# RL-PTQ: RL-based Mixed Precision Quantization for Hybrid Vision Transformers

Eunji Kwon[†], Minxuan Zhou[‡], Weihong Xu[‡], Tajana Rosing[*‡] and Seokhyeong Kang[*†]
[†]Pohang University of Science and Technology, Pohang, Republic of Korea
[‡]University of California San Diego, CA, United States

## ABSTRACT

Existing quantization approaches incur significant accuracy loss when compressing hybrid convolution and transformer models with low bit-width. This paper presents **RL-PTQ**, a novel post-training quantization (PTQ) framework utilizing reinforcement learning (RL). Our focus is on determining the most effective bit-width and observer for quantization configurations tailored for mixed precision by grouping layers and addressing the challenges of quantization of hybrid transformers. We achieved the highest quantized accuracy for MobileViTs compared to the previous PTQ methods [5–7]. Furthermore, our quantized model on Processing In Memory (PIM) architecture exhibited an energy efficiency enhancement of 10.1× and 22.6× compared to the baseline model, on the state-of-the-art PIM accelerator [15] and GPU, respectively.

## 1 INTRODUCTION

Transformers have achieved remarkable breakthroughs in various fields, but employing them poses challenges due to their memory and computation-intensive properties. Strategies such as transfer learning, pruning/quantization, and the development of transformer variants address these issues [1]. Transformer variants, in particular, modify the original architecture with alternative operators to enhance computational efficiency. As a lightweight and fast version of the Vision Transformer (ViT) [2], MobileViTs [3, 4] are a noteworthy example of transformer variants. They offer superior performance with fewer parameters compared to traditional CNNs.

However, compressing MobileViTs becomes more burdensome due to modifications in their transformer architectures and the incorporation of diverse convolution layers for reducing the number of model parameters (< 6M) as shown in Fig. 1. The quantization of these hybrid transformers, which include both convolution and transformer layers, proves to be more demanding than optimizing a standard transformer architecture [5]. Consequently, existing quantization methods for ViT [6, 7], which quantize uniformly all layers, may not be effective for MobileViT architectures. More specifically, the model incorporates a variety of layer types, each exhibiting distinct redundancy of bit representations. For instance, depth-wise convolutions tend to learn representations with widely fluctuating dynamic ranges and significant intra-layer distributional mismatch [8]. Consequently, depth-wise convolution performs less effectively in quantization compared to regular CNNs.

These insights advocate for the adoption of mixed precision across different layers in models. The cutting-edge DNN hardware accelerators have already embraced a mixed precision approach:

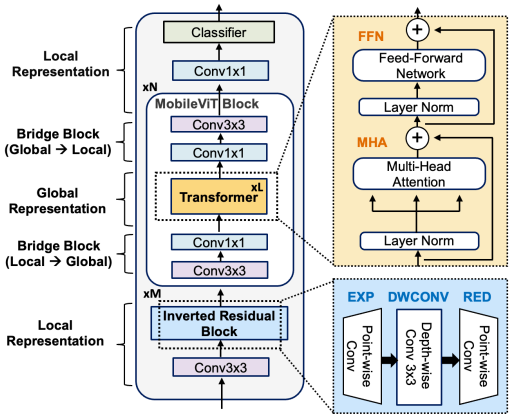


**Figure 1: MobileViTv1 architecture.**

NVIDIA GPUs support various mixed precision operations, such as FP32, FP16, INT8, and Tensor Core functionalities [9]. Furthermore, Processing-in-Memory (PIM), which employs bit-serial processing, presents a viable solution for accelerating mixed precision models.

However, determining the optimal bit-width configurations poses a substantial challenge [10]. First, exploring the expansive design space, which includes considerations such as accuracy, model size, latency, and energy consumption, requires a considerable investment of time and energy. Second, this exploration may not always yield the most optimal outcomes. As a result, there is a need for an accurate and automated quantization framework to explore and identify the optimal quantization scheme.

Motivated by this, we developed a novel post-training quantization (PTQ) framework based on reinforcement learning (RL), which we refer to as **RL-PTQ**, to search for the optimal quantization configuration in MobileViTs. Our framework supports two modes to enhance model accuracy: 1) determining the quantization bit-width and 2) identifying the appropriate quantization observer [11–14]. The quantization observer such as min-max monitors parameters during the PTQ and determines the optimal scaling factors and quantization ranges. The main contributions are as follows:

- **The RL-PTQ framework conducts mixed precision and mixed observer quantization by grouping layers based on the layer type.** We take into account the inter-module relationships in hybrid transformers (Fig. 4), to address the cost-intensive challenges with RL searches. As a result, we can compress the MobileViT parameters by an average of 5.1× with only a 1% drop in the accuracy of the baseline model.
- **We demonstrate RL-PTQ's superior capability in identifying Pareto-optimal combinations of accuracy and model compression.** Our framework achieved the highest accuracy compared to previous works [5–7], even outperforming the state-of-the-art method, Q-hyViT [5], by improving quantized accuracy by an average of 0.4%.

*These corresponding authors equally contributed to this work.


*DAC '24, June 23–27, 2024, San Francisco, CA, USA*

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3656231

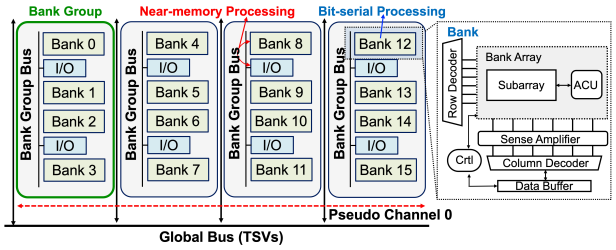


**Figure 2: Hierarchical architecture of a pseudo channel in HBM PIM.**

**Table 1: Qualitative comparison table with previous work, HAQ [10]**

| | HAQ [10] | RL-PTQ |
|---|---|---|
| Target Model | MobileNets | MobileViTs |
| Layer Type | MobileNet Block | MobileNet Block<br>Multi-head Attention<br>Separable Self-Attention |
| Quantization Method | Mixed Precision by Layer | Mixed Precision by Grouping Layers |
| Search Algorithm | DDPG<br>Deterministic Policy | REINFORCE [16]<br>Stochastic Policy |
| Search Objective | Quantization Bit-Width | Quantization Bit-Width<br>Quantization Observer |
| Hardware Acceleration | BISMO [17]<br>Bit-Serial Accelerator | HBM-PIM [18]<br>Architecture (Fig. 2) |

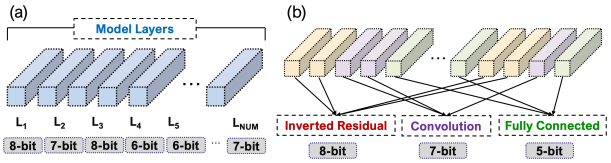


**Figure 3: (a) Mixed precision quantization by layer [10] and (b) Mixed precision quantization by grouping layers based on the layer type.**

- **We utilized the PIM architecture to accelerate compressed models, fully leveraging the advantages of mixed precision models.** We propose a new processing framework on PIM architecture, M-ViT PIM, specifically designed to support hybrid transformers with mixed precision efficiently. This resulted in a 10.1× and 22.6× improvement in average energy efficiency compared to the baseline model on TransPIM [15] and the Titan RTX GPU, respectively, with a 1% accuracy drop.

## 2 PRELIMINARY

### 2.1 Post-Training Quantization of ViT

EasyQuant [6] and PTQ4ViT [7] are prior methods designed for quantizing ViT with uniform bit-width. Their explorations of scaling factors for each layer use Cosine and Hessian-based similarity to minimize quantization error. However, these rule-based methods focus on the PTQ of the vanilla transformer architecture, posing challenges in its application to transformer variants. Q-hyViT [5] introduces the difficulty in the quantization of hybrid transformers such as highly dynamic activation range and zero-point overflow in the bridge block, and developed the first PTQ method based on the Hessian approach tailored for hybrid transformers.

### 2.2 Autonomous PTQ Framework

*2.2.1* ***Qualitative Comparison with HAQ.*** HAQ [10] performed quantization of MobileNets by employing layer-by-layer mixed precision quantization using Deep Deterministic Policy Gradient (DDPG) (Table 1). However, MobileViTs boast diverse layer types compared to MobileNets. Thus, we conducted PTQ with mixed precision by grouping layers utilizing *REINFORCE* [16]. In the action selection method, DDPG uses a deterministic policy, meaning it outputs a fixed action for a given state whereas *REINFORCE* uses a stochastic policy, learning a probability distribution over actions and selecting actions accordingly. The search objective of **RL-PTQ** is to not only determine suitable quantization bit-width but also identify fitting quantization observers for specific module types within the model. Finally, HAQ used the BISMO [17], a bit-serial accelerator for hardware acceleration. We accelerated the quantized model by using the high-bandwidth memory (HBM)-PIM architecture (Fig. 2) [18], which employs bit-serial processing, handling data at the bit level to enhance computational efficiency.

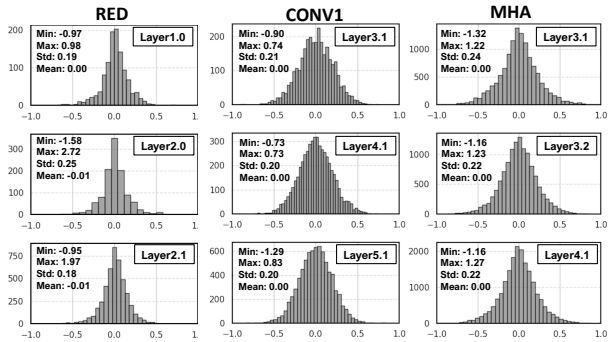


**Figure 4: Weight distributions of the adjacent three layers of each layer type in MobileViT-v1 (point-wise convolution in inverted residual block (RED), point-wise convolution in bridge block (CONV1) and multi-head self-attention (MHA)).**

*2.2.2* ***Limitation of HAQ.*** If a model consists of a total of $L_{num}$ layers and implements a mixed precision search framework for each layer [10] (Fig. 3-(a)), it is necessary to conduct a total of $L_{num}$ time steps within a single epoch. This process incurs a considerable temporal investment to converge to an optimal policy. For example, if mixed precision quantization for weights is conducted at the layer level, each of the 72 layers in MobileViTv1 can be quantized with six specific bit-width options ranging from 4 to 9 bits. This results in a total search space of $6^{72}$. Therefore, applying the previous approach is difficult for models with a large number of layers.

## 3 PROPOSED RL-PTQ FRAMEWORK

MobileViT architectures are hybrid vision transformers so that they have various layer types such as inverted residual blocks for local representation, standard convolution in bridge block, and transformer encoder architecture for global representation (Fig. 1). We develop a pragmatic post-training quantization (PTQ) approach by grouping layers by layer type (Fig. 3-(b)) to address the previously encountered cost-intensive challenges in the RL-based quantization framework. Convolution layers predominantly capture spatial patterns within images, while fully connected layers tend to globally connect the output of the previous layer to understand the overall structure. Accordingly, applying quantization tailored to each layer type can lead to efficiently optimized results.

We propose a straightforward **1) mixed precision** (MP) and **2) mixed observer** (MO) approach leveraging the simple RL algorithm to capture the different redundancies of bit representation in multiple layer types and explore the compression capabilities of a hybrid transformer. The purpose of model optimization is to increase model compression ratio as much as possible without lowering it below the target accuracy. Model compression ratio is defined as the data byte size in full precision divided by the data byte size after quantization.

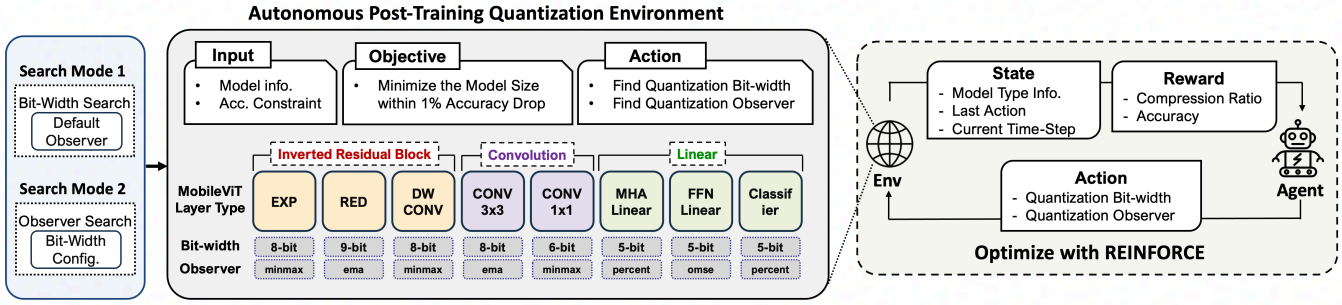


**Figure 5: An autonomous RL-PTQ framework using *REINFORCE* algorithm for MobileViT models, enabling optimal bit-width search as the primary exploration and optimal observer search as the finetuning process to enhance accuracy.**

Our autonomous framework consists of two modes (Fig. 5): 1) finding the proper quantization bit-width for model compression and 2) fine-tuning the accuracy with the proper quantization observer. We search for quantization configurations of each module type of the MobileViTs using the *REINFORCE* algorithm. The main components of *REINFORCE* are the agent, environment, state, action, and reward. The agent samples proper actions using the policy network, while the environment executes PTQ with the searched configurations (actions) to generate a reward and next state [19]. This algorithm directly learns the optimal policy, mapping states to actions, without explicitly estimating the value function.

## 3.1 Quantization by Grouping Layers

Fig. 4 illustrates the samples of weight distributions of the adjacent three layers for several layer types in a MobileViT. It becomes evident that there is a high correlation in weight distributions, such as min-max and standard deviation (std), among adjacent layers of the same module type. Therefore, we have devised an innovative framework by presuming similarity in weight redundancy within bit representations across each layer type. Our method involves partitioning the model into distinct groups and applying quantization to each group to maintain uniform precision.

We defined MobileViT as having eight layer types based on their similarity in weight distributions: 1) EXP, 2) RED, 3) DWCONV, 4) CONV3, 5) CONV1, 6) MHA, 7) FFN, and 8) CLASS. Inverted residual block follows a narrow → wide → narrow approach [20]. The block first expands with a point-wise convolution (EXP) then uses a depth-wise convolution (DWCONV) to greatly reduce the number of parameters and then uses the point-wise convolution (RED) to reduce the number of channels. We separated the layer types (EXP and RED), observing that each layer type has different weight distributions. This suggests that the role of the 1x1 convolution connected at the front and the one connected at the last place is different. The 3x3 convolutions (CONV3) and point-wise convolutions (CONV1) for the bridge block in the MobileViT block were also separated. In addition, layers for global representation including transformer architecture were also divided into layers used for multi-head attention (MHA), layers used for feed-forward natural networks (FFN), and the classifier (CLASS).

## 3.2 Agent and Environment

*REINFORCE* learns a stochastic policy for each possible action given a provided state and repeatedly improves policy parameters using gradient descent to maximize expected cumulative rewards [19]. The algorithm encodes the state vector and feeds it into the neural network as a policy function, which outputs the probabilities of actions within the action space. A recurrent neural network, LSTM was chosen as the policy network. The policy network is updated using the reward obtained by the agent in the current environment during the learning (eq. (1))

$$\theta \leftarrow \theta \ + \ \alpha \nabla \theta \ log\pi(a|s)G \tag{1}$$

In the above equation, $\theta$ is the parameter, $\alpha$ is the learning rate, $\nabla\theta \ log\pi(a|s)$ represents the gradient of policy for parameter $\theta$, and $G$ is the accumulated rewards for the epoch. The agent makes $N$ actions per epoch and gets the multi-step reward. $N$ refers to the number of layer types in the model.

More specifically, to obtain the final model compression ratio and accuracy after PTQ in the environment, determining the bit-width or observer for all layer types is essential. Therefore, multiple time steps are executed sequentially to obtain the multiple actions within a single epoch to define quantization configurations. There are eight time steps in one epoch. During one epoch, PTQ is executed once to obtain the accuracy and compressed weight size in the environment and these results are used to calculate the reward, which, in turn, updates the policy network.

## 3.3 State

$$S_t = (T_t, N_t^w, W_t^{max}, W_t^{min}, W_t^{mean}, W_t^{std}, A_t, t) \tag{2}$$

The state ($S_t$) is represented as an eight-dimensional observation space at $t^{th}$ time step. The state of the **RL-PTQ** contains the model layer type information ($T_t, N_t^w, W_t^{max}, W_t^{min}, W_t^{mean}, W_t^{std}$), the last actions ($A_t$), and the current time step ($t$). $T_t$ is the indicator of the layer type. $N_t^w$ is the number of parameters in a layer type. $W_t^{max}, W_t^{min}, W_t^{mean}$ and $W_t^{std}$ are the maximum, minimum, average, and standard deviation values of weight parameters which represent the weight distributions. $A_t^{bit} and A_t^{obs}$ are the actions of the previous layer type which we feed into the RNN controller. These dimensions constituting the state $S_t$ were normalized into the range of [-1, 1].

## 3.4 Action

At each time step, the agent makes an action ($A_t^{bit} or A_t^{obs}$). $A_t^{bit}$ is the bit-width of quantization and $A_t^{obs}$ is the quantization observer. We used symmetric linear quantization. The following equations are the linear quantization with the m-bit fixed point.

$$Q_{mbit}^{FP} = \pm\alpha \times \{0, \frac{1}{2^{m-1}-1}, \frac{2}{2^{m-1}-1}, ..., 1\} \tag{3}$$

$$\hat{w} = Q_m(w) = \alpha \cdot h^{-1}(\frac{1}{2^m-1} round((2^m-1) \cdot h(\lceil w, \alpha \rfloor)) \tag{4}$$

$$\lceil w, \alpha \rfloor = \begin{cases} -1 & w < -\alpha \\ w/\alpha & -\alpha \le w \le \alpha \\ 1 & w > \alpha \end{cases} \tag{5}$$

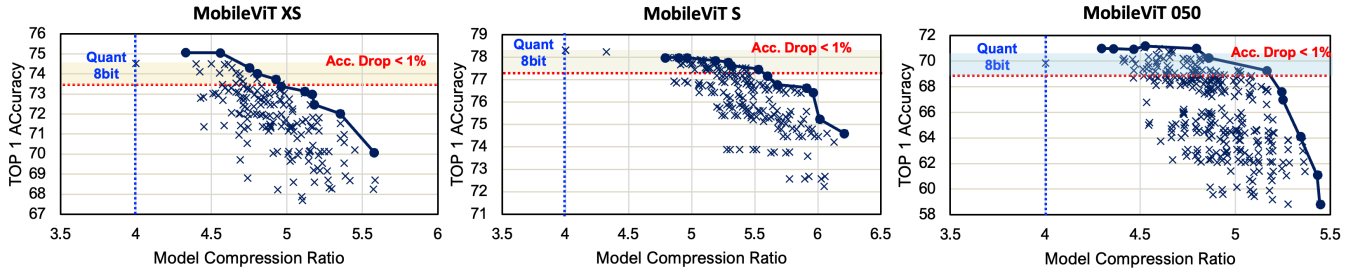


**Figure 6: Accuracy and model compression ratio results of MobileViT models within 6M parameters (v1-XS, v1-S, and v2-050) with the RL-PTQ (Bit) (Input: FP32). Our framework excelled in identifying Pareto-optimal combinations. Circles (•) represent the points on the Pareto frontier.**

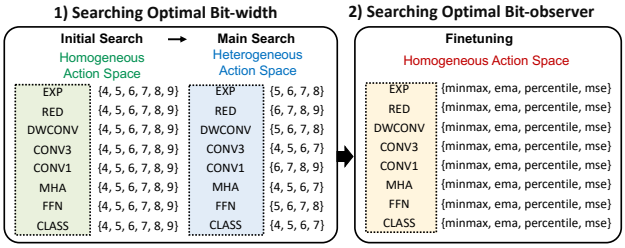


**Figure 7: Homogeneous and heterogeneous action space of RL-PTQ.**

In the above equations, $\lceil w, \alpha \rfloor$ transforms the data range into the interval [-1, 1] (eq. (5)) and $h(\cdot) = tanh(\cdot)/2 + 0.5$ transforms the data range to [0, 1]. $A_t^{bit}$ determines $m$, while $A_t^{obs}$ determines the observation method for determining the quantization range to find the appropriate scaling factor $\alpha$. The first mode of **RL-PTQ (Bit)**, involves finding the proper quantization bit-width using identical quantization observer for all types, while the second mode of **RL-PTQ (Bit+Obs)** entails fine-tuning by searching for the appropriate quantization observer using the searched optimal bit-width configurations.

*3.4.1* ***MODE 1: Bit-width Search with RL-PTQ.*** In Mode 1, two phases exist: the initial search and the main search. Initially, we construct the action space of $A_t^{bit}$ ranging from 4-bit to 9-bit. Following the initial search (as depicted in Fig. 7), we accelerated the overall search process by transitioning from a homogeneous action space to a heterogeneous action space, customized according to layer types. The action spaces defining bit-width for each layer type were condensed from 6 candidates to 4 candidates.

After conducting the initial search, configurations are sorted based on top-1 accuracy, filtering in those with an accuracy drop of 1% or less. Subsequently, from the sorted data, we select the action space candidates that have been involved in the sorted list at least once. Among these, the lowest bit-width is chosen as the reference, and configurations with lower bit-width than this reference are selected as action space candidates for the main search to aggressively compress the models.

*3.4.2* ***MODE 2: Observer Search with RL-PTQ.*** The action space of $A_t^{obs}$ comprises *minmax*, *ema*, *percentile* and *mse*. The observer of quantization is a criterion for determining critical information useful for quantization by observing the distribution of data to be quantized. In other words, as the observer type of quantization changes, the criteria for determining the quantization range change.

- minmax [11]: setting the quantization range by referring to the min-max values of the data. This method determines quantization parameters based on the extreme values of the data distribution. It is used to cover the entire range of weights or activations.
- ema [12]: setting the quantization range by referring to the exponential moving average (EMA) value of the data. It is useful in quantizing data with high volatility.

**Table 2: Architectural parameters for HBM PIM, including HBM organization, auxiliary computing unit (ACU), and data buffer size.**

| | |
|---|---|
| HBM Organization | Channels/die = 8, Banks/channel = 32, Banks/Group = 4, Rows = 32k, Row Size=1 KB, Subarray Size = 512x512, |
| ACU | Clock = 500MHz, Psum = 16 ACUs/bank, Padd = 4 Pipelined Bit-serial Adder Tree/ACU, Adder tree width = 256, 3-stage pipelined divder |
| Buffer | Data buffer: 8 x 256b |

- percentile [13]: setting the quantization range by referring to the min and max values corresponding to the top $a$%. This method is less sensitive to extreme values such as outliers.
- mse [14]: It analyzes the distribution of data and performs quantization to minimize the Mean Squared Error (MSE) before and after the quantization. This method considers not only the central value of the data but also its variability by assessing the overall error distribution.

## 3.5 Quantization Constraint

The **RL-PTQ** employs channel-wise quantization for EXP and RED modules, while applying layer-wise granularity to quantize other modules in order to preserve accuracy. In the channel-wise approach, quantization is applied at the channel level with a consistent scaling factor, whereas in the layer-wise approach, it is performed at the layer level with a uniform scaling factor. It is recommended to employ a per-channel scaling factor for layers with varying ranges per channel to preserve specific layer characteristics [21]. Without this consideration, channels with a narrow range may face issues where all values are treated as zeros during layer-wise quantization.

## 3.6 Mapping models on PIM accelerator

The optimized models from the **RL-PTQ** have mixed precision ranging from 4 to 9 bits. It is difficult to achieve the expected speedup using NVIDIA GPUs; they have limitations in accelerating our models due to their support solely for power-of-2 mixed precision operations such as INT8. Consequently, we accelerate our models using the PIM architecture, which employs bit-serial processing.

The baseline PIM architecture is HBM-PIM [18] as shown in Fig. 2. Our baseline processing framework is TransPIM [15] which is the state-of-the-art HBM-PIM accelerator for transformers. TransPIM utilizes an existing vector computation method within Compute-DRAM [22], where vectors are serialized in a bit-serial format with $\boldsymbol{n}$ rows of the same bit line for $\boldsymbol{n}$-bit data.

We propose a new processing framework, **MViT-PIM**, to efficiently support mixed precision ViT on PIM architectures. **MViT-PIM** incorporates lightweight modifications to the baseline processing framework in TransPIM [15], including support for: 1) arbitrary mixed precision data processing, and 2) mapping flow of various layer types such as inverted residual block and separable self-attention. For both multi-head and separable self-attention, we used an efficient dataflow mechanism utilizing a token-based sharding system from TransPIM.

**Table 3: Fine-tuning result of quantization observer using RL-PTQ. In $(A, B)$, $A$ represents the quantization bit-width and $B$ represents the quantization observer ($1$ : *minmax*, $2$ : *ema*, $3$ : *percentile* and $4$ : *mse* ). Top1 (Bit) is the accuracy of the MobileViTs after finding the appropriate bit-width of the model. Top1 (Bit+Obs) is the result of finding the appropriate quantization observer based on the optimal bit-width.**

| Model | Inverted Residual | | | Bridge Block | | Transformer | | Classifier | Comp. | Top1 | Top1 | Accuracy | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | exp | red | dwconv | conv3×3 | conv1×1 | mha linear | ffn linear | | Ratio | (Bit) | (Bit+Obs) | Drop | Improvememt |
| v1-XS | (8,2) | (8,2) | (8,4) | (8,2) | (7,4) | (6,2) | (6,3) | (5,1) | **4.93** | 72.92 | 73.74 | 0.96 | 0.82 ↑ |
| v1-S | (8,1) | (8,1) | (8,3) | (7,2) | (7,3) | (5,2) | (6,4) | (5,4) | **5.29** | 77.18 | 77.33 | 0.97 | 0.15 ↑ |
| v2-050 | (7,1) | (7,2) | (8,2) | (7,1) | (9,3) | (5,3) | (7,2) | (4,4) | **5.17** | 68.58 | 69.13 | 1.03 | 0.55 ↑ |
| v2-125 | (6,2) | (8,1) | (8,4) | (8,1) | (9,1) | (5,4) | (7,3) | (5,3) | **4.85** | 78.69 | 79.32 | 0.28 | 0.63 ↑ |
| v2-175 | (6,2) | (6,1) | (7,2) | (7,2) | (9,1) | (5,3) | (7,3) | (4,3) | **5.11** | 78.60 | 80.16 | 0.64 | 1.56 ↑ |
| v2-200 | (6,1) | (6,2) | (8,4) | (9,1) | (9,2) | (6,3) | (6,4) | (4,2) | **5.16** | 78.46 | 80.54 | 0.66 | 2.08 ↑ |

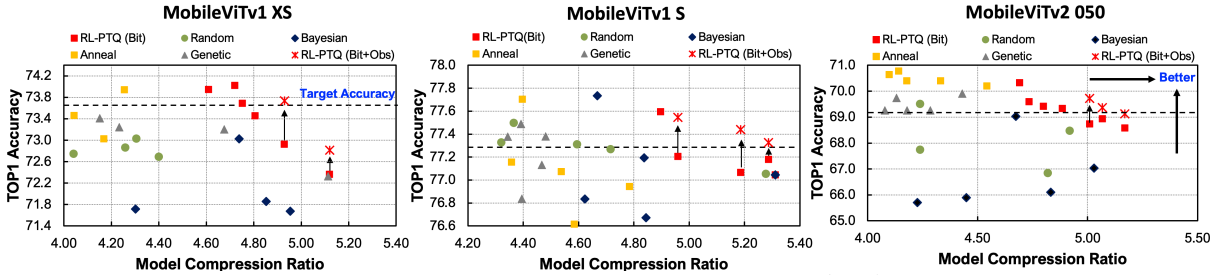


**Figure 8: Comparisons of accuracy and model compression ratio for v1-XS, v1-S, and v2-050 (< 6M) in MobileViTs after mixed precision PTQ with proposed RL-PTQ (Bit), RL-PTQ (Bit+Obs), and various search algorithms (random, Bayesian, annealing, and genetic algorithm). We applied input quantization with INT8. In the graph, the dashed line represents the target accuracy (accuracy drop < 1%).**

**Table 4: PTQ results of MobileViTs using the RL-PTQ (Bit).**

| Model | Compression Ratio | Weight (MP) | | | Input (INT8) / Weight (MP) | | |
|---|---|---|---|---|---|---|---|
| | | TOP1 | TOP5 | Acc. drop | TOP1 | TOP5 | Acc. drop |
| v1-XS | 4.75 | 74.31 | 92.12 | 0.39 | 73.69 | 90.82 | 1.01 |
| v1-S | 4.90 | 77.99 | 94.17 | -0.41 | 77.59 | 93.91 | 0.71 |
| v2-050 | 4.89 | 69.76 | 89.75 | 0.40 | 69.33 | 89.53 | 0.83 |
| v2-125 | 4.85 | 79.10 | 94.64 | 0.50 | 78.69 | 94.65 | 0.91 |
| v2-175 | 4.92 | 80.52 | 95.20 | 0.28 | 80.33 | 95.10 | 0.47 |
| v2-200 | 5.11 | 80.64 | 95.04 | 0.56 | 80.57 | 94.93 | 0.63 |

We break down input tokens into distinct shards and assign them to separate memory partitions, with memory banks serving as the primary partitions. We also apply the token-sharding flow to the convolutional layer, where we calculate output elements in different memory partitions and fully parallelize computations across all partial products. Each memory partition autonomously handles its corresponding token shards across multiple layers. We accelerated an inverted residual block by sequentially stacking convolution layers.

# 4 EXPERIMENTAL SETUP

## 4.1 Search Framework Implementations

**Implemenatation of RL-PTQ:** The baseline accuracy of MobileViTs is obtained from *ml-cvnets* [3] (ImageNet-1K datasets) and the PTQ environment is implemented using the libraries in FQ-ViT [11]. The *REINFORCE* and other search algorithms are implemented using PyTorch *v1.13.1* [16]. The **RL-PTQ** framework was trained for a total of 400 epochs using a single GPU core of the NVIDIA-TITAN RTX. In Mode 1 (RL-PTQ (Bit)), we only find the appropriate bit-width configurations using a min-max observer. In Mode 2 (RL-PTQ (Bit+Obs)), we selected several candidates from Mode 1 with lower accuracy than the target for fine-tuning 40 epochs to identify the optimal observer.

**Other Search Frameworks:** HAQ [10] faces a significantly large search space of $6^{72}$ when applied to the MobileViT-v1 models (72 layers). This complexity makes comparisons challenging, as it requires 72 time steps per epoch. We conducted a comparison with several other search algorithms: 1) random search, 2) Bayesian optimization (BO), 3) simulated annealing (SA), and 4) genetic algorithm (GA). All search algorithms were set to determine quantization configurations by grouping layers based on the type within the same training epochs (*Eps*). We configured the hyperparameters for other search frameworks according to the experimental setup in [16].

## 4.2 Previous PTQ methods

We compared our **RL-PTQ**'s results of mixed precision with an average of 8-bit (MP8), with existing rule-based PTQ methods with INT8, yielding a compression ratio of 4×, such as EasyQuant [6], PTQ4ViT [7], and Q-hyViT [5]. In all scenarios, input and attention data were quantized with INT8, and softmax and layer normalization retained their floating-point precision (FP32).

## 4.3 PIM Simulation

We created an in-house PIM simulator for the **MViT-PIM** processing framework, specifically to emulate the latency and energy characteristics of optimized MobileViTs within the PIM architecture. We assume the same memory configuration as TransPIM [15] in Table 2, which serves as the baseline PIM architecture. **MViT-PIM** accommodates various optimized MobileViT models utilizing arbitrary mixed precision (MP) data types. Due to the inability to efficiently map arbitrary mixed precision models on GPU or TransPIM, our mixed precision models were mapped to our **MViT-PIM**.

# 5 EVALUATION

## 5.1 Search Results of RL-PTQ

**RL-PTQ (Bit), Bit-width Search:** The **RL-PTQ** is advantageous for identifying Pareto-optimal combinations as shown in Fig. 6. We facilitate flexible exploitation of accuracy and model compression ratio trade-offs with mixed precision. TABLE 4 presents our best search results of **RL-PTQ (Bit)**. As a result, quantization strategies customized for each layer type can yield optimized results.

**RL-PTQ (Bit+Obs), Observer Search:** In TABLE 3, the presented finetuning results showcase the performance of the mixed quantization observer for each layer type. Introducing a mixed observer (MO) method improves the accuracy, *Top1 (Bit+Obs)*, as compared to the obtained accuracy results in mode 1, *Top1 (Bit)*, with a min-max observer across all module types. As a result, we can improve the accuracy as shown in TABLE 3. When allowing for an accuracy drop of up to 1% compared to the baseline model, we can achieve model compression ratios of 5.1 on average for various MobileViTs.

**Table 5: A comparison of four PTQ methods for image classification on ImageNet-1K using MobileViT Models: 1) EasyQuant [6], 2) PTQ4ViT [7], 3) Q-hyViT [5], and 4) RL-PTQ (ours). The proposed RL-PTQ applied 8-bit activation and mixed precision (MP8) weight quantization, achieving a weight compression of more than 4 times.** ***Acc. Improve*** **shows the accuracy improvement compared to the state-of-the-art method, Q-hyViT.**

| Model | Model Information | | | | EasyQuant [6] | PTQ4ViT [7] | Q-hyViT [5] | RL-PTQ (Ours) | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | #Params | #Flops | Model Type | Baseline Acc. (FP32) | Rule-based (INT8, Cosine) | Rule-based (INT8, Hessian) | Rule-based (INT8, Hessian) | Learning-based (MP8, MO) | Comp. Ratio | Acc. Improve |
| MobileViTv1-xs | 2.3M | 1,028M | Hybrid | 74.70 | 73.16 | 65.52 | 74.31 | **74.77** | 4.31 | 0.46 ↑ |
| MobileViTv1-s | 5.6M | 2,004M | Hybrid | 78.30 | 74.21 | 68.19 | 77.92 | **77.93** | 4.32 | 0.01 ↑ |
| MobileViTv2-050 | 1.4M | 1,048M | Hybrid | 70.16 | 66.80 | 39.39 | 69.89 | **70.64** | 4.43 | 0.75 ↑ |
| MobileViTv2-125 | 7.5M | 6,345M | Hybrid | 79.60 | 77.31 | 67.39 | 79.31 | **79.36** | 4.40 | 0.05 ↑ |
| MobileViTv2-175 | 14.3M | 12,360M | Hybrid | 80.80 | 79.93 | 72.30 | 80.63 | **81.25** | 4.32 | 0.62 ↑ |
| MobileViTv2-200 | 18.6M | 16,112M | Hybrid | 81.20 | 80.04 | 75.50 | 80.94 | **81.43** | 4.41 | 0.49 ↑ |

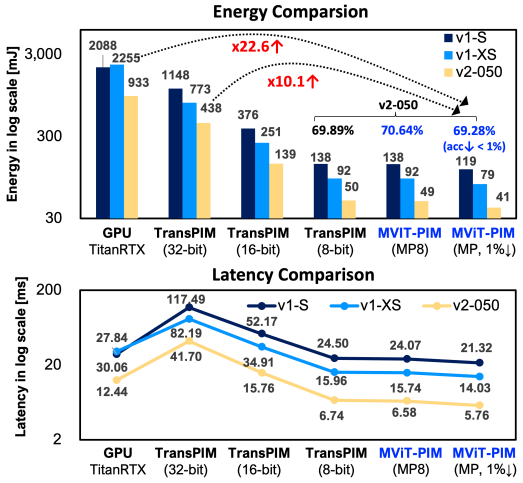


**Figure 9: Comparison of energy and latency (logarithmic scale) among MobileViTs on the GPU, TransPIM [15] and MViT-PIM.**

## 5.2 Comparison with Other Search Algorithms

Fig. 8 presents the trade-off relationships between accuracy and model compression ratio with ours and other search algorithms. The dashed line represents the target accuracy. When visualizing the top four or five design points explored by these search algorithms, the outcomes reveal that **RL-PTQ (Bit)** and **RL-PTQ (Bit+Obs)** consistently demonstrate higher accuracy compared to the target accuracy with the highest compression ratios (4.93×, 5.29×, and 5.17×) for v1-XS, v1-S and v2-050.

## 5.3 Comparison with Other PTQ Methods

Table 5 shows the comparison of four PTQ methods. Both EasyQuant [6] and PTQ4ViT [7] exhibit a significant drop in accuracy. Conversely, Q-hyViT [5] achieved less than a 0.5% accuracy drop with 8-bit quantization on the MobileViT models. Ultimately, **RL-PTQ** yields the highest accuracy with negligible accuracy drop among the previously developed PTQ methods. We improved the accuracy by 0.4% on average compared to the Q-hyViT. This demonstrates the effectiveness of mixed precision.

## 5.4 HW Acceleration on PIM

We conducted a comparison of latency and energy among several optimized models, encompassing fixed-point (32-bit, 16-bit, and 8-bit), along with our mixed precision (MP) quantization on the Titan RTX GPU, TransPIM [15], and the **MViT-PIM** (Fig. 9). The average latency of our models on the **MViT-PIM** improved by 1.82× and 6.16×, with energy consumption gains of 22.6× (Titan RTX GPU) and 10.1× (TransPIM [15]), within a 1% accuracy drop compared to the baseline model (FP32). When compared to quantization using 8-bit fixed-point, mixed precision with an average of 8-bit (MP8) demonstrates higher accuracy than the state-of-the-art accuracy of Q-hyViT [5], along with similar energy consumption and reduced latency. Additionally, our mixed precision model, with a 1% accuracy drop (MP, 1%↓), exhibits an average energy consumption reduction of 15% compared to the 8-bit fixed point case.

# 6 CONCLUSION

We propose an RL-based mixed precision and mixed observer approach by grouping layers based on the type to explore the hybrid transformer's compression capabilities. We achieved an average 5.1× model compression with 1% accuracy loss. Our quantized model on **MViT-PIM** shows energy efficiency improvements of 10.1× and 22.6× compared to the baseline PIM accelerator and GPU, respectively. In addition, the **RL-PTQ** framework has high extensibility to various hybrid transformers, such as MobileFormer and EfficientFormer due to their similarities in layer structures, such as inverted residual block, transformer, or bridge blocks [5].

# 7 ACKNOWLEDGEMENT

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT).(No. RS-2023-00222085, Development of memory module and memory compiler for non-volatile PIM) and was funded by PRISM, one of seven centers in JUMP 2.0 (an SRC program sponsored by DARPA), SRC Global Research Collaboration (GRC) grant, and NSF grants #2112167, #2003279, #2100237, #2112665, #2008365, and #2052809.

# REFERENCES

[1] J. Mao et al., "Tprune: Efficient transformer pruning for mobile devices", *ACM Trans. on Cyber-Physical Systems*, 2021, 5(3), pp. 1-22.
[2] A. Dosovitskiy et al., "An image is worth 16x16 words: transformers for image recognition at scale", *arXiv preprint arXiv:2010.11929*, 2020.
[3] S. Mehta et al., "Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer", *arXiv preprint arXiv:2110.02178*, 2021.
[4] S. Mehta et al., "Separable self-attention for mobile vision transformers", *arXiv preprint arXiv:2206.02680*, 2022.
[5] J. Lee et al., "Q-hyViT: Post-training quantization for hybrid vision transformer with bridge block reconstruction", *arXiv preprint arXiv:2303.12557*, 2023.
[6] D. Wu et al., "Easyquant: Post-training quantization via scale optimization", *arXiv preprint arXiv:2006.16669*, 2020.
[7] Z. Yuan et al., "Ptq4vit: Post-training quantization for vision transformers with twin uniform quantization", *In Proc. ECCV*, 2022, pp. 191-207.
[8] S. Yun et al., "Do all mobilenets quantize poorly? gaining insights into the effect of quantization on depthwise separable convolutional networks through the eyes of multi-scale distributional dynamics", *In Proc. CVPR*, 2021, pp.2447-2456.
[9] *https://docs.nvidia.com/deeplearning/tensorrt/support-matrix/index.html#hardware-precision-matrix*
[10] K. Wang et al., "Haq: Hardware-aware automated quantization with mixed precision", *In Proc. CVPR*, 2019, pp. 8612-8620.
[11] Y. Lin et al., "Fq-vit: Post-training quantization for fully quantized vision transformer" *arXiv preprint arXiv:2111.13824*, 2021.
[12] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference", *In Proc. CVPR*, 2018, pp. 2704-2713.
[13] R. Li et al., "Fully quantized network for object detection", *In Proc. CVPR*, 2019, pp. 2810-2819.
[14] Y. Choukroun et al., "Low-bit quantization of neural networks for efficient inference", *In Proc. ICCVW*, 2019, pp. 3009-3018.
[15] M. Zhou and W. Xu et al., "Transpim: A memory-based acceleration via software-hardware co-design for transformer", *In Proc. HPCA*, 2022, pp. 1071-1085.
[16] S.C. Kao et al., "Confuciux: Autonomous hardware resource assignment for dnn accelerators using reinforcement learning" *In Proc. MICRO*, 2020, pp.622-636.
[17] Y. Umuroglu,et al., "Bismo: A scalable bit-serial matrix multiplication overlay for reconfigurable computing", *In Proc. FPL*, 2018.
[18] JEDEC Standard JESD235: High Bandwidth Memory (HBM) DRAM, JEDEC Solid State Technology Association, Virginia, USA, Standard, 2013.
[19] R. S. Sutton et al., "Reinforcement Learning: An Introduction" MIT Press, Cambridge, 2018.
[20] M. Sandler et al., "Mobilenetv2: Inverted residuals and linear bottlenecks", *In Proc. CVPR*, 2018, pp. 4510-4520.
[21] H. Wu et al., "Integer quantization for deep learning inference: Principles and empirical evaluation", *arXiv preprint arXiv:2004.09602*, 2020.
[22] F. Gao et al., "Computedram: In- memory compute using off-the-shelf drams", *In Proc. MICRO*, 2019, pp. 100–113.