HaLo-FL: Hardware-Aware Low-Precision Federated Learning

Yeshwanth Venkatesha, Abhiroop Bhattacharjee, Abhishek Moitra, and Priyadarshini Panda
Department of Electrical Engineering, Yale University, USA
{yeshwanth.venkatesha, abhiroop.bhattacharjee, abhishek.moitra, priya.panda}@yale.edu

***Abstract*—Applications of federated learning involve devices with extremely limited computational resources and often with considerable heterogeneity in terms of energy efficiency, latency tolerance, and hardware area. Although low-precision training methods have demonstrated effectiveness in accommodating the device constraints in a centralized setting, their applicability in distributed learning scenarios featuring heterogeneous client capabilities has not been well explored. In this work, we design a hardware-aware low-precision federated training framework (HaLo-FL) tailored to heterogeneous resource-constrained devices. In particular, we optimize the precision for weights, activations, and errors for each client's hardware constraint using a precision selector (named HaLo-PS). To validate our approach, we propose HaLoSim, a hardware evaluation platform that enables precision reconfigurability and evaluates hardware metrics like energy, latency, and area utilization on a crossbar-based In-memory Computing (IMC) platform.**

***Index Terms*—In-memory Computing, Process-in-Memory, Federated Learning, Quantization**

I. Introduction

The advent of Federated Learning (FL) has facilitated the training of models on a large scale, leveraging distributed data sources [1]. FL has demonstrated effectiveness in various application domains, including healthcare [2], [3], IoT [4], [5], autonomous driving [6], [7] among others. Given that FL is deployed in resource-constrained devices, often each with different constraints and priorities, it becomes imperative to address the specific training and inference requirements of the client devices [8]. For example, a drone will have energy constraints, whereas a smartwatch will be constrained in terms of area.

Quantization [9], [10], which involves employing reduced bit representations to represent model parameters, and network pruning [11], which focuses on creating sparse models, are prominent techniques widely adopted to reduce model complexity. Although primarily aimed at enhancing inference, these techniques also effectively reduce the training complexity of client devices. Notably, employing low-precision representations (e.g., INT8 or lower) for model parameters has demonstrated significant reductions in memory usage and computational requirements during training and inference [12]–[15]. Integrating low-precision training into federated learning offers unique benefits by leveraging varied hardware capabilities through customized precision on each client. However, choosing the right parameter precision for each client is non-trivial. The parameters during training—weights, activations, and errors can each be represented with different precision. Note, that errors refer to the gradients calculated during backpropagation. Each combination will have a unique impact on how the computation is mapped on the underlying hardware which in turn affects the energy, latency, and area utilization. For example, in Fig. 1 we vary the precision of each dimension individually while keeping the other two fixed at 8-bit on a hardware simulator. It demonstrates how energy, latency, and area utilization are influenced when adjusting the precision of each parameter.

To tailor the precision of weights, activations, and errors individually for each client, we propose our Hardware-Aware Low-Precision Federated Learning framework (HaLo-FL). HaLo-FL consists of a precision selection module (HaLo-PS) that selects the precision of weights, activations, and errors for each client. Halo-PS is guided by the energy, latency and area metrics provided by the hardware simulator (HaLoSim) that takes into consideration the model architecture and maps it onto the underlying hardware. We use an analog crossbar-based In-memory computing (IMC) architecture as our precision-reconfigurable hardware simulator. Note that while we use a crossbar-based IMC hardware simulator, our method is agnostic to the underlying hardware architecture and will be amenable with any hardware that supports precision reconfigurability. Furthermore, the utilization of reduced precision on the client yields a notable reduction in the amount of data that needs to be communicated during the federated learning process. Overall, our contributions are as follows:

- We design a federated learning framework with a hardware-aware precision selector, HaLo-PS, that takes into consideration the heterogeneous constraints of different clients to determine the most suitable precision for their training based on certain hardware metrics.
- We introduce HaLoSim, a hardware evaluation platform specifically designed for obtaining essential hardware metrics, including energy consumption, latency, and area utilization, in the context of precision-reconfigurable In-memory Computing (IMC) platform. HaLo-PS is guided by HaLoSim.
- We show the effectiveness of our FL framework with 20 clients on CIFAR10, EMNIST, and TinyImagenet datasets. On CIFAR10, our approach achieves a 36% improvement in energy, 22% improvement in latency, and 22% improvement in area utilization for the clients with respective constraints.

979-8-3503-4859-0/DATE24/© 2024 EDAA

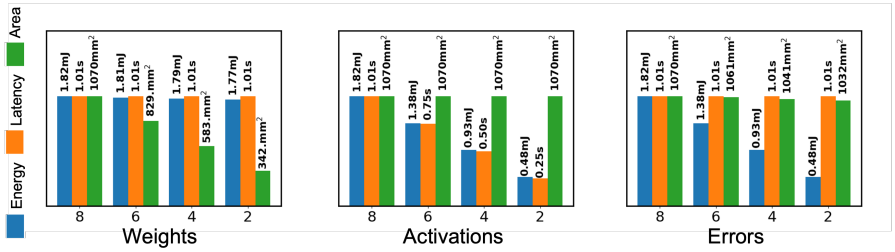
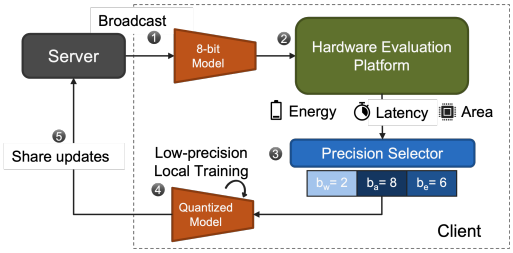
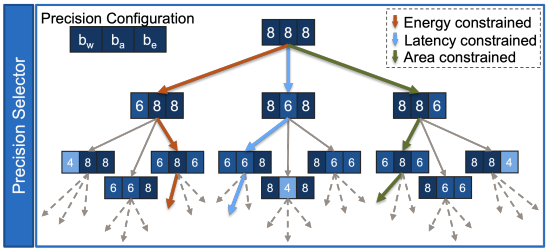


Fig. 1: Effect of precision of weights, activations, and errors on Energy, Latency, and Area utilization on a crossbar-based IMC simulator. We vary one dimension at a time while keeping the other two constant at 8-bits. Varying the precision of weights has a substantial impact on the utilized area, while energy and latency remain unchanged. The precision of errors affects the energy but not the latency and area utilization, whereas the precision of activations affects both energy consumption and latency.



(a) HaLo-FL Framework.



(b) Precision selector module (HaLo-PS).

Fig. 2: Overview of our method. (a) A visual representation illustrating the different components within our framework and their interactions. (b) Illustration depicting HaLo-PS showcasing the impact of each constraint on precision within the clients.

II. Background on Federated Learning

A federated learning system involves a central server and a set of N clients. To start the training process, the central server broadcasts the initial model parameters (denoted as $M_{(s)}^{(0)}$) to all clients. Each client ($c = 1, 2, ..., N$) has their own private dataset ($D^{(c)}$), which they use to train a local model ($M_{(c)}$). The clients communicate the locally trained model to the central server periodically for aggregation. This process of exchanging models between clients and the server constitutes a federated learning round. At each round (r), a subset of participating clients ($S^{(r)}(\ll N)$) sends their models ($M_{(c)}^{(r)}$) to the central server for aggregation. The central server performs a weighted average of the gradients to obtain the updated global model as shown in Eq. 1.

$$M_{(s)}^{(r+1)} = \frac{1}{\sum_{c \epsilon S^{(r)}} |D_{(c)}^{(r)}|} \sum_{c \epsilon S^{(r)}} |D_{(c)}^{(r)}| M_{(c)}^{(r)}, \quad (1)$$

where $|D_r^{(c)}|$ denotes the number of data samples used for local training in client c at round r.

III. HaLo-FL Methodology

As illustrated in Figure 2a, our proposed HaLo-FL framework follows a sequence of steps:

1) **Initialization:** The federated learning process is initiated by the server, which broadcasts the 8-bit model to all participating clients.
2) **Hardware Evaluation Platform:** Each client is equipped with a hardware evaluation platform (HaLoSim) that estimates the energy consumption, latency, and area utilization.
3) **Precision Selection:** The precision selector module (HaLo-PS) determines the optimal precision levels for weights, activations, and errors, taking into account the specific limitations imposed by each client.
4) **Local Training:** The models are then trained locally at each client, utilizing the precision levels determined by the HaLo-PS module.
5) **Model Aggregation:** The model updates from each client are communicated back to the server, where aggregation occurs at 8-bit precision.

A. HaLo-PS

In a typical federated learning system, comprising a server and a set of N clients, the training process occurs over R communication rounds. The precision level refers to the number of bits allocated for representing the weights (b_w), activations (b_a), and errors (b_e) in the model. Initially, all clients are set to have a precision of 8 bits. A step-by-step reduction in precision is then applied to each dimension. This progressive reduction in precision creates a tree-like structure, as depicted in Figure 2b, representing the various combinations of precisions available within the system. The selection of precision within this tree is guided by the feedback provided by the underlying hardware.

The HaLo-PS method is presented in Algorithm 1 outlining the steps involved. To summarize, we first get the constraints of the clients (Steps 1-3). Then we assign a precision of 8 bits to all clients (Steps 4-5). During training, we obtain the choice of precision reduction along each dimension (Step 9). Based on the constraint of each client, we reduce the precision of the dimension that yields maximum gain in efficiency (Steps 10-17). We perform low-precision local training (Step 19) and federated aggregation in the server (Step 21 as per Eq. 1). Note that while we describe a generalized method, there is a limit on how low we can reduce the precision of each dimension depending on the dataset without suffering from a significant loss in accuracy.

Algorithm 1 Precision Selector algorithm.

Input: Clients $c \in [N]$, No. of rounds R, Local datasets $D_c \forall c \in [N]$

Output: Trained model

for client $c = 1, ...N$ **do**
  $constraint_c \leftarrow \{energy, latency, area\}$ // obtain client constraints
end for
$b_w = 8$, $b_a = 8$, $b_e = 8$
$precision_{1...C} \leftarrow \{b_w, b_a, b_e\}$ // Initialize the starting precision
for round $r = 1, ...R$ **do**
  $M_{1...C} \leftarrow M$ // Broadcast the model
  for clients $c = 1, ...C$ **do**
    $choices \leftarrow [\{b_w - 2, b_a, b_e\}, \{b_w, b_a - 2, b_e\}, \{b_w, b_a, b_e - 2\}]$ // Choices of precsion reduction.
    if $constraint_c == energy$ **then**
      $precision_c \leftarrow \text{argmin}_e \text{HaLo-PS}(choices)$
    else if $constraint_c == latency$ **then**
      $precision_c \leftarrow \text{argmin}_l \text{HaLo-PS}(choices)$
    else if $constraint_c == area$ **then**
      $precision_c \leftarrow \text{argmin}_a \text{HaLo-PS}(choices)$
    else
      $precision_c \leftarrow precision_c$
    end if
    $M_c \leftarrow LocalTraining(M)$
  end for
  $M \leftarrow \text{FedAvg}(M_{1...C})$ (Eq. 1)
end for

TABLE I: Hardware properties of the HaLoSim simulator

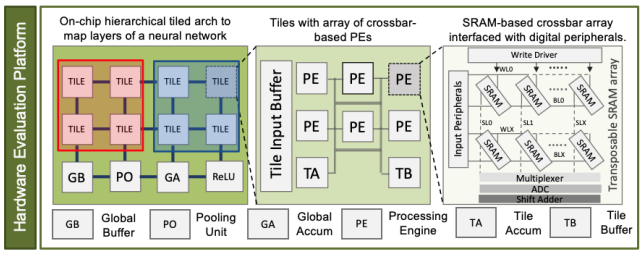
HaLoSim architectural parameters	
Technology	32nm CMOS
No. of PEs per Tile	64
No. of ADCs per PE	8
ADC precision (bits)	6
Crossbar-level parameters [16]	
IMC Device	8T-SRAM
Bits/cell	1
Crossbar size	64×64
Crossbar area	0.07 mm^2
Read energy per crossbar	29 pJ
Write energy per crossbar	13 pJ
Read delay per crossbar	0.018 μs
Write delay per crossbar	0.018 μs

B. HaLoSim

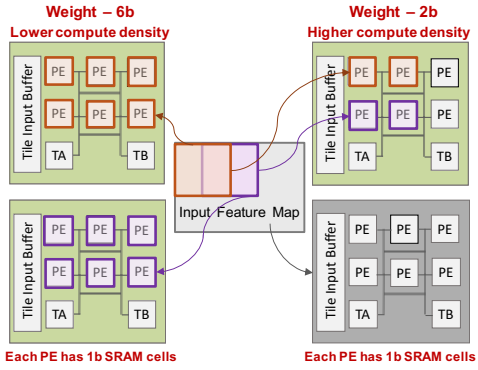
We emulate all Multiply-and-Accumulate (MAC) operations during model training on processing engines (PEs) consisting of SRAM-based IMC crossbar-arrays. Considering Fig. 3a, our underlying hardware platform to carry out training, termed as HaLoSim, consists of an on-chip array of Tiles, each Tile having an array of PEs to perform MAC operations. Inside each PE, there is a crossbar-array of size 64×64 based on 8T-SRAM cells [16], interfaced with digital peripherals such as, input decoders, ADCs, shift-adders and so forth. The relevant hardware details pertaining to HaLoSim are listed in Table I.

For a layer l, the different MAC operations, in order of succession, during model training are:

$$A_{l+1} = W_l * A_l \quad (Forward\ pass) \tag{2}$$



(a) Hardware evaluation platform (HaLoSim).



(b) Precision reconfigurability.

Fig. 3: (a) Illustration of HaLoSim showing the design of hardware evaluation platform. (b) An illustration of precision-reconfigurability in the PEs of HaLoSim. A reduced weight/error precision implies better utilization of the available IMC area to perform MACs in parallel, thereby increasing the compute density.

$$E_l = E_{l+1} * W_l \quad (Backward\ pass) \tag{3}$$

$$\Delta W_l = E_{l+1} * A_l \quad (Backward\ pass) \tag{4}$$

Here, the model weights, activations and errors are respectively denoted as W, A and E. In HaLoSim, a set of Tiles (highlighted in blue in Fig. 3a) is reserved for the MAC operations in the forward pass (Eq. 2) as well as the MAC operations between the errors and weights (Eq. 3) in the backward pass. The model weights (W) mapped onto the SRAM crossbars can be reused across both the rounds of MAC operations. The PEs inside the aforementioned Tiles consist of transposable SRAM-based crossbars [16]. Another set of Tiles (highlighted in red) are reserved for MAC operations between the activations and errors (Eq. 4) in the backward pass, wherein the errors computed using Eq. 3 are mapped onto the SRAM crossbars in the corresponding PEs. A layer l is deployed onto one or more Tiles based on the dimension of W_l for the Tiles highlighted in blue and the dimension of E_{l+1} for the Tiles highlighted in red [16]. Note, no two layers are mapped into a single Tile.

Precision-Reconfigurability: HaLoSim caters to reconfigurable precisions of W_ls and E_{l+1}s deployed on the crossbars that helps improve the MAC compute density. Say, we have 6 bit weights that would require total 12 PEs to map the weights along different strides of the input feature map (see Fig. 3b(left)). With reduced 2 bit weights, we can accommodate more weight mapping and increase the parallelism of MAC processing across the entire input feature map (Fig. 3b(right)).

TABLE II: Results Summary on CIFAR10, EMNIST, and TinyImagenet using a 20-client System. We present the final model accuracy and average energy consumption, latency, and area across all clients. System performance metrics reflect client constraints in energy (HaLo-FL-E), latency (HaLo-FL-L), and area (HaLo-FL-A). Note: Top1 accuracy is reported for CIFAR10 and EMNIST, while Top5 accuracy is reported for TinyImagenet.

Dataset	Precision	Acc (%)	E (mJ)	L (s)	A (mm^2)
CIFAR10	8-8-8	80.20	1.83	1.01	1070.93
	HaLo-FL	80.14	1.45	0.93	976.44
	HaLo-FL-E	80.14	1.16	1.01	1024.18
	HaLo-FL-L	80.14	1.42	0.78	1070.93
	HaLo-FL-A	80.14	1.76	1.01	834.21
	2-4-4	78.37	0.45	0.51	313.56
EMNIST	8-8-8	86.69	1.26	0.87	935.86
	HaLo-FL	86.55	0.99	0.81	853.64
	HaLo-FL-E	86.55	0.80	0.87	893.16
	HaLo-FL-L	86.55	0.98	0.67	935.86
	HaLo-FL-A	86.55	1.21	0.87	731.90
	2-4-4	83.91	0.31	0.44	279.79
TinyImagnet	8-8-8	54.67	26.07	1.97	2204.57
	HaLo-FL	54.81	20.62	1.82	1995.75
	HaLo-FL-E	54.81	16.47	1.97	2047.79
	HaLo-FL-L	54.81	20.18	1.53	2204.57
	HaLo-FL-A	54.81	25.22	1.97	1734.90
	2-4-4	47.45	6.48	0.99	641.59

IV. Experiments

We use the popular image classification benchmarks CIFAR10 [17], EMNIST [18], and TinyImagenet [19] to evaluate our method. Our setup employs 20 clients with independent and identically distributed (IID) data partitioning across the clients. The VGG7 model is utilized, and the experiment is conducted over 100 rounds. The learning rate is 8.0 for the first 60 rounds, 1.0 for the next 20 rounds, and 1/8 for the last 20 rounds.

In Table II, we present the final accuracy, average energy, latency, and utilized area on the three datasets. In practice, since the total area remains constant, lower area utilization translates to higher compute density in $TOPS/mm^2$. We compare to a baseline with standard 8-bit precision (8-8-8) for weights, activations, and errors. Additionally, we present results for a heuristically determined lowest precision configuration, 2-4-4, with 2 bits for weights ($b_w = 2$) and 4 bits for activations and errors ($b_a = 4$ and $b_e = 4$). Further precision reduction risks training divergence, causing accuracy to drop to random guessing levels. Our approach matches the accuracy of an 8-bit model while achieving energy efficiency, reduced latency, and minimal area. We tailor the reduction in each metric to meet the specific constraints of individual clients, classified into HaLo-FL-E (energy-constrained), HaLo-FL-L (latency-constrained), and HaLo-FL-A (area-constrained). Among 20 clients, the first 7 prioritize energy, the next 7 prioritize latency, and the remaining 6 prioritize area. We provide average energy, latency, and area metrics for each client type, demonstrating how precision choices align with their constraints. For example, on the CIFAR10 dataset, an energy-constrained client (HaLo-FL-E) averages 1.16 mJ energy, 1.01s latency, and 1024.18 mm^2 area utilization. On the other hand, an area-constrained client (HaLo-FL-A) has the smallest area (834.21 mm^2) at the expense of higher energy and latency.

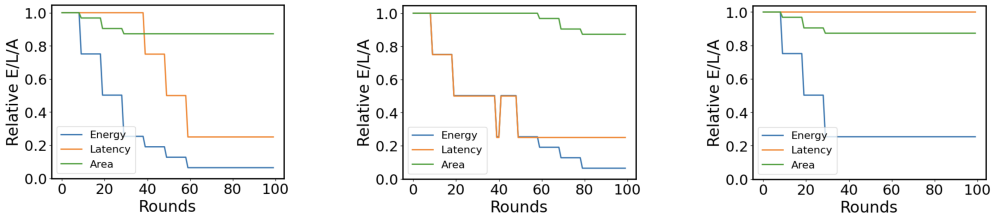
Precision Selector: This section delves into how the precision selector prioritizes the specified hardware metric across rounds. Fig. 4 depicts the normalized energy, latency, and area relative to an 8-bit model across 100 rounds of training. For energy-constrained clients, our method prioritizes aggressive energy reduction, followed by area and latency optimization (Fig. 4a). In contrast, latency-constrained clients focus on minimizing latency by reducing activation precision, simultaneously reducing energy consumption (Fig. 4b). Area-constrained clients prioritize minimizing IMC area utilization, leading to a decrease in both area and energy consumption as a secondary effect (Fig. 4c).

Further, depicted in Fig. 5, we present a precision tree that outlines the available precision choices at different stages. We track distinct client paths on the precision tree, observing varying precision selections. Clients with energy constraints initially reduce both activation and error dimensions, but later find that lowering error precision significantly reduces energy consumption (Fig. 5a), mainly due to MAC operations. Clients constrained by latency prioritize reducing precision in the activation dimension (Fig. 5b). Activations serve as inputs to the crossbar in both forward and backward propagation, creating a latency bottleneck. In contrast, as weights are programmed on the crossbar, they significantly impact the area. Therefore, clients constrained by area choose an initial precision reduction in the weight dimension (Fig. 5c), as this strategy maximally reduces the area.

Hardware Evaluation Breakdown: In Fig. 6, we examine hardware metrics for individual components within the network, utilizing the 8-bit VGG7 model on CIFAR10. Specifically, Fig. 6a illustrates read and write energy in forward and backward computations across each layer of a VGG7 network for a single data sample. The backward pass consumes nearly two orders of magnitude more energy than the forward pass. Additionally, from Eq. 2, 3, and 4, reducing the precision of activations or errors lowers overall dynamic MAC computation and processing energy in HaLoSim, as weights can be reused for MAC operations. This clarifies why energy-constrained clients prioritize reducing the precision of activations and errors over weights (Fig. 5a).

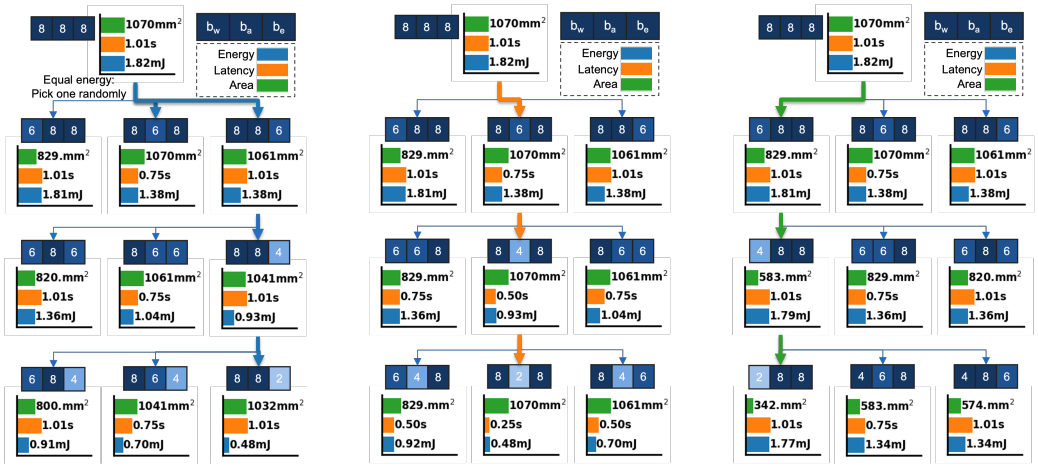
Fig. 6b presents the latency resulting from forward and backward computations. Similar to the energy analysis, the backward pass outweighs the forward pass by more than two orders of magnitude. Lastly, Fig. 6c illustrates the crossbar area required to accommodate the computation of each layer with Area (W) and Area (E) denoting the area required for programming the weights and errors, respectively. Notably, as we progress deeper into the network, the area requirement escalates due to the increased number of channels resulting in a large number of weights. This supports the fact that the precision of weight has a larger impact on area over the precision of error (Fig. 5c).

Robustness to Noise: The non-idealities in crossbar-based architectures can significantly impact their performance [16], [20]–[23]. We investigate the robustness of HaLo-FL to hardware non-idealities and input noise in crossbar-based architectures. We program weights and errors on the crossbars and pass activations as input current. We simulate the hardware noise by adding Gaussian noise proportional to 5% and 10%



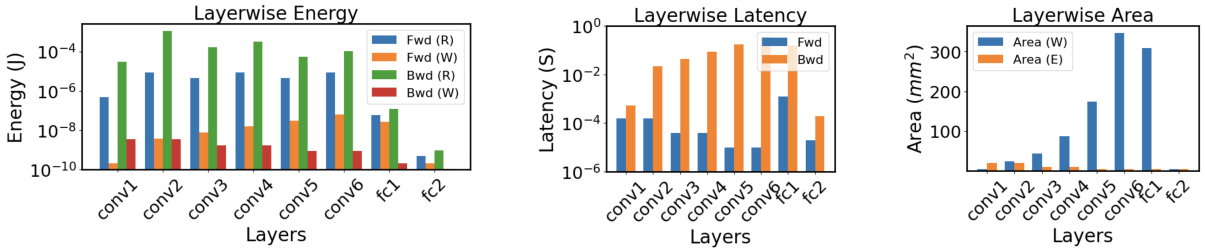
(a) Energy Constrained Client. (b) Latency Constrained Client. (c) Area Constrained Client.

Fig. 4: Energy, latency, and area trend across rounds for clients with different constraints. Observe how the clients constrained in each dimension adjusts the precision resulting in different trends in energy, latency, and area. Reduction in energy follows along all the dimensions. All the values are normalized to 8-bit precision across all dimensions.



(a) Energy Constrained Client. (b) Latency Constrained Client. (c) Area Constrained Client.

Fig. 5: Path taken by clients with different constraints on precision tree. Note that latency constrained client obtains maximum gains by reducing the activation dimension b_a. Reducing the weight dimension has a significant effect in reducing the area. Whereas energy reduction is mainly obtained by reducing the precision of errors b_e.



(a) Layerwise read and write energy for both forward and backward passes.

(b) Layerwise latency introduced in the computation of forward and backward passes.

(c) IMC area utilized for the computation of each of the layers.

Fig. 6: Energy, Latency and Area breakdown per layer in VGG7. Note, Energy and latency are shown in log scale.

TABLE III: Robustness to hardware and input noise.

Noise	Accuracy (%)
Baseline (HaLo-FL)	80.14
Hardware Noise (5%)	79.98
Hardware Noise (10%)	79.39
Input Noise (5%)	79.73
Input Noise (10%)	79.59

of the programmed weight values in the crossbars [20], [24]. The experimental results are presented in Table III. Similarly, we add noise to the input images and evaluate the robustness of our system. We observe that our method does not incur a significant drop in accuracy ($< 1\%$) across varying levels of noise, showcasing its robustness to both hardware non-idealities and input noise.

V. Related Work

Though low-precision research focuses on inference [9], [25], there is a trend toward efficient training by employing low-precision weights and activations. Previous works [12]–[14] propose different methods to incorporate quantization into the training process. These methods are trained on a single device and do not consider the challenges posed by training in a federated environment especially the heterogeneity in the capabilities of the devices. Moreover, these methods assume constant precision throughout the training process.

Prior research in quantization for federated learning has predominantly emphasized communication efficiency [26]–[34]. In Table IV, we provide a summary of the performance of previous works on CIFAR10, wherever applicable, comparing the accuracy and communication efficiency. The average

TABLE IV: Communication cost comparison to related work. We provide the accuracy and the communication cost per round for training on CIFAR10 wherever it is reported. Rest are marked as NA. Additionally, we show if the method uses quantized training (QT) and heterogenous quantization (HQ).

Method	Acc (%)	Comm. Cost	QT	HQ
QA-FL [36]	74.90%	NA	✗	✗
UVeQFed [28]	76%	NA	✗	✗
TernGrad [29]	82.80%	6 MB	✗	✗
LFL [30]	85%	NA	✗	✗
FedHQ [31]	86%	NA	✗	✓
AdaQuantFL [32]	69.12%	NA	✗	✓
LAQ [34]	87.96%	0.65 MB	✗	✗
JoPEQ [37]	71%	NA	✗	✗
HSQ [35]	93.77%	0.67 MB	✗	✗
FL-PQSU [38]	90.9%	3.17 MB	✗	✗
HaLo-FL (Ours)	80.14%	1.52 MB	✓	✓

communication cost per round is calculated by multiplying the model size with the precision of weights.

Previous works such as LAQ [34] and HSQ [35] reduce the communication overhead in federated learning, achieving superior accuracy and communication efficiency compared to our approach. In our work, the utilization of varying precision for individual parameters enables more effective optimization of the training process, particularly when handling a range of heterogeneous devices. This advancement is particularly relevant and beneficial for emerging domains such as IoT devices, on-device learning, and privacy-preserving distributed learning, where resource-constrained devices are prevalent.

VI. Conclusion

We introduced a Hardware-Aware Low-Precision Federated Learning (HaLo-FL) framework, to optimize the training process based on the specific hardware capabilities of individual clients in a heterogeneous environment. To achieve this, we developed HaLoSim, a hardware evaluation platform supporting precision reconfigurability, to obtain crucial hardware metrics such as energy consumption, latency, and area utilization. By determining appropriate precision levels for different components of the training process with feedback from HaLoSim, a hardware-in-the-loop co-optimization approach was adopted, maximizing both accuracy and efficiency.

Acknowledgement

This work was supported in part by CoCoSys, a JUMP2.0 center sponsored by DARPA and SRC, the National Science Foundation (CAREER Award, Grant #2312366, Grant #2318152), TII (Abu Dhabi), and the DoE MMICC center SEA-CROGS (Award #DE-SC0023198).

References

[1] B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*. PMLR, 2017.
[2] J. Xu *et al.*, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, 2021.
[3] N. Rieke *et al.*, "The future of digital health with federated learning," *NPJ digital medicine*, 2020.
[4] D. C. Nguyen *et al.*, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2021.
[5] L. U. Khan *et al.*, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys & Tutorials*, 2021.
[6] Z. Du *et al.*, "Federated learning for vehicular internet of things: Recent advances and open issues," *IEEE Open Journal of the Computer Society*, 2020.
[7] H. Zhang *et al.*, "End-to-end federated learning for autonomous driving vehicles," in *IJCNN*, 2021.
[8] Y. Venkatesha *et al.*, "Divide-and-conquer the nas puzzle in resource-constrained federated learning systems," *Neural Networks*, 2023.
[9] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv:1510.00149*, 2015.
[10] Y. Gong *et al.*, "Compressing deep convolutional networks using vector quantization," *arXiv:1412.6115*, 2014.
[11] D. Blalock *et al.*, "What is the state of neural network pruning?" *MLSys*, 2020.
[12] Q. Zhou *et al.*, "Octo: Int8 training with loss-aware compensation and backward quantization for tiny on-device learning." in *USENIX Annual Technical Conference*, 2021.
[13] S. Wu *et al.*, "Training and inference with integers in deep neural networks," *arXiv:1802.04680*, 2018.
[14] J. Lin *et al.*, "On-device training under 256kb memory," *arXiv:2206.15472*, 2022.
[15] A. Bhattacharjee *et al.*, "Mime: adapting a single neural network for multi-task inference with memory-efficient dynamic pruning," in *DAC*, 2022.
[16] X. Peng *et al.*, "Dnn+ neurosim v2. 0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE TCAD*, 2020.
[17] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," 2009.
[18] G. Cohen *et al.*, "Emnist: Extending mnist to handwritten letters," in *IJCNN*, 2017.
[19] Y. Le *et al.*, "Tinyimagenet visual recognition challenge," *CS231N*, 2015.
[20] X. Sun *et al.*, "Impact of non-ideal characteristics of resistive synaptic devices on implementing convolutional neural networks," *IEEE JETCAS*, 2019.
[21] A. Bhattacharjee *et al.*, "Neat: Nonlinearity aware training for accurate, energy-efficient, and robust implementation of neural networks on 1t-1r crossbars," *IEEE TCAD*, 2021.
[22] Bhattacharjee *et al.*, "Examining the role and limits of batchnorm optimization to mitigate diverse hardware-noise in in-memory computing," in *GLS-VLSI*, 2023.
[23] A. Moitra *et al.*, "Spikesim: An end-to-end compute-in-memory hardware evaluation tool for benchmarking spiking neural networks," *IEEE TCAD*, 2023.
[24] A. Agrawal *et al.*, "Cash-ram: Enabling in-memory computations for edge inference using charge accumulation and sharing in standard 8t-sram arrays," *IEEE JETCAS*, 2020.
[25] A. Zhou *et al.*, "Incremental network quantization: Towards lossless cnns with low-precision weights," *arXiv:1702.03044*, 2017.
[26] J. Sun *et al.*, "Communication-efficient distributed learning via lazily aggregated quantized gradients," *NeurIPS*, 2019.
[27] A. Reisizadeh *et al.*, "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization," in *AISTATS*. PMLR, 2020.
[28] N. Shlezinger *et al.*, "Uveqfed: Universal vector quantization for federated learning," *IEEE TSP*, 2020.
[29] W. Wen *et al.*, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *NeurIPS*, 2017.
[30] M. M. Amiri *et al.*, "Federated learning with quantized global model updates," *arXiv:2006.10672*, 2020.
[31] S. Chen *et al.*, "Dynamic aggregation for heterogeneous quantization in federated learning," *IEEE Transactions on Wireless Communications*, 2021.
[32] D. Jhunjhunwala *et al.*, "Adaptive quantization of model updates for communication-efficient federated learning," in *ICASSP*, 2021.
[33] Y. Mao *et al.*, "Communication-efficient federated learning with adaptive quantization," *ACM TIST*, 2022.
[34] J. Sun *et al.*, "Lazily aggregated quantized gradient innovation for communication-efficient federated learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.
[35] X. Dai *et al.*, "Hyper-sphere quantization: Communication-efficient sgd for federated learning," *arXiv:1911.04655*, 2019.
[36] K. Gupta *et al.*, "Quantization robust federated learning for efficient inference on heterogeneous devices," *arXiv:2206.10844*, 2022.
[37] N. Lang *et al.*, "Joint privacy enhancement and quantization in federated learning," *IEEE TSP*, 2023.
[38] W. Xu *et al.*, "Accelerating federated learning for iot in big data analytics with pruning, quantization and selective updating," *IEEE Access*, 2021.