. Leveraging the observation that the area of bespoke MLP circuits is intrinsically correlated to the MLP’s coefficients, we propose a printing-friendly MLP retraining that selects area-efficient coefficients and achieves 3.3x lower area, on average, over 10 MLPs. By printing-friendly retraining we refer to a bespoke hardware-aware retrain technique, suitable for ultra-resource constrained ML circuits such as our targeted printed ones. Our approach can be seamlessly extended to any printed ML circuit with trainable coefficients. Though, we focus on MLPs since they constitute the most complex ML classifier for printed applications [2]. Finally, in order to further optimize the resource efficiency of our circuits, we also approximate the summation of the neuron’s products by systematically keeping the most relevant bits and discarding the rest. Overall, compared to the exact bespoke baseline, we achieve, on average, 6x (5.7x) area (power) reduction with less than 1% accuracy loss.

- G. Armeniakos and D. Soudris are with the School of Electrical and Computer Engineering, National Technical University of Athens, Athens 15780, Greece. G. Armeniakos is also with the Chair for Embedded System (CES), Department of Computer Science at Karlsruhe Institute of Technology (KIT), Karlsruhe 76131, Germany.
- G. Zervakis is with the Computer Engineering & Informatics Dept., University of Patras, Patras 26504, Greece. This research was done when he was with the Karlsruhe Institute of Technology (KIT).
- M. B. Tahoori is with Chair of Dependable Nano Computing (CDNC), Department of Computer Science at KIT, Karlsruhe 76131, Germany.
- J. Henkel is with Chair for Embedded System (CES), Department of Computer Science at KIT, Karlsruhe 76131, Germany.

This work is partially supported by the German Research Foundation through the project “ACCROSS: Approximate Computing aCROSS the System Stack” HE 2343/16-1.

Corresponding Author: Giorgos Armeniakos (georgios.armeniakos@kit.edu)
Manuscript received Mars 24, 2022.

Our novel contributions within this work are as follows:

- 1) This is the first work that proposes, implements, and evaluates a software/hardware co-design for printed MLPs while approximating both multiplication and addition operations.
- 2) We propose a printing-friendly ML retraining technique.
- 3) Our evaluation shows that our framework paves the way, for the first time, towards highly accurate battery powered MLPs, suitable for ultra-resource constrained printed applications¹.

2 RELATED WORK

In the past years we have experienced a significant explosion in the printed electronics research at a variety of application domains. Targeting applications such as radio-frequency identification (RFID) tags, a pseudo-CMOS logic design for high performance thin-film circuits was presented in [9], while in [10] a 2-input neuron that can be used to support also a MAC operation was fabricated.

Recently, a flexible 32-bit microprocessor with over 18,000 gates was fabricated by ARM [11]. Exploring efficient architectures for printed microprocessors, [1] pruned the ISA (Instruction Set Architecture) and the respective microarchitecture accordingly and generated tiny printed microprocessors. However, research on printed ML applications is still at its infancy [2], since they are limited only to few neurons implementations rather than more complex ML circuits. Acknowledging the need for printed ML inference engines, the authors in [2] exploited the area-efficiency of bespoke architectures [12] and implemented fully customized ML circuits for printed technologies. Nevertheless, [2] considered only simple ML models such as Decision Trees and Support Vector Machine Regression due to the high hardware overheads of more complex models such as MLPs. Moreover, [13] described an automated methodology to generate also bespoke classifiers, but no quantification was performed, while further system integration with hardware machine learning processor for an odour recognition application was fabricated in [14]. Targeting printed MLPs, [15] employed Stochastic Computing (SC) to reduce their area and power at the cost, however, of a prohibitive accuracy loss in most cases. On the other hand, [8] designed approximate MLPs and achieved low accuracy loss but the power reduction was insufficient.

An active research field (also known as TinyML) investigates vigorously resource-efficient ML models which can run on constrained hardware such as IoT-sized devices. Recently, Google’s collaborated CFU Playground [16] proposed a full-stack framework in which bespoke and co-optimized architectures for embedded ML with TinyML focus can be explored. Although significant speedups can be attained by exploring different custom function units, optimization space for power efficiency has not been comprehensively studied yet. Furthermore, Kumar et. al. [17], targeting low-power devices, developed a tree-based classification algorithm trying to fit in the available memory of such tiny IoT devices. Similarly, aiming to minimize the required working memory, ProtoNN [18] pro-

¹.Our implementations are available at <https://github.com/garmeniakos/Ax-Printed-ML-Classifiers>

TABLE 1: Qualitative comparison of related works.

Domains	Reference	Bespoke ¹	AC ² SC	HW/SW ³ Co-Design	Battery ⁴ Operation
Printed Electronics	[1], [2], [14]	✓	✗	✓	
	[8], [15]	✓	✓	✓	
CMOS	[12]	✓	✗	✓	✗
FPGAs	[16]	✓	✗	✓	✗
TinyML (IoT)	[17], [18]	✗	✗	✓	✗
AC DNN	[20] – [24]	✗	✓	✓	✗
PE	Ours	✓	✓	✓	

¹Customized implementations. ²AC: Approximate Computing, SC: Stochastic Computing. ³HW/SW: Hardware/Software. ⁴Printed batteries \leq 30mW

posed a k-Nearest Neighbour based algorithm for resource-constrained devices. However, memory requirements and transfers of both [17] and [18] consume significant power that acts prohibitively for printed devices.

Designing approximate arithmetic units such as adders and multipliers has attracted significant research interest [19]. Though, they target non-bespoke arithmetic circuits, unsuitable for ultra resource constrained printed electronics [2]. Finally, vast research focuses on approximate neural network accelerators [20], [21], [22], [23], [24]. However, such deep networks are unrealistic to be considered for printed applications [9].

Table 1 summarizes the above discussion and provides a qualitative comparison of the related works in the field of severe resource-constrained ML inference. Although several co-design frameworks have been proposed (e.g., for IoT, FPGAs, TinyML) printed circuits have a much tighter resource constraint and the existing frameworks have not been evaluated on such extreme cases. Hence, it is highly unclear if existing frameworks will need extremely long time or even fail to find a valid solution. Finally, we propose a fine-grained design methodology that is bespoke-specific and could also be applied to other technologies that would support such high customizations. Our work distinguishes from most state-of-the-art works and can be only classified along with [8], [15] since they all: i) exploit the efficiency of bespoke implementations, ii) employ a systematic co-design methodology to boost the efficiency of the generated circuits, iii) apply non-conventional computing (Stochastic or Approximate Computing) to maximize the hardware savings, and iv) target the unmatched domain in ultra-limited resources of printed ML applications. In Section 4, we thoroughly evaluate our framework against [2], [8], [15]. In other words, to the best of our knowledge, we compare our work against the only available printed MLP works today.

3 APPROXIMATE PRINTED MLPs

In this section, we present our automated co-design framework for approximate printed MLP circuits. Briefly, our framework receives as input a trained model (e.g., dumped from scikit-learn) and performs a printing-friendly retraining in which the MLP coefficients are replaced with more area-efficient ones. Next, our framework generates the Verilog RTL description of the respective bespoke MLP circuit and approximates the summation operation of each neuron by systematically dropping the less significant bits of the

TABLE 2: Evaluation of Bespoke Printed MLPs. Most MLPs feature unsustainable hardware overheads.

Dataset	Topology	#MACs	Cpd [ms]	Acc	Area [cm ²]	Power [mW]
WhiteWine (WW)	(11,4,7)	72	198	0.54	31	98
Cardio (CA)	(21,3,3)	72	199	0.88	33	97
RedWine (RW)	(11,2,6)	34	199	0.56	18	53
Pendigits (PD)	(16,5,10)	130	201	0.94	67	213
Vertebral Col. 3C (V3)	(6,3,3)	27	200	0.83	8.9	36
Balance Scale (BS)	(4,3,3)	21	199	0.91	9.3	36
Seeds (SE)	(7,3,3)	30	200	0.94	9.9	41
Breast Cancer (BC)	(9,3,2)	33	188	0.98	12	40
Vertebral Col. 2C (V2)	(6,3,2)	24	114	0.90	3.5	13
Mammographic (MA)	(5,3,2)	21	197	0.86	6.8	27

No Adequate Power Supply
Printed Battery Operation

summands (i.e., products inputs by coefficients). Finally, our framework, runs a full search design space exploration (DSE), in which all the approximate circuits are synthesized using EDA tools and the printed Process Design Kit (PDK), to obtain the Pareto-optimal approximate MLPs.

3.1 Bespoke MLPs

The ultra low-cost and on-demand in-situ fabrication in printed electronics enables bespoke implementations, tailored to specific dataset or usecase, that enable circuits with phenomenal area reduction compared to the respective conventional general purpose ones [2]. Leveraging the ultra low-cost and on-demand in-situ fabrication, bespoke implementations prevail as the most prominent solution to realize ultra-resource constrained printed circuits [1], [2]. Driven by this potential, we also embrace the bespoke design paradigm for printed MLPs. In such highly customized circuits, the coefficients of the ML model (MLPs in our case) are hardwired in the circuit implementation itself [2].

For our analysis, we consider 10 MLP classifiers (see Table 2) trained on varying datasets of the UCI ML repository [25]. These datasets are selected similarly to [2] and [15]. To train the MLPs, scikit-learn and the randomized parameter optimization with 5-fold cross validation are used. Inputs are normalized to $[0, 1]$ and we use a random 70%/30% train/test set split. The topology of the MLPs is $\#input \times L \times \#output$ with $L \leq 5$ and the ReLU is used for activation function. L is selected so that each MLP achieves close to maximum accuracy while L (i.e., the number of its hidden nodes) is minimized. At the final stage an argmax function is used to translate and map numerical predictions (i.e., values of output neurons) to a class, from which classification accuracy can be obtained.

Table 2 presents the evaluation of the examined MLP circuits. All circuits in Table 2 are implemented following the bespoke fully-parallel (i.e., 1 inference/cycle) state-of-the-art design methodology of [2]. In the remainder, these circuits will be referred to as baseline circuits. Fixed-point arithmetic is used with 4 bits for the inputs and 8 bits for coefficients, achieving close to floating-point accuracy. Nevertheless, since coefficients are hardwired in the circuit, we use the bare-minimum precision for each coefficient

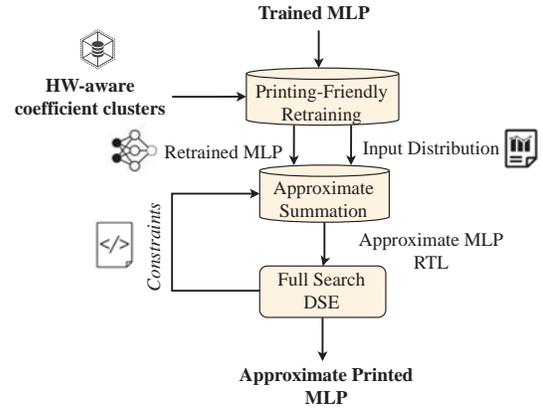


Fig. 1: Abstract overview of our framework.

(e.g., “3” uses only 2 bits). Circuit synthesis is performed with Synopsys Design Compiler and targeting the Electrolyte Gated Transistor (EGT) library [1], which is a low-voltage inkjet-printed technology that allows battery powered printed circuits. Since our objective in our optimization problem is to further improve the area efficiency rather than the performance, all MLPs are synthesized and simulated at relaxed timing constraints, i.e., 250ms per inference for Pendigits and 200ms for the rest ones. These performance values comply with typical operating frequencies in printed electronics [6], while delay gains due to our approximations are presented in Section 4.1. Power estimation is performed using Synopsys PrimeTime and switching activity obtained from circuit simulations with Questasim.

Due to the nature of printed applications, they all pose tight area requirements. As a rule of thumb and similar to [12], we consider the 10cm^2 and 30mW (i.e., maximum power of a single printed battery) as a hard constraint for most printed applications. As shown in Table 2 the average area of the MLP circuits is prohibitive for such applications [12]. Moreover, only the Vertebral Column 2C and the Mammographic MLPs can be powered by an existing printed battery (e.g., Molex 30mW battery) while for the rest MLPs it doesn’t exist any adequate power supply [2].

3.2 Printing-Friendly MLP Retraining

Our framework (Fig. 1) enhances bespoke circuits with approximate computing to further improve the hardware efficiency of the former. Although bespoke circuits form the most promising solution for printed ML applications, bespoke implementations open new rooms for optimizations in printed circuits that are barely explored up to now. For example, in Fig. 2a we perform a 1000-point Monte Carlo analysis of the area of bespoke neurons w.r.t. the values of the coefficients. Neurons are the building block of MLPs. As shown, irrespectively of the size of the neuron ($\#inputs$), the area of the neuron features very high variation. For example, in Fig. 2a the average standard deviation is 63mm^2 or else 175 gates. Therefore, there is a high potential, with a proper coefficient selection, to keep the bespoke neuron’s area minimal. We further investigate this and present in Fig. 2b the area of bespoke multipliers $a \cdot w$. Multipliers constitute the core components of a neuron. The value w is

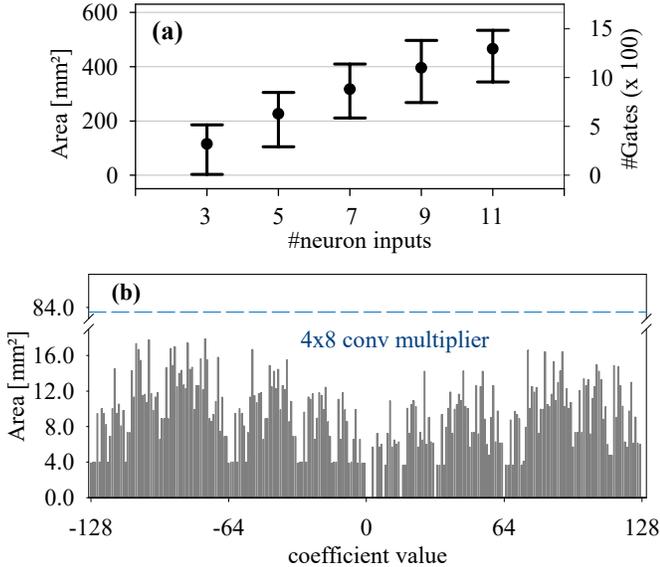


Fig. 2: a) 1000-point Monte Carlo analysis of the area of bespoke neurons w.r.t. the values of the coefficients. b) Area of bespoke printed multipliers with 4-bit inputs and coefficients in $[-128,127]$.

hardwired in each bespoke multiplier, the input a is 4 bits and the coefficient w is up to 8 bits, i.e., $w \in [-128, 127]$. It is noteworthy that the area of the bespoke multipliers is more than $5\times$ lower than the area of the respective 4×8 conventional multiplier. From Fig. 2b, it is evident that there is an intrinsic correlation between the value of the coefficient w and the area of the bespoke multiplier and consequently, the area of the neuron. Importantly, in the cases that the coefficient is a power of two, the multiplier’s area is nullified (i.e., multiplier is replaced by only simple wiring).

Motivated by the above observations we leverage our bespoke hardware analysis to implement a printing-friendly retraining. Our main goal is to replace the coefficients of the MLP with more area-efficient ones and as a result, minimize the area of the required multipliers while maintaining high accuracy in the meantime.

As a step towards enabling printing-friendly retraining, we need to distinguish the coefficients based on their area-efficiency. To achieve this, we use K-means and cluster the coefficients with respect to the area of the respective bespoke multiplier. Without loss of generality, we consider up to 8 bits for the coefficients and we cluster the coefficients into four groups C_0 - C_3 . Group C_0 comprises only powers of two, resulting thus to zero-area multipliers, while the multipliers generated by the coefficients of C_i feature larger area than the multipliers generated by the coefficients of C_j if $i > j$. For example, we clustered the coefficients w , $\forall w \in [0, 127]$, considering bespoke multipliers with 4 bits input. Fig. 3 presents the area of the bespoke multipliers of each cluster. As shown in Fig. 2b, the negative coefficients produce multipliers with larger area than the respective positive ones. Nevertheless, as we will explain later, during retraining we assume that the positive and negative coefficient multipliers feature the same area. For this reason, we perform the clustering only for the positive coefficients. Moreover, we clustered the coefficients using several input

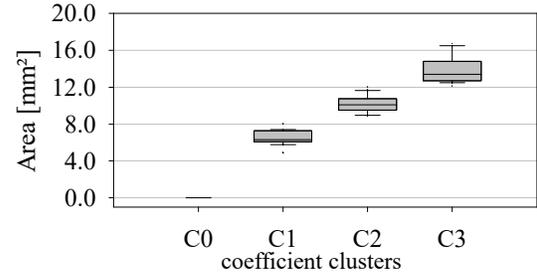


Fig. 3: Area analysis of the clustered coefficients for 4-bit inputs and coefficients in $[0,127]$.

sizes from 4 up to 16 bits and obtained identical results. The latter is explained by the fact that increasing the input size impacts all the bespoke multipliers similarly, irrespectively of the coefficient value. Therefore, although the neurons of the hidden and output layers might feature different input sizes, the same clustering can be used. Finally, to cluster the coefficients, we need to synthesize (once for all MLPs) all the positive bespoke multipliers. In our case, for 128 bespoke multipliers, it required less than a minute using 10 threads in a Xeon E5-2650 server with 64-GB RAM.

Algorithm 1 presents an abstract overview of our printing-friendly MLP retraining. Our framework receives as inputs a trained model, the training dataset, and a user-defined accuracy loss threshold that remains constant throughout the retraining process and could be relaxed for higher potential gains. Note that accuracy loss is defined as: $Accuracy_{exact} - Accuracy_{approx}$. Given an already trained MLP with fixed topology (namely MLP0), our algorithm initiates our MLP’ to MLP0 and retrains MLP’ trying to assign all its coefficients to C_0 ($VC \leftarrow \{w, -w : w \in C_0\}$). If after m epochs the accuracy is below a given threshold then i) we reset the training (MLP’ \leftarrow MLP0), ii) we increase the number of available values for the coefficients by gradually considering more clusters, and iii) we repeat the training for another m epochs each time. A solution always exists since at the worst case all the coefficient clusters will be consumed. During the m epochs retraining, if the accuracy is unacceptable and no coefficients are updated, we increase the learning rate to allow jumps [22]. The latter is crucial since if the distance of the available values in VC is large, then with a small learning rate the coefficients might always be mapped to the same value in VC . Finally, to enable area-awareness during training we add a small penalty at the obtained accuracy based on the area of the required bespoke multipliers. This is crucial, when using more than one clusters, in order to guide retraining towards selecting more coefficients from the lower (more area efficient) clusters. To achieve this, we calculate the following score function:

$$S = \alpha \cdot \frac{\text{accuracy}(\text{MLP}')}{\text{accuracy}(\text{MLP0})} + (1 - \alpha) \times \frac{\text{AR}(\text{MLP0}) - \text{AR}(\text{MLP}')}{\text{AR}(\text{MLP0})}, \quad (1)$$

where AR is the sum of the area of the bespoke multipliers instantiated by each MLP. These area values are calculated based on the input sizes of each neuron and are stored in a

Algorithm 1: Printing-Friendly Retraining Pseudocode

Input: 1) Trained Model: MLP0, 2) Accuracy Loss Threshold: T ,
 3) Train Dataset
Output: 1) Printing-Friendly Model: MLP'
 1: $C_i, \forall i \leftarrow$ Cluster Coefficients
 2: $VC \leftarrow \{\}$
 3: **for** $0 \leq i < \#$ Clusters
 4: $VC \leftarrow VC \cup \{w, -w : w \in C_i\}$
 5: MLP' \leftarrow MLP0
 6: retrain MLP' for m epochs

- feedforward: evaluate score function Eq. (1)
- coefficient update: convert coefficients to fixed point & map each coefficient to its closest value in VC
- adjust learning: if no coefficient updated \rightarrow increase learning rate

 7: **if** (accuracy(MLP') \geq accuracy(MLP0) $- T$) **break**
 8: **return** MLP'

look-up table to be used during retraining. Again the time required is negligible (5min at the worst case examined) and for the negative coefficients we use the area of the respective positive ones. When MLP' and MLP0 feature the same accuracy and MLP' uses only C_0 (only power of two), $AR(MLP')$ becomes zero and (1) takes its maximum value, i.e., $S = 1$. When MLP' and MLP0 are the same (e.g., initial assignment), S in (1) equals α . In our work, targeting high accuracy printed MLPs, we set $\alpha = 0.8$. Nevertheless, the area-accuracy tradeoff w.r.t. α needs to be explored more comprehensively in the future. Finally, we set the retraining epochs $m = 10$ to constraint the execution time spend in retraining (i.e., 40 epochs at most). On average, 4min were required by our printing-friendly retraining.

Overall, our coefficient cluster-based retraining approach constrains the search space of printing-friendly coefficients and helps exploring early and with high confidence area-efficient solutions. As more clusters are gradually used in retraining, the penalty imposed by our score function guides the training algorithm to limit the number of coefficients selected from the higher clusters and instead select more area-efficient coefficients from the lower clusters.

3.3 Approximate Bespoke Neuron

Each neuron calculates a weighted sum:

$$S = \sum_{\forall i} a_i \cdot w_i, \quad (2)$$

where w_i are the neuron's coefficients (or weights) and a_i are its inputs. After retraining, the coefficients of the MLP are replaced with more printing-friendly ones, leading to reduced hardware requirements w.r.t. the neuron's bespoke multipliers. Still, the summation of the products of the bespoke multipliers results in considerable hardware overhead. Fig. 4 depicts the implementation of our approximate neuron. Overall, we calculate the following:

$$S' = \underbrace{\sum_{\forall i: w_i \geq 0} a_i \cdot w_i}_{S_p} + \left(\sim \underbrace{\sum_{\forall i: w_i < 0} a_i \cdot |w_i|}_{S_n} \right), \quad (3)$$

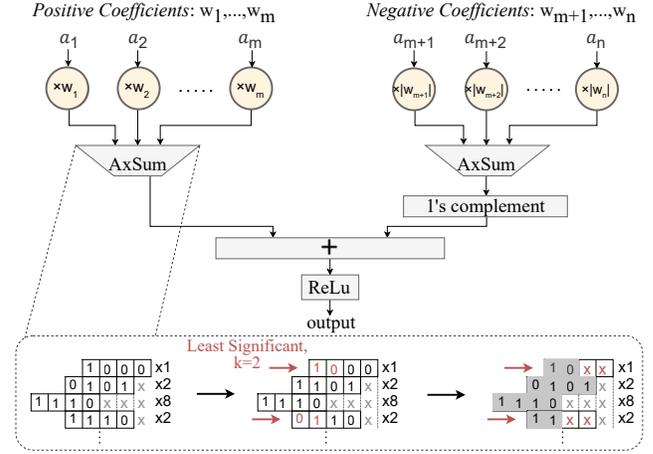


Fig. 4: Overview of our approximate bespoke neuron.

where \sim refers to logical NOT (i.e., 1's complement). Exploiting that the inputs of each neuron are positive (see Section 3.1), we know a priori the sign of each product (i.e., same with the sign of the corresponding coefficient). Hence, we split the coefficients into positives and negatives and for the negative ones we use their absolute value to generate their bespoke multipliers. A different adder tree is used to sum each set of products and 1's complement (instead of 2's complement) is used to negate the sum of the negative coefficients (S_n). Finally, S_p and $\sim S_n$ are added together. Calculating (3) requires only positive bespoke multipliers that feature significantly lower area than the negative ones. Moreover, with (3) we eliminate many full adders that would be required just for sign extension. If the neuron doesn't have any negative coefficients, the right parts of (3) and of Fig. 4 are omitted. Again, such high customization is feasible only in bespoke circuits.

To further reduce the area complexity, we approximate the adders that produce the sums S_p and S_n . To achieve this, we keep only some MSBs of the least significant summands (products $a_i \cdot w_i$) and we discard the rest. Hence, the more summands are approximated, the higher area reduction is achieved. An illustrative example is depicted in Fig. 4. Motivated by our cluster-based retraining, many coefficients are assigned to a power of two. Intuitively, inputs multiplied by high powers of two will generate considerably more significant products for the final result compared to inputs multiplied by small powers of two. However, despite this intuitive observation the significance of the product $a_i \cdot w_i$ depends also on of a_i . Thus, we define the significance of each product as follows:

$$G_i = \left| w_i \frac{E[a_i]}{\sum_{\forall i} (E[a_i] \cdot w_i)} \right|. \quad (4)$$

In other words, (4) calculates the ratio of the average value of each product $a_i \cdot w_i$ over the average sum of the neuron. For each neuron, $G_i, \forall i$, is easily calculated by just capturing the inputs distribution during training. Exploiting this high-level information, we approximate accordingly, at design time, the summation operations (S_p and S_n). For each product $a_i \cdot w_i$, if G_i is less or equal to a given threshold

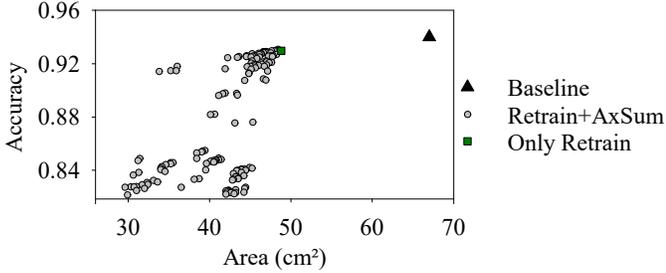


Fig. 5: Accuracy-Area Pareto space of the PD MLP.

G , we keep only the k MSBs with $k \in [1, 3]$. Hence, for each neuron, the approximate sum (AxSum) is given by:

$$S' = S_p + (\sim S_n)$$

$$S_p = \sum_{\substack{\forall i: w_i \geq 0, \\ G_i > G}} p_i + \sum_{\substack{\forall i: w_i \geq 0, \\ G_i \leq G}} p_i [n_i - 1 : n_i - k] 2^{n_i - k}, \quad (5)$$

$$S_n = \sum_{\substack{\forall i: w_i < 0, \\ G_i > G}} p_i + \sum_{\substack{\forall i: w_i < 0, \\ G_i \leq G}} p_i [n_i - 1 : n_i - k] 2^{n_i - k},$$

$$p_i = a_i \cdot |w_i|, \quad n_i = \text{size}(|w_i|) + \text{size}(a_i), \forall i$$

where $n_i, \forall i$, is the size of each product, e.g., for $w_i = \pm 7$ and 4-bit inputs, n_i is 7 bits. Note that, (5) refers to each neuron, i.e., different neurons might feature different k and G values. To reduce the size of the design space, we consider the same k value for all neurons and one G value per layer. The latter is based on the fact that different layers feature different sensitivity to approximation [24].

Finally, we perform an exhaustive DSE w.r.t. the value $k \in [1, 3]$ and all the possible values of G for each layer. Each design point is synthesized and simulated and a Pareto analysis is performed to obtain the Area-Accuracy Pareto Front. Unlike conventional silicon VLSI, in printed electronics the examined ML models are rather small in size. Hence, synthesis and simulation of each design requires a couple of minutes at most. Thus, we can obtain fast enough the Pareto optimal designs through an exhaustive DSE. On average, DSE required only 7 min using 10 threads (i.e., limit of our available EDA licenses). At the worst case, the DSE required 1h for the PD MLP, which is however very large to be considered for a printed application. For example, Fig. 5 presents the Accuracy-Area Pareto space for the PD MLP and up to 20% accuracy loss. The green square is the design that applies only our printing-friendly retraining, while gray circles are approximate derivatives of the green square and are derived from different approximate configurations of k and G . Overall, the designs generated using “Retrain+AxSum” constitute the Pareto front and achieve 2x area reduction for only 2% accuracy loss.

Note that in (2)-(5) we didn’t consider a neuron bias term to simplify the presented analysis. If a neuron has a bias, then similarly to the coefficients, the bias is hardwired in the circuit itself. Moreover, if the bias is positive, then it is added in S_p among with the positive products. Similarly, if it is negative its absolute value is added in S_n .

4 RESULTS

In this section, we investigate the effectiveness of our proposed co-design framework in enabling printed MLPs with minimal accuracy loss. We evaluate the area, power, and accuracy of our approximate MLPs against the state-of-the-art exact bespoke circuits [2] (see Table 2) and we also compare our framework against the stochastic MLPs [8], [15] that also trade accuracy for area and power gains. To the best of our knowledge, [2], [8], [15] are the only available works on printed ML circuits. Overall, in our AxSum DSEs, we evaluated more than 600 circuits in order to extract the optimal approximate printed MLPs. Note that the accuracy reported in this section is on the test dataset while all our optimizations are performed on the train dataset.

4.1 Comparison With Exact Baseline

Fig. 6 presents the area and power gains of our framework (“Retrain+AxSum”) w.r.t. the exact bespoke circuit [2]. Both our MLPs and [2] are synthesized with the same timing constraints. In Fig. 6, we consider three accuracy loss thresholds: 1%, 2% and 5%. For each threshold, we selected the Pareto-optimal MLP from our DSE that satisfies it. In addition, we also report the gains obtained when applying only our printing-friendly retraining (“Only Retrain”). The examined thresholds refer to the overall accuracy loss, i.e., due to printing-friendly retrain and AxSum. However, since multipliers consume the most area and power in MLPs [20], we assign all the available accuracy loss budget to our retraining algorithm. Then, if there is still room for further approximation, we apply our AxSum. Though, as Fig. 6 shows, AxSum is always used. Compared to [2], our framework delivers very significant area and power gains. Specifically, we achieve 6.0x (5.7x), 9.3x (8.4x), and 19.2x (17.4x) lower area (power), for up to 1%, 2% and 5% accuracy loss, respectively. The corresponding values when using only our printing friendly retraining are 3.30x (2.72x), 3.78x (3.03x), and 3.80x (3.04x). Therefore, both our retrain as well as our AxSum methods contribute significantly towards the final area and power gains. Nevertheless, the area and power savings of retraining saturate after 2% accuracy loss. Hence, above 2%, further gains are subject only to our AxSum. For 1% accuracy loss, after retraining, WhiteWine and BreastCancer MLPs used the first two coefficient clusters, Pendigits used all the clusters, while the rest MLPs used only C_0 (i.e., maximum area reduction w.r.t. multipliers). Apart from PD, all MLPs used only C_0 for 2% and 5% accuracy loss. For 2%, PD used the first three clusters, while for 5% PD used the first two. Since for 2% and 5% almost all MLPs were retrained only with coefficients from C_0 , our printing-friendly retraining achieved the maximum possible savings and thus, this explains why its average area/power gains saturate after 2%. Overall, the PD MLP elucidates the different aspects of our retraining methodology. For tight accuracy constraints, all the coefficient clusters are used and our score function still ensures hardware gains, while for relaxed constraints less clusters are needed. It is noteworthy, that the baseline PD MLP uses only 21 coefficients from C_0 , while for 1% accuracy loss, the score function (1) forced our retraining to select 64 coefficients from C_0 . This explains why our “Only Retrain” PD MLP achieves 1.75x lower area

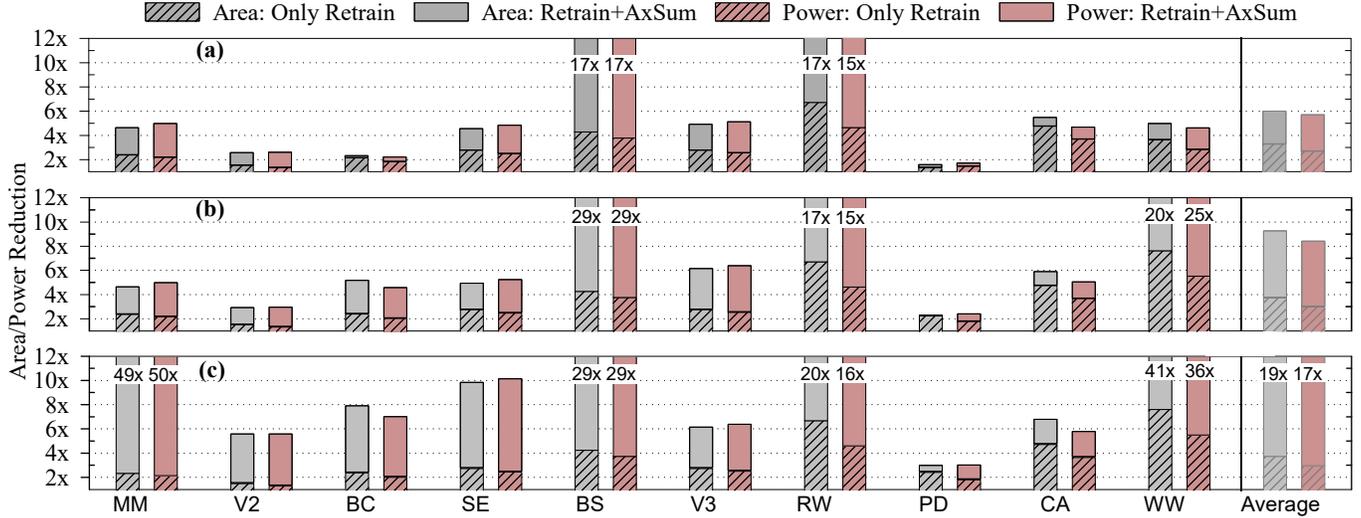


Fig. 6: Area and power reduction our proposed approximate MLPs compared to the exact state-of-the-art bespoke baseline [2]. Three accuracy loss thresholds T are examined for our approach: a) 1%, b) 2%, and c) 5%.

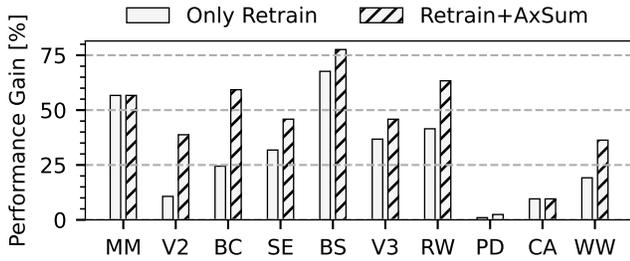


Fig. 7: Critical path delay gains for our approximate MLPs compared to the exact bespoke baseline [2] and for 1% accuracy loss threshold.

than [2], although both of them use coefficients from all the clusters (i.e., impact of our score function). Finally, applying both our retraining and AxSum approximation opens the road also for higher performance. Fig. 7 illustrates the delay reduction for each model, when applying our proposed approximations, compared to their exact (baseline) designs. On average, 44% CPD reduction is achieved for less than 1% accuracy loss.

Fig. 8 quantifies in an illustrative manner the tangible impact of our work. In Fig. 8, up to 5% accuracy loss is considered. As shown, the current state-of-the-art cannot support printed MLPs. For example, for 8/10 MLPs there is no existing adequate power supply. On the other hand, our framework enables 9/10 battery powered printed MLPs. Most of the MLPs can now be powered by only a Zinergy 15mW battery, while 3/10 can use a Blue Spark 3mW battery. Similarly, significant gains in area can now enable the practical and realistic printed applications (see unattainable area of the exact designs in Table 2). *This huge shift on supported printed MLPs may open new horizons for the realization of smart printed applications.*

4.2 Comparison With AC- and SC- based Printed MLPs

In Fig. 9 we compare our framework against the state-of-the-art stochastic printed MLPs [15] and approximate MLPs

of [8]. In Fig. 9, we included only the common MLPs examined in our work and [15]. Since in [8] only a few datasets are considered, we followed the respective methodology to generate the approximate MLPs of the additional (i.e., not included in [8]) datasets. Since the stochastic MLPs mainly feature high accuracy degradation, for our MLPs we used the 5% accuracy loss threshold (see Fig. 6c). Similarly, for [8], we selected the approximate designs that feature the lowest area (and power) and up to 5% accuracy loss. As shown, our approximate bespoke MLPs outperform [15] and [8] at all the metrics examined (i.e., power, area, accuracy). Specifically, we achieve 3.4x lower area, 3.7x lower power, and 7.7x lower accuracy loss than stochastic circuits [15], on average. Similarly, compared to [8], our gains increase to 8.8x lower area, 7.8x lower power, and 1.2x lower accuracy loss. All the aforementioned gains refer to similar performance since [15] required a stochastic bitstream of length 1024. In Fig. 9, our MLPs require 200ms per inference, while [15] requires 220ms to 230ms. As in our case, the approximate designs of [8] operate at 200ms per inference. Finally, we should mention that our co-design framework and [15] require some extra training epochs while [8] proposed a post-training approximation procedure. Though, since printed electronics allow only relatively small MLPs with a limited number of parameters, the time overhead required for re-training is still negligible.

5 CONCLUSION

Printed electronics offers a promising solutions to address limitations of silicon-based systems w.r.t. applications that require low cost, conformity, nontoxicity, etc. However, the ultra-resource constraint nature of printed circuits prohibits the realization of complex circuits, such as machine learning classifiers. In this work, we propose, for the first time, a software-hardware co-design framework for approximate printed MLPs. Through our printed-friendly MLP retraining and approximate summation, we design for the first time high accuracy battery powered printed MLPs, paving the way towards smart complex printed applications.

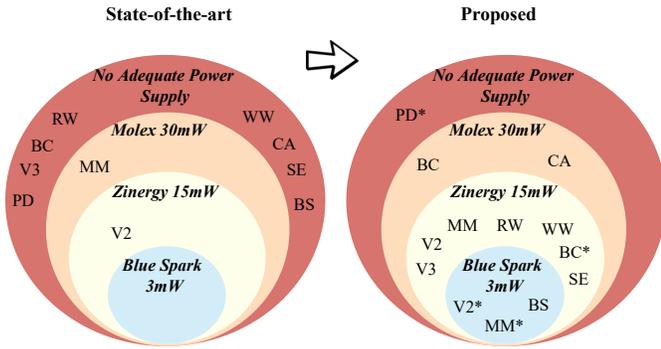


Fig. 8: Power supply classification of printed MLPs w.r.t. existing printed batteries. a) State-of-the-art bespoke MLPs [2], b) ours. All our MLPs feature up to 1% accuracy loss, except the * denoted ones that feature 5%.

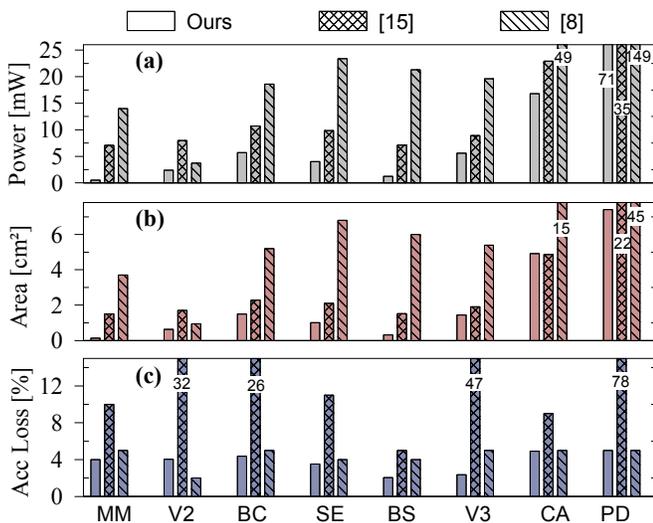


Fig. 9: a) Area, b) Power, and c) Accuracy comparison of our approximate printed MLPs vs the stochastic [15] and approximate [8] ones.

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, D. D. Weller, N. Bleier, M. Tomei, J. Aghassi-Hagmann, M. B. Tahoori, and R. Kumar, "Printed machine learning classifiers," in *Annu. Int. Symp. Microarchitecture (MICRO)*, 2020, pp. 73–87.
- [3] V. Subramanian, J. B. Chang, A. de la Fuente Vornbrock, D. C. Huang, L. Jagannathan, F. Liao, B. Mattis, S. Molesa, D. R. Redinger, D. Soltman, S. K. Volkman, and Q. Zhang, "Printed electronics for low-cost electronic systems: Technology status and application development," in *European Solid-State Circuits Conference (ESSCIRC)*, 2008, pp. 17–24.
- [4] P.-Y. Chen, C.-L. Chen, C.-C. Chen, L. Tsai, H.-C. Ting, L.-F. Lin, C.-C. Chen, C.-Y. Chen, L.-H. Chang, T.-H. Shih *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.
- [5] M. G. Mohammed *et al.*, "All-printed flexible and stretchable electronics," *Advanced Materials*, vol. 29, no. 19, p. 1604965, 2017.
- [6] G. Cadilha Marques, S. K. Garlapati, S. Dehm, S. Dasgupta, H. Hahn, M. Tahoori, and J. Aghassi-Hagmann, "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.
- [7] T. Lei, L.-L. Shao, Y.-Q. Zheng, G. Pitner, G. Fang, C. Zhu, S. Li, R. Beausoleil, H.-S. P. Wong, T.-C. Huang *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.
- [8] G. Armeniakos, G. Zervakis, D. Soudris, M. B. Tahoori, and J. Henkel, "Cross-Layer Approximation For Printed Machine Learning Circuits," in *Design, Automation Test in Europe Conference & Exhibition (DATE)*, 2022, [Online]. Available: <https://arxiv.org/abs/2203.05915>.
- [9] H. Çeliker, W. Dehaene, and K. Myny, "Dual-input pseudo-cmos logic for digital applications on flexible substrates," in *European Solid State Circuits Conference (ESSCIRC)*, 2021, pp. 255–258.
- [10] D. D. Weller, M. Hefenbrock, M. B. Tahoori, J. Aghassi-Hagmann, and M. Beigl, "Programmable neuromorphic circuit based on printed electrolyte-gated transistors," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 446–451.
- [11] J. Biggs, J. Myers, J. Kufel, E. Özer, S. Craske, A. Sou, C. Ramsdale, K. Williamson, R. Price, and S. White, "A natively flexible 32-bit arm microprocessor," *Nature*, vol. 595, pp. 532–536, 07 2021.
- [12] 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.
- [13] E. Ozer, J. Kufel, J. Biggs, G. Brown, J. Myers, A. Rana, A. Sou, and C. Ramsdale, "Bespoke machine learning processor development framework on flexible substrates," in *2019 IEEE International Conference on Flexible and Printable Sensors and Systems (FLEPS)*, 2019, pp. 1–3.
- [14] E. Özer, J. Kufel, J. Myers, J. Biggs, G. Brown, A. Rana, A. Sou, C. Ramsdale, and S. White, "A hardwired machine learning processing engine fabricated with micron metal-oxide thin-film transistors on a flexible substrate," *Nature Electronics*, vol. 3, pp. 1–7, 07 2020.
- [15] D. D. Weller, N. Bleier, M. Hefenbrock, J. Aghassi-Hagmann, M. Beigl, R. Kumar, and M. B. Tahoori, "Printed stochastic computing neural networks," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, 2021, pp. 914–919.
- [16] S. Prakash, T. Callahan, J. Bushagour, C. Banbury, A. V. Green, P. Warden, T. Ansell, and V. J. Reddi, "Cfu playground: Full-stack open-source framework for tiny machine learning (tinyml) acceleration on fpgas," 2022.
- [17] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 kb ram for the internet of things," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017, p. 1935–1944.
- [18] C. Gupta, A. S. Suggala, A. Goyal, H. V. Simhadri, B. Paranjape, A. Kumar, S. Goyal, R. Udupa, M. Varma, and P. Jain, "Protonn: Compressed and accurate knn for resource-scarce devices," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, 2017, p. 1331–1340.
- [19] 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.
- [20] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," *Int. Conf. Computer-Aided Design*, 2016.
- [21] V. Mrazek, Z. Vasicek, L. Sekanina, M. A. Hanif, and M. Shafique, "Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining," *Int. Conference on Computer-Aided Design (ICCAD)*, Nov. 2019.
- [22] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-Efficient Neural Computing with Approximate Multipliers," *J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 2, Jul. 2018.
- [23] G. Zervakis, O. Spantidi, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Control variate approximation for dnn accelerators," in *Design Automation Conference (DAC)*, 2021, pp. 481–486.
- [24] Z. G. Tasoulas, G. Zervakis, I. Anagnostopoulos, H. Amrouch, and J. Henkel, "Weight-oriented approximation for energy-efficient neural network inference accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, pp. 4670–4683, 12 2020.
- [25] D. Dua and C. Graff, "UCI machine learning repository," 2017.