Quasar-ViT: Hardware-Oriented Quantization-Aware Architecture Search for Vision Transformers

Zhengang Li[1], Alec Lu[2], Yanyue Xie[1], Zhenglun Kong[1], Mengshu Sun[†], Hao Tang[3], Zhong Jia Xue[2], Peiyan Dong[1], Caiwen Ding[4], Yanzhi Wang[1], Xue Lin[1], Zhenman Fang[2]
[1]Northeastern University, [2]Simon Fraser University, [3]ETH Zurich, [4]University of Connecticut
E-mail: [1]{li.zhen, xie.yany, kong.zhe, dong.pe, yanz.wang, xue.lin}@northeastern.edu
[2]{alec_lu, zjxue, zhenman}@sfu.ca

ABSTRACT

Vision transformers (ViTs) have demonstrated their superior accuracy for computer vision tasks compared to convolutional neural networks (CNNs). However, ViT models are often computation-intensive for efficient deployment on resource-limited edge devices. This work proposes Quasar-ViT, a hardware-oriented quantization-aware architecture search framework for ViTs, to design efficient ViT models for hardware implementation while preserving the accuracy. First, Quasar-ViT trains a supernet using our row-wise flexible mixed-precision quantization scheme, mixed-precision weight entanglement, and supernet layer scaling techniques. Then, it applies an efficient hardware-oriented search algorithm, integrated with hardware latency and resource modeling, to determine a series of optimal subnets from supernet under different inference latency targets. Finally, we propose a series of model-adaptive designs on the FPGA platform to support the architecture search and mitigate the gap between the theoretical computation reduction and the practical inference speedup. Our searched models achieve 101.5, 159.6, and 251.6 frames-per-second (FPS) inference speed on the AMD/Xilinx ZCU102 FPGA with 80.4%, 78.6%, and 74.9% top-1 accuracy, respectively, for the ImageNet dataset, consistently outperforming prior works.

ACM Reference Format:
Zhengang Li[1], Alec Lu[2], Yanyue Xie[1], Zhenglun Kong[1], Mengshu Sun[†], Hao Tang[3], Zhong Jia Xue[2], Peiyan Dong[1], Caiwen Ding[4], Yanzhi Wang[1], Xue Lin[1], Zhenman Fang[2], [1]Northeastern University, [2]Simon Fraser University, [3]ETH Zurich, [4]University of Connecticut, E-mail: [1]{li.zhen, xie.yany, kong.zhe, dong.pe, yanz.wang, xue.lin}@northeastern.edu, [2]{alec_lu, zjxue, zhenman}@sfu.ca. 2024. Quasar-ViT: Hardware-Oriented Quantization-Aware Architecture Search for Vision Transformers. In *Proceedings of the 38th ACM International Conference on Supercomputing (ICS '24), June 4–7, 2024, Kyoto, Japan.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3650200.3656622

[†] MS is now afflicted with Beijing University of Technology; she contributed to this work during Ph.D at Northeastern University.

ICS '24, June 4–7, 2024, Kyoto, Japan

ACM ISBN 979-8-4007-0610-3/24/06…$15.00
https://doi.org/10.1145/3650200.3656622

1 INTRODUCTION

ViTs [10, 35, 42, 62] incorporate the attention mechanism [46] to fulfill various computer vision tasks, by allowing all the pixels in an image to interact through transformer encoder blocks and thus achieving higher accuracy compared to CNNs. Table 1 compares representative CNN and ViT models, i.e., ResNet [12]/ResNeXt [56] and DeiT [42] for the ImageNet dataset. DeiT-small (DeiT-S) with a comparable number of parameters and GMACs as ResNet-50 achieves even higher accuracy than ResNeXt-101, whose size is around 4× as that of DeiT-S. DeiT-base (DeiT-B) with comparable size as ResNeXt-101 achieves 2.54% higher top-1 accuracy.

Table 1: Comparison of ResNets, ResNeXt, and DeiTs on ImageNet dataset. We choose DeiT without distilling token here, which represents state-of-the-art ViTs, as it can be directly trained on ImageNet-10k without pre-training on a massive dataset.

Model	#Params (M)	MACs (G)	Top-1 Acc.	Top-5 Acc.
ResNet-18	11.69	1.82	69.76%	89.08%
ResNet-50	26.56	4.14	76.13%	92.86%
ResNet-152	60.19	11.61	78.31%	94.05%
ResNeXt-101	88.79	16.59	79.31%	94.53%
DeiT-S	22.10	4.60	79.85%	94.97%
DeiT-B	87.50	17.60	81.85%	95.59%

Despite ViTs' significant accuracy improvement, it is non-trivial to deploy ViT inference on resource-limited edge devices due to their huge model size and complex architectures. For example, even the lightweight ViT model DeiT-S [42] has a model size of $22.10M$ parameters $\times 4Bytes$ per floating-point parameter = $88.4MB$, presenting an overwhelming computing load and memory size for most edge devices.

The basic transformer encoder with multi-headed self-attention (MSA) and multi-layer perceptron (MLP) blocks is shown in Figure 1, consisting of multiple different computation components, including linear layer, attention, residual addition, matrix reshape operation, GELU, and layer norm. To further understand the bottleneck of the current ViT model structure, we profile the runtime of each component of ViT on a Xeon(R) Silver 4214 CPU [16] using Pytorch Profiler [33] as shown in Figure 1. We use the same color to indicate the same component in both the transformer block structure and profiling figures. It shows matrix multiplication operations dominate the processing time (94.7% and 87.3% for DeiT-B [42] and DeiT-S [31], respectively) of execution cycles.

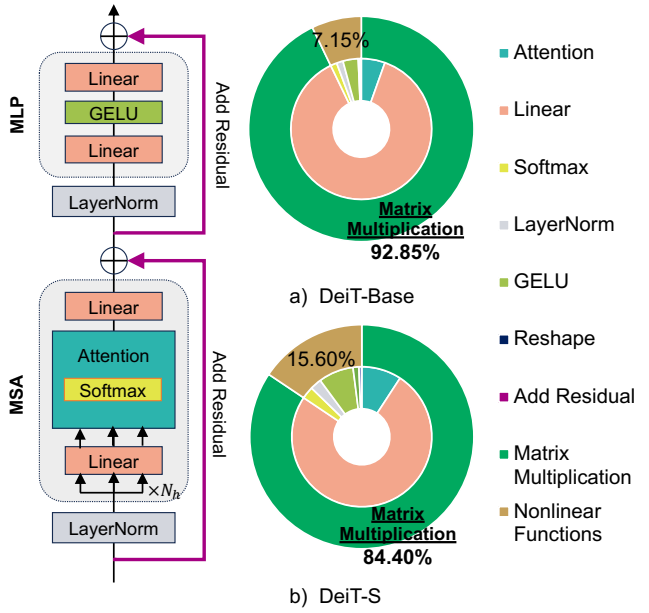

Figure 1: Transformer encoder block structure and ViT model execution performance profiling on CPU for a) DeiT-Base and b) DeiT-S with 12 encoders on ImageNet dataset.

Contemporary acceleration methods mainly focus on reducing the practical inference latency of matrix multiplication operations. They primarily fall into two categories: 1) neural architecture search (NAS) that searches the lighter-weight model; and 2) model compression, especially model quantization that reduces the per-parameter bit-width. However, there are two major challenges when applying these methods on hardware. The first challenge is associated with model quantization. It has been revealed that the most suitable quantization schemes/bit-widths depend on model sizes and architectures [49, 54], and there is a vast design space in the quantization of both weights and activations for each layer on different models and hardware. As ViT models become deeper, the design space increases exponentially, resulting in poor performance of rule-based strategies. Although recent studies explored automated quantization techniques for a given ViT architecture [45, 49, 54], they did not integrate model quantization with NAS together, which could result in suboptimal performance. In this paper, we propose the framework of model quantization and NAS co-design for ViTs towards improved performance compared to treating NAS and quantization separately.

The second challenge is the gap between the theoretical computation throughput and the practical inference speed on actual hardware. For example, layer-wise (inter-layer) mixed-precision quantization (for CNNs) [49, 54] quantizes each layer with a different bit-width and therefore executes layers through distinct hardware components sequentially, leading to low resource utilization. Furthermore, kernel-wise mixed-precision quantization (for CNNs) [29] assigns different bit-widths down to the kernel level, significantly diversifying the computing pattern and is inefficient for hardware implementation.

Recent work FILM-QNN [40] and Auto-ViT-Acc [22] leverage the intra-layer mixed quantization to achieve good performance for both model accuracy and throughput on FPGA. By applying two different quantization bit-widths/schemes for different channels and limiting the same mixed-precision ratio across each layer, FPGA can efficiently handle different computations on different hardware resources sharing the same hardware design. However, existing approaches suffer from a manually configured uniform mixed-precision ratio across all layers, potentially compromising quantized model accuracy. Moreover, architectural design considerations are often neglected, limiting the overall model performance.

To address these problems comprehensively, we propose Quasar-ViT, an integration of a hardware-oriented quantization-aware architecture search targeting ViT. First, to fully unleash the computation potential of FPGA resources, we investigate a hardware-friendly row-wise mixed-precision quantization scheme. At the algorithm level, different from FILM-QNN [40] and Auto-ViT-Acc [22], we quantize different channels within each layer into lower and higher bit-widths with the flexibility of different mix-ratios for layers, which achieves a more fine-grained architecture to maintain the accuracy. At the hardware level, we propose the FPGA-based model-adaptive design, including 4-bit atomic computation and hybrid signed/unsigned DSP packing, which set basic hardware units for the lower-bit computation, and decompose the higher-bit computation to lower-bit ones to reuse the resources. Second, during the supernet training, we propose the mixed-precision weight entanglement mechanism, such that different transformer blocks in subnets can share weights for their common parts in each layer to enable efficient quantization during architecture search and reduce training memory cost. On top of that, we establish the corresponding FPGA latency and resource modeling to estimate the inference latency and combine it with an efficient hardware-oriented evolution search method. Based on the above, we integrate with the one-shot NAS algorithm to efficiently find the most accurate quantized model under the given inference latency. We also explore the layer scaling in CaiT [43] and extend it to the supernet architecture to improve the training efficiency and model accuracy. To demonstrate the compatibility of our proposed framework with knowledge distillation (KD) and further improve our searched model accuracy, we integrate KD [15] into the training process. Finally, on the hardware side, we implement the basic computing units for 4-bit weight and 6-bit activations with hybrid signed/unsigned DSP packing optimization to enable efficient FPGA implementation.

The contributions of our work are summarised as follows:

- An end-to-end hardware-oriented quantization-aware architecture search framework (Quasar-ViT) for ViTs, achieving superior accuracy and inference speed over prior studies. Latency/resource modeling of the hardware accelerator design is integrated into the search process.
- Hardware-friendly quantization techniques—such as flexible row-wise mixed-precision quantization and mixed-precision weight entanglement—in the architecture search, towards high accuracy, low training cost, and efficient implementation.

- Real FPGA implementations of our model-adaptive design, with our proposed 4-bit weight atomic computation and hybrid signed/unsigned DSP packing.
- Integration of proposed supernet layer scaling (SLS) in our framework, achieving further accuracy improvement. Our ablation study also demonstrates our framework's good compatibility with knowledge distillation (KD).

Quasar-ViT achieves 101.5, 159.6 and 251.6 FPS on the AMD/Xilinx ZCU102 FPGA board with 80.4%, 78.6%, and 74.9% top-1 accuracy for ImageNet, respectively.

Compared to the representative ViT training-aware quantization [20] and the post-training quantization [27], at a similar model size, our model achieves 2.1% and 5.2% higher top-1 accuracy, respectively. Compared with Auto-ViT-Acc [22], a state-of-the-art FPGA accelerator for ViT with mixed-scheme quantization (without NAS), we achieve 1.7% better top-1 accuracy with a similar FPS, and 1.6× better FPS with a similar level of model accuracy.

2 RELATED WORK

2.1 Vision Transformers

First proposed in [10], the vision transformer (ViT) is a groundbreaking work that uses transformer blocks for vision tasks. Unlike traditional CNN architectures that use a fixed-size window with restricted spatial interactions, ViT interprets an image as a sequence of patches and adopts the self-attention mechanism [46]. This allows all the positions in an image to interact through transformer blocks, which provides the extraordinary capability to capture relations at the pixel level in both spatial and temporal domains. However, the original ViT requires pre-training with large-scale datasets such as ImageNet-21k and JFT-300M. To tackle the problem, many variants such as DeiT [42] and T2T-ViT [62] were proposed, which can be well trained with only ImageNet-10k. ViTs improve model accuracy at the cost of increased volume of computation and structural complexity. In ViTs, the main model architecture is transformer encoder blocks with multi-headed self-attention (MSA) and multilayer perceptron (MLP) blocks. These blocks involve large matrix multiplications, which incur the most computational cost. These complex architectures and enormous computation/storage demand make it hard to deploy ViTs on resource-limited edge devices.

Therefore, we quantize all layers involved in matrix multiplication, but not the non-linear functions, e.g., layer normalization, due to their low computational cost and potential effects on accuracy.

2.2 Non-Transformer DNN Model Quantization

2.2.1 Quantization Scheme. To compress model size and improve inference speed, model quantization has been widely explored for deep neural networks (DNNs). Existing quantization research can be categorized according to quantization schemes, such as binary [8, 36], ternary [13], and low-bit-width fixed-point [7, 7, 68, 68] quantize models with the same interval between each quantization level. Although binary and ternary quantization reduce operations and simplify hardware implementation to the extreme, they introduce large accuracy loss due to insufficient bit-width. For example, based on reports from the above works, accuracy typically degrades by $> 5\%$ under binary quantization and $2-3\%$ for ternary quantization. To overcome the large accuracy loss coming from insufficient bit-width, the fixed-point quantization is proposed, applying moderate and adjustable quantization bit-width, to maintain accuracy. This quantization scheme was implemented with different methods and algorithms, such as DoReFa-Net [68] and PACT [7].

Finally, there are also non-linear quantization schemes, such as power-of-two (PoT) [17] and additive PoT [19]. They replace the multiplication with shifting operations where the distribution of quantization levels becomes unbalanced, having higher precision around the mean and less precision at the two sides.

2.2.2 Mixed-Precision/Scheme Quantization. To explore more quantization potential while preserving the model accuracy, Besides the single scheme quantization, some works [9, 39, 45, 49, 54] explore inter-layer mixed-precision quantization by assigning different precisions to layers. For example, HAQ [49] determines the bit-width of each layer by an agent trained with reinforcement learning. DNAS [54] used NAS to search layer-wise bit-width. Furthermore, [29] explored intra-layer mixed quantization to enable different precisions or schemes within each layer. Based on them, hardware designs [5, 40] leveraged the intra-layer mixed-precision/mixed-scheme to enable uniformity within each layer, guaranteeing inference acceleration. However, they need to set the same mixed ratio for layers, which limits the model's accuracy.

2.3 Transformer and ViT Quantization

Quantization has also been studied for transformers, especially for natural language processing (NLP) tasks [1, 64, 65]. Q8BERT [64] finetuned BERT through 8-bit quantization-aware training. TernaryBERT [65] implemented an approximation-based and loss-aware ternary quantization on BERT. BinaryBERT [1] proposed a ternary weight splitting strategy to derive binary BERT with performance as the ternary one. Inspired by those, [27] and [22] studied quantization on ViT in computer vision tasks. PTQ [27] evaluated the post-training quantization on ViT and achieved comparable accuracy to the full-precision version. Auto-ViT-acc [22] proposed an FPGA-aware framework with mixed-scheme quantization for ViT, which we will compare in the evaluation. FQ-ViT [24] proposed power-of-two factor and log-int-softmax to proceed with the ViT quantization. Q-ViT [20] used the switchable scale to achieve head-wise ViT mixed quantization. However, these works are all based on full-precision pre-trained models and do not include the dimension of network architecture search.

2.4 Neural Architecture Search

2.4.1 NAS Strategies. There has been a trend to design efficient DNNs with NAS. In general, NAS can be classified into the following categories according to its search strategy. First, reinforcement learning (RL) methods [2, 4, 25, 32, 67, 69, 70] use recurrent neural networks as predictors validating the accuracy of child networks over a proxy dataset. Second, evolution methods [30, 37] develop a pipeline of parent initialization, population updating, generation, and elimination of offspring to find desired networks. Third, one-shot NAS [3, 11, 60] trains a large one-shot model containing all operations and shares the weight parameters with all candidate models. Based on the above work, weight-sharing NAS has become popular due to training efficiency [38, 47, 61]. One over-parameterized supernet is trained with weights shared across all

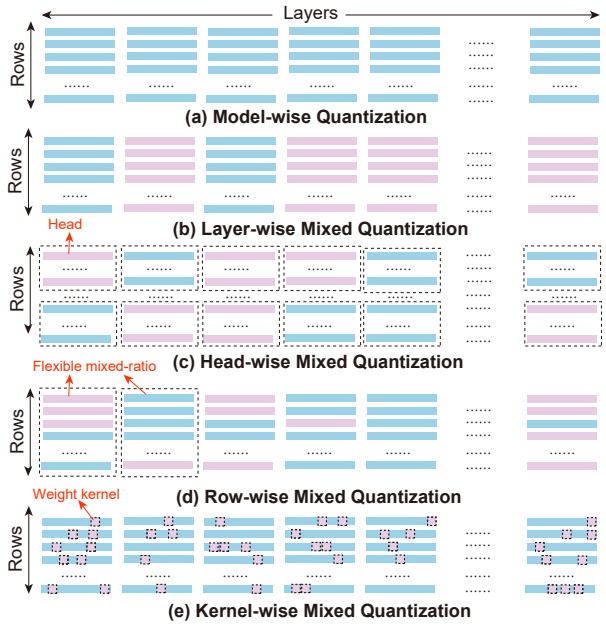

Figure 2: Comparison of mixed-precision quantization under different granularity. We use the example of two different bit-widths, represented as blue and pink colors. We propose the row-wise flexible mixed-precision quantization in (d).

sub-networks in the search space. This significantly reduces the computational cost during the search. Although most of the above work focuses on the traditional CNN architectures, such as [61] and [38], some works have started investigating the search for efficient ViT networks [6, 18, 48, 55]. Among them, Autoformer [6] entangles the model weights of different ViT blocks in the same layer during supernet training with an efficient weight-sharing strategy to reduce training model storage consumption as well as training time.

2.4.2 Hardware-Oriented NAS. Some recent works realize the gap between theoretical computation improvement and practical inference speedup. They investigate the algorithm/hardware co-design and incorporate the inference latency into NAS [23, 41, 53], which is more accurate than intuitive volume estimation by MAC operations. For example, MnasNet [41] and NPAS [23] utilize the latency on mobile devices as the reward to perform RL search, where gradient-based NAS work FBNet [53] adds a latency term to the loss function. However, these works neither target ViTs nor exploit quantization in the hardware-aware ViT search.

3 HARDWARE-ORIENTED QUANTIZATION-AWARE NAS FOR VITS

3.1 Row-Wise Flexible Mixed-Precision Quantization with Hardware/Software Co-design

Figure 2 classifies quantization with different levels of granularity. Model-wise quantization [7, 68] uses a unified quantization bit-width for the whole model and thus misses some quantization opportunities. On the other hand, mixed-precision quantization, as discussed in related work, explores more quantization potential (i.e., quantizing each component to a bit-width as low as possible) while preserving the accuracy of the model. Specifically, layer-wise (inter-layer) mixed-precision quantization [49, 54] sets each layer with a specific quantization bit-width. Besides that, Q-ViT [20] proposed a head-wise mixed-precision quantization scheme, which assigns different bit-widths to different attention heads. Both the layer-wise and head-wise quantization schemes suffer from limited quantization flexibility without considering the variance inside each layer. Moreover, fixed row-wise (intra-layer) mixed-precision quantization is proposed in prior work [40], which uses different quantization bit-widths for different channels in each CNN layer and limits the same mixed-precision ratio across different CNN layers, and thus multiple layers can share the same hardware design, making it more hardware-friendly. Finally, kernel-wise mixed-precision quantization [29] assigns different quantization bit-widths down to the kernel level, which greatly diversifies the computing pattern and makes it inefficient to implement on hardware.

Based on the above discussion, we use the row-wise flexible mixed-precision quantization scheme for ViTs, as shown in Figure 2(d), which preserves the quantization flexibility among layers for better accuracy while maintaining the hardware uniformity for more efficient implementation. Different from [40] that limits the same mixed-precision ratio across CNN layers, for ViTs, we have to provide the flexibility to obtain different mixed ratios in different layers to maintain the model accuracy. To maintain hardware uniformity and avoid hardware under-utilization, we propose to design the basic hardware units for the lower-bit computation, decompose the higher-bit computation into lower-bit ones, and reuse the basic hardware units (described in Section 4.2 and Section 4.3). As a result, we have preserved the uniformity of the hardware design and enabled the flexible bit-width mixed-ratio among ViT layers. We explain the hardware details in Section 4.

3.2 Intra-layer Mixed-Precision Weight Entanglement

In classical one-shot NAS, the weights of each sample candidate are shared with the supernet during training. However, as shown in Figure 3 (a), when using the classical weight-sharing strategy, the building blocks from multiple subnets, even in the same layer, are isolated. Therefore, it leads to higher memory costs and slower training convergence.

To address this problem, weight entanglement is proposed in [6] to reduce the supernet model size: as shown in Figure 3 (b), different transformer blocks can share their common weights in each layer. It also allows each block to be updated more times than the previous independent training strategy, thus achieving faster convergence. However, this structure is hard to combine with mixed quantization since one shared weight cannot be trained into two different bit-widths at the same time (i.e., bit-width conflict).

In this paper, we propose the mixed-precision weight entanglement, as shown in Figure 3 (c), to incorporate the quantization search while preventing the potential bit-width conflicts problem in the shared weight. Mixed-precision weight entanglement block

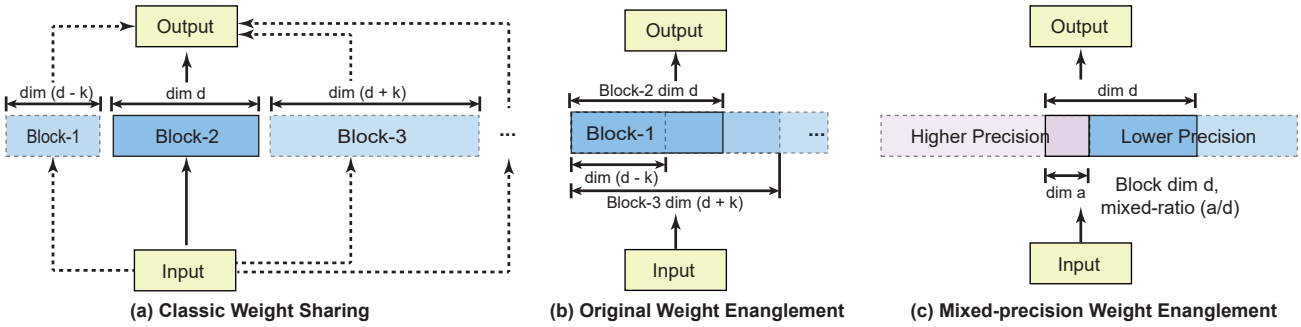

Figure 3: Classic weight sharing and original weight entanglement versus our proposed mixed-precision weight entanglement.

contains two parts of weights with different precisions. Different quantization mixed ratios can be achieved by extracting different percentages of weights over these two parts. In the implementation, for each layer, we only need to store the weights of the largest of the n homogeneous candidate blocks with both the 4-bit and 8-bit weights. The remaining smaller building blocks can extract the weights directly from the largest building block. The different quantization mixed ratios can be reached by moving the position of the selected block.

3.3 Supernet Layer Scaling (SLS) Structure

The layer scaling structure proposed by CaiT [43] improves the stability of the optimizations when training transformers for image classification, thus improving the model accuracy. We explore this structure and extend it to supernet layer scaling (SLS).

Layer scaling is a per-channel multiplication of the vector produced by each residual block. For ViT, this layer is deployed after the multi-head self-attention (MSA) and multi-layer perceptron (MLP) modules in each encoder block. The objective is to group the updates of the weights associated with the same output channel. Layer scaling can be denoted as a multiplication by a diagonal matrix diag $(\lambda_{l,1}, \ldots, \lambda_{l,d})$ on the output of l-th residual block, where d is the corresponding number of output channels in the model. All λs are learnable weights.

To fit our mixed-precision weight entanglement strategy, different from the original CaiT [43] implementation that uses the whole layer scaling in every training iteration, our SLS extracts the corresponding elements synchronized with the output dimension of the selected subnet while keeping the other weights frozen. As shown in Figure 4, using the residual block of MLP as an example, assuming that the current MLP's output dimension starts from m-th channel and ends at n-th channel, the supernet layer scaling computation can be formulated as:

$$y_l = x_l + \text{diag}\left(\lambda_{l,m}, \ldots, \lambda_{l,n}\right) \times \text{MLP}\left(\text{LN}\left(x_l\right)\right), \tag{1}$$

where x_l and y_l denote the input and output, respectively; LN means the layer normalization.

3.4 End-to-End Quasar-ViT Framework

3.4.1 One-shot NAS Algorithm. Our one-shot NAS algorithm consists of two steps:

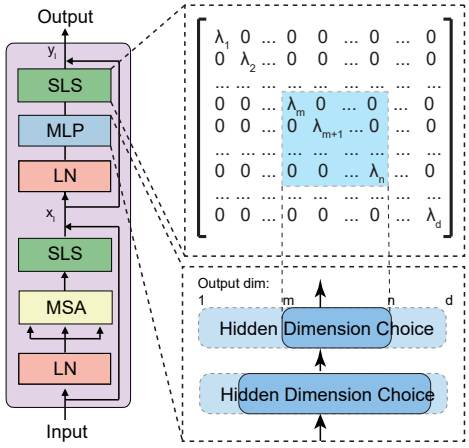

Figure 4: Supernet layer scaling (SLS) in Quasar-ViT encoder block. We use the SLS after MLP as an example.

- We train a supernet to directly sample different quantized architectures as child models for training. The supernet is trained with SLS and KD techniques. The search space is encoded in the supernet, and the parameters of all candidate networks in the search space are optimized simultaneously by our proposed weight-sharing during training.
- We select architectures from the pre-trained supernet using the hardware-oriented evolution search method to find the most accurate model under the given hardware resource constraints. We search based on the hardware latency/FPS and resource modeling illustrated in Section 4.4.

Here, we show a toy example of the supernet training process including candidate sampling and the corresponding searched results for different targeting FPS in Figure 5. Figure 5 (a) illustrates one iteration of the supernet training process, where the pink area indicates the sampled high precision values and the blue area indicates the sampled low precision values in the supernet. The light blue area indicates the frozen values (currently not sampled) in this iteration. After the supernet training and the hardware-oriented evolution search, we could obtain different models targeting different frames per second (FPS) as shown in Figure 5 (b). For the

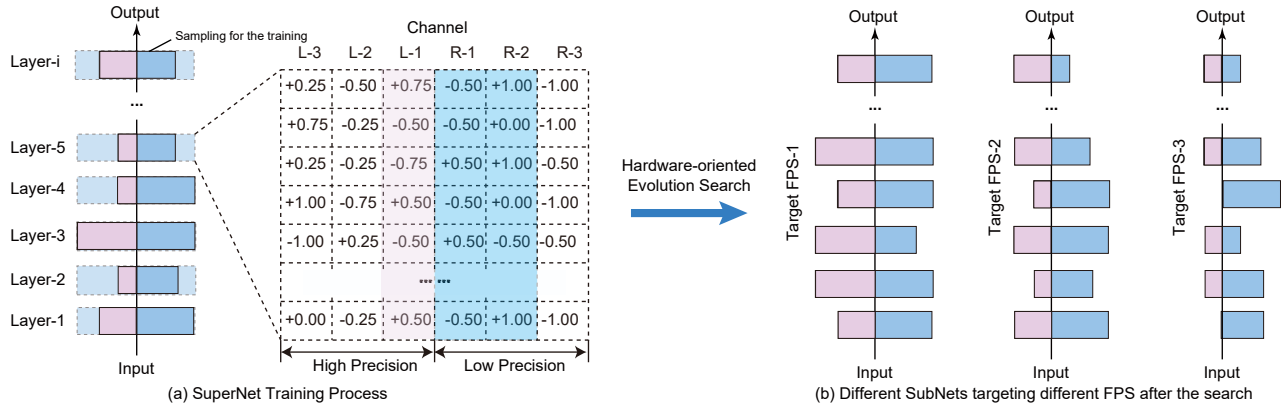

Figure 5: SuperNet training process including candidate sampling and the searched results for different targeting FPS. We use a model with layers of 6 channels as a toy example.

sake of brevity, we only show the quantized value here. The scaling factor along with other related structures is omitted.

3.4.2 Search Space. We show our search space design in Table 2. Our search components include the overall embedding dimension, the number of transformer layers, the quantization mixed-ratio (i.e., the percentage of 8-bit weights mixed in the layer) for each linear layer, and the hidden dimension and expansion ratio (for MLP) in each ViT encoder block.

To accelerate the supernet training process and improve the overall model performance, we partition the large-scale search space into two sub-spaces and encode them into two independent supernets for QUASAR-Small and QUASAR-Large, respectively. By splitting and customizing search space for supernets of different sizes, we mitigate the training interference caused by the huge subnets' difference. This training strategy has been proved in [66]. Such partition allows the search algorithm to concentrate on finding models within a specific hardware inference latency, which can be specialized by users according to their available resources and application requirements. It also reduces gradient conflicts between large and small sub-networks trained via weight-sharing due to gaps in model sizes.

Table 2: An illustration of our search space: It is divided into two independent supernets within the different parameter ranges to satisfy different resource constraints.

	QUASAR-Small	QUASAR-Large
Embed Dimension	(192, 216, 240)	(320, 384, 448)
Hidden Dimension	(192, 256)	(320, 384, 448)
8-bit Mixed-ratio	(0%, 25%, 50%)	(0%, 25%, 50%)
Expansion Ratio	(3.5, 4)	(3, 3.5, 4)
Number of Layers	(12,13,14)	(12,13,14)

3.4.3 Supernet Training. In each iteration, we randomly select a quantized ViT architecture from the search space. Then we obtain its weights from the supernet and compute the losses of the subnet.

Algorithm 1 Supernet Training.

Input: Training epochs N, search space $\mathcal{P}$, supernet $\mathcal{S}$, loss function L, train dataset D_{train}, initial supernet weights $\mathcal{W}_{\mathcal{P}}$, candidate weights $\mathcal{W}_p$
for i in N epochs **do**
  for data, labels in D_{train} **do**
    Randomly sample one quantized ViT architecture from search space $\mathcal{P}$
    Obtain the corresponding weights $\mathcal{W}_p$ from supernet $\mathcal{W}_{\mathcal{P}}$
    Compute the gradients based on L
    Update the corresponding part of $\mathcal{W}_p$ in $\mathcal{W}_{\mathcal{P}}$ while freezing the rest of the supernet $\mathcal{S}$
  end for
end for
Output $\mathcal{S}$

Finally, we update the corresponding weights with the remaining weights frozen. The architecture search space P is encoded in a supernet denoted as $\mathcal{S}(P, W_P)$, where W_P is the weight of the supernet that is shared across all the candidate architectures. Algorithm 1 illustrates the training procedure of our supernet.

3.5 Hardware-Oriented Evolution Search

In our hardware-oriented evolution search for crossover, two random candidate architectures are first picked from the top candidates. Then we uniformly choose one block from them in each layer to generate a new architecture. For mutation, a candidate mutates its depth with probability P_d first. Then it mutates each block with a probability of P_m to produce a new architecture. Newly produced architectures that do not satisfy the constraints will not be added for the next generation. To evaluate the candidates, we perform hardware latency and resource modeling based on the proposed row-wise flexible mixed-precision quantization scheme. The details of the modeling have been discussed in Section 4.4.

3.6 Integration with Knowledge Distillation (KD)

To demonstrate the compatibility of our proposed framework with knowledge distillation (KD) and further improve the accuracy of our supernet, we also integrate KD [15] in our training process. We use the pre-trained RegNetY-32G [34] with 83.6% top-1 accuracy as different teacher models. We also apply the soft distillation method. Soft distillation [15] minimizes the Kullback-Leibler divergence between the softmax of the teacher and the softmax of the student model. The distillation loss is:

$$L_{soft} = (1-\alpha)L_{CE}(\psi(Z_s), y) + \alpha\tau^2 L_{KL}(\psi(\frac{Z_s}{\tau}), \psi(\frac{Z_t}{\tau})), \quad (2)$$

where Z_t and Z_s are the logits of the teacher and student models, respectively. ψ is the softmax function. τ is the temperature for the distillation, α is the coefficient balancing the Kullback–Leibler divergence loss (L_{KL}), and the cross-entropy (L_{CE}) on the ground truth labels y in the distillation.

4 FPGA HARDWARE DESIGN FOR QUASAR-VIT

4.1 Overall Hardware Design for Quasar-ViT

Figure 6 presents the overall hardware architecture of the Quasar-ViT accelerator on the ARM-FPGA platform. Below is how each module in ViT is mapped to the hardware in Figure 6. The most time-consuming MSA and MLP modules are accelerated by our GEMM engine on the FPGA, which is similar to the recent Auto-ViT-Acc work [22]. The lightweight SLS modules right after MSA and MLP layers are also accelerated on the FPGA to avoid time-consuming execution on the ARM CPU. The less time-consuming modules including layer normalization and activation functions (i.e., Softmax or GELU) are executed on the ARM CPU, due to their complex structure for FPGA implementation. The hardware engines on the FPGA and software modules on the ARM CPU exchange data via the shared off-chip memory.

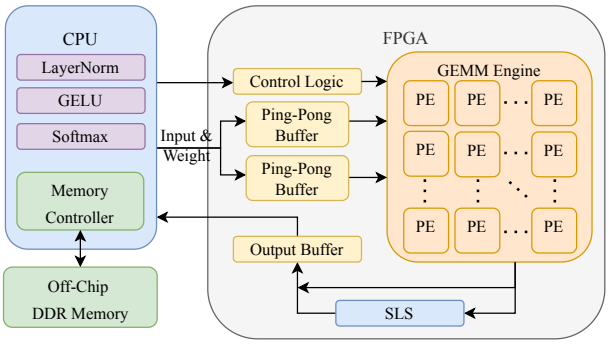

Figure 6: Quasar-ViT hardware architecture.

As previously mentioned, we mainly focus on the most time-consuming GEMM engine design. Due to the limited on-chip memory capacity and computing resource on the FPGA, for each ViT layer (i.e., MSA and MLP), our GEMM engine processes the input, weight, and output data in tiles: a small tile of the input (tokens) and weight of each ViT layer are first loaded from the off-chip DDR memory to the on-chip buffers, then they are processed by the GEMM engine all on-chip. To improve the performance, the double buffering technique is applied again to overlap the off-chip memory accesses and GEMM computation, shown in Figure 6.

Next, we present our design of the basic hardware units in the GEMM engine and the corresponding DSP (digital signal processor) packing optimization, as well as the hardware resource and latency modeling for the tiled GEMM design.

4.2 Unification of Atomic Computation

One major challenge in the FPGA accelerator design is to efficiently support flexible mixed ratios of different bit-width computations across ViT layers. On one hand, putting multiple copies of hardware accelerator designs for each mixed-ratio (i.e., each layer) simultaneously on the FPGA leads to severe hardware resource contention and under-utilization, since layers are executed sequentially. On the other hand, pre-synthesizing multiple copies of hardware accelerator designs for each layer and reconfiguring the FPGA for each layer incurs significant FPGA reconfiguration overhead.

Inspired by the approach proposed in QGTC [52] to support arbitrary bit-width computation for quantized graph neural networks on GPUs, in our FPGA hardware design, we unify the basic processing elements to process 4-bit weight atomic computations and construct the 8-bit weight data computations using two 4-bit weight data operations as such: for multiplication between an N-bit activation value (act_N) and an 8-bit weight value (wgt_8), we derive the corresponding product as:

$$act_N \cdot wgt_8 = act_N \cdot wgt_{h4} << 4 + act_N \cdot wgt_{l4}, \quad (3)$$

where wgt_{h4} and wgt_{l4} represent the higher and lower 4-bit data of wgt_8, respectively. The multiplication result between act_N and wgt_{h4} are left shifted by 4 bits.

Based on this unification, we propose hybrid signed/unsigned DSP packing to handle the 4-bit weight atomic computation.

4.3 Proposed Hybrid Signed/Unsigned DSP Packing

To fully exploit the potential of DSP resources on FPGAs, we pack multiple low-bit multiplications within each DSP block following [57, 58]. Each DSP block (DSP48E2) on the AMD/Xilinx ZCU102 FPGA board could support the computation of P=(A+D)×B, where both A and D are 27-bit operands, B is an 18-bit operand, and P is the 45-bit output. In our study, we explore the following two DSP packing schemes and discuss their design trade-offs. The activation bit-width N is set to 6 to fully exploit the DSP for computation.

- **Packing factor 3 (3 weights sharing 1 activation).** In Figure 7 (a), three 4×6-bit multiplications are packed into a single DSP block, by holding one 6-bit signed activation in port B and three 4-bit weight values in port D. To pack three weights into a single 27-bit port D, look-up tables (LUTs) are utilized to first combine two weights and then integrate them with the third weight data. With this DSP packing scheme, for the W4A6 (i.e., 4-bit weight and 6-bit activation) computation, we could pack three 4-bit weights that share the same activation. And for the W8A6 computation, we could use two DSPs to process the upper and lower 4-bit

of three 8-bit weights in parallel. Note that after the 8-bit weights are decomposed into two 4-bit weights, only the upper 4-bit weights contain the sign bits and should be sign-extended (required by DSP for output correctness [57, 58]), the lower 4-bit weights should be treated as unsigned values.

- **Packing factor 4 (2 weights sharing 2 activations).** In Figure 7 (b), four 4×6-bit multiplications are packed into a single DSP block, by holding two 6-bit signed activation values in port D and two 4-bit weights in port B. It is worth mentioning the port placement of the activation and weight values are swapped from the packing factor 3 scheme, to increase the packing strength. With this DSP packing scheme, for the W4A6 computation, we could pack a pair of two 4-bit weights that share the same pair of activation values. And for the W8A6 computation, a similar technique as the previous packing scheme is used to separately handle the upper and lower 4-bit values of an 8-bit weight. Again for the two decomposed 4-bit weights, only the upper 4-bit weights contain the sign bit and should be sign-extended (required by DSP for output correctness [57, 58]), but the lower 4-bit weights should be treated as unsigned values.

4.4 Hardware Resource and Latency Modeling

Here, we present our hardware resource and latency modeling used in the hardware-oriented evolution search.

Table 3: Notations for Quasar-ViT accelerator

Notation	Description
M (N)	Number of output (input) channels
F	Number of token sequences
T_n	Tiling size for data in input channel dimension for each head
T_m	Tiling size for data in output channel dimension
N_h	Total number of heads
P_F	Parallel factor along the number of tokens
D_{act}	Number of data packed as one for activations
D_{wgt}	Number of data packed as one for weights
A_{in} (A_{out}, A_{wgt})	Number of AXI ports used for data transfer of input (output, weight) tile
L_{in} (L_{wgt}, L_{out}, L_{cmpt})	Number of clock cycles for input transfer (weight transfer, output transfer, computation) for a tile
S_{dsp} (S_{lut})	Available number of DSPs (LUTs) on FPGA
C_{dsp}	DSP cost for each MAC operation
C_{lut}	LUT cost for each MAC operation
C_{lut}^{dsp}	Number of LUTs used by a multiplication executed on DSPs
N_{dsp}	Number of multiplication executed on DSPs
N_{lut}	Number of multiplication executed on LUTs
N_{tot}	The total number of multiplication on FPGA
γ_{dsp} (γ_{lut})	DSP (LUT) utilization threshold
f	FPGA accelerator frequency
FPS	Frames per second

4.4.1 Resource Modeling. To help guide the neural architecture search, we provide details of the resource and latency models of our FPGA accelerator design (mainly the GEMM engine). Table 3 lists the notations used in our models. We design our FPGA accelerator to fully leverage the available FPGA computing resources (i.e., DSPs and LUTs), on-chip memory (i.e., BRAMs), and off-chip memory bandwidth. To fully exploit the computing capability with the available hardware resources, We maximize the total number of parallel basic hardware compute units using both DSPs (i.e., N_{dsp}) and LUTs (i.e., N_{lut}) for the datapath of our accelerator as

$$N_{tot} = \text{maximize}\left\{N_{dsp} + N_{lut}\right\}, \tag{4}$$

while satisfying the following resource constraints

$$N_{dsp} \cdot C_{dsp} \leq S_{dsp} \cdot \gamma_{dsp}, \tag{5}$$

$$N_{lut} \cdot C_{lut} + N_{dsp} \cdot C_{lut}^{dsp} \leq S_{lut} \cdot \gamma_{lut}, \tag{6}$$

where constraints 5 and 6 bound the DSP and LUT utilization to be under the allowable thresholds, i.e., γ_{dsp} and γ_{lut} with the total resource amounts denoted by S_{dsp} and S_{lut}. The hardware costs of each multiplication implementation are denoted as C_{lut} and C_{lut}^{dsp}.

In order to choose the best implementation method for the basic hardware units of our design, we characterize the FPGA resource consumption using Xilinx Vivado 2020.1 [59] for the three cases in Table 4, i.e., (a) multiplications executed on DSPs with packing factor of 3, (b) multiplications executed on DSPs with packing factor of 4, and (c) multiplications executed purely using LUTs. As observed in Table 4, we derive the hardware costs of each multiplication implementation, particularly, the LUT costs for pure-LUT-based and DSP-based methods correspond to C_{lut} and C_{lut}^{dsp}. Note that the DSP-based approach also consumes LUTs, due to data packing on the input operands and output data construction operations, such as bit shifting and data accumulation. Regarding the efficiency lost by decomposing 8-bit to 4-bit, for the composed W8A6 computation, on average, we can achieve one computation with 25.8 LUTs and 0.5 DSP or purely 66.7 LUTs. In contrast, direct W8A6 computation used in [40] (i.e., packing two W8A6 operations within one DSP method) requires 21.5 LUTs and 0.5 DSP or purely 62.2 LUTs. Since most of the weights are in 4-bit, using decomposing does not affect the overall performance much by a slight increase in the LUT utilization. In terms of the efficiency of DSP packing, a single DSP can pack four W4A6 operations at most theoretically, which is achieved by our approach.

For the LUT usage, shown in Table 4, we have:

$$C_{lut}^{dsp,pack3} < C_{lut}^{dsp,pack4} < C_{lut}. \tag{7}$$

In the final implementation, regarding which DSP packing scheme to use and whether to use the pure-LUT-based method for the basic hardware compute units, there are several situations according to the available FPGA resources.

Situation-1: When S_{lut} is limited and insufficient to hold the LUT consumption of full utilization of DSP packing with a factor of 3, denoted as:

$$S_{lut} \cdot \gamma_{lut} \leq 3 \cdot S_{dsp} \cdot \gamma_{dsp} \cdot C_{lut}^{dsp,pack3}. \tag{8}$$

In this case, to fully utilize the computation resources, we directly allocate DSP-based computations with a packing factor of 3 as much as possible until we reach the LUT resource limit.

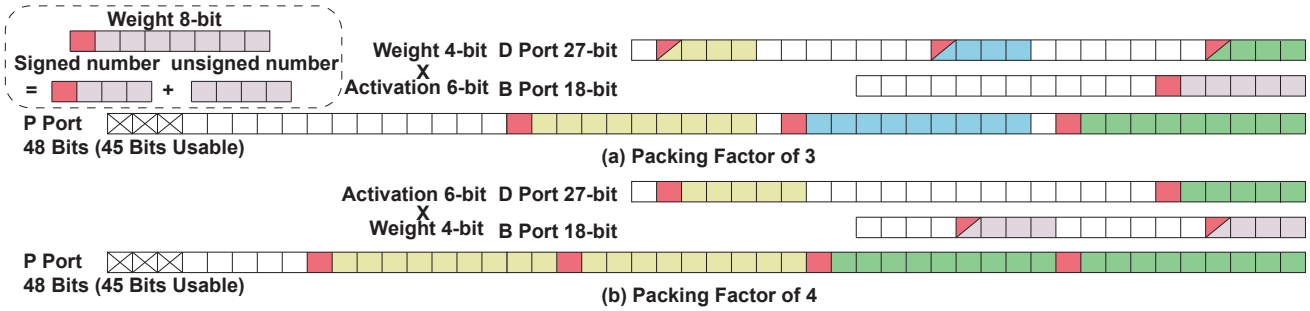

Figure 7: Illustration of DSP multiplication packing schemes for (a) 3 weights sharing 1 activation with packing factor of 3, and (b) 2 weights sharing 2 activations with packing factor of 4.

Table 4: FPGA resource consumption for a single DSP-based or LUT-based basic hardware compute unit. W4A6 denotes 4-bit weight and 6-bit activation; W8A6 denotes 8-bit weight and 6-bit activation.

		pure LUT-based	DSP-based	
			packing factor 3	packing factor 4
W4A6	C_{LUT}	33.3	10.9	12.9
	C_{DSP}	0	0.33	0.25
W8A6	C_{LUT}	66.7	21.9	25.8
	C_{DSP}	0	0.67	0.5
W8A6 (direct)	C_{LUT}	62.2	21.5	
	C_{DSP}	0	0.5	

Situation-2: When S_{lut} is enough to hold all the LUT consumption from DSP packing with the factor of 4, satisfying:

$$4 \cdot S_{dsp} \cdot \gamma_{dsp} \cdot C_{lut}^{dsp,pack4} \leq S_{lut} \cdot \gamma_{lut}. \tag{9}$$

Based on Eq. 4, 5, 6 and 9, we can conclude that using DSP packing with the factor of 4 is more efficient if it meets the following condition:

$$(4 \cdot C_{lut}^{dsp,pack4} - 3 \cdot C_{lut}^{dsp,pack3}) \cdot S_{dsp} \cdot \gamma_{dsp} \leq S_{lut} \cdot \gamma_{lut} \cdot C_{lut}. \tag{10}$$

If it does not satisfy Eq. 10, we perform DSP-based computations with a packing factor of 3. Besides that, for the remaining available LUT resources, pure-LUT-based computation is also conducted in both cases.

Situation-3: When S_{lut} is between the LUT demands by fully deploying DSP computations with packing factors of 3 and 4, satisfying:

$$3 \cdot S_{dsp} \cdot \gamma_{dsp} \cdot C_{lut}^{dsp,pack3} \leq S_{lut} \cdot \gamma_{lut} \leq 4 \cdot S_{dsp} \cdot \gamma_{dsp} \cdot C_{lut}^{dsp,pack4}. \tag{11}$$

Based on Eq. 4, 5, 6, and 11, we can conclude that using DSP packing with the factor of 4 is more efficient if it meets the following condition:

$$\frac{S_{lut} \cdot \gamma_{lut} + 3 \cdot S_{dsp} \cdot \gamma_{dsp} \cdot (C_{lut} - C_{lut}^{dsp,pack3})}{C_{lut}} \leq \frac{S_{lut} \cdot \gamma_{lut}}{C_{lut}^{dsp,pack4}}, \tag{12}$$

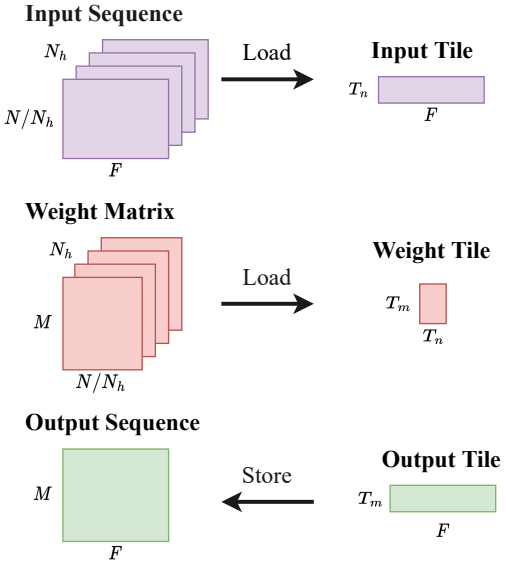

Figure 8: Data tiling in ViT computations.

In this case, we conduct only DSP-based computations with a packing factor of 4. Otherwise, if it does not satisfy Eq. 12, we perform DSP-based computations with a packing factor of 3 and pure LUT-based computation for the remaining available LUT resource.

4.4.2 Latency Modeling. As discussed in the Hardware Design Section, our GEMM hardware engine accelerates the major MSA and MLP modules and processes their input, weight, and output data in tiles, as shown in Figure 8. our GEMM hardware engine accelerates the major MSA and MLP modules and processes their input, weight, and output data in tiles, as shown in Figure 8. Each MSA can be seen as multiple parallel matrix multiplications. The accelerator is designed to process parallel computations within each head. This input channel splitting is also done for fully connected (FC) layers, each containing one matrix multiplication for compatibility, and the results need to be accumulated from all the input channels in all the heads. Furthermore, the computing engine exploits fine-grained data and operation parallelisms and can process $T_m \cdot T_n$ multiply-accumulate (MAC) operations in parallel.

We model the inference latency of our hardware design based on the number of clock cycles. For a layer i in ViT, since data packing can be used to transfer multiple (i.e., D_{act} or D_{wgt}) values at the same time in each AXI port, the clock cycles for loading one input/weight tile and storing one output tile, are calculated as:

$$\begin{aligned} L_{\text{in}} &= \left\lceil \frac{T_n}{D_{act}} \right\rceil \cdot \left\lceil \frac{F}{A_{\text{in}}} \right\rceil, \\ L_{\text{wgt}} &= \left\lceil \frac{T_n}{D_{wgt}} \right\rceil \cdot \left\lceil \frac{T_m}{A_{\text{wgt}}} \right\rceil, \\ L_{\text{out}} &= \left\lceil \frac{T_m}{D_{act}} \right\rceil \cdot \left\lceil \frac{F}{A_{\text{out}}} \right\rceil, \end{aligned} \tag{13}$$

where L_{in}, L_{wgt} and L_{out} indicate the number of the clock cycles of input, weight, and output transfer for one corresponding tile, respectively.

The clock cycle number to compute one tile is

$$L_{\text{cmpt}} = \max\left\{ \left\lceil \frac{F}{P_F} \right\rceil, \frac{T_n \cdot T_m \cdot F}{N_{tot}} \right\}, \tag{14}$$

where the first term is calculated by the latency model. The total number of multiplications needed to compute one tile of matrix multiplication is $T_n \cdot T_m \cdot F$. Each tile has two levels of parallelism: one is along the T_n and T_m dimension and the parallel factor is $T_n \cdot T_m$ (i.e., fully parallel); and the other is along the F dimension and the parallel factor is P_F. Therefore, the first term is calculated as $\left\lceil \frac{F}{P_F} \right\rceil$. The second term is limited by the resource constraint, where we can support at most N_{tot} parallel multipliers (Eq. 4) due to the resource constraint.

With the double buffers overlapping the data loading and computation of the tiles, the overall clock cycle number for processing one tile is $L_1 = \max\{L_{\text{in}}, L_{\text{wgt}}, L_{\text{cmpt}}\}$.

To obtain the accumulation of output results, this process is performed multiple times (i.e., processing $\left\lceil \frac{N}{T_n} \right\rceil$ input tiles). The clock cycle number for calculating the whole output tile is $L_2 = \max\left\{L_1 \cdot \left\lceil \frac{N}{T_n} \right\rceil + L_{\text{cmpt}}, L_{\text{out}}\right\}$.

Since there are $\left\lceil \frac{M}{T_m} \right\rceil$ number of output tiles, the total number of clock cycles for a ViT layer i is described by

$$L^i_{\text{tot}} = \left\lceil \frac{M}{T_m} \right\rceil \cdot L_2 + L_{\text{out}}. \tag{15}$$

Under a working frequency f, the FPS is calculated as:

$$FPS = \frac{f}{\sum_i L^i_{\text{tot}}}. \tag{16}$$

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

Our supernet training process takes 700 epochs with a batch size of 2048. The learning rate is set to 5×10^{-4} initially and decayed with a cosine annealing schedule. The AdamW [28] optimizer is used with the epsilon value of $1e^{-8}$ and weight decay of 0.05. Additional training optimizations, such as warmup and label smoothing are performed during training. The number of warmup epochs and label smoothing factor are set as 20 and 0.1, respectively. After supernet training, we perform the hardware-oriented evolution search, without subnet retraining.

Our training is conducted on 8 NVIDIA Ampere A100 GPUs with CUDA 11.0 and PyTorch 1.7 frameworks on the Ubuntu operating system. To test the effectiveness of our framework, we also implement Quasar-ViT framework on the Xilinx ZCU102 embedded FPGA platform with quad-core ARM Cortex-A53 and XCZU9EG FPGA chip. The FPGA working frequency is set to 150 MHz for all the designs implemented via Xilinx Vitis and Vitis HLS 2020.1.

5.2 Accuracy Results

Here we analyze the accuracy results of our Quasar-ViT framework. The weight precision is mixed with 4-bit and 8-bit, as mentioned earlier, and the activation bit-width is determined by the hardware feature. Without loss of generality, we use activation of 8-bit to evaluate the accuracy in the ablation study of knowledge distillation and supernet layer scaling.

5.2.1 Ablation Study of Knowledge Distillation and Supernet Layer Scaling. To evaluate the compatibility of knowledge distillation and our proposed SLS, we conduct an ablation study on both of them. Without loss of generality and to prevent interference from different model sizes and different quantization mixed ratios from the searched subnet, we unify the search constraint with pure W8A8 (8-bit for both weight and activation) quantization implementation.

As shown in Table 6, we conduct four different settings of Quasar-ViT with the unified model size and quantization scheme. The accuracy of the four cases is 74.1% (w/o distillation and SLS), 75.6% (only distillation), 74.9% (only SLS), and 76.1% (with both of them), respectively. Knowledge distillation and SLS strategies are orthogonal to each other, and both improve the model accuracy. Using them together provides a better result. Given the observed effectiveness of our proposed SLS strategy and the seamless compatibility of our framework with knowledge distillation, we opt to incorporate both strategies in our following experiments. Here we also quantize the baseline DeiT-T [42] and compare it with our method without SLS and distillation. Even without SLS and distillation, our quantization NAS approach achieves a much better model accuracy than the full-precision and the quantized (W8A8) DeiT-T models.

5.2.2 Ablation Study of Mixed-Ratio and Quantization Scheme. To assess the efficacy of our row-wise flexible mixed-precision quantization scheme, we conducted an ablation study examining both the quantization scheme itself and the 8-bit mixed ratio, as outlined in Table 7. Since Quasar-ViT automatically searches the mixed ratios, here we pick up the best model from the search stage for different mixed ratios and compare them with the counterparts under the fixed row-wise mixed quantization scheme. The results indicate a consistent improvement in accuracy across different 8-bit mixed-ratio levels with our flexible mixed scheme, underscoring the efficiency of our proposed quantization scheme.

5.2.3 Overall Accuracy Results. Table 5 compares representative ViT-based works with our proposed Quasar-ViT. Since many ViT-based works do not incorporate model quantization, we also consider the bit-width in the model size and the equivalent number of total bit operations (BOPs).

Table 5: Comparison of representative ViT-based works with our proposed Quasar-ViT.

Model	Quantization Mixed Type	#Params (M)	Model Size (MB)	MACs (G)	MAC Bit-width Weight	MAC Bit-width Activation	Equivalent BOPs (G)	Top-1 Accuracy (%)
DeiT-T [42]	-	5.7	22.8	1.3	32	32	1.3×10^3	72.2
T2T-T [63]	-	4.3	17.2	1.1	32	32	1.1×10^3	71.7
PiT-T [14]	-	10.6	42.4	1.4	32	32	1.4×10^3	72.4
LocalViT-T [21]	-	5.9	23.6	1.3	32	32	1.3×10^3	74.8
FQ-ViT [24]	Model-wise	5.7	5.7	1.3	8	8	80.6	71.6
Q-ViT [20]	Head-wise	5.7	-	-	4-8	4-8	-	72.8
QUASAR-S (ours)	Row-wise	5.9	**4.1**	1.4	4 & 8	6	**45.6**	**74.9**
PVT [51]	-	24.5	98.0	3.8	32	32	3.9×10^3	79.8
DeiT-S [42]	-	22.9	91.6	4.6	32	32	4.8×10^3	79.9
Swin-T [26]	-	28	112.0	4.5	32	32	4.7×10^3	81.2
BossNAS [18]	-	-	-	3.4	32	32	3.5×10^3	80.5
PTQ [27]	Layer-wise	22.9	16.6	4.6	6-10	6-10	-	75.1
FQ-ViT [24]	Model-wise	22.9	22.9	4.6	8	8	294.4	79.1
Q-ViT [20]	Head-wise	22.9	-	-	4-8	4-8	-	80.1
QUASAR-L1 (ours)	Row-wise	14.7	**9.8**	3.2	4 & 8	6	**103.8**	**78.6**
QUASAR-L2 (ours)	Row-wise	22.6	**15.8**	4.8	4 & 8	6	**163.2**	**80.4**

Table 6: Comparison between different settings of QUASAR-Small with and without knowledge distillation (KD) and supernet layer scaling (SLS) on ImageNet dataset.

Model	Setting KD	Setting SLS	#Params (M)	Quantization Scheme	Top-1 Acc. (%)
DeiT-T [42]	No	No	5.7	W32A32	72.2
DeiT-T (quant)	No	No	5.7	W8A8	71.5
Ours	No	No	5.9	W8A8	74.1
Ours	Yes	No	5.9	W8A8	75.6
Ours	No	Yes	5.9	W8A8	74.9
Ours	Yes	Yes	5.9	W8A8	76.1

Table 7: Comparison between different settings of 8-bit quantization mixed-ratio and quantization schemes.

Scheme	Model Size (MB)	8-bit mixed-ratio (%)	Acc. (%)
Fixed row-wise	3.6	23	73.5
Flexible row-wise	3.6	23	74.2
Fixed row-wise	4.1	39	74.1
Flexible row-wise	4.1	39	74.9

As shown in Table 5, we present three Quasar models for different accuracy levels: QUASAR-S searched within the QUASAR-Small supernet, and QUASAR-L1, QUASAR-L2 searched within the QUASAR-Large supernet.

Our QUASAR-S achieves 74.9% top-1 accuracy with only 4.1 MB model size and 1.4 GMACs, Compared with the representative ViT-based model LocalViT-T [21] under a similar accuracy, our model size is only 17.4% of that in LocalViT-T; although the GMACs numbers are similar, our MAC unit is much more hardware efficient as it is for a 4-bit/8-bit weight and a 6-bit activation instead of a 32-bit floating-point MAC unit in LocalViT-T. For a higher model accuracy level, our QUASAR-L1 and QUASAR-L2 achieve 78.5% and 80.4% top-1 accuracy, respectively. Among them, QUASAR-L2 only has a 15.8 MB model size with a computation volume of 4.8 GMACs, which obtains the smallest BOPs with a similar level of accuracy compared with other baselines. Specifically, compared with PTQ [27] (16.6 MB, top-1 75.1%), QUASAR-L2 achieves a similar model size and GMACs with 5.2% higher accuracy. Compared with ViT NAS framework BossNAS [18], we additionally achieve low-bit quantization with a much smaller BOPS and similar accuracy. Compared with quantization-aware training framework Q-ViT [20] using multiple quantization bit-widths in the range of 4 to 8, which incurs inefficiency and hardware under-utilization, our results show better accuracy with a more unified and hardware-friendly quantization scheme.

Table 8: Comparisons of FPGA implementations for ViTs on ImageNet, including DeiT-S and Auto-ViT-Acc from [22], and our Quasar-ViT, all running at 150MHz on the same AMD/Xilinx ZCU102 embedded FPGA platform.

	DeiT-S	Auto-ViT -Acc	Quasar-ViT L2	Quasar-ViT L1	Quasar-ViT S
Quant.	No	Mixed Scheme	Mixed Precision	Mixed Precision	Mixed Precision
Weight	32	4 & 8	4 & 8	4 & 8	4 & 8
Act.	32	8	6	6	6
Top-1 Acc.	79.9	78.7	80.4	78.6	74.9
kLUT	47%	67%	66%	66%	65%
FF	-	-	31%	31%	30%
DSP	69%	62%	69%	69%	69%
BRAM	-	-	44%	44%	44%

5.3 Comparison of Hardware Results on FPGA

We implement a proof-of-concept hardware accelerator for our Quasar-ViT on the AMD/Xilinx ZCU102 embedded FPGA platform. We also compare our results to Auto-ViT-Acc [22], the state-of-the-art FPGA accelerator for ViT with mixed-scheme quantization (without NAS). We retrieve the hardware results for Auto-ViT-Acc (which is quantized from DeiT-S) and the original DeiT-S on the same Xilinx ZCU102 FPGA platform from [22].

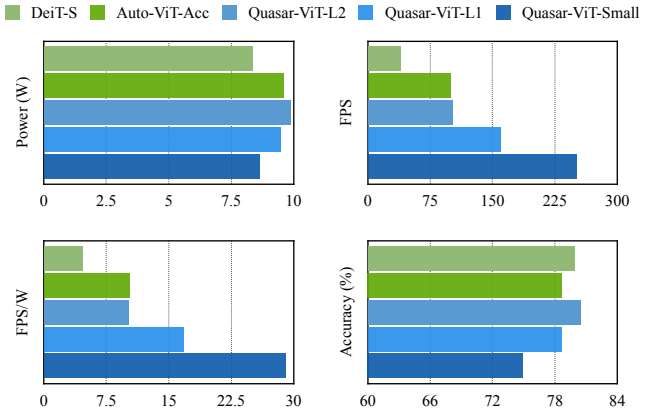

Figure 9: Comparisons between DeiT-S, Auto-ViT-Acc from [22], and our Quasar-ViT.

As shown in Table **??** and Figure 9, our approach consistently outperforms the previous work. Specifically, compared with DeiT-S [42], our QUASAR-L2 achieves 2.6× higher inference frames per second (FPS) with 0.5% better accuracy. Compared with Auto-ViT-Acc [22], our QUASAR-L1 achieves 1.6× higher FPS (159.6) with a similar model accuracy level, and our QUASAR-L2 achieves a similar level of FPS with 1.7% better top-1 accuracy.

The improvement in model accuracy and inference performance within our framework is attributed to two key factors. Firstly, our approach involves the training and search for a customized network architecture, specifically tailored for both the mixed-precision quantization schemes and the targeted inference latency. This strategy enhances adaptability and efficiency, surpassing the achievements of previous methodologies.

Secondly, our novel supernet training algorithm, coupled with the proposed hybrid DSP packing design, allows for distinct quantization mixed ratios across various model layers. This fine-grained model achieves better flexibility than the previous approaches, unleashing the full potential of mixed-precision quantization.

With regard to the efficiency of our hardware accelerator design, the performance is mainly limited by DSP, LUT, and off-chip memory bandwidth. On par with Auto-ViT-Acc [22], our design achieves 150MHz frequency with about 66% usage of LUTs and 69% DSPs without timing violations. Note a typical FPGA design usually utilizes approximately 60% to 70% of the available FPGA resources; otherwise, it may fail during the placement and routing phase due to congestion or result in a lower operating frequency. Without considering the timing violation, the maximum theoretical expected performance is based on the 100% utilization ratio for both DSP, LUTs, and bandwidth, which can achieve about 1.47x of our reached FPS for the same model.

5.4 Other Transformer-based Model Accuracy Results

To demonstrate the scalability and versatility of our methods, we applied them across various datasets and applications, notably deploying them on a large language model (LLM). This choice is motivated by two key factors. Firstly, LLM shares a transformer-based architecture similar to that of Vision Transformer (ViT), aligning with the framework that we propose. Secondly, LLM is frequently integrated with ViT in text-to-image/video applications, making it an ideal candidate to showcase the scalability of our approach across both models and its potential for real-world applications.

Our comparative analysis, presented in Table 9, utilizes the renowned LLM model, LLaMA, as the foundation for our supernet. We juxtapose our optimized results with those of LLaMA-7B [44] on the commonly used WikiText-2 dataset for LLMs, with perplexity score (PPL) serving as the evaluation metric, where lower scores indicate superior performance. According to the comparison results, our method shows a constant pattern, achieving a similar level of PPL with a much smaller model size.

Table 9: Result comparison on large language model.

Model	Model Size (GB)	W # Bits	A # Bits	PPL
LLaMA-7B [44]	26.8	FP32	FP32	5.68
Ours	6.7	INT8	INT8	5.73
Ours	4.8	INT4 & 8	INT8	5.91

5.5 Training Cost Comparison

Prior co-design frameworks, such as APQ [50], have also delved into the integration of neural architecture search (NAS) and quantization techniques. Please note that APQ is based on the convolutional neural network (CNN) and BitFusion platform [50]. To the best of our knowledge, we compare our Quasar-ViT models (both small and large variants) and the APQ result. As detailed in Table 10, our approach demonstrates superior FPS performance while maintaining comparable or even higher model accuracy, achieved at a reduced training cost. Compared with the 2,400 GPU hours training cost of APQ [50], our approach only consumes 768 and 1,344 GPU hours for the small and large versions of Quasar-ViT, respectively. Our training setting has been illustrated in Section 5.1.

Table 10: Training cost and accuracy comparison with other NAS and quantization co-design.

Method	Training cost (GPU hours)	FPS	Acc.(%)
APQ [50]	2400	82.2	75.1
QUASAR-S	768	101.5	74.9
QUASAR-L	1344	251.6	80.4

6 CONCLUSION

In this work, we propose Quasar-ViT, a hardware-oriented quantization-aware network architecture search framework to enable efficient ViT deployment on resource-constrained edge devices. First, we proposed hardware-friendly quantization techniques including flexible row-wise mixed-precision quantization scheme and intra-layer mixed-precision weight entanglement in architecture search towards high accuracy and low training cost for efficient implementation. Second, we propose 4-bit weight atomic computation and hybrid signed/unsigned DSP packing for FPGA implementation, then incorporate latency/resource modeling to enable the hardware-oriented architecture search. Third, we extend the supernet layer scaling technique to further improve the training convergence and supernet accuracy. We also demonstrate the compatibility of our proposed framework with knowledge distillation during supernet training. Finally, we developed an efficient hardware-oriented search algorithm—integrated with hardware latency and resource modeling—to search the efficient subnet with high accuracy under a given inference latency target and implemented the searched model on real FPGA hardware for validation. From the experiment evaluation results, our approach achieves 101.5, 159.6, and 251.6 FPS on the AMD/Xilinx ZCU102 FPGA board with 80.4%, 78.6%, and 74.9% top-1 accuracy for ImageNet, respectively, consistently outperforming prior works.

ACKNOWLEDGMENTS

This work was supported in part by NSERC Discovery Grant RGPIN-2019-04613, DGECR-2019-00120, Alliance Grant ALLRP-552042-2020; CFI John R. Evans Leaders Fund; BC Knowledge Development Fund; Army Research Office/Army Research Laboratory via grant W911-NF-20-1-0167 to Northeastern University; National Science Foundation CCF-1937500, CNS-1909172, and IIS-2310254. No external funding support to CD and MS for this project.

REFERENCES

[1] Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jing Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. BinaryBERT: Pushing the Limit of BERT Quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2017. Designing Neural Network Architectures using Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and simplifying one-shot architecture search. In *Proceedings of the International Conference on Machine Learning (ICML)*. 550–559.

[4] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[5] Sung-En Chang, Yanyu Li, Mengshu Sun, Runbin Shi, Hayden K-H So, Xuehai Qian, Yanzhi Wang, and Xue Lin. 2021. Mix and Match: A novel FPGA-centric deep neural network quantization framework. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. 208–220.

[6] Minghao Chen, Houwen Peng, Jianlong Fu, and Haibin Ling. 2021. Autoformer: Searching transformers for visual recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 12270–12280.

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. Pact: Parameterized clipping activation for quantized neural networks. *arXiv:1805.06085* (2018).

[8] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in Neural Information Processing Systems (NeurIPS)*. 3123–3131.

[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2019. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=YicbFdNTTy

[11] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single path one-shot neural architecture search with uniform sampling. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 544–560.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778.

[13] Zhezhi He and Deliang Fan. 2019. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. 2021. Rethinking spatial dimensions of vision transformers. *arXiv:2103.16302* (2021).

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv:1503.02531* (2015).

[16] Intel. 2024. Intel® Xeon® Silver 4214 Processor. https://ark.intel.com/content/www/us/en/ark/products/193385/intel-xeon-silver-4214-processor-16-5m-cache-2-20-ghz.html. Last accessed Jan. 18, 2024.

[17] Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. 2018. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[18] Changlin Li, Tao Tang, Guangrun Wang, Jiefeng Peng, Bing Wang, Xiaodan Liang, and Xiaojun Chang. 2021. Bossnas: Exploring hybrid cnn-transformers with block-wisely self-supervised neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 12281–12291.

[19] Yuhang Li, Xin Dong, and Wei Wang. 2020. Additive Powers-of-Two Quantization: An Efficient Non-uniform Discretization for Neural Networks. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=BkgXT24tDS

[20] Yanjing Li, Sheng Xu, Baochang Zhang, Xianbin Cao, Peng Gao, and Guodong Guo. 2022. Q-ViT: Accurate and Fully Quantized Low-bit Vision Transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*. https://openreview.net/forum?id=fU-m9kQe0ke

[21] Yawei Li, Kai Zhang, Jiezhang Cao, Radu Timofte, and Luc Van Gool. 2021. LocalViT: Bringing Locality to Vision Transformers. arXiv:2104.05707 [cs.CV]

[22] Zhengang Li, Mengshu Sun, Alec Lu, Haoyu Ma, Geng Yuan, Yanyue Xie, Hao Tang, Yanyu Li, Miriam Leeser, Zhangyang Wang, Xue Lin, and Zhenman Fang. 2022. Auto-vit-acc: An fpga-aware automatic acceleration framework for vision transformer with mixed-scheme quantization. In *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 109–116.

[23] Zhengang Li, Geng Yuan, Wei Niu, Pu Zhao, Yanyu Li, Yuxuan Cai, Xuan Shen, Zheng Zhan, Zhenglun Kong, Qing Jin, et al. 2021. NPAS: A Compiler-aware Framework of Unified Network Pruning and Architecture Search for Beyond Real-Time Mobile Acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 14255–14266.

[24] Yang Lin, Tianyu Zhang, Peiqin Sun, Zheng Li, and Shuchang Zhou. 2022. FQ-ViT: Post-Training Quantization for Fully Quantized Vision Transformer. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*. 1173–1179.

[25] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*. 19–34.

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

[27] Zhenhua Liu, Yunhe Wang, Kai Han, Siwei Ma, and Wen Gao. 2021. Post-Training Quantization for Vision Transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[28] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=Bkg6RiCqY7

[29] Qian Lou, Feng Guo, Minje Kim, Lantao Liu, and Lei Jiang. 2019. AutoQ: Automated Kernel-Wise Neural Network Quantization. In *International Conference on Learning Representations (ICLR)*.

[30] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, et al. 2019. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 293–312.

[31] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. 2021. IA-RED2: Interpretability-Aware Redundancy Reduction for Vision Transformers. *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021), 24898–24911.

[32] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *Proceedings of the International Conference on Machine Learning (ICML)*. 4095–4104.

[33] PyTorch. 2024. PYTORCH PROFILER. https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html. Last accessed Jan. 18, 2024.

[34] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10428–10436.

[35] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. 2021. Do Vision Transformers See Like Convolutional Neural Networks?. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[36] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 525–542.

[37] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4780–4789.

[38] Manas Sahni, Shreya Varshini, Alind Khare, and Alexey Tumanov. 2021. CompOFA: Compound Once-For-All Networks for Faster Multi-Platform Deployment. *arXiv preprint arXiv:2104.12642* (2021).

[39] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34:05. 8815–8821.

[40] Mengshu Sun, Zhengang Li, Alec Lu, Yanyu Li, Sung-En Chang, Xiaolong Ma, Xue Lin, and Zhenman Fang. 2022. Film-qnn: Efficient fpga acceleration of deep neural networks with intra-layer, mixed-precision quantization. In *Proceedings of the 2022 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 134–145.

[41] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2820–2828.

[42] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. 2021. Training data-efficient image transformers & distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 10347–10357.

[43] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. 2021. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 32–42.

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[45] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. 2020. Mixed Precision DNNs: All you need is a good parametrization. In *International Conference on Learning Representations (ICLR)*.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*. 5998–6008.

[47] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. 2021. Attentivenas: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 6418–6427.

[48] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. 2020. Hat: Hardware-aware transformers for efficient natural language processing. *arXiv:2005.14187* (2020).

[49] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. HAQ: Hardware-Aware Automated Quantization with Mixed Precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[50] Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. 2020. Apq: Joint search for network architecture, pruning and quantization policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2078–2087.

[51] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[52] Yuke Wang, Boyuan Feng, and Yufei Ding. 2022. QGTC: accelerating quantized graph neural networks via GPU tensor core. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 107–119.

[53] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10734–10742.

[54] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. 2018. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv:1812.00090* (2018).

[55] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. 2021. Cvt: Introducing convolutions to vision transformers. *arXiv preprint arXiv:2103.15808* (2021).

[56] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[57] Xilinx. 2017. Deep Learning with INT8 Optimization on Xilinx Devices. https://docs.xilinx.com/v/u/en-US/wp486-deep-learning-int8. Last accessed Mar. 28, 2022.

[58] Xilinx. 2020. Convolutional Neural Network with INT4 Optimization on Xilinx Devices. https://docs.xilinx.com/v/u/en-US/wp521-4bit-optimization. Last accessed Mar. 28, 2022.

[59] Xilinx. 2020. Vivado Design Suite. https://www.xilinx.com/products/design-tools/vivado.html. Last accessed Aug. 28, 2022.

[60] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. 2020. GreedyNAS: Towards Fast One-Shot NAS with Greedy Supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 1999–2008.

[61] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. 2020. Bignas: Scaling up neural architecture search with big single-stage models. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 702–717.

[62] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. 2021. Tokens-to-Token ViT: Training Vision Transformers From Scratch on ImageNet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 558–567.

[63] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Zi-Hang Jiang, Francis E.H. Tay, Jiashi Feng, and Shuicheng Yan. 2021. Tokens-to-Token ViT: Training Vision Transformers From Scratch on ImageNet. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 558–567.

[64] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. 2019. Q8bert: Quantized 8bit bert. In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*. 36–39.

[65] Wei Zhang, Lu Hou, Yichun Yin, Lifeng Shang, Xiao Chen, Xin Jiang, and Qun Liu. 2020. Ternarybert: Distillation-aware ultra-low bit bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[66] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. 2021. Few-shot neural architecture search. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 12707–12718.

[67] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2018. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2423–2432.

[68] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv:1606.06160* (2016).

[69] Barret Zoph and Quoc V. Le. 2017. Neural Architecture Search with Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

[70] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8697–8710.