Energy-Efficient ReRAM-based ML Training via Mixed Pruning and Reconfigurable ADC

Chukwufumnanya Ogbogu[1], Mohapatra Soumen[1], Biresh Kumar Joardar[2], Janardhan Rao Doppa[1], Deuk Heo[1], Krishnendu Chakrabarty[3], Partha Pratim Pande[1]. [1]School of EECS Washington State University, Pullman WA, USA. [2]University of Houston, Houston TX, USA, [3]Arizona State University, Tempe AZ, USA.

2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED) | 979-8-3503-1175-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISLPED58423.2023.10244258

Abstract— **Machine learning (ML) models have gained prominence in solving real-world tasks. However, implementing ML models is both compute- and memory-intensive. Domain-specific architectures such as Resistive Random Access Memory (ReRAM)-based Processing-in-Memory (PIM) platforms have been proposed to efficiently accelerate ML training and inference. However, existing ML workloads require a high amount of area and power for training. A major contributor to the area and power overheads is the Analog-to-Digital Converter (ADC). In this work, we propose a mixed pruning technique along with a novel reconfigurable ADC design to improve the power consumption profile. Overall, the pruned model with the reconfigurable ADC achieves ~50% reduction in power for training compared to existing state-of-the-art ReRAM-based architectures.**

Keywords—ReRAM, CNN training, Pruning, ADC

I. Introduction

Training machine learning (ML) models at the edge (on-chip training or on end-user devices) can address many pressing challenges associated with data privacy/security, increase the accessibility of ML applications to different parts of the world by reducing the dependence on communication fabric and the cloud infrastructure, and meet the real-time requirements of AR/VR applications. However, existing edge platforms do not have sufficient computing capabilities to support complex ML tasks such as training deep Convolutional Neural Networks (CNNs). ReRAM-based processing-in-memory (PIM) offers high-performance yet energy-efficient computing platforms for on-chip CNN training. This is due to their inherent capability to perform energy-efficient and high-throughput matrix-vector multiplication (MVM). Despite these advantages, deep CNNs with multiple layers require high power for training.

Recent work has proposed quantization and pruning to reduce the storage and power overheads of implementing deep CNNs on ReRAM-based PIM platforms [1] [2]. However, they are mostly targeted toward CNN inferencing and are not suited for training at the edge. Unlike inferencing, we also need to store the intermediate activations (in addition to the weights) during training. Simply pruning/quantizing weights does not help to reduce the storage and power overhead for activations. Moreover, existing techniques do not adequately reduce the number of ReRAM peripherals (e.g., ADCs). The ADCs in particular contribute between 50-70% of the power in a ReRAM-based PIM accelerator [3] [4]. Since, the power cost of ADCs scales exponentially with their precision, reducing the precision of ADCs without compromising the accuracy is key to saving power.

In this work, we achieve this by proposing a mixed-weight pruning technique that maximally prunes weights. In addition to the mixed pruning technique, we leverage dropout to further reduce the number of activations that must be stored in memory. The mixed pruning method involves a coarse-grained structured weight pruning where the granularity varies from filters to channels to index of the CNN weights. This is followed by fine-grained unstructured pruning. Hence, we refer to this mixed pruning method as coarse-to-fine-grained pruning (**C2F**). As we show later, C2F also prunes some activations. Dropout is also used as a mechanism to sparsify activations further. The C2F pruning along with Dropout enables the reduction of both the number and precision of required ADCs.

This work was supported, in part by the US National Science Foundation (NSF) under grants CNS-1955353, and CNS-1955196.

As we show later, different CNN layers achieve different levels of sparsity using C2F pruning and Dropout; this necessitates variable ADC precisions, which is not possible with conventional ADC designs. Hence, to support the mixed weight pruning (structured + unstructured) and Dropout, we propose to use a reconfigurable ADC. In this work, we propose a reconfigurable time-based ADC design [5]. The reconfigurability of the ADC stems from its ability to reconfigure the number of bits (precision) by disabling the lower LSB bits of the ADC. Hence, this approach enables a flexible and power-efficient design for ReRAM-based architectures with varying ADC requirements. The main contributions of this paper are:

- We propose a weight pruning technique called C2F, which along with dropout, sparsifies both weights and activations thereby reducing the power required for CNN training on ReRAM-based PIM architectures.
- We propose a reconfigurable ADC to efficiently support the C2F pruned CNN model during training to achieve greater power savings
- Experimental results indicate that the proposed solution achieves significant power reduction compared to existing architectures for CNN training.

II. Related Prior Work

ReRAM crossbars are well-suited for performing efficient MVM operations, which are predominant in CNN workloads [6]. Recent work has proposed ReRAM-based architectures for CNN inference. However, these architectures cannot efficiently support CNN training especially as the CNNs grow larger. Some ReRAM-based PIM architectures for CNN training have been proposed [7] [8]. However, they require a high bandwidth memory hierarchy for off-chip activation storage, which results in high latency and power costs [3].

Pruning is a popular technique for removing redundant weights and effectively reducing the computation and memory costs of CNN training and inferencing. Unstructured pruning does not result in a commensurate area or power savings compared to the amount of sparsity. Recent work has proposed

Authorized licensed use limited to: Washington State University. Downloaded on September 29,2024 at 15:13:56 UTC from IEEE Xplore. Restrictions apply.

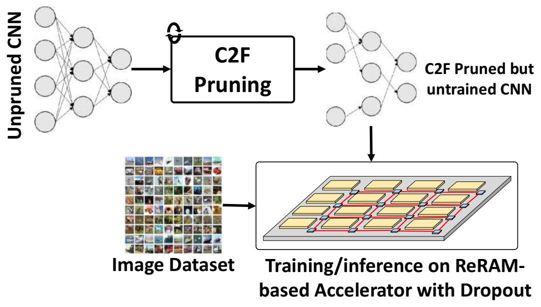

Unpruned CNN
C2F Pruning
C2F Pruned but untrained CNN
Image Dataset
Training/inference on ReRAM-based Accelerator with Dropout

Fig. 1: Overview of the C2F pruning enabled energy-efficient deep CNN training/inference on ReRAM-based PIM platforms.

crossbar-aware pruning methodologies that remove weights in regular (or structured) shapes to enable more hardware savings [4] [9] [10]. These existing pruning techniques when applied naively to ReRAM-based architectures, prune out only the CNN model weights and do not sparsify activations during training. Existing pruning strategies also do not reduce the hardware overhead due to peripheral circuit components such as ADCs [4] [1]. As a result, ADCs with high power and area overheads are still required for ReRAM crossbars even after pruning.

Recent work has proposed ADC-aware optimization techniques for ReRAM crossbars [4] [11] [2]. However, they assume lower but uniform ADC precision requirements across CNN layers. This can lead to accuracy loss when low-bit ADCs are used for precision-critical CNN layers as we show later. Recently, a load balancing optimization technique for CNN inferencing was proposed, which leverages a reconfigurable ADC design for ReRAM-based accelerators [12]. However, the load balancing-based method is also focused on inferencing and is not suitable for the input-dependent nature of activations during CNN training. Therefore, it is important to explore peripheral-circuit-aware techniques that efficiently accelerate CNN training on ReRAM-based architectures without any accuracy loss. In this work, we address these limitations by proposing the C2F pruning technique with Dropout, which along with the reconfigurable ADCs reduces power.

III. Joint Weights and Activations Pruning

In this section, we discuss the important features of the coarse- to fine-grained pruning technique (C2F). Also, we complement C2F with dropout specifically for activation pruning. Overall, we aim to enable training CNN models at the edge. CNN training requires storing both weights and activations. The C2F pruning requires several training iterations to generate the pruned model. Hence, we implement the C2F pruning offline (i.e., on GPUs) to first obtain the pruned but untrained CNN model as shown in Fig. 1. However, activations are input-dependent and cannot be implemented offline. Hence, we implement the dropout-enabled activation pruning online during the training with the pruned CNN model on the ReRAM-based architecture. We implement Dropout using Linear Feedback Shift Registers (LFSRs), which require little area and power. Next, we present more details about sparsifying both weights and activations.

A. Coarse- to Fine-grained (C2F) Pruning Technique

As mentioned earlier, existing pruning methods can be categorized as either structured or unstructured. Unstructured pruning can remove more weight than the structured counterpart, but it is oblivious to the underlying crossbar structure. Hence, it often does not translate to a significant area or power savings [4]. On the other hand, structured pruning does not achieve the same level of sparsity as the unstructured counterpart, but it saves more area and power than unstructured pruning. C2F synergistically combines both structured and unstructured pruning and hence is more effective.

The C2F technique achieves this by adopting an iterative magnitude pruning approach to find a highly sparse and trainable model following the Lottery Ticket Hypothesis (LTP) [13]. Figs. 2 (a)-(d) show the overall C2F pruning process. C2F prunes (in order) (a) filter-wise, to reduce the number of columns across multiple ReRAM crossbars required for storing weights, (b) channel-wise to ensure that columns within a ReRAM crossbar are reduced, (c) index-wise, which prunes entries along rows of a crossbar, and (d) element-wise (finest pruning) which prunes individual weights in an unstructured fashion to maximize sparsity. The filter-, channel-, and index-wise pruning constitute the structured part of the C2F pruning. C2F prioritizes filter-wise pruning as it reduces both the weights and activations. As shown in Fig 2(b), pruning out an entire filter from a CNN layer ensures that an entire column is pruned in all four crossbars. This produces sparse output (activations) from the crossbars. However, the relatively coarse granularity of pruning does not lead to significant sparsity without accuracy loss. Once C2F fails to prune filter-wise without significant accuracy drop, it switches to channel-wise, followed by index-wise, and then finally unstructured pruning (Fig. 2(c)-(d)) to maximize sparsity with less than 1% accuracy drop. As mentioned earlier, the iterative C2F pruning process is implemented on conventional computing platforms such as GPU/CPU offline. This pruning phase is a one-time process.

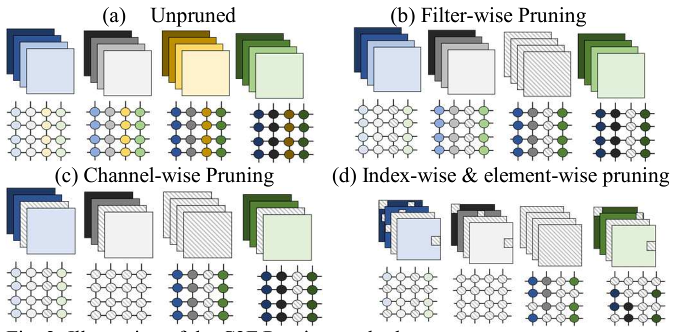


Fig. 2: Illustration of the C2F Pruning method.

Algorithm 1. C2F Pruning

Input: Unpruned CNN model, pruning percentage p

Output: C2F Pruned CNN model

Algorithm:

```
1:  Initialize: $\theta^l \leftarrow \theta_{initial}$;
2:  While $itr < N$ and no $accuracy\_drop$ do
3:      Train for $E$ epochs
4:      Prune $p\%$ of filters based on magnitude ($|\theta^l|$)
5:      If $new\_accuracy < Baseline\_accuracy$ do
6:          Undo last pruning step
7           Switch to finer pruning strategy (Channel-wise
            -> Index-wise -> element-wise pruning)
8:      Reinitialize remaining weights with $\theta_{initial}$
9:  Return Hardware-friendly C2F Pruned Model
```

Authorized licensed use limited to: Washington State University. Downloaded on September 29,2024 at 15:13:56 UTC from IEEE Xplore. Restrictions apply.

Once the pruned model is obtained, it can be reused since LTP-enabled pruned models are largely dataset-agnostic [14]. Here, we deploy the previously pruned model to the ReRAM accelerator as shown in Fig. 1 for future training tasks with any dataset [14], thereby amortizing the cost (time/energy) for the offline C2F pruning itself.

Algorithm 1 presents the high-level details of the C2F pruning strategy. We start by initializing the CNN weights (θ) using Xavier or Kaiming initialization (line 1) [7], then we train the CNN for $\boldsymbol{E}$ epochs. Next, we first prune the p% of the low magnitude CNN weights using coarse filter-wise pruning, and progressively switch to finer punning strategies (channel-, index-, and element-wise pruning), with <1% accuracy loss (lines 4-7). Finally, the C2F pruned (but untrained) model can then be used to train on resource-constrained edge devices with little area and power.

B. *Activation Pruning using Dropout*

Weight pruning does not automatically result in activation sparsity. Only the filter-wise pruning in C2F introduces some sparsity in the activations. To further sparsify the activations, we incorporate the dropout method on the activations. Dropout introduces unstructured sparsity without loss of accuracy. Dropout is a well-known regularization technique for CNN training which solves the overfitting problem and improves the model's generalizability to unseen data [15]. Dropout randomly prunes (temporarily) output activations of layers, which can be leveraged to save power as well. Traditionally, dropout is applied to the activations of the last few fully connected layers of CNNs only [15]. However, activation sizes typically decrease as the input passes through the various CNN layers. The last few CNN layers have a few activations. Hence, dropout only on these layers does not result in any hardware savings.

In reality, the initial layers of the CNN produce most of the activations stored on-chip during the pipelined training of CNNs on the ReRAM architectures [7]. For example, in VGG11 the output activations of the first two layers make up more than 70% of the total activations required for training as we show later. This happens because the initial layers process larger-sized activations. In addition, due to the pipelined execution of CNN training on ReRAM-based platforms, the memory requirement is especially high for the first few layers [6] [7]. The memory required for activation storage of a given CNN layer at depth (l) for pipelined training is multiplied by a factor of $(2(L-l)+1)$ (where L = number of CNN layers) [7]. As is clear from the equation, the initial layers need much more storage for activations compared to the later layers. Hence, in this work, we apply dropout on these first few layers to significantly sparsify activations. As typically done in practice, we also incorporate dropout on the last fully connected layers to prevent overfitting and improve prediction accuracy. However, adding dropout to the last few layers does not help with reducing activation storage significantly.

Other activation pruning methods such as magnitude-based activation pruning can also be used too. However, we show later that magnitude activation pruning leads to a significant accuracy drop during training. Similarly, applying dropout to every CNN layer also leads to significant accuracy degradation. Therefore, to effectively reduce the number of activations stored on the ReRAM crossbars during CNN training without accuracy loss, dropout should be applied to a few initial layers only. This reduces the ADC precision requirements for ReRAM crossbars that store the activations of the first few layers.

C. *ADC Design for C2F Pruned CNNs on ReRAM Platforms*

An ADC is necessary to interface between the ReRAM crossbar and the digital peripherals. The minimum ADC precision ($ADC_{precision}$) required for a ReRAM crossbar is determined by the number of rows activated (r), the number of input bits processed per cycle (v), and the number of weight or activation bits (w) stored per ReRAM cell. The required ADC precision (following [6]) is computed as follows [6]:

$$ADC_{precision} = v + w + \log_2(r) - 2 \qquad (1)$$

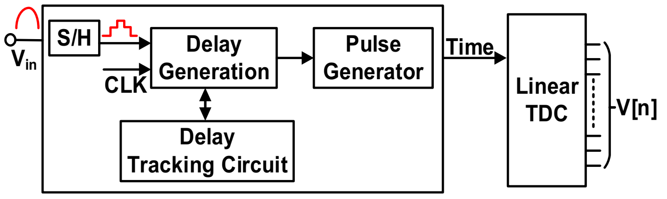
Following equation (1), a ReRAM crossbar of size 128×128 (r = 128), with 1-bit input per cycle (v = 1), and 2-bits stored per cell (w = 2), would require an 8-bit ADC in an unpruned scenario. It is well known that large ADCs (e.g., 8-bit) are power-hungry and this problem must be addressed. Besides the precision, having too many ADCs incurs significant power overhead. A single ADC is often shared by multiple ReRAM columns (one ADC is shared by 128 columns as in [6]). Therefore, it follows that the number of ADCs required for c ReRAM columns is given by: $\lceil 128/c \rceil$.

The structured pruning part of C2F removes (prunes) all the weights mapped to selected columns while leaving all the weights in other columns intact, i.e., structured pruning reduces the value of c only. This enables us to use fewer ADCs. However, it does not affect the precision of ADCs. Hence, we will have a few, but high precision ADCs with structured pruning. On the other hand, unstructured pruning removes weights from every column but may not prune all the weights in a column i.e., unstructured pruning reduces the effective value of r. This allows us to use ADCs with lower precision. However, it does not reduce the number of ADCs required. Hence, unstructured pruning results in many but lower precision ADCs. C2F combines the benefits of both these pruning techniques and maximizes power savings by reducing both the number and size of ADCs.

In addition, dropout can be seen as a form of unstructured pruning. Hence, we can further reduce the precision of ADCs for activations following equation (1). For example, if only 8 out of 128 weights or activations are left on a column after C2F pruning and dropout, this effectively activates only 8 rows (r = 8). Thus, a 4-bit ADC can be used instead of an 8-bit ADC (as in the unpruned model) for the computation. This is a non-trivial reduction in peripheral power as the ADC power increases rapidly with the bit precision [3]. The power consumption increases by approximately 1.5-2× for every additional bit of resolution. Overall, C2F pruning combined with dropout can reduce both the precision and the number of ADCs required.

However, the amount of pruning varies with each CNN layer. The first few layers in the CNN have few weights that cannot be easily pruned, while the later convolution layers can be pruned to a greater extent [13]. As a result, C2F (and other pruning techniques) achieve different levels of sparsity for different CNN layers. This necessitates ADCs with different precisions for each layer. Hence, we propose to incorporate a

Authorized licensed use limited to: Washington State University. Downloaded on September 29,2024 at 15:13:56 UTC from IEEE Xplore. Restrictions apply.



reconfigurable ADC design (discussed next), that can enable different precisions as needed.

D. *Time-Based ADC Design for ReRAM Crossbars.*

Existing Successive Approximation Register (SAR) and Flash ADC architectures have been adopted for the ReRAM-based accelerators [3]. SAR ADCs are commonly used for ReRAM crossbars due to their relatively simple structure and low area/power overhead compared to Flash ADCs [3]. However, SAR ADCs do not scale with technology, due to their analog sub-block circuits. On the other hand, Flash ADCs provide higher conversion speed, but they suffer from limited precision, which is not suitable for CNN training.

Time-based ADCs offer superior time resolution and scalability, making them attractive for designing complex PIM systems [16] [17]. The time-based ADC design is composed of two primary blocks, namely the voltage-to-time converter (VTC) and the time-to-digital converter (TDC). The analog input signal is sampled and processed by the VTC to produce an output pulse width that corresponds to the input voltage. This timed signal is subsequently converted to the digital domain by the TDC, as illustrated in Fig. 3. In this context, a timed signal implies a time-delayed signal where differently delayed clock signals are generated based on the input voltage.

The proposed ADC features a new VTC with a low-power consumption, process, and temperature-insensitive design, high linearity, and dynamic range [16]. The VTC consists of three main components: a sampler, a delay generation circuit with self-tracking, and a pulse generator. The sampler generates a discrete-time voltage signal from the input voltage, which is then used by the delay generation circuit to produce a delayed pulse width modulation (PWM) signal. The proposed VTC design, with its delay generation circuit featuring self-tracking, achieves superior linearity. This makes it highly effective, regardless of temperature or other environmental factors. We reduce this ADC architecture's dynamic power consumption by limiting the ramp signal, making it a power-efficient solution. Furthermore, the delay increases linearly with the sampled input voltages, ensuring accuracy and stability in detecting input signals. Incorporating a self-tracking circuit and limiting the ramp makes the proposed ADC architecture a highly effective solution for ReRAM-based architectures.

IV. Experimental Evaluation

A. *Experimental Setup*

We evaluate the C2F pruning method on the CIFAR-10 dataset using diverse CNN models: VGG-11 (V11), VGG-19 (V19), ResNet-18 (R18), ResNet-34 (R34), and GoogleNet (GN) [13]. The C2F pruning is implemented on an NVIDIA Titan GPU with 24GB of memory. The C2F pruning is offline and it results in a pruned but untrained CNN, which is then mapped to a ReRAM-based PIM architecture and trained with dropout for 50 epochs and a learning rate of 0.01 as an example. In this work, we simulate the on-chip training process on the ReRAM-based architecture using NeuroSim [18]. We follow the hierarchical ReRAM tile configuration presented in [6]. Each ReRAM tile consists of multiple 128×128 crossbar arrays that can be used for storage and computation with both weights and activations [6]. The weights and activations are stored using 16-bit fixed-point precision. Each tile contains a 16-bit reconfigurable LFSR operating at 1GHz for implementing dropout, which consumes less than 1% of the ReRAM tile area and power [19]. We summarize the hardware specifications of the ReRAM-based PIM architecture in Table I.

TABLE I. ReRAM-based PIM Architecture Specifications

16 PEs, 4 tiles per PE	
ReRAM Tile	96 Time-based ADCs, 128×96 DACs (1-bit), 96 crossbars, 128×128 crossbar array, 2-bit/cell resolution, 0.34W, 0.38 mm^2 [19] 1 Programable LFSR (16-bit)

TABLE II. Time-Based ADC Power Requirements

$ADC_{precision}$	*3-bits*	*4-bits*	*5-bits*	*6-bits*	*7-bits*	*8-bits*
Power (mW)	0.18	0.20	0.24	0.31	0.46	0.77

ADC Power and Area: The power consumption of the proposed time-based reconfigurable ADC is shown in Table II for various bit precisions obtained through Cadence simulations. The proposed Time-based reconfigurable ADC operates at a 1.2GHz sampling frequency and occupies an area of around 0.0013mm^2 in the TSMC-28nm technology node. The VTC's size and power consumption are independent of the number of bits, as only the number of TDC increases based on the number of bits and its area. Therefore, the power behavior is not linear with respect to the number of bits. The proposed VTC architecture, along with the constant field scaling and time interleaving, offers a promising solution for reducing the power consumption of ReRAM-based architectures while maintaining high accuracy and performance.

Baseline Pruning: To ensure a fair and thorough evaluation, we consider the unpruned CNN model as a baseline (BL). We benchmark the performance of the C2F pruned models trained with dropout, with two existing techniques. We consider the existing standard LTP method as the representative unstructured pruning method [13], and a recently proposed crossbar-aware structured pruning (CSP) technique [10]. The CSP method utilizes a multi-group LASSO algorithm to prune groups of weights that would otherwise be mapped along a column in a ReRAM crossbar. We implement iterative pruning (as shown in Algorithm 1) in all the methods to ensure that maximum sparsity can be achieved without incurring significant accuracy loss.

B. *Accuracy of C2F Pruned Models with Dropout*

As mentioned earlier, we must prune weights and activations as they both must be stored on-chip for training. C2F inherently prunes weights and some activations. We further increase the sparsity on the activations using Dropout to reduce ADC precision and number; this step leads to high power savings. Dropout is applied to the fully connected layers to improve inferencing accuracy. However, the last fully connected layers generate fewer activations than the initial convolution layers. Hence, having dropout only on the last few layers is not very useful for reducing storage requirements.

Authorized licensed use limited to: Washington State University. Downloaded on September 29,2024 at 15:13:56 UTC from IEEE Xplore. Restrictions apply.

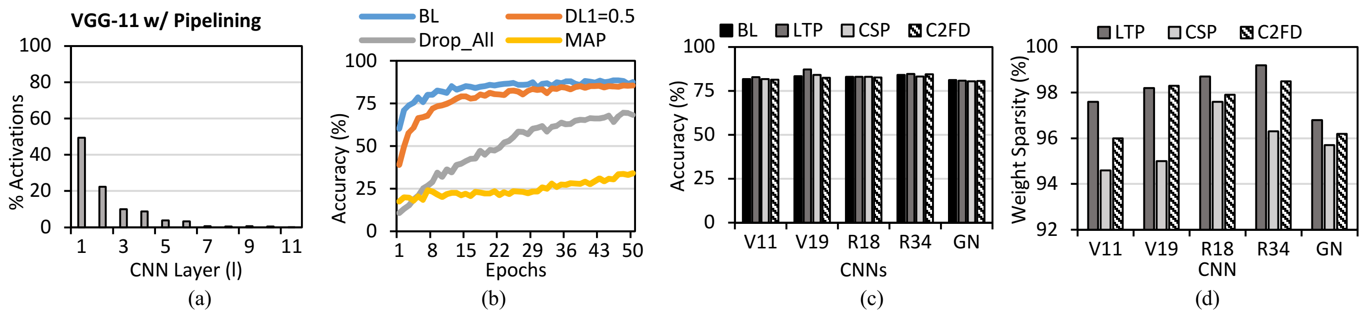


Fig. 4: (a) Distribution of activations across multiple VGG11 layers during pipelined training, (b) prediction accuracy of VGG-11 C2F pruned model with dropout on the first layer (D_{L1}), all layers (*Drop_All*), magnitude pruning (MAP) compared to the baseline (BL), (c) Overall accuracy of pruned CNNs using LTP, CSP, & C2FD compared to the unpruned model BL, (d) average sparsity of pruned CNNs using LTP, CSP, & C2FD.

Fig 4(a) shows the distribution of activations across layers during the pipelined training of VGG11 on the ReRAM-based architecture. As mentioned earlier in Section IIIB, the first layer of VGG-11 generates ~50% of the activations during pipelined training on the ReRAM-based architecture, while the last few layers generate less than 2%, as shown in Fig. 4(a). Hence, simply applying dropout to just the last few fully connect layers only is insufficient. Applying dropout to all layers leads to diminishing returns as the majority of the activation storage is used to store the data from the first CNN layer. Hence, we apply a dropout ratio of 0.5 on the activations of the first layer (D_{L1}=0.5) while training the C2F pruned model on the ReRAM-based architecture. We set the dropout ratio in the first layer to be 0.5 (D_{L1}=0.5), as this is sufficient to ensure a 1-bit reduction in the ADC precision requirement.

We compare the accuracy of this selective Dropout with two other activation pruning methods (a) pruning activations using simple magnitude-based pruning (MAP) and (b) a Dropout of 50% is incorporated in all CNN layers (*Drop_All*). Both MAP and *Drop_All* include C2F for weight pruning. Fig. 4(b) shows the accuracy of the C2F pruned model trained on the ReRAM-based architecture with different activation pruning methods incorporated. As shown in Fig.4(b), pruning activations using MAP causes the CNN model to train poorly (<30% accuracy). Moreover, implementing MAP on-chip requires sorting of the activations based on their magnitude, which requires additional hardware overhead. It is also evident that *Drop_All* leads to poorer training performance compared to the baseline unpruned model (BL). Hence, in this work, we use dropout in the first layer (D_{L1}=0.5), which is a simple and effective method to prune activations without drastic accuracy degradation. Following existing work, we also apply dropout on the last fully connected layers to prevent overfitting and ensure no accuracy loss [15]. C2F reduces both the number of weights and activations and Dropout reduces the number of activations further. Overall, this leads to significantly lower energy requirements. Henceforth, we refer to the C2F pruned model with dropout as **C2FD**. In all the subsequent analyses, we employ C2FD as the overall pruning method.

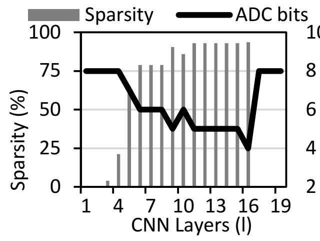
Fig. 4(c) and Fig. 4(d), compare the effectiveness of the C2FD in terms of the prediction accuracy and the achievable sparsity respectively with other pruning methods. The CNN models are pruned using each pruning technique (LTP, CSP, & C2F), and then we map the pruned CNNs to the ReRAM-based platform for the training evaluation. The goal of the iterative pruning approach (Algorithm 1) is to find the sparsest model that can be trained from scratch with minimal accuracy loss (<1%). As shown in Fig. 4(c), the C2FD pruned model achieves comparable accuracy with other pruning methods for all CNN models. In Fig 4(d), we show the percentage of weights pruned (sparsity) using each method (UP, CSP, & C2F). LTP achieves the highest sparsity (99.20%) due to its unstructured nature, while CSP achieves the least sparsity because of its structured pruning (94.60%). C2FD combines both unstructured and structured pruning to achieve high sparsity (98.40%) as shown in Fig 4(d). Moreover, as discussed earlier, in addition to pruning the weights, C2FD also prunes activations due to its filter-wise pruning and dropout. For example, in V19, C2FD achieves an average activation sparsity of 53.27%, while LTP and CSP achieve a lower activation sparsity of 30.25% and 33.41% respectively. Next, we present the power savings when training the C2FD-enabled CNN on the ReRAM-based PIM.

C. *Overall Power Analysis*

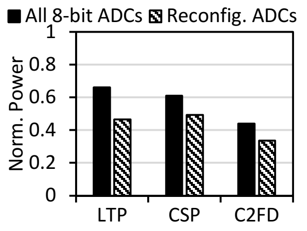
It is well known that the high precision ADC peripheral circuits in ReRAM-based PIM architectures contribute significantly to the overall chip power consumption [3] [6]. Hence, the reconfigurable ADC design proposed in this work enables optimizations that focus on reducing the bit precision of ADCs, thereby reducing the overall power consumption. As discussed earlier, the C2F pruning approach leads to varying sparsity levels across different CNN layers as shown in Fig. 5(a). Highly sparse CNN layers tend to have fewer weights per ReRAM crossbar column left on average after pruning (smaller r in eqn. (1)). Hence, the ADCs required for such layers can be reconfigured to use a lower bit precision to save power. In Fig. 5(a), we show the minimum ADC requirements and per-layer sparsity of the weights in the C2FD pruned VGG-19 CNN as an example. Here, we observe that the initial and final layers have less sparsity (i.e., more weights remaining), and hence require high-precision ADCs (8 bits). Meanwhile, the intermediate layers are significantly pruned and thus require low-precision ADCs (4-6 bits). This necessitates a reconfigurable ADC design for the ReRAM-based architecture as proposed in this work.

Fig. 5(b) compares the power consumption of two types of ReRAM-based PIM architectures: with uniform 8-bit ADC design, and the proposed architecture with reconfigurable ADC design. Here, we map the VGG19 pruned model obtained using the offline pruning techniques (LTP, CSP, & C2FD) to both architectures. We observe that for each pruned model, the reconfigurable ADC design inherently consumes less power compared to the all 8-bit ADC counterpart. Moreover, the C2FD

Authorized licensed use limited to: Washington State University. Downloaded on September 29,2024 at 15:13:56 UTC from IEEE Xplore. Restrictions apply.



(a)



(b)

Fig. 5: (a) ADC requirements and sparsity distribution for C2F pruned VGG19, (b) Power consumption for LTP, CSP and C2FD pruned VGG-19 mapped to an all 8-bit ADC design vs the proposed Reconfigurable ADC design, all normalized w.r.t. the unpruned VGG-19 CNN model.

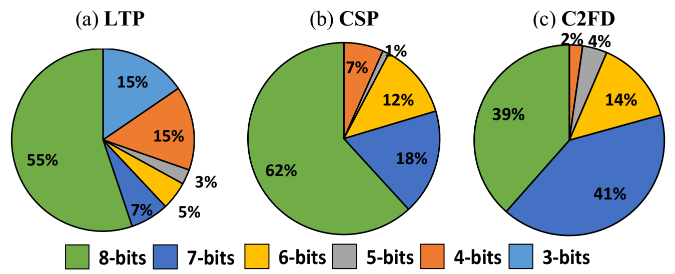


Fig. 6: Breakdown of ADC requirements for ResNet18 after pruning using; (a) unstructured pruning (LTP), (b) Crossbar-aware structured pruning (CSP) (c) the proposed coarse- to -fine- grained pruning with dropout (C2FD).

pruned model consumes the least power (<40%) when it is mapped to the reconfigurable ADC design.

Fig. 6 shows the breakdown of the ADC precision requirements when the ResNet-18 CNN is pruned using all three methods (LTP, CSP, & C2FD) as an example. Each pruned version of ResNet-18 is mapped onto the ReRAM architecture with reconfigurable ADCs. As seen in Fig. 6, the LTP and CSP pruned models require more high-precision ADCs compared to C2FD. For example, 8-bit ADCs constitute 55% and 62% of the total number of ADCs for LTP and CSP respectively, while for C2FD, this number is only 39%. This is because the finer-grained pruning in C2FD reduces the number of elements left in each ReRAM crossbar column, thereby reducing r (as in eqn. (1)) and the ADC precision. Moreover, the dropout incorporated in C2FD further reduces the number of 8-bit ADCs from 48% in C2F only to 39% as we show in Fig. 6(c). Hence, the C2FD-enabled pruned model requires fewer 8-bit ADCs and more lower precision ADCs (7bits, 6bits, etc.). This precision reduction leads to overall power savings.

We evaluate the overall power consumption of the ReRAM-based architecture with reconfigurable ADC design. Fig. 7 presents the overall power consumption of the C2FD pruned model compared to other pruning methods for the different models considered in this work. The C2FD pruned model achieves ~50% reduction in power consumption compared to the unpruned model (BL). Moreover, the C2FD pruned model consumes ~10% and ~18% less power compared to the CSP and LTP methods respectively. Overall, the C2FD pruning enables significant power savings on the ReRAM-based architecture with the proposed reconfigurable ADC design.

V. Conclusion

In this work, we demonstrate that coarse-to-fine weight pruning and dropout-enabled activation pruning enable us to

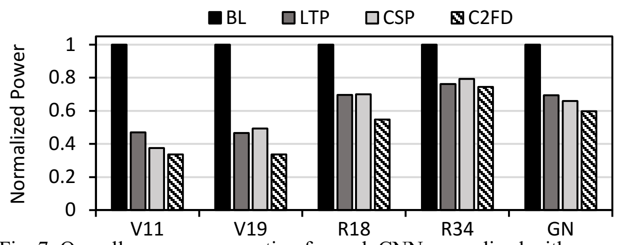


Fig. 7: Overall power consumption for each CNN normalized with respect to their unpruned versions (BL).

reduce both the number and precision of power-hungry ADC circuits in ReRAM-based architectures for training CNNs. By incorporating a reconfigurable ADC architecture, we can reduce the ADC requirements (both number and precision) significantly without any noticeable loss in model accuracy while training the CNN models. This approach effectively reduces the overall on-chip power consumption by ~50% compared to the unpruned counterpart.

Refrences

[1] G. Yuan et al., "FORMS: Fine-grained Polarized ReRAM-based In-situ Computation for Mixed-signal DNN Accelerator," in *ISCA*, 2021.

[2] S. Huang et.al, "Mixed Precision Quantization for ReRAM-based DNN Inference Accelerators," in *ASP-DAC*, 2021.

[3] K. Roy et al., "In-Memory Computing in Emerging Memory Technologies for Machine Learning: An Overview," in *(DAC)*, 2020.

[4] G. Yuan et. al, "TinyADC: Peripheral Circuit-aware Weight Pruning Framework for Mixed-signal DNN Accelerators," in *DATE*, 2021.

[5] S. Mohapatra et al., "Low-Power Process and Temperature-Invariant Constant Slope-and-Swing Ramp-Based Phase Interpolator," *IEEE JSSC,* 2023.

[6] A. Shafiee et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars Ali," in *ISCA*, 2016.

[7] L. Song, X. Qian, L. Hai and Y. Chen, "PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning," in *IEEE HPCA*, 2017.

[8] A. Ankit et. al, "PANTHER: A Programmable Architecture for Neural Network Training Harnessing Energy-Efficient ReRAM," *IEEE Transactions on Computers,* 2020.

[9] C. Ogbogu et. al, "Accelerating Large-Scale Graph Neural Network Training on Crossbar Diet," *IEEE TCAD,* 2022.

[10] J. Meng et al., "Structured Pruning of RRAM Crossbars for Efficient In-Memory Computing Acceleration of Deep Neural Networks," *IEEE Transactions on Circuits and Systems II: Express Briefs,* vol. 68, no. 5, pp. 1576-1580, 2021.

[11] Y. He et. al, "InfoX: An Energy-Efficient ReRAM Accelerator Design with Information-Lossless Low-Bit ADCs," in *IEEE DAC*, 2022.

[12] D. Kim et. al, "SAMBA : Sparsity Aware In-Memory Computing Based Machine Learning Accelerator," *IEEE Transactions on Computers* , 2023.

[13] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *ICLR*, 2019.

[14] A. Morcos, Y. Haonan, M. Paganini and Y. Tian, "One ticket to win them all: Generalizing lottery ticket initializations across datasets and optimizers," NeurIPS, 2019.

[15] N. Srivastava et. al, "Dropout: A simple way to prevent neural networks from overfitting," *JMLR,* 2014.

[16] K. Ohhata, "A 2.3-mW, 1-GHz, 8-Bit Fully Time-Based Two-Step ADC Using a High-Linearity Dynamic VTC," *IEEE JSSC,* 2019.

[17] M. Zhang et.al, "A 20GS/s 8b Time-Interleaved Time-Domain ADC with Input-Independent Background Timing Skew Calibration," in *Symposium on VLSI Circuits*, 2021.

[18] X. Peng et al., "DNN+NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *arXiv:2003.06471,* 2020.

[19] A. Arka et al., "DARe: DropLayer-Aware Manycore ReRAM architecture for Training Graph Neural Networks," in *ICCAD*, 2021.

Authorized licensed use limited to: Washington State University. Downloaded on September 29,2024 at 15:13:56 UTC from IEEE Xplore. Restrictions apply.