공학석사학위논문

Optimization of Mixed-precision Neural Architecture with Knowledge Distillation

2020년 2월

서울대학교 대학원

전기 · 정보 공학부

Nguyen Tuan Nghia

Optimization of Mixed-precision Neural Architecture with Knowledge Distillation

지도교수 이 혁 재

이 논문을 공학석사학위논문으로 제출함

2020년 2월

서울대학교 대학원

전기 · 정보 공학부

Nguyen Tuan Nghia

Nguyen Tuan Nghia 의 석사학위논문을 인준함

2020년 2월

위 원 장 김 태 환 (인)

부 위 원 장 이 혁 재 (인)

위 원 류 수 정 (인)

Abstract

Quantization is an essential process in the deployment of deep neural networks on edge devices which only have limited memory and computation capacity. However, it is widely known that the straightforward uniform-precision quantization method, which applies the same quantization scheme including bit-width and transformation function to all layers, suffers a severe performance degradation. Meanwhile, finding and assigning different quantization schemes to different layers is challenging as the number of candidates is exponential to the the number of layers. To address this problem, this study proposes a method that utilizes Knowledge Distillation technique to efficiently explore the search space in linear time. In particular, the proposed method formulates a per-layer loss function to estimate the impact of a quantization scheme on a target layer. Based on the assumption that the dependence among layers is minimal, the assignment is then decided for each layer separately to minimize the performance loss. Experiments are conducted for both image classification and object detection task, using only hardware-friendly quantization schemes. The results show that the most efficient mixed-precision ResNet20 model with 13.65 times smaller size can still achieve up to 93.62% accuracy on CIFAR-10 dataset, which is only 0.19% lower than the baseline full-precision model. On VOC dataset, the proposed method generates a mixed-precision Sim-YOLOv2-FPGA model with a mean average precision of 63.87, which outperforms all uniform-precision models with the same compression rate. The proposed method is practically simple to carry out while still achieving a comparable efficiency to other state-of-the-art approaches on mixed-precision quantization.

Keywords: Neural Network Quantization, Mixed-precision, Knowledge Distillation, Neural Architecture Search, Deep Learning, AI Accelerator

Student Number: 2018-21525

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

A deep neural network model usually has a huge amount of learnable parameters. Deploying the model on any device requires considerably large memory to store its parameters, as well as intermediate computational results which are usually called feature maps. This problem become an obvious challenge for edge devices due to their limited high-speed on-chip memory. While using external memory may help, one should consider the negative impact on processing speed and power consumption. These are motivations for network quantization techniques.

Many researches on quantization have proven that the 32-bit floating-point representation of neural network is inefficient. Instead, deep neural network can work equally well or even better with much smaller bit-width. Quantization has been intensively focused for many years. At the early stage, quantization methods basically went towards the idea of low-precision representation which allows neural network to practically work on FPGA and mobile devices. People soon realized the trade-off between complexity and accuracy that using fewer bits would result in lower accuracy. To minimize this trade-off, many novel ideas were proposed. However, the more improvement those methods can make, the more complicated they are.

As we roll back to the early stage of quantization research, those hardware-friendly methods such as uniform quantization did not work very well due to the fact that they have been applied with little consideration. A single quantization scheme is applied to all target units, such as layers or blocks of layers, ignoring an important fact that different units may have different characteristics as well as impacts on the

overall accuracy. For example, if a parameter set A has a different range of values compared to a parameter set B, using the same quantization scheme might lead to data loss in either A or B and further lead to overall accuracy loss. To minimize this loss, the known mixed-precision should be applied, in which two different schemes must be selected for A and B separately. It is clear that mixed-precision is more general and appropriate for use to obtain a compact yet accurate network model.

Mixed-precision problem aims to find an optimal assignment of a quantization scheme to each unit in the target network. An optimal assignment must satisfy some given constraints on accuracy and compression ratio. For example, to obtain minimal accuracy loss, some layers in a network should be represented by two bits while some others need four. It is clear that using four bits for all layers is inefficient even though the model is able to achieve the desired accuracy. On the other hand, using two bits for all layers reduces the size better but will likely degrade the accuracy. From the example, we also find that the target network usually contains many layers, and each layer has several quantization options to choose. As a consequence, the pool of choices contains an exponentially large number of possible configurations. Exhaustive searching through all the configurations is impossible in practice. Therefore, it is necessary to find an efficient approach to explore the candidate space.

In this study, we propose Knowledge Distillation Mixed Precision (KDMP), an efficient method to address the mixed-precision problem. The proposed method explores the impact of a quantization scheme to a layer. Given a baseline network with finite quantization choices, we iteratively apply one scheme to one target layer while keeping all other layers unchanged. The new network is then sufficiently trained with knowledge distillation technique to recover the original accuracy before

it is ready for validation on the task objective. The changes in accuracy of the obtained model compared to the baseline are measured which reflects the impact we want to know. Based on the evaluation, mixed-precision candidate architectures are generated. The candidates satisfy the constraint on compression ratio while still having high accuracy on the desired task.

The efficiency of the proposed method is demonstrated through experiments. With a search space for quantization options containing only hardware-friendly methods, the mixed-precision ResNet20 model generated by KDMP has a negligibly low accuracy loss on CIFAR-10 dataset compared to the baseline, while being 13.65 times smaller in model size and requiring 7.03 times less memory to store the intermediate feature maps. Another experiment on VOC dataset shows that our mixed-precision Sim-YOLOv2-FPGA model outperforms other uniform-precision ones in terms of accuracy while having the same compression rate. The mAP on object detection task is also comparable to another mixed-precision model with bigger size obtained by random pick.

Chapter 2: Related Work

This chapter walks through the concept of neural network transformation including quantization and distillation. We highlight the importance of these techniques in the deployment of neural network on edge devices. Their imperfection is the motivation for the proposed method in the next chapters. We also introduce two simple quantization methods which directly involve in our experiments. Finally in the last section, we introduce an automated approach to generate the network architecture for a specific task that satisfies various design constraints. This approach is named Neural Architecture Search.

2.1. Quantization techniques

Quantization is a well-known technique to represent neural network parameters and feature maps by a specific number of bits, aiming to relax the computational complexity. While these parameters are typically 32-bit floating-point numbers in most of the currently available frameworks, a quantized neural network model may have its parameters stored in the memory by 16 bits, 8 bits or even 1 bit. To perform quantization, one needs to define the bit-width and a transformation function mapping an input value to an output in a set of finite values. A set including bit-width, transformation function and other factors is called a quantization scheme.

Neural network quantization has been an active research area for many years. In practical, quantization causes some degradation to the performance since the set and distribution of values to represent the network parameters are strictly limited. There are several quantization methods with complicated transformation function

exploiting various characteristics [11-13] of the target input to avoid accuracy loss. However, as the ultimate goal for using quantization is to make the implementation of neural network on edge devices possible, an appropriate method should balance the trade-off between accuracy and complexity.

In this work, we prove that simple quantization methods when selectively applied can prevent accuracy degradation. This is the core idea that will be introduced in the next chapter. The next sections introduce two simple ways to quantize weights and activations. These methods have been successfully used in hardware design of neural network [2].

2.2. Uniform quantization

Uniform quantization is the simplest form of quantization. The set of output values are uniformly distributed. Transformation function maps an input to an output by comparing the input with thresholds. Mathematical functions such as “round”, “floor” and “ceiling” are three commonly used uniform transformation functions. In case of finite output set, if the input is smaller than the smallest threshold, it will be mapped to the smallest output value. If the input is larger than the largest threshold, it will be mapped to the largest output value. It is worth to note that thresholds are also uniformly distributed. Typically, two adjacent output values or two adjacent thresholds are different by a fixed value, and this difference is called step or step size. Figure 2.1 illustrates uniform quantization mapping input **x** to output **y** by a transformation function **q**.

The output value set, as mentioned before, is finite. The number of value is decided by the number of bits, or bit-width. If we quantize the input by 4 bits, there are 2^4 or

16 output values to which an input value can be mapped. Given a uniform quantization scheme specified by a bit-width **n**, a positive value step size **s**, and a smallest (starting) value q_1, the mathematical formula of uniform quantization can be expressed as equation (1). In the equation, q_i belongs to the output value set Q = $\{q_i \mid i = 1 \text{ to } 2^{\mathbf{n}} \text{ and } q_{i+1} = q_i + \mathbf{s}\}$.

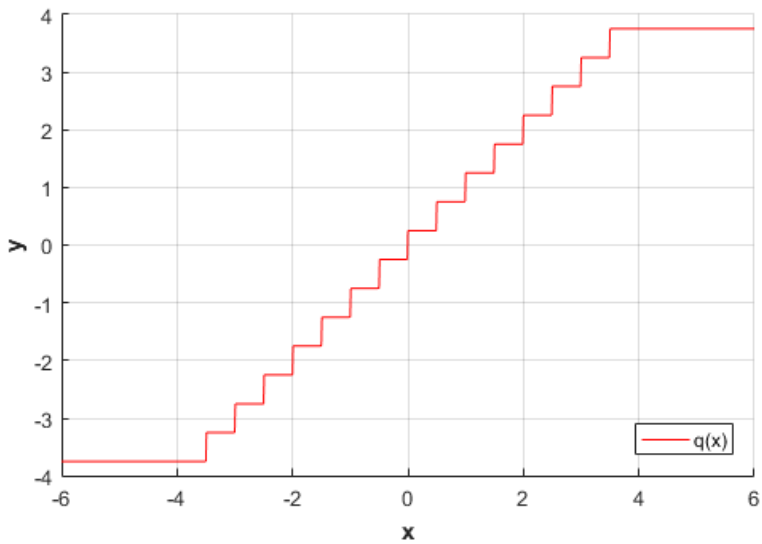

Figure 3.1. Uniform quantization with 4 bits and step of 0.5

$$y = q(x) = \begin{cases} q_i \text{ if } x > q_{i-1} \text{ and } x \leq q_i \\ q_1 \text{ if } x \leq q_1 \\ q_n \text{ if } x > q_{n-1} \end{cases} \quad (1)$$

2.3. Shifter quantization

Shifter quantization is a variance of incremental quantization [4], in which an input is mapped to an output in the form of multiplication between a scale factor **s** and a representing value $\pm 2^k$. The name shifter is derived from the fact that multiplying

with the represented value can be exploited as a shifting operation in the computation. This quantization scheme is useful for hardware implementation as multiplication is usually much more expensive than shifting. An early form of shifter quantization is introduced in XNOR-net [3]. In XNOR-net, the represented value is only a single bit and scale factor is mean of absolute values of each weight tensor. This method is further developed into a more general form with **n** bits. Scale factor is also allowed to be max of absolute values of each weight tensor as well.

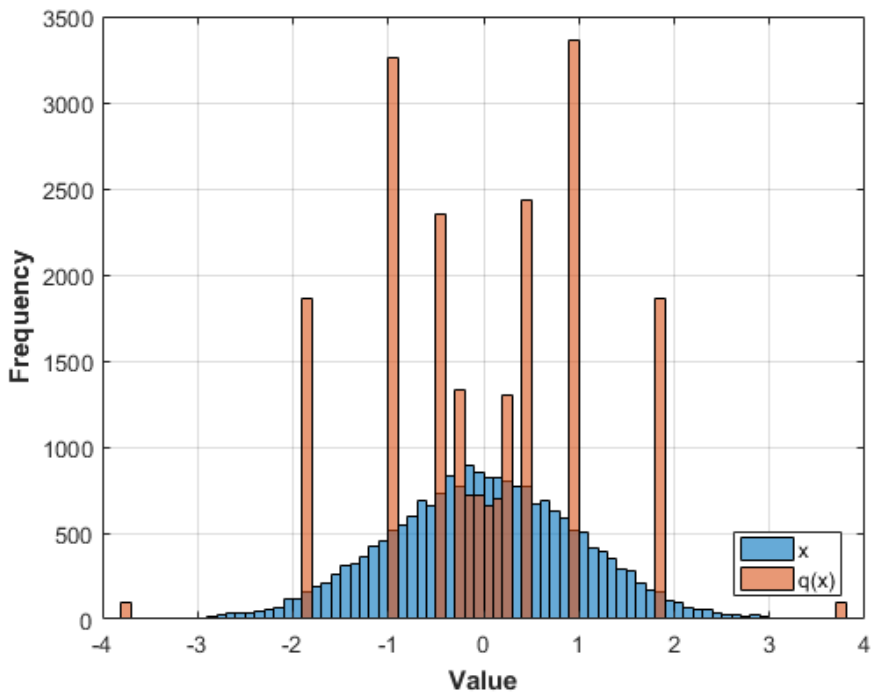

Figure 2.2. Histogram of values before and after quantization with max-shifter

The transformation function of max-shifter quantization is given in equation (2). The scale factor **s** is the maximum absolute value of a weight tensor. In case of mean-shifter quantization, **s** is the mean of all absolute values. It is necessary to specify the

bit-width **n** which will restrict the number of values in the output set. The size of the output set is $2^{\mathbf{n}}$, and the choice of k ranges from 0 to $(1 - 2^{\mathbf{n}-1})$. For example, with **n** = 3, the output set contains 8 values: ±**s**×1, ±**s**×0.5, ±**s**×0.25, and ±**s**×0.125. Any input that is out of bound will be mapped to its nearest value in the set. Figure 2.2 shows a value histogram of a tensor before and after 4-bit max-shifter quantization.

$$y = q(x) = \text{sgn}(x) \times s \times 2^{k}$$
$$\text{where } k = \text{round}\left(\log_2 {\text{abs}(x)}/{s}\right) \tag{2}$$

2.4. Knowledge Distillation

Knowledge distillation (KD) [14] aims to transfer the behavior of one or multiple huge networks, also named the teachers, to a smaller one, the student. In other words, the student network is trained with the guide of the teachers to reproduce the same output with less complexity. KD has been successfully applied in network compression. Comparing to quantization, KD targets the change of network architecture instead of the representation of network parameters. Because training a small network directly on task objective does not guarantee a good convergence, guiding with KD is necessary, especially at the early stage of the training phase.

A conventional setup for training with KD is illustrated in Figure 2.3. During training, forward pass is performed on both teacher and student while backward is done only on the student network. The student network will update itself based on not only the task loss but also the distillation loss. Distillation loss is evaluated at the output of some intermediate layers in the student network to guide the training better. Loss is

based on the difference of two output feature maps coming from the teacher network and the student network. There is no restriction for the student network to generate its own output at other layers, but the same behaviors at those distillation points are expected. Popular choice for distillation loss is L1, L2 or smooth L1.

KD works well in such case where only one or few layers involve in the training process. For example, if one early layer in a deep architecture needs to be replaced, it is better to train the new layer to mimic the output of the old one by KD instead of using task loss. The reason is that deep network is likely to suffer gradient vanishing that could make the training inefficient. In addition, training the whole network also consumes a lot more time and resources than training only the target layer with KD.

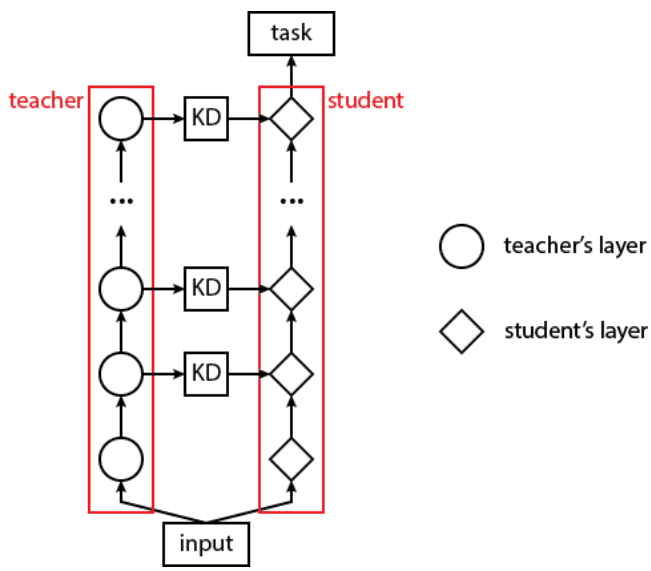

Figure 2.3. Knowledge distillation training setup

Sufficiently trained quantized layers are necessary for the assignment of the mixed-precision problem. However, training takes a lot of time and resources. In this work,

we apply KD to help the quantized layer reach a sufficient convergence point in an efficient yet still effective way. We prove that the proposed method is scalable such that training can be done on a single GPU even with large network.

2.5. Neural Architecture Search

Neural Architecture Search (NAS) becomes an active research area in the recent years. Before the era of architecture search, network architectures are manually decided. People then have interest in an automated approach to generate network architecture for a specific task. One of the early NAS methods [8] uses reinforcement learning which requires a huge amount of computing resources. In 2018, ENAS [9] is proposed that can generate network architecture with much lower computational power by using reinforcement learning to sample a good child from a super network. ENAS is based on the idea that weights can be shared between the super network and its child networks. Training to sample is further developed in DARTS [10].

Recently, NAS has gained the attention of researchers working in deep learning for hardware. In mixed-precision neural architecture search, one of the earliest approaches is DNAS [1]. Taking the idea of DART and ENAS, DNAS constructs a super-net by stacking all quantization schemes in the search space and then trains it so that all quantized layers are optimized. Though DNAS is effective and efficient, the method is not scalable with large network as training the super-net requires a lot of computing resources. Our proposed method is more flexible, in which it allows each quantized layer to be trained separately instead. In case of having enough resources, it is possible to train all quantized layers in parallel, thus obtaining at least the same efficiency as DNAS.

Chapter 3: Knowledge-Distillation Mixed-Precision

In the previous chapter, we have discussed the importance of quantization technique. There are many quantization methods, and each method has its own advantages and disadvantages in terms of accuracy and complexity. In general, a method which guarantees a better accuracy has higher complexity that makes it difficult to implement on hardware. It is obvious and reasonable for this trade-off to exist. However from another view, hardware-friendly quantization methods have been applied to all layers in a uniform manner. In other words, all target layers share the same quantization scheme including bit-width, transformation function and other related factors. This approach has oversimplified the problem, ignoring some unique characteristics of each layer such as statistics or value distribution and leading to high accuracy loss.

In this work, we propose Knowledge-Distillation Mixed-Precision (KDMP) framework to address the mixed-precision problem, which aims to find an optimal assignment of different quantization schemes to different layers. In the proposed method, knowledge distillation is applied to assess the impact of each quantization scheme on each layer. We then decide the assignment based on this information before fine-tuning the architectures in order to obtain the optimized performance.

3.1. Complexity challenge and solution

Mixed-precision neural architecture search is a branch of the more general problem called neural architecture search. With a baseline model and a search space containing multiple quantization schemes, the goal is to assign one of those schemes

to each layer of the model. A typical flow to solve the problem involves two phases: searching and training. In the searching phase, we need to generate the network architecture that satisfies some constraints such as compression ratio. Then, in the second phase, the candidate architecture will be trained so that the final model can recover its original accuracy. An illustration of the flow is shown in Figure 3.1.

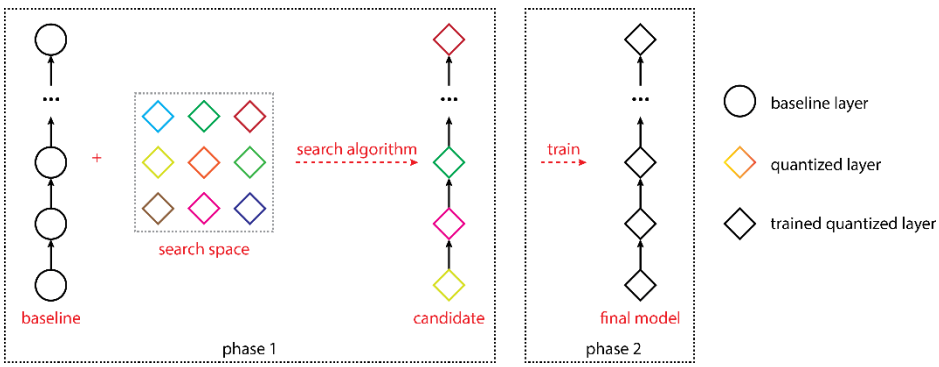

Figure 3.1. Mixed-precision neural architecture search. Search space contains many candidate quantized layers (different colors denote different quantization schemes)

One challenge of this problem is the exponentially large candidate configuration space. Provided that there are **N** layers in the baseline model and **M** quantization schemes in the search space, the candidate space will contain $\mathbf{M}^{\mathbf{N}}$ possible configurations (assuming that each layer can take any of **M** quantization schemes). Any brute force algorithm should go through all configurations to find the best one. This is impractical due to the fact that training a deep network is an expensive task. If each of **N** layers can take one of **M** quantization schemes, we have **N**×**M** candidate quantized layers. The impact of a quantization scheme on a layer is generally unknown until applied, thus a reasonable solution should go through all **N**×**M** candidates at least once. In that case, the complexity is now reduced linear time.

The proposed method is based on this observation to overcome the complexity challenge. While going through all **N**×**M** candidates, a metric is used to measure the impact of the current quantization scheme on the target layer. This metric we mentioned is simply the task performance. In case of image classification task, we measure the accuracy of the target model. In case of object detection task, performance is measured by mean average precision (mAP).

Applying quantization to a target layer can be considered as binding it with a handicapped condition. The idea of impact assessment is inspired by biological implants, in which a patient, having a part of his body replaced by a medical device, tries to adapt to it. The patient needs some time to recover from the surgery and to get along with the replaced part. Similarly, the network model with one or more quantized layers needs a sufficient training or fine-tuning to recover its original accuracy. Depending on how well the accuracy is recovered, decision to use a quantization scheme for a layer can be made.

A naïve method for impact assessment is presented in Algorithm 1. For each layer and each quantization scheme, we first quantize the layer with the selected scheme while keeping other layers unchanged. Next, retraining is performed on the new model with the task objective. Change in accuracy will then be measured on the obtained quantized model. Note that in the obtained model, all layers, except the selected layer for quantization, keep their baseline configuration (for example, if their parameters are originally represented by 32-bit floating-point numbers, they will still be 32-bit floating-point numbers). In some cases, it is also safe to “freeze” these layers during training and update only the selected layer. This method is illustrated in Figure 3.2.

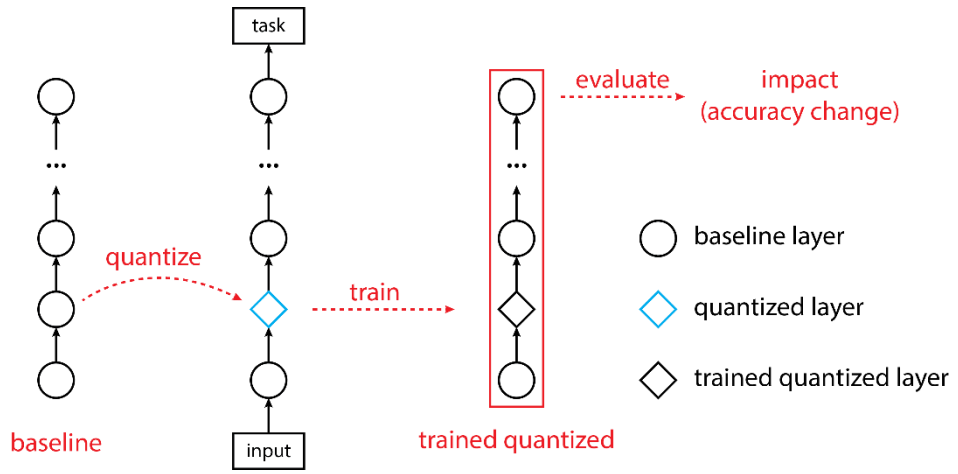

Figure 3.2. Impact assessment. This process is repeated **N**×**M** times

---

Algorithm 1: Naïve impact assessment

Input: network with **N** layers (L_1 to L_N) and **M** quantization schemes (Q_1 to Q_M)

initialize: an **N**×**M** accuracy table T

for i = 1 to **N do**

  for j = 1 to **M do**

    load baseline model

    quantize layer L_i with scheme Q_j

    train the new network model

    validate the model to obtain the accuracy $\mathbf{a}_{ij}$

    add the accuracy $\mathbf{a}_{ij}$ to the table T

Output: table of accuracy T

Since retraining is an expensive task, especially for very deep network, assessing all candidates is another challenge in the search process that the naïve method cannot

address. To overcome the expensiveness of training, we apply knowledge distillation to help the quantized model recover accuracy. The detail of this method is shown in the next section.

3.2. Impact assessment via Knowledge Distillation

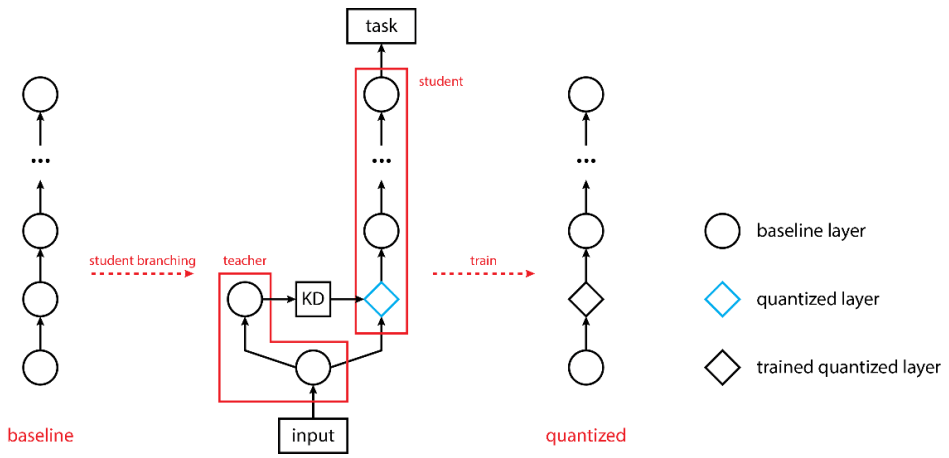

Figure 3.3. Impact assessment with distillation and task objective loss

In the impact assessment process, when an early layer (the layer following the input data for example) is quantized and retrained, the gradient vanishing problem is likely to happen. In order to prevent this, an objective function should be applied right after that layer. Since a baseline model is given, knowledge distillation is a good approach to do this. Without loss of generality, assuming that layer L_i is selected for impact assessment and L_{i-1} is prior to L_i such that the output of L_{i-1} is the (only) input of L_i, we add a branch so that the output of L_{i-1} flows to both quantized version of L_i and original L_i. Distillation loss is applied on the output of L_i and its quantized version. Output of the quantized layer continues to pass through the rest in the architecture. This student branch which contains quantized L_i is also trained on task objective

function. This idea is illustrated in Figure 3.3.

The previous idea, however, actually increases the expensiveness during training. To further optimize this process, we detach the task objective loss and train the network with only distillation loss. Output of the quantized layer does not need to pass through any other layers. This approach works due to the fact that we are training on a single layer (or a small part of the baseline network) and we expect the quantized network to have similar performance to its baseline. Figure 3.4 illustrates the idea.

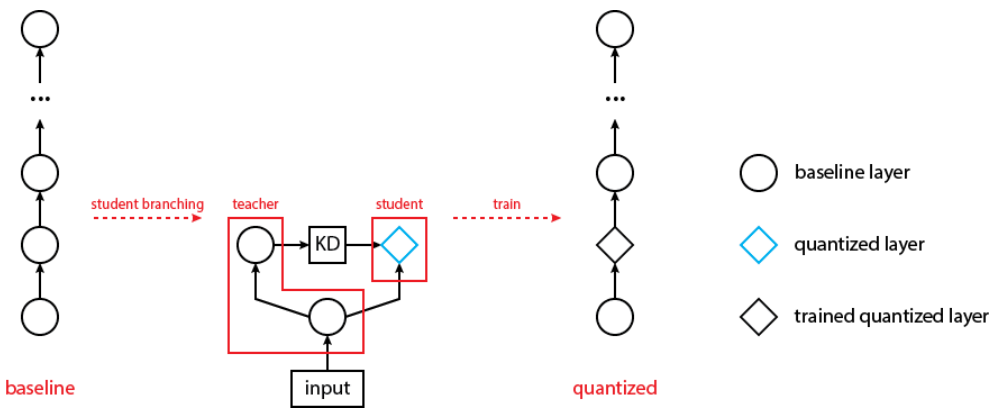

Figure 3.4. Impact assessment with only distillation loss

During training with knowledge distillation, output feature maps of the teacher and student layer are compared with each other. If the output feature map of the student layer is quantized, the difference loss is impossible to converge to zero in most cases. It is an unexpected behavior, thus, KD training setup for activation quantization should be configured in a different way. Based on a previous work [7], instead of calculating the difference between the baseline and the quantized output feature maps, loss is evaluated at the next few layers. That means, the quantized feature map is not necessarily the same as the original one but its succeeding output should be.

3.3. Parallel Knowledge Distillation training

In practice, accuracy recovery via KD can be accelerated with parallelization. If there are **M** options for each layer, it is possible to train all **M** quantized layers at once. If the training takes **K** iterations, individual training setup requires **K** forward passes through the teacher network and **K** forward-backward passes through the student network. Therefore, to train all **M** quantized layers, it requires **K**×**M** forward passes through the teacher and **K**×**M** forward-backward passes through the student. With this parallel setup, only **K** forward pass through the teacher is required, and the output feature map after each iteration can be reused to update all **M** layers at once.

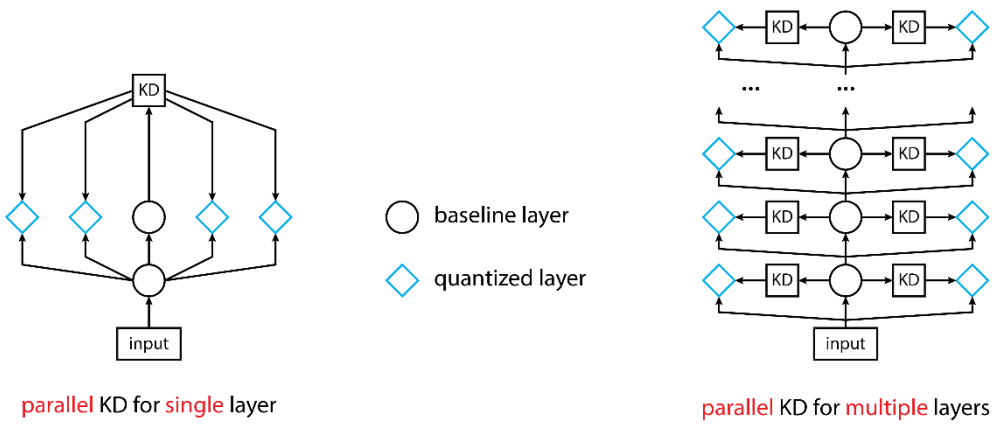

Figure 3.5. Parallel training setup for single layer (left) and multiple layers (right)

Parallelization can be applied for one or multiple layers as illustrated in Figure 3.5. The highest level is parallelization for all quantized layers in the network, which is similar to the super-net setup in DNAS. Generally, higher level of parallelization guarantees better data reuse. However depending on available resources, this can be done on small network only in most cases. For large network, one can design the setup such that the training process utilizes all the available resources for the highest

achievable efficiency. This proves that the proposed method is more flexible and scalable than previous works.

3.4. Candidate architecture generation

The previous phase, impact assessment, generates a table of accuracy. Table I is an example in which we quantize each of 9 residual blocks with one of 4 quantization schemes. Value at row **i** and column **j** denotes the accuracy of the model which has block **j** quantized with scheme **i**. This table shows the impact of each quantization scheme on each layer. Based on this information, we assign the scheme to each layer to generate a candidate architecture.

Table I. Impact-assessment table of ResNet20 model with 4 quantization schemes

block	1	2	3	4	5	6	7	8	9
1-bit	92.56	93.53	92.43	91.45	92.75	92.70	91.75	92.57	92.42
2-bit	92.70	93.73	92.76	92.19	92.92	92.97	92.36	92.88	92.83
3-bit	93.66	93.69	93.70	93.64	93.75	93.76	93.70	93.73	93.75
4-bit	93.74	93.80	93.77	93.78	93.81	93.78	93.80	93.82	93.72

In detail, KDMP adopts Greedy selection for scheme assignment. Note that, the candidate architecture should satisfy a given constraint. In our work, the constraint is compression ratio $\mathbf{C_W}$ of weights and/or $\mathbf{C_A}$ of activations, or in other words, the candidate architecture must have at least $\mathbf{C_W}$ times smaller model size and require $\mathbf{C_A}$ times less memory to store the activations. Before assigning, we calculate the efficiency score by equation (3). This score denotes how accuracy changes as a number of bits are truncated. Selection based on efficiency score tends to prioritize layers having high number of parameters or large activations maps.

$$\text{efficiency} = \frac{\Delta\text{accuracy}}{\Delta n_{\text{bit}}} \tag{3}$$

An example of selection for ResNet20 will be explained. We start with an initial assignment (step i = 1) where each layer is assigned with the highest-efficiency scheme as in Table II. In this step, residual blocks 1 to 8 are assigned with 4-bit schemes while block 9 has 3-bit. The selected schemes are highlighted in green. The compression ratio of this model is 8.59. We assume that this is not the desired compression ratio, thus, it is necessary to further reduce the bit-width for some layers. In Table II, the 4-bit scheme for block 9 (highlighted in red) is not selectable in the next step. The reason is that it has higher bit-width than the currently selected one. On the other hand, the scheme highlighted in yellow will be selected since it is the next highest-efficiency one.

Table II. Efficiency table and selection at step i = 1, compression ratio: 8.59

block	1	2	3	4	5	6	7	8	9
1-bit	-93.13	-21.12	-103.06	-58.68	-19.78	-20.73	-12.84	-5.77	-6.49
2-bit	-85.82	-6.55	-81.35	-41.69	-17.10	-16.14	-9.34	-4.47	-4.74
3-bit	-11.81	-9.41	-8.85	-4.65	-1.20	-1.08	-0.70	-0.42	-0.31
4-bit	-5.70	-0.50	-3.06	-0.85	0.02	-0.54	-0.04	0.05	-0.49

Table III shows the assignment of step i = 2 (with selected schemes highlighted in green). Similarly, compression ratio will be computed for this step. If the constraint is not yet satisfied, we keep on reducing bit-width by assigning block 7 with 3-bit scheme. The process is repeated until the generated candidate architecture reaches the desired compression ratio.

Table III. Efficiency table and selection at step i = 2, compression ratio: 9.28

block	1	2	3	4	5	6	7	8	9
1-bit	-93.13	-21.12	-103.06	-58.68	-19.78	-20.73	-12.84	-5.77	-6.49
2-bit	-85.82	-6.55	-81.35	-41.69	-17.10	-16.14	-9.34	-4.47	-4.74
3-bit	-11.81	-9.41	-8.85	-4.65	-1.20	-1.08	-0.70	-0.42	-0.31
4-bit	-5.70	-0.50	-3.06	-0.85	0.02	-0.54	-0.04	0.05	-0.49

It is noticeable that the proposed method ignores the dependence of a layer on the others. In addition, training with distillation loss alone causes the impact assessment result (or the accuracy) to fluctuate. Thus, the proposed method does not guarantee that the selection of all best-performed schemes will be the optimal assignment. Instead, that selection has high probability to perform as well as expected. In practice, it is better to generate a number of candidates by accepting small difference in efficiency score between two schemes.

3.5. Final model fine-tuning

In this final step, candidate architectures are trained or fine-tuned on the task objective. In the impact assessment phase of KDMP, the quantized layers are already trained up to a point, thus, the fine-tuning effort is reduced in this step. It is possible to form a half-trained model by combining the pre-trained weights of the quantized layers after the impact-assessment step. Figure 3.6 shows the overall of this process in KDMP. We observe that the model composed from high-loss schemes will require longer fine-tuning time while one composed from low-loss schemes requires less. In some extreme cases, the model may not need fine-tuning. This also implies that computational overhead in KDMP is reduced.

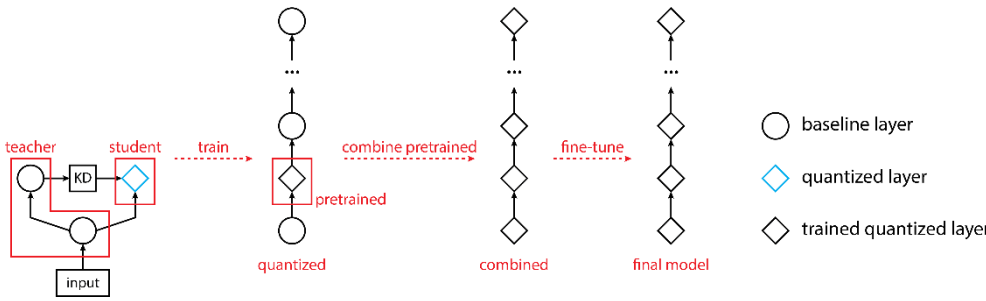

Figure 3.6. Training and reusing of weights in KDMP

3.6. Summary

We summarize the flow of KDMP framework in Algorithm 2. In overall, KDMP consists of 3 main steps. Models are half-way trained in the first step for the assignment in step 2. When the architecture has been decided, qualified candidates continue to be trained to reach their final accuracy. In the next chapter, we show that KDMP is flexible and efficient, yet still effective through various experiments.

Algorithm 2: Knowledge Distillation Mixed Precision framework

Input: network with **N** layers (L_1 to L_N) and **M** quantization schemes (Q_1 to Q_M)

initialize: an **N**×**M** accuracy table T and a set of weights S

/* Step 1: impact assessment */

for i = 1 to **N do**

    for j = 1 to **M do**

        load baseline model

        quantize layer L_i with scheme Q_j

        train only weights $\mathbf{w}_{ij}$ of the quantized layer L_i

        evaluate the model to obtain the accuracy $\mathbf{a}_{ij}$

        add the accuracy $\mathbf{a}_{ij}$ to the table T

        add the weights $\mathbf{w}_{ij}$ to set S

/* Step 2: candidate generation */

calculate efficiency score from table T

for i = 1 to N **do**

    assign highest-efficiency scheme Q_{j*} to L_i

while constraint is not satisfied

    select the next highest-efficiency scheme with reduced bit-width

/* Step 3: fine-tuning */

for i = 1 to **N do**

    load the corresponding pre-trained weights $\mathbf{w}_{ij}$ for layer L_i quantized with Q_j

fine-tune the architecture

Output: mixed-precision models

Chapter 4: Experimental Results

This chapter shows the experimental results of the proposed method. In each experiment, we describe our baseline model and search space before showing the results. We perform various experiments with different goals to prove the different advantages of using KDMP.

4.1. Final model fine-tuning time reduction

Table IV. Weight quantization on ResNet32

Model	Accuracy (%)	Compression (times)	Note	Training Effort (epochs)
baseline	95.02	1.00		-
KDMP_1	83.18	8.36	quantize without retraining	250
KDMP_1	94.54	8.36	combination of pre-trained weights	50
KDMP_1	94.69	8.36	final model	-
KDMP_2	33.85	12.55	quantize without retraining	400
KDMP_2	93.62	12.55	combination of pre-trained weights	150
KDMP_2	94.58	12.55	final model	-

In this experiment, we prove that KD training in the impact assessment is more than just a computational overhead. In other words, we show that the model weights have been half-way trained in this step, leading to less fine-tuning effort in the last step.

The model used in this experiment is ResNet32 on CIFAR-10 dataset. The baseline model is full-precision and the targets are weights of 15 residual blocks, each of

which contains 2 layers. Each block can take one of 8 quantization schemes which are combination of max and mean shifter factor and bit-width of 1 up to 4. In summary:

- Baseline model: full-precision ResNet32 on CIFAR-10
- Target: weights of 15 residual blocks
- Quantization schemes: {1, 2, 3, 4 bits} × {max shifter, mean shifter}

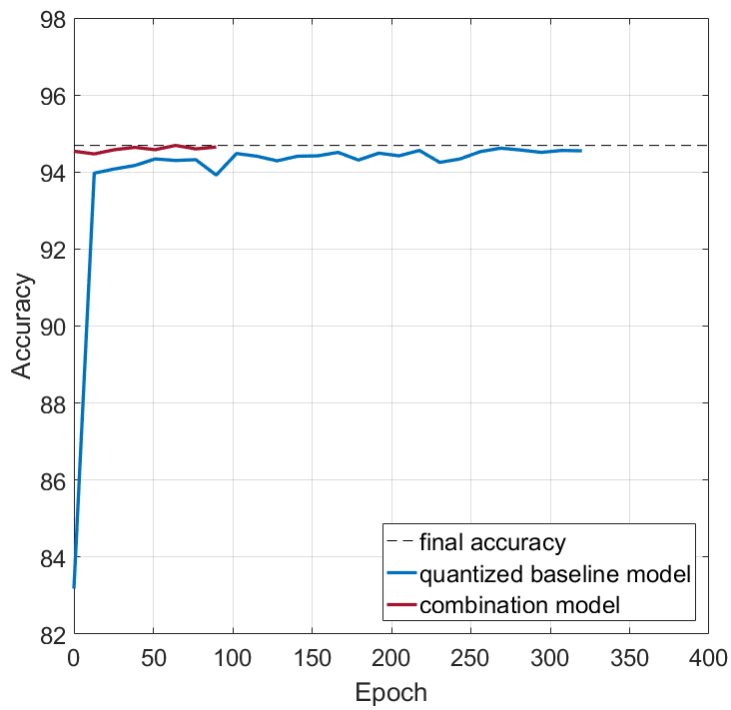

Figure 4.1. Convergence time comparison of pre-trained and non-pre-trained KDMP_1 models

The results are shown in Table IV. KDMP framework generates 2 candidate architectures with different compression ratios: KDMP_1 and KDMP_2. It is clear that the models composed from pre-trained weights have better initial accuracies

than those being quantized without retraining. The additional training effort for the pre-trained KDMP_1 to reach its final accuracy is 5 times less than its counterpart. Similarly, it takes 2.7 times less effort to fine-tune the pre-trained KDMP_2 model. The convergence time comparison of pre-trained and non-pre-trained KDMP_1 models is shown in Figure 4.1. It is also reasonable that the pre-trained KDMP_1 has higher initial accuracy than KDMP_2 as it is composed from low-loss schemes. In addition, it explained why the training effort of KDMP_1 is lower than KDMP_2.

4.2. Scalable and flexible training

This experiment shows that training with knowledge distillation is flexible. The model used in this experiment is Sim-YOLOv2-FPGA, which has been implemented on FPGA in [2]. Sim-YOLOv2-FPGA is a simplified version of YOLOv2 [5], having 17 convolutional (CONV) layers. In the implementation of this model on hardware, all CONV layers except the last one is represented by a single bit (in detail, the author used 1-bit mean shifter scheme). This binary-weight model is the baseline in this experiment. We perform uniform quantization on the activations with 4 bits and 6 different step sizes. The final model has each of 16 binary CONV layers assigned with a quantization scheme for activations. In summary:

- Baseline model: binary-weight Sim-YOLOv2-FPGA
- Target: activations of first 16 layers (last layer is kept unchanged)
- Search space: {4 bits} × {step 0.0625, 0.125, 0.25, 0.5, 1, 2}

Sim-YOLOv2-FPGA is a large model. The input to this network is 416 × 416 and training this network requires 9,167 MB GPU memory with a batch size of 32. Building and training the super-net with DNAS [1] consumes at least 6 times more

memory, which requires a lot of GPU resources. With KDMP however, the framework is able to utilize a single NVIDIA RTX 2080Ti GPU with 11GB memory to search for a mixed-precision architecture with high performance.

Table V shows two mixed-precision assignments that has better mean average precision (mAP) than the uniform-assignment models. Compared to the baseline model, we achieve 8 times activation compression with a degradation of 3.75 mAP. Note that we only use uniform quantization with a power-of-two hardware-friendly step size to make use shifting operation instead of multiplication. Our mixed-step model is also comparable to the mixed-precision-activation model in [2] which uses 6 bits for almost all layers. On the other hand, the best uniform-step 4-bit assignment achieves only 61.38 mAP, which is 2.49 mAP lower than our mixed-step model. The mixed-step model is clearly better without introducing any additional complexity.

Table V. Activation quantization on Sim-YOLOv2-FPGA

Model	**mAP**	**Compression (times)**	**Note**
32-bit activations	67.62	1.0	baseline model
4-6 bits activations [2]	64.16	5.5	
4-bit activations, step 0.0625	-	8.0	unable to converge
4-bit activations, step 0.125	-	8.0	unable to converge
4-bit activations, step 0.25	41.86	8.0	
4-bit activations, step 0.5	61.02	8.0	
4-bit activations, step 1	61.38	8.0	
4-bit activations, step 2	-	8.0	unable to converge
4-bit activations, mixed step	63.87	8.0	proposed method
4-bit activations, mixed step	62.50	8.0	selected by L2 loss

It is also worth to note that distillation loss can be used to assess the impact of a quantization scheme. In fact, distillation loss simply tells us how similar the output feature maps of the student network is to the ones of the teacher. In a perfect scenario, the loss is zero, meaning that the student branch has been able to mimic exactly the behavior of the teacher. In the experiment, we also select an assignment based on this loss. The obtained model works reasonably well with a mAP of 62.50, higher than uniform-assignment model but lower than the model selected based on accuracy. Currently, selection based on accuracy is preferred. Distillation loss is a promising and can be used in conjunction with accuracy, but we will leave this idea for the future work.

4.3. Effectiveness of Knowledge Distillation Mixed Precision

We perform weight quantization on full-precision ResNet20 on CIFAR-10 dataset. Except the first and last layer, 18 other layers are divided into 9 residual blocks. Following DNAS [1], we target only those 9 blocks such that different blocks may have different quantization schemes while layers in the same block have the same scheme. The search space contains 8 quantization schemes which are combination of max and mean shifter factor and bit-width of 1 up to 4. In summary:

- Baseline model: full-precision ResNet20 on CIFAR-10
- Target: weights of 9 residual blocks
- Quantization schemes: {1, 2, 3, 4 bits} × {max shifter, mean shifter}

In this experiment, we select 12 assignments targeting different compression ratios. Results are shown in Table VI. It can be seen from the table that mixed_5 is the most accurate model, achieving up to 93.97% on CIFAR-10. mixed_2 and mixed_11 are

two remarkable networks with good accuracy and compression ratio. Within the rest, some models have similar performance to the uniform-precision models while others perform worse. This is because of the fluctuation during training with KD.

Table VI. Weight quantization on ResNet20

Model	**Accuracy (%)**	**Weight Compression**	**Activation Compression**
baseline	93.81	1.00	1.00
uniform 1-bit max shifter	91.73	32.00	1.00
uniform 1-bit mean shifter	91.99	32.00	1.00
uniform 2-bit max shifter	92.24	16.00	1.00
uniform 2-bit mean shifter	92.86	16.00	1.00
uniform 3-bit max shifter	93.64	10.67	1.00
uniform 3-bit mean shifter	93.03	10.67	1.00
uniform 4-bit max shifter	93.94	8.00	1.00
uniform 4-bit mean shifter	92.80	8.00	1.00
mixed_0	92.81	17.51	1.00
mixed_1	93.03	16.00	1.00
mixed_2	93.77	14.28	1.00
mixed_3	93.73	11.90	1.00
mixed_4	93.78	9.23	1.00
mixed_5	93.97	9.19	1.00
mixed_6	93.42	14.97	1.00
mixed_7	92.34	17.35	1.00
mixed_8	93.71	9.14	1.00
mixed_9	93.67	8.67	1.00
mixed_10	93.63	10.03	1.00
mixed_11	93.83	13.65	1.00

We compare KDMP with other state-of-the-art methods such as DNAS. Results are shown in Table VII. Note that the proposed work uses Darknet framework [6] to train the model while DNAS and others use PyTorch [17] and Tensorflow [18]. There are some other mismatches leading to the difference of two baseline models. However, the results are still comparable in terms of accuracy loss. In addition, our baseline model has higher accuracy than the other, implying that our models are better converged. The experiment is further extended to activation quantization with 9 options, which use 4, 6 and 8 bits combined with power-of-two step sizes.

Table VII. Weight quantization on ResNet20

Model	**Accuracy (%)**	**Weight Compression**	**Activation Compression**
baseline	92.35	1.00	1.00
DoReFa [15]	89.90	10.67	10.67
PACT [16]	91.10	10.67	10.67
DNAS most accurate [1]	92.72	11.60	1.00
DNAS most efficient [1]	92.00	16.60	1.00
HAWQ [13]	92.22	13.11	8.00
LQ-Nets [12]	92.00	10.67	1.00
our baseline	93.81	1.00	1.00
mixed_5	93.97	9.19	1.00
mixed_2	93.77	14.28	1.00
mixed_2_act	93.46	14.28	7.03
mixed_11	93.83	13.65	1.00
mixed_11_act	93.62	13.65	7.03

While using only simple quantization methods as candidates, our mixed_11_act model has weight compression and activation compression ratio of 13.65 and 7.03

times respectively. The accuracy loss of this model is 0.19%, comparable to HAWQ [13] with similar compression ratio and 0.13% accuracy loss. It is also safe to imply that our model is better than DoReFa [15], PACT [16] and LQ-Nets [12].

4.4. Intra-layer mixed precision

We perform an extra experiment with KDMP framework in which we assign different quantization schemes to different weight channels. In this experiment, each weight tensor is divided into multiple groups. The division strategy is also simple such that weights are grouped by the output-channel indices as illustrated in Figure 4.2. Using simple division strategy highlights the effectiveness of KDMP in searching and generating mixed-precision architecture. In addition, the implementation requires no indexing to find which group uses what bit-width.

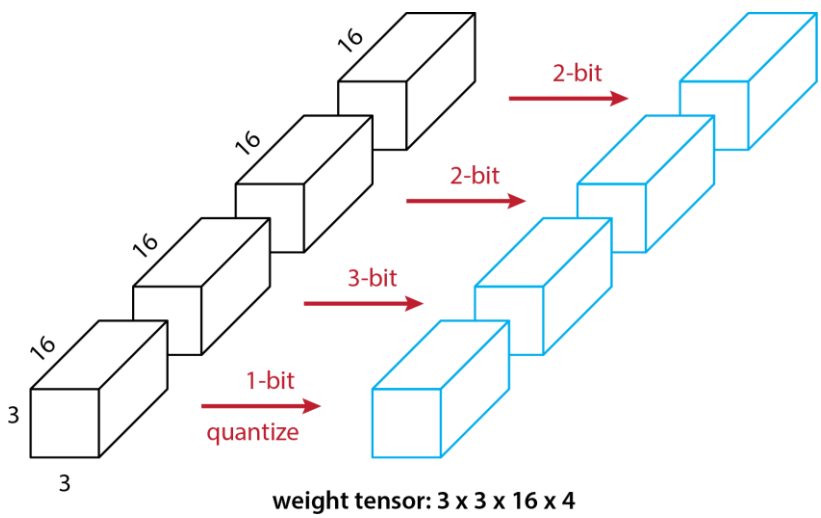

Figure 4.2. Intra-layer weight division and quantization

All settings for this experiment is the same with the effectiveness experiment. We use full-precision ResNet20 as baseline model and target 36 groups of weights of 9

residual blocks. Each group has 8 options which are combination of max and mean shifter and bit-width of 1 up to 4. In summary:

- Baseline model: full-precision ResNet20 on CIFAR-10
- Target: groups of weights of 9 residual blocks
- Quantization schemes: {1, 2, 3, 4 bits} × {max shifter, mean shifter}

The results in Table VIII shows that intra-layer mixed-precision can further increase the compression ratio with negligible accuracy loss. In detail, we generate a model with almost the same accuracy, 93.75% compared to 93.77% of mixed_2. However, the intra-layer mixed-precision model has higher weight compression ratio in times, 14.85 compared to 14.28%. The trade-off is 0.02% accuracy loss for 0.57 more unit of compression (times).

Table VIII. Intra-layer weight quantization on ResNet20

Model	**Accuracy (%)**	**Weight Compression (times)**	**Note**
baseline	93.81	1.00	
mixed_2	93.77	14.28	(inter-layer) mixed-precision
intra_mix	93.75	14.85	intra-layer mixed-precision

Going deeper into the network architecture for change is an interesting and promising approach. However, we realize that further splitting small weights into groups might cause negative effects due to the hidden interaction between weights. As KDMP algorithm assumes that the dependence between layers is minimal, units for applying mixed-precision should not be too small to make sure that each of them have contribution to the model accuracy.

Chapter 5: Conclusion and Future Work

In this work, we propose Knowledge Distillation Mixed Precision (KDMP) framework to solve the mixed-precision problem. The KDMP framework efficiently trains all candidate quantized layers to evaluate the impact of each quantization scheme. Qualified candidate architectures are then generated by Greedy algorithm and fine-tuned on the task objective. KDMP applies knowledge distillation in training to accelerate the search process while removing task loss further reduces the expensiveness of training. KDMP also allows multiple candidates to be trained in parallel. One can flexibly set up the training to utilize the available resources. Trained weights can be reused in fine-tuning step to cut down time and effort in reaching the final accuracy. Experimental results prove all the listed advantages of the proposed framework. In addition, KDMP has comparable effectiveness to other state-of-the-art methods in generating highly-accurate, low-size and hardware-friendly models.

Despite being efficient and effective, we are aware of the imperfection in the KDMP framework. Currently in the impact assessment, training quantized layers is unstable which results in slight difference between two runs and further leads to the generation of different candidate architectures. Stabilizing training may be done with selective distillation such that the student layer mimics the output of the corresponding teacher layer only if the teacher network predicts the input correctly. Another problem lies within the assignment strategy. Reinforcement learning can be adopted at the selection step so that the dependence between layers is considered. The imperfections are left for the future work of this research.

Appendix

A. Experiment with Sim-YOLOv2-FPGA on VOC dataset

The detailed impact assessment is showed in table A.1. All quantization schemes are 4-bit uniform with different step sizes. They are applied on activations.

Table A.1. Impact of 6 quantization schemes on 16 layers of Sim-YOLOv2-FPGA

Layer/Step	**0.0625**	**0.125**	**0.25**	**0.5**	**1**	**2**
1	32.37	54.64	54.58	61.78	55.82	15.10
2	1.49	13.15	37.26	63.54	64.60	62.56
3	22.41	56.27	62.04	65.36	65.32	48.87
4	0.02	4.14	41.68	65.57	64.50	51.80
5	1.27	0.79	11.44	58.02	63.39	48.33
6	14.30	54.28	63.78	66.58	57.67	6.06
7	2.04	17.60	54.43	66.48	62.84	33.40
8	3.00	16.90	58.72	64.89	52.13	8.70
9	50.93	44.12	59.78	63.46	11.94	0.06
10	29.55	58.16	65.07	66.84	60.16	17.43
11	43.60	59.81	65.86	61.49	36.25	4.18
12	25.24	57.47	64.95	66.31	57.99	12.32
13	39.92	51.92	62.50	64.89	28.56	0.03
14	51.52	62.44	64.78	61.92	42.17	0.00
15	44.83	56.03	65.61	65.66	52.40	14.27
16	33.61	52.25	64.97	60.99	56.63	28.36

The final selected configuration is:

- Layer 11, 14, and 16: 4-bit uniform step 0.25

- Layer 2 and 5: 4-bit uniform step 1
- Others: 4-bit step 0.5

B. Experiment with ResNet20 and ResNet32 on CIFAR-10

We set-up a parallel knowledge distillation (KD) training for ResNet20 and ResNet32. The original training set of CIFAR-10 contains 50,000 images. In our work, we use the whole training set for training and the test set of 10,000 images is used to measure the impact. For official benchmarking however, this training set should be divided into two sets, training and validation with 45,000 and 5,000 images respectively. The impact will be measured on 5,000-image validation set. After generating some candidates, models are fine-tuned on the training set and report the accuracy on test set of CIFAR-10.

KD training uses ADAM optimization. Batch size can be set to 32 or 128 images. Quantized layers are trained with KD for 13 epochs.

Reference

[1] Wu, Bichen, 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] Nguyen, Duy Thanh, Tuan Nghia Nguyen, Hyun Kim, and Hyuk-Jae Lee. "A High-Throughput and Power-Efficient FPGA Implementation of YOLO CNN for Object Detection." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2019).

[3] Rastegari, Mohammad, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. "Xnor-net: Imagenet classification using binary convolutional neural networks." In *European Conference on Computer Vision*, pp. 525-542. Springer, Cham, 2016.

[4] Zhou, Aojun, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. "Incremental network quantization: Towards lossless cnns with low-precision weights." *arXiv preprint arXiv:1702.03044* (2017).

[5] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.

[6] Redmon, Joseph. "2016. Darknet: Open Source Neural Networks in C." (2013).

[7] Yu, Joonsang, Sungbum Kang, and Kiyoung Choi. "Network Recasting: A Universal Method for Network Architecture Transformation." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5701-5708. 2019.

[8] Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement

learning." *arXiv preprint arXiv:1611.01578* (2016).

[9] Pham, Hieu, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. "Efficient neural architecture search via parameter sharing." *arXiv preprint arXiv:1802.03268* (2018).

[10] Liu, Hanxiao, Karen Simonyan, and Yiming Yang. "Darts: Differentiable architecture search." *arXiv preprint arXiv:1806.09055* (2018).

[11] Park, Eunhyeok, Sungjoo Yoo, and Peter Vajda. "Value-aware quantization for training and inference of neural networks." In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 580-595. 2018.

[12] Zhang, Dongqing, 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)*, pp. 365-382. 2018.

[13] Dong, Zhen, Zhewei Yao, Amir Gholami, Michael Mahoney, and Kurt Keutzer. "HAWQ: Hessian AWare Quantization of Neural Networks with Mixed-Precision." *arXiv preprint arXiv:1905.03696* (2019).

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

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

[16] Choi, Jungwook, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang,

Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. "Pact: Parameterized clipping activation for quantized neural networks." *arXiv preprint arXiv:1805.06085* (2018).

[17] Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch." (2017).

[18] Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin et al. "Tensorflow: A system for large-scale machine learning." In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265-283. 2016.

초 록

Quantization은 메모리와 계산 능력이 제한된 edge device에서 deep neural network를 수행하기 위해 필수적인 프로세스이다. 그러나 bit-width 및 transformation function을 포함하여 똑같은 quantization 을 모든 레이어에 그대로 적용하는 uniform-precision quantization은 심각한 성능 저하를 겪는 것으로 널리 알려져 있다. 한편 각각의 레이어에 서로 다른 quantization 을 찾아서 적용하는 것은 후보군의 수가 레이어 수에 따라 기하급수적으로 증가하기 때문에 적용하기 어렵다. 이 문제를 해결하기 위해, 본 연구에서는 Knowledge Distillation 기법을 활용하여 선형시간 내에 검색 공간을 효율적으로 탐색하는 방법을 제안한다. 특히, 제안된 방법은 대상 레이어에 대한 quantization 의 영향을 추정하기 위해 레이어별로 loss function을 공식화하였다. 레이어간에 의존성이 최소라는 가정에 근거하여 각 레이어별 적용을 개별적으로 결정해서 성능의 손실을 최소화한다. 하드웨어 친화적인 quantization 만 사용하여 image classification과 object detection 에 대한 실험을 실시하였다. 그 결과, CIFAR-10 데이터 세트에서 크기가 13.65배 작은 가장 효율적인 mixed-precision의 ResNet20 모델로 93.62%의 정확도를 달성하였으며, 이는 기본 full-precision 모델보다 0.19% 낮은 수준이다. VOC 데이터 세트에서, 제안된 방법은 평균 precision 이 63.87% 인 mixed-precision Sim-YOLOv2-FPGA 모델을 생성하였고, 동일 압축률의 모든 uniform-precision 모델보다 성능이 뛰어나다. 제안하는 방법은 다른 최첨단 mixed-precision quantization 접근법과 유사한 효율을 달성하면서도 실행은 단순하다.

주요어: Neural Network Quantization, Mixed-precision, Knowledge Distillation, Neural Architecture Search, Deep Learning, AI Accelerator

학 번: 2018-21525

Acknowledgement

I would like to express my sincere gratitude to my advisor, Prof. Lee Hyuk Jae for his kind and continuous support of my study and research.

From the deepest of my heart, I would like to say thank you to my family, my girlfriend and my friends for the encouragement, understanding and sympathy.

I also want to thank all members of CAPP Lab. The working environment that CAPP Lab provided fits me well, and the support from my fellows are just great.

Last but not least, I would like to thank my thesis reviewers, Prof. Kim Tae Hwan and Prof. Ryu Soo Jung, for giving me valuable comments on my research.