

DyBit: Dynamic Bit-Precision Numbers for Efficient Quantized Neural Network Inference

Jiajun Zhou^{1*}, Jiajun Wu^{1*}, Yizhao Gao¹, Yuhao Ding¹, Chaofan Tao¹, Boyu Li¹

Fengbin Tu², Kwang-Ting Cheng², Hayden Kwok-Hay So¹, Ngai Wong^{1†}

¹ Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong

² Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong

{jjzhou, jjwu, yzga, yhding}@eee.hku.hk, {cftao, liboyu}@connect.hku.hk

tufengbin@gmail.com, timcheng@ust.hk, {hso, nwong}@eee.hku.hk

arXiv:2302.12510v1 [cs.LG] 24 Feb 2023

Abstract—To accelerate the inference of deep neural networks (DNNs), quantization with low-bitwidth numbers is actively researched. A prominent challenge is to quantize the DNN models into low-bitwidth numbers without significant accuracy degradation, especially at very low bitwidths (< 8 bits). This work targets an adaptive data representation with variable-length encoding called DyBit. DyBit can dynamically adjust the precision and range of separate bit-field to be adapted to the DNN weights/activations distribution. We also propose a hardware-aware quantization framework with a mixed-precision accelerator to trade-off the inference accuracy and speedup. Experimental results demonstrate that the inference accuracy via DyBit is 1.997% higher than the state-of-the-art at 4-bit quantization, and the proposed framework can achieve up to $8.1\times$ speedup compared with the original model.

Index Terms—Deep Neural Networks, Quantization, Accelerator

I. INTRODUCTION

There is an ever-growing need of accelerating deep neural network (DNN) inference. While the de facto industrial standard is to represent network weights as single-precision (32-bit) floating-point (FP) numbers in pre-trained DNN models, inference hardware commonly relies on reduced bitwidth fixed point arithmetic circuits (e.g., `int8`) instead for their superior speed, area, and energy efficiency over their floating-point counterparts. To operate with these fixed point hardware, the original FP models must first be quantized into the target low-precision linear fixed-point representations offline based on the training data [1]. Although the use of low-bitwidth hardware can significantly speed up DNN inference, this approach suffers from significant accuracy degradation, especially on very low bitwidth settings (< 8 bits), because it is challenging for the *fixed* and *linear* range of the conventional fixed-point format to capture the complex *dynamic* parameter distribution changes in a DNN model during run time. A number of recent works attempted to address this challenge by introducing mixed-precision quantization that employs fixed-point numbers of different bitwidth in different parts of the neural network [2]–[4]. Unfortunately, obtaining the optimal configuration for mixed-precision quantization that minimizes accu-

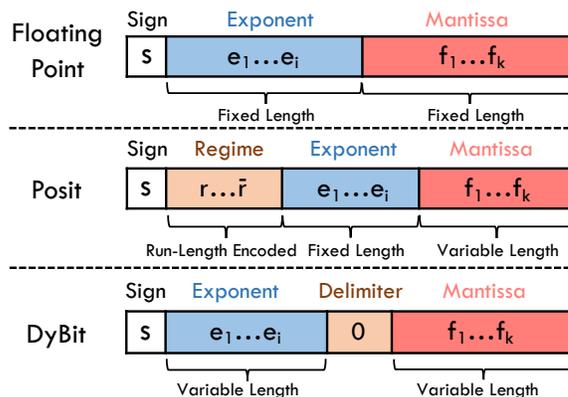


Fig. 1: An illustration of different N -bit numerical arithmetic formats including FP, Posits and DyBit numbers.

racy loss remains an unsolved problem, making it difficult to justify the hardware and speed overhead of supporting mixed-precision operations in hardware [2]. To address the low-bitwidth quantization challenge caused by the linear mapping in conventional fixed-point representations, recent works have begun to investigate in tailored made number presentations for neural network inference that reduce the representation error of low-bitwidth quantization [5]–[7]. Instead of affine mapping, they leverage additional mechanisms to adjust precision in different ranges. For instance, Posit [5] uses run-length encoding to dynamically define the exponent and mantissa ranges in each data, while Adaptivfloat [6] assigns different exponent lengths to different data blocks as an adjustment for precision ranges. These approaches are often designed in a way that can better represent the network based on their distribution properties due to the dynamic precision range with lower bitwidths than the standard FP format. However, existing adaptive data types require additional variables for adjusting the dynamic range. In this regard, a hardware-efficient data format that can dynamically represent the tensor distribution without extra variables is of research and practical value.

To this end, we propose a hardware-efficient data representation called *DyBit* for low-bitwidth quantization with a variable length in the exponent bit-field to adapt to the distribution of DNN models. Furthermore, an efficient mixed-

*Both authors contributed equally to this research

†Corresponding author

precision quantization framework is developed to tradeoff between quantization error and latency speedup. Thanks to the dynamically adaptive representation, the framework can quantize activations and weights to the lowest 4 bits and 2 bits, respectively, while maintaining high accuracy. The proposed framework can also be adapted to different application requirements using different constraints on quantization error or speedup. Finally, we design and implement a run-time configurable mixed-precision accelerator that can efficiently decode the DyBit and reuse computation units for different bitwidths. The key contributions of this work are:

- We propose *DyBit*, an adaptive data representation that has efficient variable-length exponent bits and can also adjust its precision at the tensor level. Evaluation results show the proposed representation can be adapted to the data distributions in various DNN models and layers.
- We have developed a run-time configurable mixed-precision accelerator that supports *DyBit* operations, which fuses multiple multiply-accumulate (MAC) operations into one processing element to speed up the DNN inference and reduce memory access in low-bitwidth quantization.
- We propose a hardware-aware mixed-precision quantization framework based on the adaptive *DyBit* to trade-off between the inference accuracy and hardware speedup. The proposed framework searches for optimal layer-wise quantization based on two strategies for different application scenarios.

II. BACKGROUND AND RELATED WORK

A. Quantization Method

Many studies have extensively explored DNN compression and optimization on hardware using quantization. For efficient edge deployment, binary neural networks (BNNs) exclusively make use of the logical XNOR operation that obviates regular multipliers binarized the network weights into $\{-1,+1\}$ [8] and replace multiplication with addition or bit-shift operations. Jacob [1] made use of fixed-length integers to quantize weights and activations. Many approaches only quantize static weights with on-device storage considerations but do not deliver verifiable computational efficiency improvements on real hardware. The survey paper by Qualcomm AI research [9] contains more details about hardware-motivated methods for quantization. Nonetheless, these conventional quantization methods simply assign separate quantizers per group of weights and activations, whereas the proposed framework herein automates fused multiple bits for efficient calculations.

B. Mixed-Precision Hardware Accelerator

To efficiently support mixed-precision quantization, previous works have explored different architectural designs that can achieve scalable performances on different precisions. Prior mixed-precision accelerators can mainly be divided into *spatial-based* and *temporal-based* architectures depending on how the precision-scaling operations are mapped [10]. The spatial-based accelerators, e.g., BitFusion [11], are generally

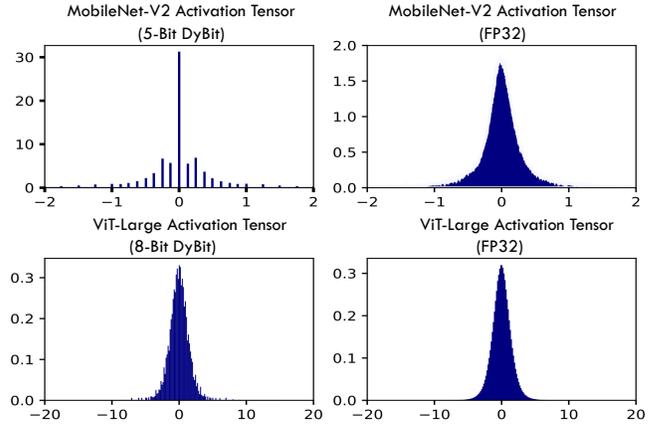


Fig. 2: The diagram of the proposed DyBit quantization.

based on a configurable multiplier composed of low-bitwidth multiply units. By splitting the full-precision input into several low-precision data, the multiplier can process multiplications with different bitwidths. On the other hand, temporal-based accelerators usually leverage bit-serial MAC operations to achieve efficient computation [12]. In general, it is challenging to achieve good tradeoffs between bit-level compatibility, energy efficiency, and performance without causing significant overhead to support mixed-precision operations. In this work, an accelerator based on a spatial-based architecture is proposed to efficiently support DyBit format under low precision (<8 bits). A cycle-accurate simulator is also developed to foster hardware-aware mixed-precision quantization.

III. METHODOLOGY

We now present more details of the proposed DyBit representation and the efficient hardware accelerator, as well as the mixed-precision framework and the quantization algorithm.

TABLE I: 4-Bit Unsigned DyBit Value Table

| Binary | Value | Binary | Value | Binary | Value | Binary | Value |
|---------|-------|---------|-------|---------|-------|---------|-------|
| 0 0 0 0 | 0 | 0 1 0 0 | 0.5 | 1 0 0 0 | 1.0 | 1 1 0 0 | 2 |
| 0 0 0 1 | 0.125 | 0 1 0 1 | 0.625 | 1 0 0 1 | 1.25 | 1 1 0 1 | 3 |
| 0 0 1 0 | 0.25 | 0 1 1 0 | 0.75 | 1 0 1 0 | 1.5 | 1 1 1 0 | 4 |
| 0 0 1 1 | 0.375 | 0 1 1 1 | 0.875 | 1 0 1 1 | 1.75 | 1 1 1 1 | 8 |

A. Variable-Length Datatype

The variable-length DyBit number representation scheme contains a mandatory sign, multiple dynamical exponent bits, and mantissa bits. We follow the generic representation to illustrate any DyBit value in Fig. 1. To efficiently decode all bitwidth data points, only shift or add operations are required to compute the bit-level number system. Specifically, the tapered DyBit representation is defined in Eqn. (1).

$$f(\mathbf{x}) = \begin{cases} 0, & 0 \\ 2^n, & max \\ (-1)^s \times 2^{i-1} \times \left(1 + \frac{x}{2^k}\right), & others \end{cases} \quad (1)$$

where n refers to the total number of bits, i stands for the variable length of exponent bits, k is the variable length of the

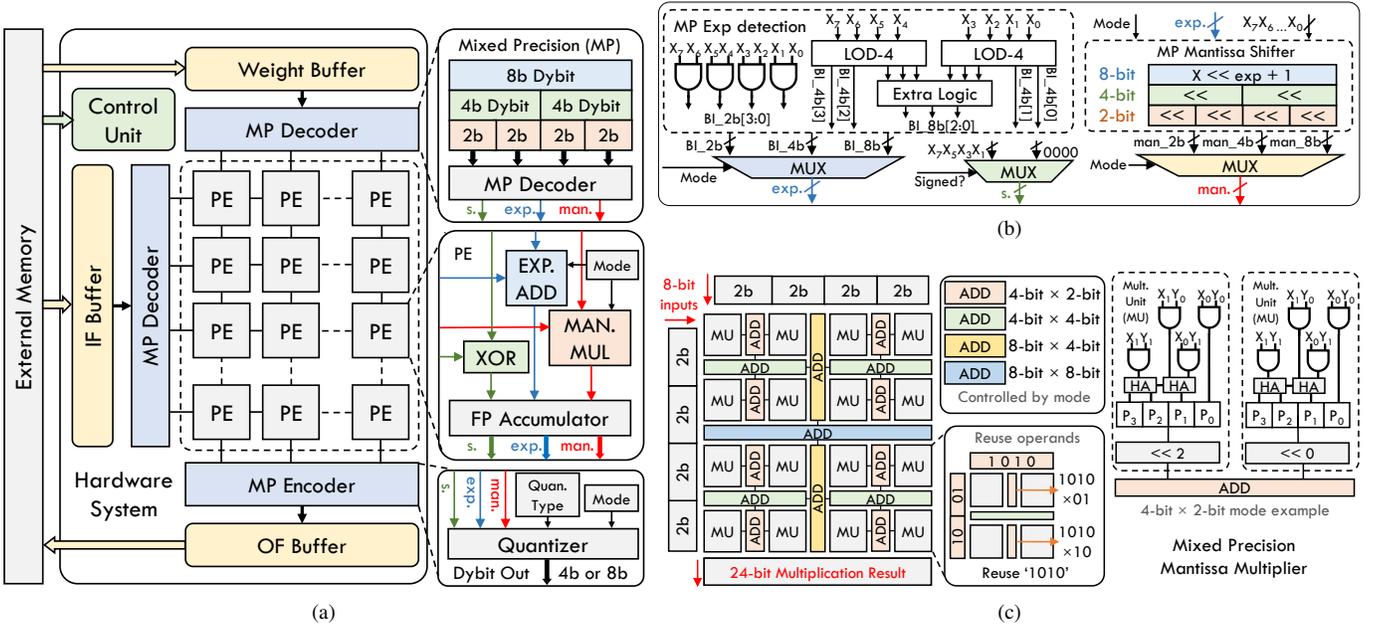


Fig. 3: Mixed-precision hardware system based on the proposed DyBit representation. (a) Hardware architecture based on the systolic array, (b) mixed-precision decoder (MP Decoder), and (c) mixed-precision mantissa multiplier (MAN. MUL).

power-of-2 scaling mantissa bits of encoded variable-length range bits, and x represents the decimal number of the fraction field. In the formula, when the start bit is the digit 1, the variable-length exponent bit i is used to encode the number of 1s and combines the hardware-oriented characteristics of leading one detector (LOD), which counts the number of 1s before the next zero bit. If the start bit is the digit 0, only variable-length fraction bit k represents the actual value within $\{-1,1\}$. In this way, the exponent region of DyBit is a variable-length encoding method instead of a fixed-length one. Meanwhile, the fraction bits are also adaptively changed due to the shifting of the exponent bit. We further explain this encoding results of non-uniform distributions with a 4-bit truth Table I that maps small and large values to tensor distributions. Thanks to the variable-length method, DyBit is suitable for DNN quantization as it can be adapted to tensor distributions of the original models (cf. Fig. 2).

B. Hardware Design

To support the DyBit-based quantization and inference, we propose a run-time configurable mixed-precision hardware accelerator. This section introduces how the architecture and circuit design efficiently support the configurable mixed-precision requirement.

1) *Architecture*: The proposed hardware architecture is based on a systolic array with an input feature (IF) buffer, a weight buffer, and an output feature (OF) buffer, shown in Fig. 3a. Based on the systolic dataflow, all partial results can remain FP for MAC operations. Thereby, all processing elements (PE) share the same decoder per row/column and the same encoder per column so that the decoders and encoders do not exist in PEs, which reduces the hardware overhead.

The FP intermediate results will be quantized to DyBit format before being written back to the external memory.

2) *Decoder & Encoder*: Due to the mixed-precision support, decoding the input data into unified floating-point formats will be easier for processing. As in Fig. 3b, the proposed mixed-precision decoder extracts the exponent (exp) by detecting the number of the leading 1s. Then the decoder left-shifts exp to get the mantissa and inserts the normalized 1 in the MSB. Take an unsigned 8-bit DyBit data 11001010 as an example, the decoded data will be exponent(001), mantissa(10101000). To reduce the mixed-precision overhead, we reuse the 4-bit leading one detector (LOD-4) for 8-bit DyBit input, and we also reuse the logic in the dynamic shifter for the mantissa. For the encoder part, the process is the opposite of the decoding part, in which the circuit will insert ($exp + 1$) number of 1s in the MSB and select the remaining bits of mantissa to fill the DyBit output.

3) *Mixed-precision PE*: As illustrated before, the data processed inside PEs fit well with variable-length separate bit-field. Implementing individual exponent adders and mantissa multipliers for different data widths will cause huge overhead as no computation resources are reused. For the mantissa multiplier (MAN. MUL), we modified the BitFusion [11] architecture to support four different multiplication modes. It is worth noting that based on this fused strategy, the PE can process multiple multiplications in parallel with data reuse (cf. Fig. 3c). For the exponent adder (EXP. ADD), it is natural and trivial to reuse the low-precision adder to build up a high-precision adder with a small amount of overhead in the carry chain. The run-time instructions can control the PE working on different modes. With such mixed-precision PEs, when an $N \times N$ systolic array is working on $P_1 \times P_2$ (< 8 -bit) mode, it

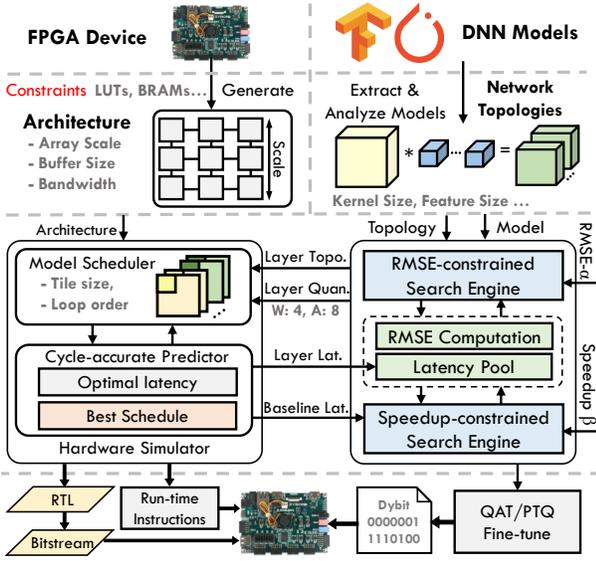


Fig. 4: DyBit-Based hardware-aware quantization framework.

is equivalent to achieving $(8/P_1)N \times (8/P_2)N$ scale based on this fixed systolic array. Therefore, our hardware design can achieve high speedup in low-precision modes.

C. Hardware-aware Quantization Framework

Fig. 4 presents the proposed novel hardware-aware quantization framework based on the search-based method. The framework first estimates the maximum hardware resource utilization based on the DNN models and given hardware constraints (e.g., LUTs and BRAMs in FPGAs). Then, it searches the layer-wise quantization bitwidths based on two variant-constrained strategies. The hardware-aware framework uses a cycle-accurate hardware simulator to provide latency results to do layer-wise mixed-precision quantization dynamically. Finally, the pre-trained 32-bit floating-point (FP32) models are quantized into DyBit according to the layer-wise search results using quantization-aware training (QAT) to retain accuracy. The post-quantization DNN models can then be deployed to our hardware accelerator.

1) *Quantization Metrics*: According to previous works, Root Mean Squared Error (RMSE) is a common metric to effectively evaluate the accuracy of the post-quantization DNN models [6]. The smaller the RMSE, the higher accuracy a quantized model can potentially achieve. Here we use RMSE as a metric to measure the quantization error and facilitate the search process, defined as:

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n \left(\frac{x - \hat{x}}{\sigma_i} \right)^2}, \quad (2)$$

x and \hat{x} are respectively the original FP32 and quantized values, and σ_i is the standard deviation of the tensor distribution.

2) *Two Search Strategies*: Based on the quantization error RMSE and speedup ratio, we propose two different optimization strategies to adapt to different application scenarios. In the case of stringent real-time requirements, our framework can constrain the speedup ratio as α to ensure the hardware

performance while minimizing the quantization error RMSE, as shown in Eqn. (3).

$$\begin{aligned} \min_{\mathbf{A}} \quad & \sum_{i=1}^m RMSE_i(a, w) \\ \text{s.t.} \quad & \alpha \times \sum_{i=1}^m Lat_i(a, w) \leq \sum_{i=1}^m Lat_i(8, 8). \end{aligned} \quad (3)$$

On the other hand, if the application prioritizes accuracy, our framework can use another constraint β to limit quantization error while obtaining quantization with minimum latency, as in Eqn. (4). Note that in both search strategies of our algorithm, we select 8-bit Dybit as the baseline for latency and RMSE metrics.

$$\begin{aligned} \min_{\mathbf{A}} \quad & \sum_{i=1}^m Lat_i(a, w) \\ \text{s.t.} \quad & \sum_{i=1}^m RMSE_i(a, w) \leq \beta \times \sum_{i=1}^m RMSE_i(8, 8) \end{aligned} \quad (4)$$

3) *Quantization Search Flow*: The layer-wise mixed-precision quantization of weights and activations leads to a vast design space. In DyBit, we support the selections for weights/activations in 8-bit, 4-bit, and 2-bit for better hardware efficiency since the bitwidths non-integer powers of 2 (e.g., 6-bit) will cause additional overhead for data alignment in off-chip memory and data transfer between accelerator and memory. Assuming the DNN model has N layers, the total number of possible solutions will be $(3 \times 3)^N$. In such a space, we design a heuristic search algorithm to find near-optimal solutions efficiently.

Algorithm 1 describes the proposed heuristic search algorithm. In the speedup-constrained strategy, we get the layer-wise baseline latency performances calculated by the simulator and select the k largest layers as candidates. In other words, we intend to quantize the slowest layer first to get a better overall end-to-end speedup. Besides, to get the optimal solution with the minimum RMSE, we also calculate the RMSE of each candidate and reorder them in ascending order of RMSE. The search engine will lower the bitwidth of each candidate one by one so that the low-RMSE layers can be quantized first. Whenever the speedup ratio within the k candidates is satisfied, this iteration will stop. The engine will recalculate the latency and select the next top- k candidates in the next iteration. The overall process will stop when the end-to-end speedup constraint is satisfied. This way, we can ensure the final speedup ratio while lower the RMSE as well.

As for the RMSE-constrained strategy, the objective and condition are exchanged compared with the speedup-constrained one. Therefore, the search flow is similar to the speedup-constrained strategy, except the ordering of the candidate is based on different metrics.

4) *Hardware Simulator*: To support the hardware-aware quantization, we develop a cycle-accurate simulator, shown in Fig. 4. The simulator first generates the maximum architecture constrained by the resources of the target device. By modifying the backend of the systolic array GEMM dataflow [7] based on our hardware design, it obtains the optimal latency by

Algorithm 1 Search Flow of speedup-constrained and RMSE-constrained strategies

Input: DNN model \mathbf{M} with N layers $\{L_1, L_2, \dots, L_N\}$, search strategy m , constraint α or β , top-k parameter k

Output: Layer-wise bitwidths of weights and activations $\mathbf{W} = (W_1, W_2, \dots, W_N)$, $\mathbf{A} = (A_1, A_2, \dots, A_N)$

```

1:  $\mathbf{W}, \mathbf{A} \leftarrow (8, 8, \dots, 8)$ 
2:  $metric\_base \leftarrow TOTAL\_METRIC(\mathbf{M}, \mathbf{W}, \mathbf{A}), ratio \leftarrow 1$ 
3: while  $ratio$  does not meet  $\alpha$  or  $\beta$  do
4:    $metric \leftarrow LAYERWISE\_METRIC(\mathbf{M}, \mathbf{W}, \mathbf{A})$ 
5:   if  $m = \text{speedup}$  then
6:      $metric\_top \leftarrow LAT\_RANK(metric, k)$ 
7:      $layer\_list \leftarrow RMSE\_RERANK(metric\_top)$ 
8:   else if  $m = \text{RMSE}$  then
9:      $metric\_top \leftarrow RMSE\_RANK(metric, k)$ 
10:     $layer\_list \leftarrow LAT\_RERANK(metric\_top)$ 
11:   end if
12:    $DEGRADE\_LEVEL(list, \mathbf{W})$ 
13:    $DEGRADE\_LEVEL(list, \mathbf{A})$ 
14: end while
15:
16: procedure  $DEGRADE\_LEVEL(layer\_list, \mathbf{W}$  or  $\mathbf{A})$ 
17:   for  $l = 1 \rightarrow k$  do
18:     Degrade  $W_{layer\_list[l]}$  or  $A_{layer\_list[l]}$ :  $8 \rightarrow 4$  or  $4 \rightarrow 2$ 
19:      $ratio \leftarrow TOTAL\_METRIC(\mathbf{M}, \mathbf{W}, \mathbf{A})/metric\_base$ 
20:     break if  $ratio$  meets  $\alpha$  or  $\beta$ 
21:   end for
22: end procedure

```

calculating the latencies corresponding to all possible tiling schedules of the current layer. During the search flow, the search engines call the simulator to get the latency of each layer, which will be used for ranking layer candidates.

IV. EVALUATIONS

A. Experiment Setup

1) *Benchmark:* We conduct the experiments based on ResNet18/50 and the lightweight MobileNetV2 on ImageNet classification. We use the pre-trained 32-bit floating-point (FP32) model from PyTorch as the baseline. Based on the method in Section III-C, we train 3~5 fine-tuning epochs for QAT. To conduct a fair comparison, the training setup, and the hyper-parameters are kept the same for all types under evaluation. We also test our framework on emerging models like Vision Transformer, RegNet, and ConvNext to verify that method is universal and efficient.

2) *Baselines:* We obtained evaluation results for integer quantized models (i.e. INT4, INT8) following the same training setup. Besides, we compare our method with various fixed-precision quantization methods including PACT [13], AdaFloat [6], DSQ [14], Posit [5], Flint [7] and layer-wise mixed-precision quantization methods, such as BRECQ [3].

3) *Implementation:* The proposed mixed-precision accelerator is designed and implemented with Verilog HDL. We implemented the accelerator on the Xilinx ZCU102 platform. As discussed in Section III-C, a cycle-accurate hardware simulator is developed to support hardware-aware quantization search. We also utilize this simulator to evaluate our speedup performance compared with baselines.

TABLE II: Top-1 accuracy performance with quantization-aware training on ImageNet dataset.

| Methods (W/A) | MobileNetV2 | ResNet18 | ResNet50 |
|-------------------|-------------|----------|----------|
| FP32 | 71.79 | 69.68 | 75.98 |
| INT(4/4) | 39.78 | 66.24 | 73.04 |
| INT(8/8) | 71.658 | 69.4 | 75.92 |
| AdaFloat(4/4) [6] | — | — | 75.1 |
| BRECQ(4/4) [3] | 66.57 | 69.60 | — |
| PACT(4/4) [13] | 61.40 | 69.20 | — |
| DSQ(4/4) [14] | 64.80 | 69.56 | — |
| Flint(4/4) [7] | — | 67.50 | 74.91 |
| Posit(8/8) [5] | — | — | 73.61 |
| DyBit(4/4) | 69.31 | 69.47 | 75.87 |
| DyBit(4/8) | 68.17 | 69.57 | 75.82 |
| DyBit(8/8) | 69.47 | 69.66 | 75.93 |

TABLE III: Top-1 accuracy performance with quantization-aware training on ImageNet with emerging models

| Methods (W/A) | RegNet-3.2GF | ConvNext-Tiny | ViT-Base |
|-------------------|--------------|---------------|----------|
| FP32 | 78.364 | 82.52 | 81.07 |
| INT(4/4) | 75.9 | 0.1 | 72.19 |
| Flint(4/4) [7] | - | - | 78.33 |
| DyBit(4/4) | 77.13 | 71.9 | 79.44 |
| DyBit(8/8) | 77.844 | 80.55 | 80.82 |

B. Quantization Accuracy

To validate that the adaptive DyBit data representation can keep the accuracy in the low-precision models, we conducted quantization-aware training. In Table II, we show the Top-1 accuracy results of three models in different bitwidth on the ImageNet dataset, e.g., 4W4A stands for 4-bit activation and 4-bit weight tensor. We observe that our quantization achieves 1.997% inference accuracy higher than the state-of-the-art, viz. Flint [7], on 4-bit quantization and also surpasses other fixed-precision or mixed-precision quantization methods. Besides, it is noteworthy that 4-bit and 8-bit quantization results are provided in Table III to demonstrate that our method for larger models causes less accuracy drop compared to high-precision models after fine-tuning. FP32, INT4, and INT8 results are also provided for a fair comparison. In addition, 8-bit DyBit has only a 0.05 Top-1 accuracy drop compared with FP32 on ResNet50. Specifically, our quantization method performs better at the lower-precision bitwidth.

C. Accuracy-Speedup Trade-off

To demonstrate the proposed hardware-aware quantization framework can trade-off between accuracy and speedup, we set up different constraints and quantize the ResNet18/50 and MobileNetV2 models based on the two search strategies in Section III-C, depicted in Fig. 5. Generally, an increase in the constraint α or β leads to a speedup increase and accuracy loss, because the framework will search for more low-precision numbers to meet the demand. For the speedup-constrained

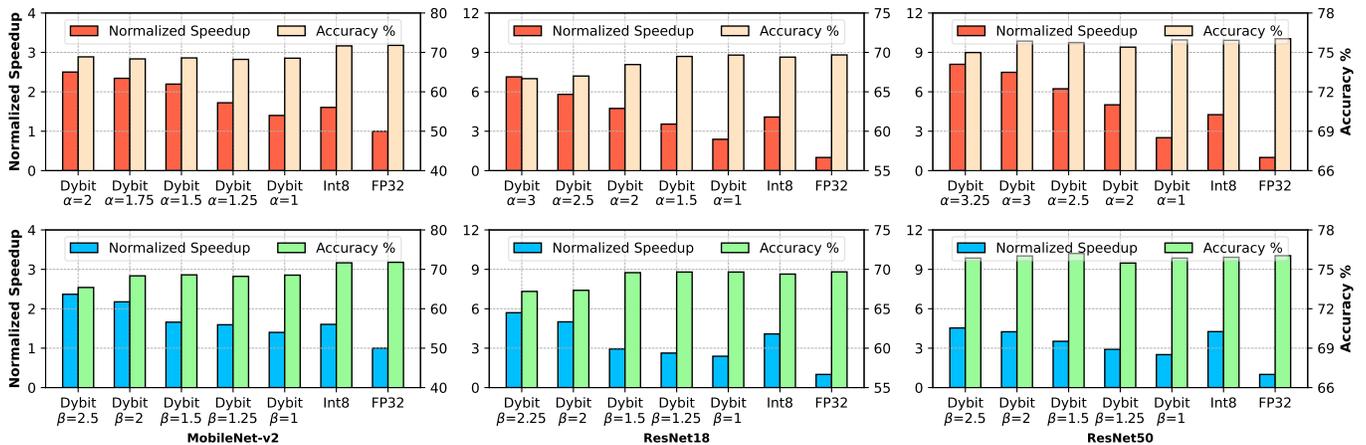


Fig. 5: Speedup and accuracy evaluations on the speedup-constrained strategy (the first row) and the RMSE-constrained strategy (the second row), based on MobileNetV2 and ResNet18/50 models. The target platform is Xilinx ZCU102.

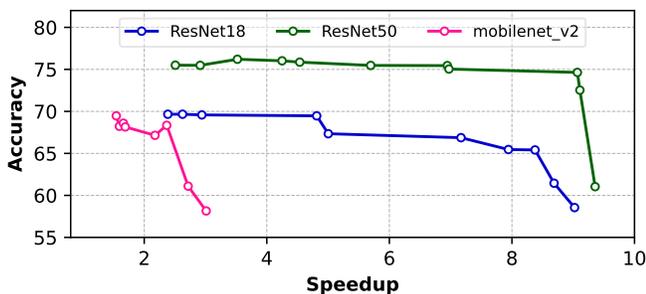


Fig. 6: Accuracy-speedup trade-off in DyBit quantization.

strategy, the quantized model tends to have a higher speedup (e.g., in ResNet50, up to $8.1\times$) with lower accuracy. On the contrary, the quantized model can maintain a closer accuracy to the original model while still delivering a decent speedup in the RMSE-constrained strategy (e.g., in ResNet50, only 0.18% accuracy drop with $4.5\times$ speedup). Therefore, the proposed framework can work for different application scenarios with the two strategies. To further present the adjustment between accuracy and speedup, we collect all results based on both strategies, as shown in Fig. 6. It can be concluded that with the growing speedup, the inference accuracy drops, and our proposed framework can quantize the DNN models with trade-offs along the curves. The speedup ratio is limited in the MobileNetV2 since depth-wise operations are not efficient based on our current GEMM systolic array.

V. CONCLUSION

This paper has proposed a novel hardware-aware quantization framework, with a fused mixed-precision accelerator, to efficiently support a distribution-adaptive data representation named DyBit. The variable-length bit-fields enable DyBit to adapt to the tensor distribution in DNNs. Evaluation results show that DyBit-based quantization at very low bitwidths (<8 bits) consistently achieves higher accuracy than competing methods. Moreover, the proposed end-to-end framework

can effectively search for the optimal solution under various constraints, thus achieving a trade-off between accuracy and hardware speedup. Experiments on various DNN models under different quantization constraints demonstrate that the framework can quantize DNN models to achieve $2.5 \sim 8.1\times$ speedup.

REFERENCES

- [1] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2704–2713.
- [2] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Hq: Hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8612–8620.
- [3] Y. Li, R. Gong, X. Tan, Y. Yang, P. Hu, Q. Zhang, F. Yu, W. Wang, and S. Gu, "Breq: Pushing the limit of post-training quantization by block reconstruction," *arXiv preprint arXiv:2102.05426*, 2021.
- [4] Z. Dong, Y. Gao, Q. Huang, J. Wawrzynek, H. K. So, and K. Keutzer, "Hao: Hardware-aware neural architecture optimization for efficient inference," in *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2021, pp. 50–59.
- [5] H. F. Langroudi, V. Karia, Z. Carmichael, A. Ziyarah, T. Pandit, J. L. Gustafson, and D. Kudithipudi, "Alps: Adaptive quantization of deep neural networks with generalized posits," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3100–3109.
- [6] T. Tambe, E.-Y. Yang, Z. Wan, Y. Deng, V. Janapa Reddi, A. Rush, D. Brooks, and G.-Y. Wei, "Algorithm-hardware co-design of adaptive floating-point encodings for resilient deep learning inference," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6.
- [7] C. Guo, C. Zhang, J. Leng, Z. Liu, F. Yang, Y. Liu, M. Guo, and Y. Zhu, "Ant: Exploiting adaptive numerical data type for low-bit deep neural network quantization," *arXiv preprint arXiv:2208.14286*, 2022.
- [8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.
- [9] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.
- [10] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 697–711, 2019.

- [11] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural network," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 764–775.
- [12] A. Li, H. Mo, W. Zhu, Q. Li, S. Yin, S. Wei, and L. Liu, "Bitcluster: Fine-grained weight quantization for load-balanced bit-serial neural network accelerators," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4747–4757, 2022.
- [13] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "Pact: Parameterized clipping activation for quantized neural networks," *arXiv preprint arXiv:1805.06085*, 2018.
- [14] R. Gong, X. Liu, S. Jiang, T. Li, P. Hu, J. Lin, F. Yu, and J. Yan, "Differentiable soft quantization: Bridging full-precision and low-bit neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4852–4861.