Automated Quantization Range Mapping for DAC/ADC Non-linearity in Computing-In-Memory

Chi-Tse Huang, Yu-Chuan Chuang, Ming-Guang Lin, and An-Yeu (Andy) Wu

Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan
{rickhuang, frankchuang, chrislin}@access.ee.ntu.edu.tw, andywu@ntu.edu.tw

Abstract—Computing-in-memory (CIM) has demonstrated the great potential of analog computing in improving the energy efficiency of matrix-vector multiplications for deep learning applications. Albeit low-power feature of CIM, the non-linearity of digital-to-analog converters (DACs)/analog-to-digital converters (ADCs) causes deviation between the computed outputs and desired values, thus degrading classification accuracy. This paper proposes Automated Quantization Range Mapping (A-QRM) mechanism to mitigate the negative effect of non-linearity on model accuracy. Instead of fixing the quantization range for quantized deep learning models, the proposed A-QRM automatically finds a better quantization range that balances the model capability and quantization errors caused by the non-linearity. Experimental results show that our proposed A-QRM achieves 89.02% and 86.93% of top-1 accuracy in ResNet20 and VGG8 on Cifar-10, respectively, under the non-linearity of DACs/ADCs.

Keywords—Computing-in-memory, non-linearity of DACs/ADCs, deep neural networks (DNNs).

I. INTRODUCTION

Deep neural networks (DNNs) have achieved remarkable success in various applications such as video analysis, image recognition, and natural language processing. In practice, most DNN accelerators are based on von-Neumann architectures, which separate processing elements (PEs) from memories. However, the growing complexity of DNNs results in frequent memory access, which requires substantial amounts of energy and poses a significant challenge to DNN-based applications on edge devices.

To break the von-Neumann bottleneck, computing-in-memory (CIM) has been proposed as new computing paradigm. It enables parallel execution of matrix-vector multiplications (MVMs), the most dominating operations in DNNs, within memory sub-arrays. Therefore, the energy consumption and latency for memory access can be dramatically reduced. The static random access memory (SRAM)-based CIM had been implemented in [1], [2], [3], to demonstrate the high energy efficiency of CIM during DNN inference.

Although SRAM-based CIM exhibits higher energy efficiency than traditional von-Neumann architecture, it suffers from reliability issues caused by the non-linearity of digital-to-analog converters (DACs) and analog-to-digital converters (ADCs). The non-linearity significantly deviates the inputs and computed outputs from desired values. These errors would accumulate and propagate in different DNN layers, eventually leading to a severe accuracy drop [3].

To address the reliability issues, prior works proposed hardware compensation [1], [2], [3] and hardware calibration methods [1], [4]. Nevertheless, both methods required extra circuit components to improve the linearity of DACs/ADCs, causing additional area overhead for CIM. On the other hand, non-linearity-aware training (NAT) that trains DNNs with analytical models of DACs/ADCs demonstrate alternative approach [3], [5]. To be exact, NAT makes DNNs adapt to the non-linearity instead of enhancing the linearity of DACs/ADCs. CxDNN [5] proposed a software training flow to make DNNs more robust to the non-linearities. The training flow consists of the following three stages: quantizing DNN’s weights and activations, modeling the non-linearities of DACs/ADCs, and, most importantly, determining suitable quantization ranges for DNN training with the non-linearities. While CxDNN achieved accuracy improvements under the non-linearities of DACs/ADCs, some challenges are still worth addressing:

1) No investigations on the trade-off between model capability and quantization errors under the non-linearity of DACs/ADCs: The quantization ranges for DACs/ADCs determine the trade-off between model capability and quantization errors caused by the non-linearities. Specifically, DNNs with a smaller quantization range can mitigate the quantization errors but restrict their numerical range for training. On the other hand, DNNs with a larger quantization range have a more flexible numerical range for achieving higher accuracy; however, the quantization errors are amplified simultaneously. To the best of our knowledge, no work investigates this trade-off in CIM.

2) The heuristic-based method for finding quantization ranges is time-consuming and infeasible: To find the suitable quantization ranges, CxDNN exhaustively tries all possible candidates in a pre-set list and selects the one with the highest classification accuracy in a layer-wise manner. However, the exploration is time-consuming and even infeasible as DNNs grow deep. Moreover, we argue that deciding the ranges in a layer-wise manner only achieves sub-optimal performance due to the lack of information about the impacts of ranges on the entire network.

In this paper, we first explore the trade-off between model capability and quantization errors. After that, considering the trade-off, automated quantization range mapping (A-QRM) is proposed to automatically optimize the quantization ranges during model training. The contributions of this work include:

1) Explore the trade-off determined by quantization ranges: The quantization ranges for DNN can be used as knobs to control the trade-off between model capability and quantization errors. We manually adjust the quantization

2) Heuristic-based method for finding quantization ranges is time-consuming and infeasible: To find the suitable quantization ranges, CxDNN exhaustively tries all possible candidates in a pre-set list and selects the one with the highest classification accuracy in a layer-wise manner. However, the exploration is time-consuming and even infeasible as DNNs grow deep. Moreover, we argue that deciding the ranges in a layer-wise manner only achieves sub-optimal performance due to the lack of information about the impacts of ranges on the entire network.
ranges and show a peak point that can strike the right balance and reach better accuracy under the non-linearities of DACs/ADCs.

2) Automated quantization range mapping (A-QRM): We propose A-QRM, a new training scheme to make DNNs automatically learn suitable quantization ranges for each layer. Inspired by [6], we introduce a trainable parameter \( \max \) for each layer representing the quantization range \([0, \max]\) for inputs. The trainable parameter \( \max \) allows models to balance the model capability and quantization errors caused by the non-linearities of DACs/ADCs. Experimental results show the effectiveness of A-QRM on the Cifar-10 dataset. Compared with CxDNN, A-QRM achieves 0.31% and 0.5% accuracy improvements in ResNet20 and VGG8, respectively.

II. BACKGROUND

A. Computing-in-memory (CIM)

A general CIM architecture is shown in Fig. 1. CIM performs computations by using the following three stages. First, load pre-trained weights \( W_{11}, ..., W_{wh} \) into the memory crossbar, where \( M \) and \( N \) are the numbers of columns and rows in the memory crossbar. Second, convert the input patterns into voltages through DACs and apply them on word lines (WLs). The memory crossbar multiplies the input with loaded weights and accumulates the resulting currents on bit lines (BLs). Finally, ADCs convert the analog currents into digital outputs, and peripheral circuits execute shift-and-add operations to obtain final outputs.

B. Related Work: CxDNN [5]

CxDNN developed in [5] is a statistics-driven approach to identify suitable quantization ranges for each DNN layer. Each layer first determines a list of possible ranges by input distribution. Then, CxDNN exhaustively tries all possible candidates in the pre-set list and selects the one with the highest classification accuracy in a layer-wise manner. When the arrangement of quantization ranges is determined, CxDNN retrains the model to recover the degraded accuracy caused by the non-linearity of DACs/ADCs. Below is the CxDNN retraining method. CxDNN first quantizes weights and activations of a floating-point (FP32)-based DNN to match the resolutions of DACs/ADCs. CxDNN also models the non-linear curves of DACs/ADCs by look-up tables (LUTs) to simulate the conditions on a crossbar-based hardware fabric. Then, CxDNN retrains the quantized DNN with the manually selected quantization ranges and the non-linearity model to make the quantized DNN adapt and be robust to the non-linearity.

III. PROPOSED AUTOMATED QUANTIZATION RANGE MAPPING

A. Trade-offs of Quantization Ranges

Since both inputs and outputs in CIM are operated in the digital domain, DNNs must be quantized to fit the resolutions of DACs/ADCs. Moreover, scaling factors are computed to recover converted digital codes to quantized values. A scaling factor \( s_{f_i} \) for inputs \( i \) in the \( i \)th layer is obtained by dividing the quantization range with the number of intervals between digital codes, which can be expressed as

\[
s_{f_i} = \frac{\max(i) - 0}{2^k - 1}. \tag{1}
\]

\( k \) is the resolution of DAC and \( \max(i) \) returns the maximum of input. Note that we use an unbounded quantization range \([0, \max(i)]\) here as an example, which means that inputs are not clipped. With the scaling factors for inputs/weights, the final output \( y_i \) of CIM can be computed as

\[
y_i = \text{ADC} \left( \text{MVM} \left( \text{DAC} \left( \text{round} \left( \frac{i}{s_{f_i}} \right) \right) \cdot W_{i} \right) \right) \cdot s_{w_{i}} \cdot s_{f_i}. \tag{2}
\]

\( W_{i} \) is quantized weights in the \( i \)th layer, \( \text{round}() \) maps inputs to the nearest integer values, \( \text{MVM}(I, W) \) performs MVMs between \( I \) and \( W \), \( s_{w_{i}} \) is the scaling factor of quantized weights in the \( i \)th layer, \( \text{DAC}() \) converts digital values into voltages, and \( \text{ADC}() \) converts accumulated currents back to the digital values.

Using the unbounded quantization range for CIM is a popular method. However, this approach is susceptible to quantization errors caused by non-linearity. As shown in Fig. 2, a larger quantization range produces a larger scaling factor as well as magnifies quantization errors caused by the non-linearity of DACs/ADCs. A smaller quantization range results in smaller scaling factors and thus reduces the quantization errors. However, it also restricts the model capability since the numerical range available for the model...
is narrowed simultaneously. To validate the trade-off of the quantization range, we adjust the upper bound of the range and fix the lower bound at 0. Fig. 3 illustrates that as the quantization range (the upper bound) becomes smaller, the accuracy of the FP32 model without quantization errors is decreased. This is because the numerical range for the model would become narrower and thus restricts the model capability. On the other hand, the quantized model affected by the non-linearity of DACs/ADCs shows a bell-shaped accuracy curve, demonstrating the trade-off between model capability and quantization errors caused by the non-linearity of DACs/ADCs. This trade-off also indicates that the quantization range should be carefully determined to achieve high accuracy in CIM.

B. Automated Quantization Range Mapping (A-QRM)

Inspired by PACT [6] that aims to find the optimal quantization ranges for low-bit-width models, we propose Automated Quantization Range Mapping (A-QRM) to make DNNs automatically learn suitable quantization ranges under the non-linearity of DACs/ADCs. A-QRM parameterize the upper bound of the quantization range in the $i^{th}$ layer as $\max_i$, and clip the input activation as

$$l_{\text{clip}} = \text{clip}(l_i, \max_i). \quad (3)$$

where $\text{clip}(x, u)$ clips the range of $x$ within the range $[0, u]$.

Algorithm 1. Training Flow of A-QRM

**Input:** $W_{\text{FP32}}$ – floating point weights, $\{\rho_w, \rho_D, \rho_{\text{ADC}}\}$ – precisions of weights, inputs, and ADC, $cs$ – crossbar size, $TrainSet$ – training data $TrainLabel$ – training label $V_{\text{init}}$ – initial value of upper bound in quantization range $lr$ – learning rate

**Output:** $W_{\text{DRM}}$ – non-linearity-aware trained weights with A-QRM $l_{\text{max}}$ – list of quantization ranges for each layer

**Forward Pass**

1: $W_{\text{CIM}}, N_c = \text{convertToCIM}(W_{\text{FP32}}, \rho_w, \rho_D, \rho_{\text{ADC}}, cs)$
2: $l_{\text{max}} = \text{Parameterize}(\{V_{\text{init}} \text{ for (i in 1: } N_{\text{layer}} \text{ )}\})$
3: $y = \text{TrainSet}$
4: for (layer in 1: $N_{\text{layer}}$) do
5: \hspace{1em} $l = y$
6: \hspace{1em} $\text{PartialSum} = []$
7: \hspace{1em} $l_{\text{clip}} = \text{A-QRM}(l, l_{\text{max}})$
8: \hspace{1em} $l_{q_i,j} = \text{DAC Split}(l_{\text{clip}}, cs)$
9: \hspace{1em} $W_{q_i,j} = \text{SplitWeight}(W_{\text{CIM}}, cs)$
10: \hspace{1em} for (j in 1: $N_c$) do
11: \hspace{2em} $\text{PartialSum}.\text{append}(\text{ADC Conv}(l_{q_i,j}, W_{q_i,j}))$
12: \hspace{1em} end for
13: \hspace{1em} $y = \text{convertToFeatureMap(PartialSum)}$
14: \hspace{1em} end for
15: $y = \text{softmax}(y)$
16: $\text{loss} = CE(y, \text{TrainLabel})$, CE: cross entropy loss function

**Backward Pass**

17: for (layer in 1: $N_{\text{layer}}$) do
18: \hspace{1em} $\text{gradient}_{l_{\text{max}}^i} = \frac{\partial \text{loss}}{\partial l_{\text{max}}^i}$
19: \hspace{1em} $l_{\text{max}}^i = l_{\text{max}}^i - lr \times \text{gradient}_{l_{\text{max}}^i}$
20: end for

The training flow of A-QRM is summarized in Algorithm 1 and depicted in Fig. 4. Since each layer is processed sequentially, we take the $i^{th}$ layer as an example in the...
following description. The three steps in the procedure are described as follows:

Step 1. Parameterize the quantization range of input: We convert an FP32 model into a fixed-point model to fit the CIM settings by $convertToF(\text{CIM})$ and parameterize the upper bound of the quantization range in the $i^{th}$ layer as $\max_i$ by $\text{Parameterize}(\cdot)$. After that, we pass the input $I_i$ into the A-QRM. This step is depicted in row 7 of Algorithm 1.

Step 2. Split activation and weights: We split the input $I_{\text{clip}_i}$ and quantized weights $W_i$ into several groups by $\text{Split}(\cdot)$ to fit the crossbar size and perform convolution individually. The multiple partial sums are obtained and are expressed as $\text{Partialsum}_i$. After that, we transfer the partial sums into the non-linear ADC model to get the output feature map $y_i$ in the $i^{th}$ layer through shift-and-add operations. This step is listed in rows 8-12 of Algorithm 1.

Step 3. Calculate gradients of $\max_i$: After each layer finishes forward pass, we compute the gradients of $\max_i$ as

$$\frac{\partial I_{\text{clip}_i}}{\partial \max_i} = \begin{cases} 0, & I_i < \max_i \\ 1, & I_i > \max_i \end{cases}$$  \hfill (4)

Specifically, we backward the partial gradient of the activation $I_i$ which is located in the range $(\max_i, \infty)$ to update the $\max_i$.

IV. EXPERIMENTAL RESULTS

A. Dataset and Simulation Settings

We verify the effectiveness of the proposed A-QRM in ResNet20 and VGG8 on CIFAR-10 [7], which is a standard benchmark for DNNs. Because quantization errors caused by DAC’s non-linearity would propagate to ADC, we can integrate the non-linearity of DACs/ADCs into one non-linear transfer curve. The non-linear curve is generated by Matlab code, which simulates a behavior model of a successive-approximation register ADC (SAR-ADC) with capacitance mismatch and distortion. The specifications of the used SAR-ADC are listed in Table I. Our experimental setting is the same as CxDNN with 6-bit weight, 6-bit input, and 10-bit resolution of ADC.

B. Comparisons of Classification Accuracy

Table II shows the accuracy of ResNet20 and VGG8 with different frameworks. ReLU indicates that inputs $I_i$ are clipped within the range of $[0, \max(I_i)]$, which means the quantization ranges are unbounded. Fixed quantization range means that we set the quantization range for the entire DNN as the same value.

Compared with the ideal CIM, CIM with the non-idealities of DACs/ADCs severely degrades the accuracy of ResNet20 and VGG8 by 51.14% and 72.22%, respectively. Fixed quantization range outperforms ReLU since the latter approach results in larger scaling factors and thus amplifies the quantization errors from the non-linearity. CxDNN achieves higher accuracy than the fixed quantization range since it considers the numerical characteristics of each layer and heuristically searches for the most suitable arrangement for the quantization ranges. Compared with CxDNN, the proposed A-QRM can further improve the accuracy by 0.31% and 0.41% on ResNet20 and VGG8, respectively. This is because A-QRM can consider the impacts of quantization ranges for an entire network during model training. Moreover, A-QRM does not require running several different settings to find the best arrangement of the quantization ranges compared with the heuristic method in CxDNN.

V. CONCLUSION

This paper investigates the trade-off of quantization ranges under the non-linearity of DACs/ADCs in CIM, which shows that the quantization ranges would trade model capability with quantization errors and significantly influence accuracy. We propose the A-QRM to automatically find the suitable quantization ranges and effectively recover the degraded accuracy. A-QRM parameterizes the quantization ranges and clips inputs to mitigate the negative impact from the non-linearity. The experimental results demonstrate that the proposed A-QRM improves the accuracy by 0.31% and 0.5% on ResNet20 and VGG8, respectively, compared with the state-of-the-art CxDNN [5].

Acknowledgments. This work was supported by the Ministry of Science and Technology of Taiwan under Grants MOST 110-2218-E-002-034-MBK.

REFERENCES


<p>| TABLE I. Specifications of ADC. |</p>
<table>
<thead>
<tr>
<th></th>
<th>Architecture</th>
<th>SAR-ADC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution [bits]</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Sample Rate [MS/s]</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>Supply Voltage [V]</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

<p>| TABLE II. Accuracy comparisons with different frameworks |
| --- | --- | --- |</p>
<table>
<thead>
<tr>
<th>Framework</th>
<th>Model</th>
<th>ResNet20</th>
<th>VGG8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Non-linearity-aware Training</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ideal CIM (upper bound)</td>
<td>91.05%</td>
<td>89.85%</td>
<td></td>
</tr>
<tr>
<td>CIM with DAC/ADC Non-linearity</td>
<td>39.91%</td>
<td>17.63%</td>
<td></td>
</tr>
<tr>
<td>With Non-linearity-aware Training</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ReLU (unbounded range)</td>
<td>73.24%</td>
<td>66.56%</td>
<td></td>
</tr>
<tr>
<td>Fixed quantization range</td>
<td>88.50%</td>
<td>86.20%</td>
<td></td>
</tr>
<tr>
<td>CxDNN [5]</td>
<td>88.71%</td>
<td>86.43%</td>
<td></td>
</tr>
<tr>
<td>Proposed A-QRM</td>
<td>89.02%</td>
<td>86.93%</td>
<td></td>
</tr>
</tbody>
</table>
