USDN: A Unified Sample-wise Dynamic Network with Mixed-Precision and Early-Exit

Ji-Ye Jeon[1] Xuan Truong Nguyen[1] Soojung Ryu[2] Hyuk-Jae Lee[1]
[1]Seoul National University [2]SK Telecom
{jyj0805, truongnx, hyuk_jae_lee}@capp.snu.ac.kr, s.ryu@sk.com

Abstract

*To reduce computation in deep neural network inference, a promising approach is to design a network with multiple internal classifiers (ICs) and adaptively select an execution path based on the complexity of a given input. However, quantizing an input-adaptive network, a must-do task for network deployment on edge devices, is a non-trivial task due to jointly allocating its computation budget along with network layers and IC locations. In this paper, we propose **U**nified **S**ample-wise **D**ynamic **N**etwork (USDN) with a mixed-precision and early-exit framework that obtains both the optimal location of ICs and layer-wise bit configurations under a given computation budget. The proposed USDN comprises multiple groups of layers, with each group representing a varying degree of complexity for input samples. Experimental results demonstrate that our approach reduces computational cost of the previous work by 12.78% while achieving higher accuracy on ImageNet dataset.*

1. Introduction

Quantization is a widely-used technique to accelerate deep neural network (DNN) inference. Conventional quantization methods [5, 8, 26] allocate the same number of bits to all layers in the network. In recent years, mixed-precision quantization methods [2, 9, 18, 19] have further reduced the computational cost of DNN inference by assigning optimal bit-widths to each convolutional layer, balancing accuracy and computational cost trade-offs. However, both uniform quantization and mixed-precision methods are typically trained and applied uniformly across an entire dataset. This can lead to *computational redundancy* due to variations in the difficulty of individual samples. For instance, an easy-to-classify sample might require less computational effort for accurate prediction than a difficult-to-classify one, meaning potential computational savings remain unexploited during inference.

Dynamic networks, as opposed to static ones, can adapt

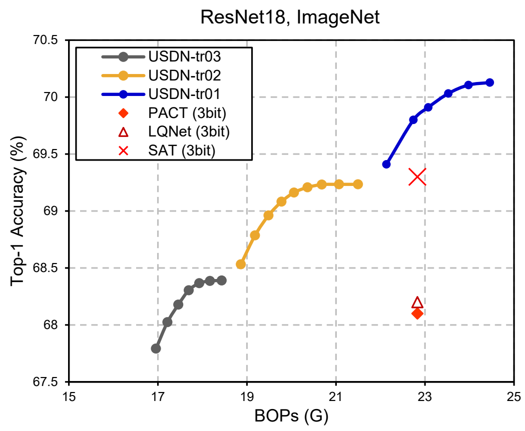


Figure 1. Comparison between the proposed method (USDN) and various state-of-the-art (SOTAs) on *ImageNet*.

their structures or parameters and select an execution path according to image-wise complexity during inference [15, 16, 19]. To dynamically choose a sample-wise execution path, many input-adaptive networks construct a *supernet* stacking multiple subnets, for example, a unified model supporting various bit-width configurations. During training, within such a supernet, multiple subnets share a single set of parameters, and a separate network, such as the recurrent neural network route [15] and policy network [16], is trained to choose an execution path for a given sample. During inference, the control unit determines a specific execution path to the input, and the selected subnet is executed on hardware. Unlike static networks, dynamic networks offer favorable properties, such as efficiency, representation power, adaptiveness, compatibility, generality, and interoperability [6].

The advantages of quantization and dynamic networks suggest a message that designing a unified mixed-precision network with multiple internal classifiers (ICs) may be a promising solution for *accuracy* and *computational efficiency*. Unfortunately, such an integrated network may suffer from accuracy degradation for various reasons. First, an IC location and bit-widths are sensitive to spatial-scale image resolution and the depth of a prediction network, which requires carefully designing an IC structure and searching for a proper IC location to balance *accuracy degradation* and *computational cost*. Second, jointly finding IC locations and layer-wise bit-widths may consume considerable training time due to large design space and result in a suboptimal solution for a weight-sharing supernet. Third, it is challenging to obtain a unified fully-quantized network, as crucial parts of the network, such as the first and last layers [19], as well as the residual connection [15], that significantly influence performance, are preserved to full precision.

To address these problems, we propose a *unified sample-wise dynamic network* (USDN), a fully-quantized input-adaptive mixed-precision network, with the following main contributions.

- *Supernet construction and a new IC design:* We construct a supernet by stacking architectural parameters for both IC locations and layer-wise bit-widths. Unlike existing EEs, we propose a scale-aware IC design to handle accuracy degradation for a large input effectively. In addition, different from conventional weight-sharing supernets, our USDN framework utilizes *dedicated model parameters* for different bit-widths during training.

- *Architectural Search and Training:* USDN introduces a compound set of architectural parameters, which consist of IC locations and layer-wise bit-widths, and cannot be explored by the conventional differential objective function [24, 25]. To address this problem, we propose a new objective function that integrates an *inference cost* for 'successful' exit rates and a *failure cost* for 'failed' exit attempts. We adopt a reinforcement learning (RL) solver to explore the architectural design space by minimizing the objective function.

- *Refinement with IC-wise Knowledge Distillation (KD)*: After the architectural search, IC locations and bit-width configurations are fixed. We refine the model by (1) training a full-precision model with the fixed IC locations and (2) applying IC-wise KD to refine the target mixed-precision student network from the full-precision teacher one.

- *Experimental Results*: To validate our USDN, we conduct a comprehensive analysis with various datasets. To the best of our knowledge, USDN is the first fully quantized input-adaptive mixed-precision network that achieves SOTA performance on the ImageNet dataset.

2. Background and Motivation

2.1. Static Quantized Networks

Weight/Activation Quantization: To compress and accelerate DNN inference, many studies proposed methods to quantize the weights and activations to low precision [5, 8, 26, 27]. In this paper, we utilized the widely-used method DoReFa-Net [27] to quantize the full-precision weights w with a bit-width of k. Once the bit-width is fixed, the weights are fine-tuned via quantization-aware-training (QAT). To quantize activations, the parameterized clipping activation (PACT) scheme [5] is employed to optimize the quantization scales. When both weights and activations are quantized to low precision, the metric of bit operations (BitOPs) is widely used to measure the computation cost of a quantized layer/network:

$$BOPs = FLOPs * k_w * k_a \tag{1}$$

Where *FLOPs* represent the number of floating-point operations, and k_w and k_a are bit-widths of weights and activations, respectively.

Mixed-Precision Quantization and Neural Architecture Search: Unlike uniform quantization, mixed precision techniques search for layer-wise bit-width configurations. [1, 13] preserve the accuracy of a quantized network by allocating different bit-widths to each layer, taking into consideration the layer's quantization sensitivity and computational cost. Considering bit-width configurations as a search space, [24, 25] construct a supernet that encompasses all possible bit-width configurations, such as {1,2,4,8} for a layer. This often involves exploring a vast design space(= 4^L,) where L represents the number of layers. Many previous studies [3, 17] train the supernet using RL, a methodology also applicable to USDN. USDN introduces a novel objective function that combines an inference cost for successful exit rates with a failure cost for failed exit attempt rates during the architecture search.

2.2. Dynamic Neural Networks

Dynamic Quantized Networks: Compared to static models, dynamic networks can adapt their structures or parameters according to different inputs during inference. [15] expands the search space to include a set of bit-widths, specifically {0,8,32}, where a bit-width "0" indicates a layer/block skipped during inference. Meanwhile, [16, 19] employ weight-sharing supernets, in which convolutions with different bit-widths use a *shared* weight set, and dedicated batch-normalization layers are assigned to corresponding bit-widths. Unfortunately, these methods require substantial training time to find *shared* model parameters among different architectural configurations. In addition, to avoid accuracy degradation, certain layers, such as the first and last layers (F/L) [19] or residual connections, are kept at full-precision [4, 15].

Early-exit (EE) Networks: Another class of dynamic networks involves EE networks that stack some internal classifiers (ICs) on intermediate feature maps [7,10,20]. For a given input, the IC outputs an intermediate prediction with its confidence score, and if this score surpasses a predefined threshold, the execution comes to a halt. Quantized EEnet is primarily explored for co-inference within distributed settings, such as edge-cloud environments. While the quantized early layers are offloaded to the edge, the rest are retained in full-precision and reside in the cloud [14]. [12] delves into the design spaces for exit location and threshold value in hardware-aware deployment. [11] focuses on the optimal split point between the edge device and the cloud. Meanwhile, [21] uses DQN to determine the choice between floating-point and quantized networks.

Anytime Inference: Anytime inference allows the network to halt computation when a sufficiently confident prediction is obtained at the IC [7, 10, 20]. Although anytime inference can effectively reduce computation costs for easy or moderately complex samples, obtaining an accurate prediction at early ICs is generally challenging, especially for a large inputs, such as 224x224 in ImageNet.

3. USDN Design Methodology

Overall Architecture: This section describes the unified sample-wise dynamic network (USDN) architecture. As depicted in Fig. 2, the USDN comprises a mixed-precision Q-baseline model and ICs. The baseline model (❶) comprises multiple convolutional layers/blocks with different bit-widths, followed by a fully connected (FC) layer and a softmax layer for the base prediction (❷). For each group of layers with the same feature map scale (❸), USDN attaches a scale-aware IC (❹). As a result, each input follows a distinct execution path based on its complexity. For example, an easy-to-classify sample is processed by the few first layers followed by an IC, while a hard-to-classify sample propagates to the base prediction layer. The sample can be classified at the IC if the confidence score at the IC prediction is larger than a exit threshold. Like instance-wise dynamic networks, USDN can adapt its structures or parameters to different inputs, leading to notable advantages in accuracy, computational efficiency, and adaptiveness [6].

Despite the above potential advantages, designing a unified sample-wise dynamic network may suffer from *accuracy degradation* and computation overhead when stacking ICs into the baseline network. To address the above problem, we introduce two simple yet effective techniques: (1) Scale-aware layer group construction and (2) Scale-aware IC design.

Scale-aware Layer Group Construction: We construct USDN by splitting the Q-baseline network along the depth,

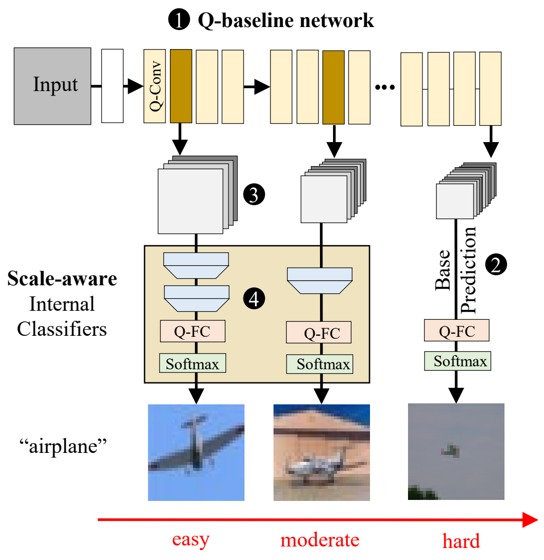
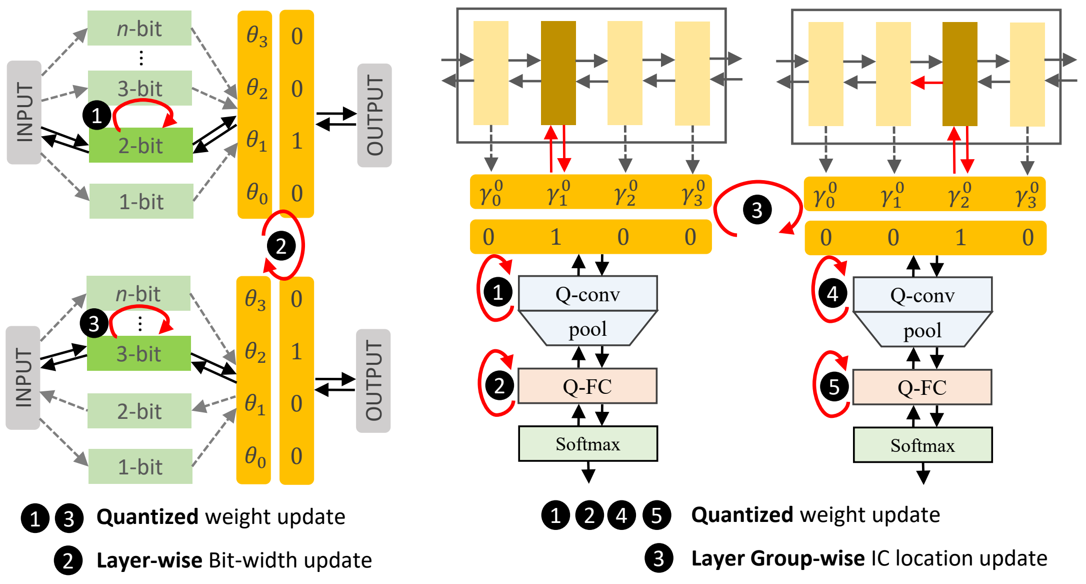


Figure 2. The diagram of the proposed method

which results in a few disjoint layer groups with a single IC. The precise IC locations within each layer group, that is, highlighted boxes at the Q-baseline network in Fig. 2, are determined by the architectural search process (See Section 4 for details.) Intuitively, our IC group construction can assess the sample difficulty in terms of computational cost. For instance, an *easy-to-classify* image requires fewer layers with shallow feature maps to output a correct prediction, whereas a *hard-to-classify* image must pass through many layers with deep feature maps to produce such a result. It is worth noting that stacking dense ICs can lead to gradient conflicts among ICs [22], resulting in accuracy degradation and increased computational costs due to exit failures. We empirically observe that using three classifiers would be sufficient.

Scale-aware IC Design: To reduce computation overhead, an IC typically includes a stride convolution or a pooling layer [10, 20], which may cause an unreliable prediction with shallow feature maps. Therefore, we design each IC so that its model size increases according to the size of the input feature map. Given an input of 224×224, the size of feature maps at the baseline prediction is 7×7. Accordingly, at each IC, we add multiple stride-2 3×3 convolutions until the feature map size decreases to 7×7. For example, when designing an IC for 56×56 feature maps, we add three convolutions. The resulting reduced feature map is then pooled to a 7×7 size and is subsequently fed into an FC layer. Notably, to avoid much computation overhead of ICs, we quantized the FC layer to eight bits, whereas the remaining components were quantized to four bits. *In practice, we observe that our ICs only account for 2.69% of the overall computation cost.*



INPUT
n-bit
3-bit
2-bit
1-bit
OUTPUT
θ_3 θ_2 θ_1 θ_0
Q-conv
pool
Q-FC
Softmax
γ_0^0 γ_1^0 γ_2^0 γ_3^0
❶❸ Quantized weight update
❷ Layer-wise Bit-width update
❶❷❹❺ Quantized weight update
❸ Layer Group-wise IC location update

Figure 3. Overview of the proposed USDN framework. (left): Bit-width search process. (right) : IC location search process.

4. Compound Architectural Search

4.1. USDN Supernet Construction

Fig. 3 presents an overview of the proposed supernet design. In the proposed method, we define two types of architecture parameters: θ for multiple bit-width candidates and γ for possible IC locations. To reduce memory consumption, we adopted binary gates from [3], which allow only a single candidate to be activated for inference. For each forward pass, binary gates are sampled from the multinomial distribution, with the probability derived from softmaxed θ and γ. Subsequently, each layer's output can be expressed as a weighted sum of the candidate operations and binary gates. More precisely, we denote the supernet as a DAG $\mathcal{N} = (V, O, Q)$, where $O = (o_0, ..., o_{K-1})$ refers to each candidate bit-width operation, and $Q = (q_0, ..., q_{M-1})$ denotes a candidate IC. Consequently, a binary gate g is plugged between the candidate operations. When K number of candidate bit-widths exist, the i-th layer output $v_i \in V$ can be expressed as follows:

$$v_i = \sum_{k=0}^{K-1} o_k(v_{i-1}) * g_k, \tag{2}$$

$$s.t. g_k \in \{0,1\}, \sum_{k=0}^{K-1} g_k = 1, g_k \sim softmax(\theta_i)$$

Similarly, the output feature of the IC is defined as the weighted sum of the IC candidates and binary gates. The architecture parameter γ_l represents possible IC locations within layer group l. When M layers exist in l-th layer group, the logit of l-th layer group $w_l \in V$ is expressed as follows:

$$w_l = \sum_{m=0}^{M-1} q_m^l(v_m) * g_m^l, \tag{3}$$

$$s.t. g_m^l \in \{0,1\}, \sum_{m=0}^{M-1} g_m^l = 1, g_m^l \sim softmax(\gamma_m^l)$$

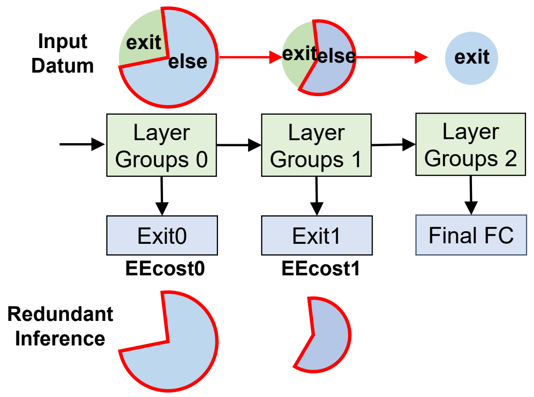


Figure 4. Anytime inference and redundant computation caused by exit failure.

4.2. USDN Supernet Training

The architecture parameters and weight parameters are trained in an interleaved manner, which implies that each set of parameters is updated in turn during the training process. The weights are trained with conventional cross-entropy loss L_{ce}. If the number of layer groups is N, we formulate the total loss L as the weighted sum of the losses of each of the ICs and their relative coefficient as follows:

$$L = \sum_{i=0}^{N-1} \lambda_i L_{ce}^i \tag{4}$$

Here, L_{ce}^i denotes the cross-entropy loss of the i-th classifier. λ is a relative coefficient that enhances the stability of the training by adjusting the scale of the gradient of each IC.

To train the supernet, we utilize the REINFORCE policy gradient algorithm [23]. For each forward pass, the binary gates g are sampled from a multinomial distribution with the probability derived from softmaxed θ and γ. Then, a candidate network $\mathcal{N}_g$ is constructed using those binary gates and the reward R is determined by performing an early-exit evaluation on the sampled network. When the optimal EEnet is defined as $\mathcal{N}(\theta_o, \gamma_o)$, the objective function is defined as:

$$\theta_o, \gamma_o = \underset{\theta,\gamma}{\operatorname{argmax}} \, \mathbb{E}_{g \sim \theta,\gamma}[R(\mathcal{N}_g(\theta^*, \gamma^*; \mathcal{D}_v))] \tag{5}$$

$$s.t. \theta^*, \gamma^* = \underset{\theta,\gamma}{\operatorname{argmin}} \, L(\mathcal{N}_g(\theta, \gamma; \mathcal{D}_t))$$

Here, $\mathcal{D}_v$ and $\mathcal{D}_t$ refer to the validation and training datasets, respectively. The nested equation can be solved by training the weight parameters using the equation defined in Eq. (4)

The architecture parameters are updated to maximize the expected reward. For simplicity, we express only θ in this equation:

$$\nabla_\theta J(\theta) = \mathbb{E}_{g \sim \theta}[R(\mathcal{N}_g(\theta^*; \mathcal{D}_v)) \nabla_\theta \log(p(g))] \tag{6}$$

$$\approx \frac{1}{P} \sum_{i}^{P} [R(\mathcal{N}_{g_i}(\theta^*; \mathcal{D}_v)] \nabla_\theta \log(p(g_i))$$

Here, $p(g_i)$ refers to the probability of sampling g_i, and P represents the length of the reward trajectory.

4.3. EE-aware Reward Function

Each sampled network has different IC locations and bit-width configurations. The reward function aims to establish a criterion for distinguishing the best configuration. In this study, we formulate an EE-aware reward function that compromises both computational cost and accuracy, as shown in the following equation:

$$R(\mathcal{N}_g) = EEAcc(\mathcal{N}_g) * (\frac{Cost(\mathcal{N}_g))}{B}))^{-tr}, \tag{7}$$

where $EEAcc$ refers to the early-exit accuracy, and B denotes the target BOPs. A trade-off ratio tr is introduced to balance the cost and accuracy.

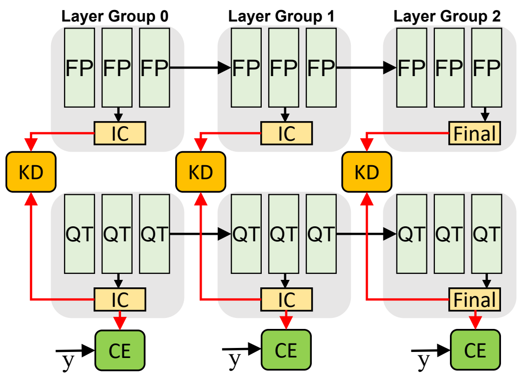


Figure 5. IC-wise knowledge distillation training. Here, y represents the true label of the input image.

To quantify the computational cost of each sampled network, we design the early-exit cost using the inference cost ($Cost_{infer}$) and the failure cost ($Cost_{fail}$).

$$Cost(\mathcal{N}_g) = Cost_{infer}(\mathcal{N}_g) + Cost_{fail}(\mathcal{N}_g) \tag{8}$$

As illustrated in Fig. 4, for images that successfully undergo early exit, further execution stops. Their inference cost consists of the accumulated cost along the inference path of each layer group ($ToClassifier$), which includes the cost of the classifier they use to exit. The total inference cost of the validation set is calculated as follows:

$$Cost_{infer}(\mathcal{N}_g) = \sum_{i=0}^{N-1} ER_i * ToClassifier(i) \tag{9}$$

Here, N refers to the number of classifiers in the network and ER_i denotes the exit rate of the i-th classifier.

If a sample fails to exit at an IC, it propagates to the subsequent layer groups, incurring redundant computation equivalent to the computational cost ($EEcost$) of that IC. For instance, assume an image takes an early exit at layer group $j(> i)$. Since the execution path of layer group i is a part of that of j, the only redundant computation comes from the BOPs of the IC at layer group i. Similarly, if an image exits at level $j + 1$, the redundant computation is the sum of the BOPs consumed by the ICs at layer groups i and j. As a result, the overall failure cost ($FailCost$) for the validation set is defined as follows:

$$Cost_{fail}(\mathcal{N}_g) = \sum_{j=0}^{N-2} (1 - \sum_{i=0}^{j} ER_i) * EEcost(j) \tag{10}$$

5. IC-wise Knowledge Distillation Training

Knowledge distillation is a method used to train a compact model by utilizing information from a larger model. Several studies have introduced knowledge distillation training for EEnet. Most of them employ self-distillation, in which each IC functions as both a student and a teacher. However, we empirically found that self-distillation is ineffective when training quantized EEnet. Instead, we leverage the higher precision EEnet as a teacher network to train the quantized EEnet. The proposed training method is illustrated in Fig. 5. Each IC of the full-precision (FP) EEnet imparts knowledge to the corresponding IC location in the quantized EEnet. As a result, the quantized EEnet is trained using knowledge distillation loss, denoted as L_{KD}, which comprises the distillation loss L_{dist} and the classification loss L_{ce}. If there are N classifiers and multiple softmax outputs $\{z_{i,q}\}^{i=0,..,N-1}$ from the quantized classifiers, then L_{KD} is defined by the following equation:

$$\begin{aligned} L_{KD} &= L_{classification} + L_{distillation} \qquad (11) \\ &= 2(1-\alpha)(\sum_{i=0}^{N-1} \lambda_i \mathcal{L}(z_{i,q}^{1/T}, \hat{y}) \\ &+\alpha T^2 \sum_{i=0}^{N-1} \lambda_i \mathcal{L}(z_{i,q}^{1/T}, z_{i,f}^{1/T}) \end{aligned}$$

Here, $z_{i,f}$ refers to the logit of the i-th classifier of FP EEnet, and α stands for the balancing parameter between the CE loss and KD loss. The logits are temperature (T)-scaled to create a margin between the logits of the quantized EEnet and the high-precision EEnet.

6. Evaluation

6.1. Dataset

ImageNet consists of 1.2 million training images and 50,000 validation images. During search process, we randomly sample 50,000 images from the training dataset and utilize it as a validation dataset. The original validation images are retained for use as the test dataset. Each image is resized to 224×224, subjected to random cropping, and flipped for data augmentation. **CIFAR** contains 50,000 training and 10,000 test images. Similarly, we randomly sample 5,000 images from the training dataset and employ them as a validation set during the search process. For both datasets, we do not apply normalization to the images. Instead, the input data is converted to the uint8 data format, as in [8, 9]. Due to the space limitations, we provide further details about the training scheme in the supplementary materials.

6.2. Searched Configuration and trade-off ratio

For a fair comparison, we explored configurations using various trade-off ratios. When employing a larger trade-off ratio, the weight of the cost function within the reward function increases. Consequently, a model with a fewer BOPs is discovered during search process. As shown in Tab. 1, the model searched with a high trade-off ratio (=0.3, USDN-tr03) exhibits lower BOPs and accuracy compared to the

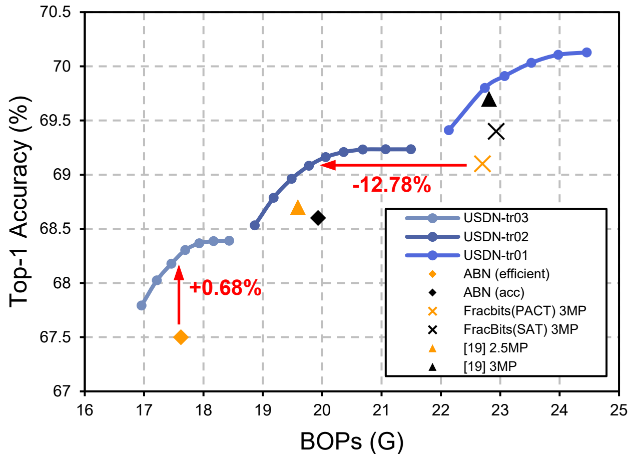


Figure 6. Comparison between the proposed method and various state-of-the-art (SOTAs) on *ImageNet*. Compared with the mixed-precision method, USDN reduces the computational cost by 12.78% without any accuracy degradation. Compared with the input-adaptive quantization method, USDN shows 0.68% absolute gains in accuracy without additional computational cost.

model searched with a lower trade-off ratio (=0.1, USDN-tr01). The specifics of the searched configurations are provided in the supplementary materials.

6.3. Comparison with quantization methods

Fig. 6 shows the results of ResNet18 tested on the *ImageNet dataset*. We compare our results with works of various quantization methods [2, 5, 8, 9, 18, 19, 26]. To demonstrate the feasibility of the USDN, we provide the results for various target BOPs. The numerical results are listed in Tab. 1. **Compared with uniform quantization**, USDN achieves higher accuracy than PACT [5], LQNet [26], and SAT [8] (69.3% vs. 69.8%) with a smaller BOPs (22.84G vs. 22.74G) consumption. **Compared with mixed-precision methods**, USDN outperforms FracBits [25] with higher accuracy (69.4% vs. 69.8%) with lower BOPs (22.93G vs. 22.74G). In the case of Tang, Chen,*et al*. [18], USDN shows (+0.3%) absolute gains in accuracy when the target bit-width is 2.5MP. USDN achieves higher accuracy (69.7% vs. 69.8%) when the BOPs target is set to 3MP.

Compared with adaptive precision networks, USDN shows superior accuracy (68.7% vs. 69.8%) with a smaller computational cost. It is worth noting that when a lower exit threshold is applied for evaluation, USDN (Ours-efficient) can reduce the computational cost by (-14.63%) even with higher accuracy (68.7% vs. 69.0%).

Compared with the input-adaptive quantization method [19], USDN improves the accuracy by 1.2% and, 0.4% absolute values when the target BOPs are set to 2.5MP and 3MP.

To demonstrate the feasibility of the proposed method

Table 1. Comparisons between our method and previous quantization approaches on ImageNet using ResNet18. The term 'th' refers to the exit threshold. Except for ABN [19], the first and last layers are quantized to 8-bit across all other methods. The calculation of BOPs in [18] differs from our method; thus, we approximate the value using the BOPs of uniform quantization. Similarly, for ABN, as they reported their BOPs as a relative cost compared to AdaBits [9], we estimate their BOPs using the cost of AdaBits.

Method	W	A	Top-1 Acc.↑(%)	BOPs↓ (G)
PACT [5]	3	3	68.1	22.83
LQNet [26]	3	3	68.2	22.83
SAT [8]	3	3	69.3	22.83
FracBits-PACT [25]	3MP	3MP	69.1	22.70
FracBits-SAT [25]	3MP	3MP	69.4	22.93
Tang, Chen,*et al*. [18]	2.5MP	3MP	68.7	19.59*
Tang, Chen,*et al*. [18]	3MP	3MP	69.7	22.81*
AdaBits [9]†	3	3	68.5	22.83
Bit-Mixer [2]†	3	3	68.7	22.83
ABN [19]	Runtime+MP		67.0	17.62*
ABN [19]	Runtime+MP		68.6	19.93*
USDN-tr03(th=0.7)	EE+2.5MP		**68.2**	**17.46**
USDN-tr02(th=0.7)	EE+3MP		**69.0**	**19.49**
USDN-tr01(th=0.7)	EE+3MP		**69.8**	**22.74**

Method	W	A	Top-1 Acc.↑(%)	BOPs↓ (G)
PACT [5]	4	4	69.2	34.70
LQNet [26]	4	4	69.3	34.70
SAT [8]	4	4	70.3	34.70
FracBits-PACT [25]	4MP	4MP	69.7	34.73
FracBits-SAT [25]	4MP	4MP	70.6	34.70
Tang, Chen,*et al*. [18]	4MP	4MP	70.8	34.68*
AdaBits [9]†	4	4	69.2	34.70
Bit-Mixer [2]†	4	4	69.4	34.70
USDN-tr01(th=0.6)	EE+4MP		**69.9**	**29.84**
USDN-tr01(th=0.9)	EE+4MP		**70.8**	**34.43**

†There is no bit-width selection w.r.t. input

across various dataset, we conduct experiments on a small network and dataset, specifically ResNet20 and CIFAR10. In this case, we adopt the IC design from [10]. As demonstrated in Tab. 2, USDN exhibits a significant cost reduction in terms of BOPs while preserving high accuracy when compared to the fixed precision methods such as DoReFa and PACT.

6.4. Comparison with quantized EEnets

We also conduct comparisons between the results of USDN and previous early-exit networks [7, 10]. Firstly, in Fig. 8, it is evident that USDN-tr03 (2.5MP) outperforms 4-bit quantized MSDnet in terms of Top-1 Accuracy in ImageNet dataset. Secondly, we present the comparison results with [10] in Fig. 9. In this case, we applied our method on MobileNetV1 baseline and evalalute its performance on the CIFAR100 dataset. The results in Fig. 9 demonstrate that the proposed USDN achieves higher accuracy (+1.1%) compared to the uniformly quantized 4-bit SDN at the same computation cost. Note that previous EEnets have more ICs ($\geq$5) than USDN(=3). The experimental results demonstrate that the existing EEnet, despite having more ICs and a higher bit-width, exhibits lower accuracy compared to the proposed USDN. This implies that merely combining EEnet and quantization cannot effectively exploit the accuracy-cost trade-off.

Table 2. Comparisons of our work and previous quantization approaches on CIFAR-10 with ResNet20. Note that all methods quantized the first and last layer to 8-bit. 'W' and 'A' are the weight/activation bit-width abbreviations. 'USDN-Acc' refers to the model in which the trade-off ratio is set to 0, so it considers accuracy only.

Method	W	A	Top-1 Acc.(%)↑	BOPs (G)↓
DoreFa [27]	4	4	90.5	0.670
PACT [5]	4	4	91.3	0.670
USDN-tr001(th=0.8)	EE+4MP		91.23	**0.514**
USDN-Acc(th=0.9)	EE+4MP		**91.68**	**0.603**

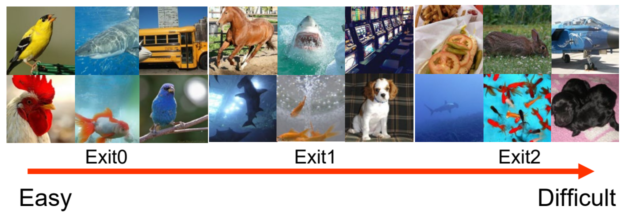


Figure 7. Early-exited images from each exit point. As the level increases, the target object becomes more complex and vague.

6.5. Early-Exited images from each Exit Point

Fig. 7 shows early-exited images from each exit point. As USDN conducts progressive inference, the image that exits from the earliest exit point will be the most straightforward case. As the exit point moves towards the back, the images gradually become more challenging. For instance, the target objects become more intricate and less distinct.

7. Discussion

7.1. Effect of IC-wise KD Training

To verify the effectiveness of IC-wise KD training, we train the USDN obtained through search process using only the cross-entropy loss. Subsequently, we compare

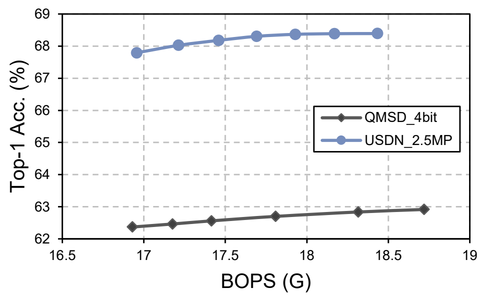


Figure 8. Comparison between uniformly quantized 4-bit MSD-net [7] and the proposed USDN-tr03 EE+2.5MP (ResNet18) evaluated on ImageNet dataset.

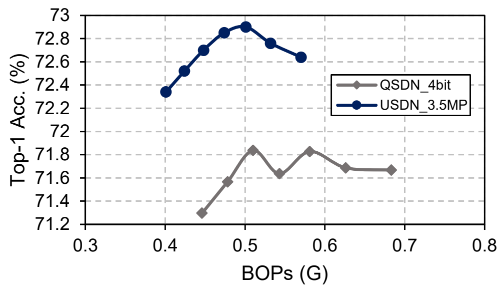


Figure 9. Comparison between uniformly quantized 4-bit SDN [10] and the proposed USDN-tr01 EE+3.5MP (MobileNetV1) evaluated on CIFAR100 dataset.

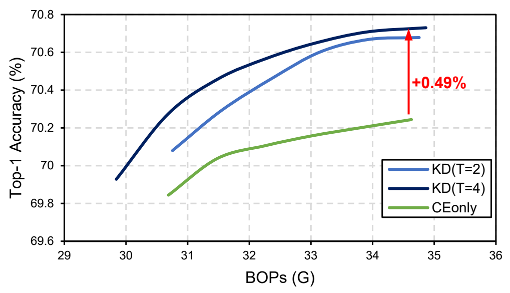


Figure 10. Comparison between using the proposed knowledge distillation loss and cross-entropy loss. The experiment is conducted on EE+4MP (ResNet18) on the ImageNet dataset.

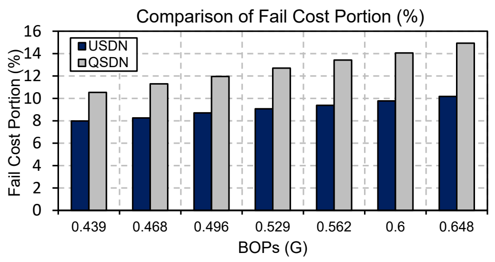


Figure 11. Comparison between uniformly quantized 4-bit SDN [10] and the proposed USDN EE+3.5MP (MobileNetV1) evaluated on CIFAR100 dataset.

this model with the KD-trained network with identical bit-widths and IC locations. The results are summarized in Fig. 10. The KD-trained USDN exhibits better accuracy-cost trade-offs than the CE-trained network on ImageNet dataset. More precisely, KD training improved the accuracy at most 0.49%. We found that each classifier's confidence calibration gets better when KD is applied, which in turn leads to better performance. Further analyses are presented in the supplementary material.

7.2. Effect of Exit Threshold

In Fig. 6, we measure accuracy and BOPs by incrementally raising the threshold from 0.5 to 0.8. Generally, as the threshold rises, only more reliable predictions are exited early, leading to a reduction in the exit rate and an improvement in accuracy. Indeed, we observe an escalation in BOPs and note a trend where accuracy tends to align with the accuracy of the final fully connected layer.

7.3. Effect of Failure cost

To assess the effect of incorporating failure cost into the objective function, we compared the failure cost portions of the conventional SDN and USDN. The failure cost portion was calculated by dividing the fail cost by the total cost and multiplying it by 100. The corresponding results are illustrated in Fig. 11. For a fair comparison, we calculated the BOPs for each model using various thresholds (0.6-0.9) and compared cases with similar BOPs. As shown in the figure, USDN exhibits a lower failure cost portion compared to quantized SDN across all BOPs ranges.

8. Conclusion

This work proposes a unified sample-wise dynamic network with internal classifiers and mixed-precision quantization. The proposed framework jointly searches IC locations and layer-wise bit configurations to find a good trade-off between computation reduction and accuracy degradation. Especially during the search process, USDN stacks multiple parallel layer-wise model candidates with a gating function, which avoids an accuracy drop caused by sharing weights among different execution paths. Meanwhile, during inference, with multiple ICs, the unified network saves memory storage and reduces computation by adaptively selecting an execution based on an instance's difficulty. To this end, USDN outperforms SOTA methods by achieving similar or better accuracy at a low bit-width computation cost on various datasets such as ImageNet and CIFAR.

Acknowledgement. This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2020-0-01080, Variable-precision deep learning processor technology for high-speed multiple object tracking) and the artificial intelligence semiconductor support program to nurture the best talents (IITP-2023-RS-2023-00256081)

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Fixed point optimization of deep convolutional neural networks for object recognition. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1131–1135. IEEE, 2015. 2

[2] Adrian Bulat and Georgios Tzimiropoulos. Bit-mixer: Mixed-precision networks with runtime bit-width selection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 5188–5197, 2021. 1, 6, 7

[3] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *International Conference on Machine Learning (ICLR)*, 2019. 2, 4

[4] Shaofeng Cai, Yao Shu, and Wei Wang. Dynamic routing networks. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3588–3597, January 2021. 3

[5] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: parameterized clipping activation for quantized neural networks. *CoRR*, 2018. 1, 2, 6, 7

[6] Yizeng Han, Gao Huang, Shiji Song, Le Yang, Honghui Wang, and Yulin Wang. Dynamic neural networks: A survey, 2021. 1, 3

[7] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q. Weinberger. Multi-scale dense networks for resource efficient image classification. In *ICLR*, 2018. 3, 7, 8

[8] Qing Jin, Linjie Yang, and Zhenyu Liao. Towards efficient training for neural network quantization. 2019. 1, 2, 6, 7

[9] Qing Jin, Linjie Yang, and Zhenyu Liao. Adabits: Neural network quantization with adaptive bit-widths. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2146–2156, 2020. 1, 6, 7

[10] Yigitcan Kaya, Sanghyun Hong, and Tudor Dumitras. Shallow-deep networks: Understanding and mitigating network overthinking. In *International Conference on Machine Learning (ICLR)*, pages 3301–3310. PMLR, 2019. 3, 7, 8

[11] Stefanos Laskaridis, Stylianos I Venieris, Mario Almeida, Ilias Leontiadis, and Nicholas D Lane. Spinn: synergistic progressive inference of neural networks over device and cloud. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–15, 2020. 3

[12] Stefanos Laskaridis, Stylianos I Venieris, Hyeji Kim, and Nicholas D Lane. Hapi: hardware-aware progressive inference. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020. 3

[13] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International conference on machine learning*, pages 2849–2858. PMLR, 2016. 2

[14] Yinghan Long, Indranil Chakraborty, and Kaushik Roy. Conditionally deep hybrid neural networks across edge and cloud. *arXiv preprint arXiv:2005.10851*, 2020. 3

[15] Jianghao Shen, Yue Wang, Pengfei Xu, Yonggan Fu, Zhangyang Wang, and Yingyan Lin. Fractional skipping: Towards finer-grained dynamic CNN inference. In *The 34th AAAI Conference on Artificial Intelligence*, 2020. 1, 2, 3

[16] Ximeng Sun, Rameswar Panda, Chun-Fu Richard Chen, Aude Oliva, Rogerio Feris, and Kate Saenko. Dynamic network quantization for efficient video inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 7375–7385, 2021. 1, 2

[17] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019. 2

[18] Chen Tang, Kai Ouyang, Zhi Wang, Yifei Zhu, Wen Ji, Yaowei Wang, and Wenwu Zhu. Mixed-precision neural network quantization via learned layer-wise importance. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pages 259–275. Springer, 2022. 1, 6, 7

[19] Chen Tang, Haoyu Zhai, Kai Ouyang, Zhi Wang, Yifei Zhu, and Wenwu Zhu. Arbitrary bit-width network: A joint layer-wise quantization and adaptive inference approach. *arXiv preprint arXiv:2204.09992*, 2022. 1, 2, 3, 6, 7

[20] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2464–2469. IEEE, 2016. 3

[21] Meiqi Wang, Jianqiao Mo, Jun Lin, Zhongfeng Wang, and Li Du. Dynexit: A dynamic early-exit strategy for deep residual networks. In *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 178–183. IEEE, 2019. 3

[22] Xinglu Wang and Yingming Li. Gradient deconfliction-based training for multi-exit architectures. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 1866–1870. IEEE, 2020. 3

[23] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992. 5

[24] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018. 2

[25] Linjie Yang and Qing Jin. Fracbits: Mixed precision quantization via fractional bit-widths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10612–10620, 2021. 2, 6, 7

[26] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 365–382, 2018. 1, 2, 6, 7

[27] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, 2016. 2, 7