Hardware-Aware Mixed-Precision Neural Networks using In-Train Quantization

Manoj Rohit Vemparala*[1]
manoj-rohit.vemparala@bmw.de
Nael Fasfous*[2]
nael.fasfous@tum.de
Lukas Frickenstein*[1]
lukas.frickenstein@bmw.de
Alexander Frickenstein[1]
alexander.frickenstein@bmw.de
Anmol Singh[1]
anmol.singh@bmw.de
Driton Salihu[2]
driton.salihu@tum.de
Christian Unger[1]
christian.unger@bmw.de
Naveen Shankar Nagaraja[1]
naveen-shankar.nagaraja@bmw.de
Walter Stechele[2]
walter.stechele@tum.de

[1] BMW AG
Munich, Germany

[2] Chair of Integrated Systems
Technical University of Munich
Munich, Germany

Abstract

Fixed-point quantization is an effective method to reduce the model size and computational demand of convolutional neural networks, by lowering the numerical precision of all layers down to a specific bit-width. Recent work shows assigning layer-wise specific bit-widths has an advantage over uniform assignments, although requiring complex, post-training search techniques and many GPU hours to identify the optimal bit-width strategy. To alleviate this, we propose an in-train quantization method that can directly learn the optimal bit-widths for weights and activations during the gradient-based training process. We incorporate hardware-awareness into the gradient-based optimization to directly improve the real hardware execution metrics. We replace the discrete and non-differentiable hardware measurements with a differentiable Gaussian process regressor. This provides accurate hardware predictions as an auxiliary loss to the gradient-descent optimizer, performing hardware-friendly in-train quantization. Our hardware-aware mixed-precision ResNet56 achieves an improvement of $1.3\times$ in execution latency compared to the uniform 4-bit quantization with no degradation in accuracy. Finally, we highlight the effectiveness of the in-train quantization method in the context of adversarial training, improving the trade-off between prediction accuracy and robustness.

* indicates that the authors have contributed equally.

1 Introduction

Convolutional neural networks (CNNs) deployed in real-time, computer-vision applications are strictly constrained in terms of inference latency, energy consumption, and model size. Model compression techniques such as quantization [3, 8, 17], pruning [9, 13, 23], and knowledge distillation [14] reduce the memory footprint of CNN models and speed up their computation. Quantization, in particular, has become a standard technique in both industry [10, 24] and academia [3, 22], typically applied before deploying CNNs in embedded settings. The benefits of quantization are manyfold, ranging from reducing the bit-width of weights and activations to shrink the model's size, to simplifying the arithmetic computation units on hardware (HW) and lowering the energy consumed by on-chip and off-chip data movement [2].

Quantization is typically applied as a post-training calibration method, which takes in a full-precision pre-trained model and compresses it to a lower bit-width representation with iterative steps of fine-tuning. Alternatively, quantization-aware training (QAT) methods are capable of producing quantized CNNs *during* the training process [3, 30]. These are typically uniform in terms of bit-width assignment, fixing the entire CNN's representation to a pre-determined number of bits. However, different layers contribute differently to the accuracy and efficiency of a network [8], justifying the use of different quantization degrees for different layers of the CNN. To obtain HW-friendly layer-wise quantization strategies, post-training quantization methods use search techniques, such as reinforcement learning (RL) [25] or evolutionary search (ES) [26], bringing back the costly post-training GPU hours.

In this paper, we reduce the execution metrics for model inference by searching for optimal bit-widths directly *during* the training process and produce **dominant solutions** in terms of prediction accuracy and model complexity, when compared to uniform bit-width assignments and post-training methods. We summarize the key contributions of this work as follows:

1. We introduce a novel training scheme which jointly learns the model parameters and the number of unique values required to represent weights and activations for all the layers, thereby identifying optimal bit-width assignments. Compared to the uniform 4-bit quantization, our approach reduces the number of bit operations (BOPs) by 1.5× for ResNet56 with minimal accuracy degradation on CIFAR-100 dataset.

2. We propose a novel gradient-based approach to append a differentiable auxiliary HW-loss objective, constraining real hardware metrics such as latency and memory accesses, using Gaussian process regression. With no degradation in task accuracy, our approach reduces the inference latency by 1.3×, compared to a uniformly quantized ResNet56.

3. We show that our quantization scheme can be infused with adversarial training, improving robustness by 1.4 pp and reducing the number of bit operations by 1.9×, when compared to a uniformly quantized ResNet56.

2 Related Work

2.1 Quantization-Aware Training

By limiting the weights and activations of CNNs to a constrained set of values, it becomes increasingly challenging to use the standard stochastic gradient descent (SGD) to update the CNN parameters during backpropagation. DoReFa-Net [30] adopts the straight-through

estimator (STE) [1] and bounds the magnitude of latent weights and activation between [0, 1]. The work in PACT [3] improves the training procedure of QNNs by learning the optimal clipping level for the activations of each layer. The dynamic clipping function allows larger representational capability than DoReFa-Net, thereby increasing the prediction accuracy. The work in LSQ [7] parameterizes the step size instead of clipping point to further improve the prediction accuracy. ABC-Nets [17] introduce multi-bit networks by approximating a full-precision convolution using multiple bases of binary weights and activations. The aforementioned methods use uniform quantization for all layers and do not directly consider the latency benefits on target HW.

2.2 Mixed-Precision Compression

ReLeQ [6], HAQ [25] and AutoQ [19] propose reinforcement learning-based exploration schemes to determine HW-aware layer-wise quantization strategies. ReLeQ searches for optimal bit-widths only for the weights of each layer, while HAQ searches for both weights and activations. AutoQ determines a fine-grained quantization strategy for each filter in every layer. The reward function is evaluated after executing the inference of the quantized CNN on a target HW. This involves finding a bit-width strategy in a large search space, demanding high training effort due to the iterative fine-tuning of every solution during the exploration. In this work, we find the optimal quantization strategy *during* the CNN's training procedure, circumventing the need for fine-tuning steps to evaluate different quantization strategies.

WaveQ [5] formulates a gradient-based optimization problem by introducing a sinusoidal regularization loss, pushing the weights to optimal quantization levels. However, the quantization level of activations is set uniformly. Wu et al. [28] learns quantization levels through a path selection-based neural architectural search formulation. This method is challenging to scale for larger CNN models, as the super-network leads to larger search and training costs. The work in LBS [18] alleviates the compute cost by devising a single-path scheme that captures different quantization and filter pruning strategies using binary gates. The work in [21] investigates a suitable parameterization of the quantization operation to learn bit-widths and avoid unbounded gradient updates. In particular, the authors learn step-size and dynamic range for quantized weights and activations to determine the optimal quantization strategy. Different to [5, 18, 21, 28], we learn the number of unique values required to represent weights and activations by progressively reducing the bit-widths using a differentiable loss formulation, capturing HW-awareness in the process.

3 In-Train Mixed Precision Quantization

Without loss of generality, the activation feature map $A^{l-1} \in \mathbb{R}^{X_i \times Y_i \times C_i}$ is considered as the input to a convolutional layer $l \in [1,...,L]$, where X_i, Y_i and C_i describe the dimensions of width, height and input channels. A^0 and A^L are the input image and the prediction of the CNN, respectively. The weight matrix $W^l \in \mathbb{R}^{K_x \times K_y \times C_i \times C_o}$ consists of kernels of shape $K_x \times K_y$, and C_o output channels. A convolution operation of tensors W^l and A^{l-1} results in the output $A^l \in R^{X_o \times Y_o \times C_o}$. The output features of the final layer A^L can be used for the computation of the task-specific accuracy ψ, by comparing it to the dataset labels.

Every layer l is associated with weight and input activation bit-widths, represented as b_w^l and b_a^l respectively. The optimized CNN can be obtained by selecting the optimal quantization tuple $(b_w^l, b_a^l)\ \forall\ l$. In this paper, we target two objectives: (1) reducing the bit-width of weights

and activations during the training process to lower the computational complexity of a neural network, and (2) improving the HW-awareness by directly finding the quantization strategy based on real HW metrics, such as latency and memory accesses. Both can be efficiently achieved by formulating a joint optimization problem as shown in Fig 1.

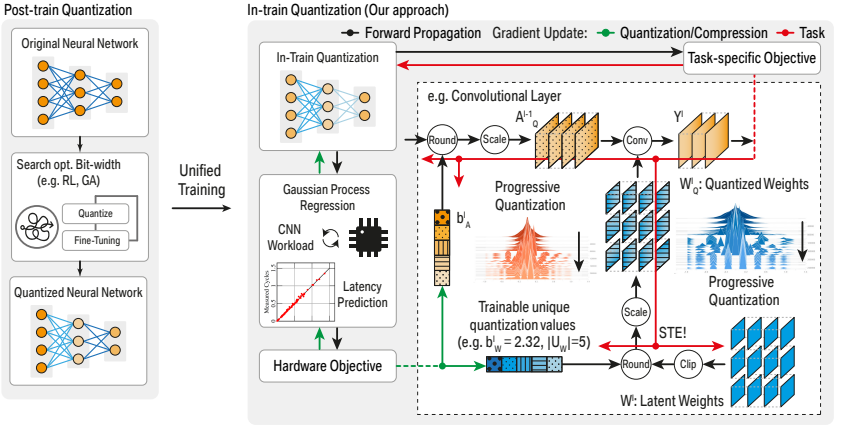


Figure 1: Depiction of post-train quantization approaches (left) in comparison to the proposed approach (right). Optimal precisions are determined through progressive quantization.

3.1 Trainable Bit-Widths for Mixed-Precision Quantization

We aim to obtain an efficient quantization strategy directly when training the network's weights W to circumvent any additional effort of post-train quantization search. More generally, b is the bit-width of the quantized datatype, weights or activations. The real-valued data are represented by 2^b unique values in the fixed-point quantization domain. The mapping of a data element x (weight or activation) onto a quantized value $\mathtt{x_q}$ is expressed in Eq. 1. This is similar to the linear quantization methods proposed in [3, 30]. Firstly, x is clipped between $[-c,+c]$, where c is a trainable variable for every layer and is determined by the task-specific loss function of the CNN model [3]. Based on the determined c for a given datatype, we define a scaling factor $s = c/(2^{b-1}-1)$. For activations, we clip the values between the range of $[0,+c]$, instead of $[-c,+c]$, due to the non-linear activation function (ReLU). $\mathtt{x_q}$ approximates the continuous domain of $\mathtt{x}$ into the discrete values $\mathtt{x_q} \in \{0,\ s,\ 2s,\ 3s, ...,\ (2^b-1)\cdot s\}$. During backpropagation, the gradient of the *Round* operation vanishes, therefore we estimate it in order to update the real-valued weights during the training phase. In the simplest case, the estimated gradient g_x could be obtained by replacing the derivative of *Round* with the identity function (see Eq. 1). This is referred to as the straight-through estimator (STE) [1].

$$\mathtt{x_q} = Round(\mathtt{Clip}(x,0,c)\cdot\frac{(2^b-1)}{c})\cdot\frac{c}{(2^b-1)} \quad ; \quad g_x \overset{\text{!STE}}{=} g_{\mathtt{x_q}}\cdot 1_{x\leq c} \tag{1}$$

Varying the number of bits b of each datatype, for every layer, can change the prediction accuracy ψ by several percentage points. One way to determine the best configuration is to perform exploration using an RL-agent or an evolutionary search algorithm [25, 26]. However, this search could lead to excessive GPU hours, given the need for iterative fine-tuning for every exploration step. This motivates *learning* the most efficient allocation of datatype precision of each layer during the training process itself. There are challenges to achieve this task due to

two important reasons: (1) Devising a training scheme which directly changes the bit-width b could lead to sudden fluctuations in the discrete weight distribution, resulting in unstable gradient updates, (2) The bit-width b only considers integer values, e.g. $b \in [1,2,3,...,8]$ for an accelerator supporting maximum of 8-bit fixed-point multiplications. Using another STE to round the parameters in the forward pass while retaining the original float values in the backward propagation could lead to further gradient approximation, producing sub-optimal prediction accuracies ψ. We tackle these challenges by determining the set of unique values U required to represent all x, as shown in Eq. 2.

$$\begin{aligned} \tilde{\mathtt{x}}_{\mathtt{q}} &= Round(\mathtt{Clip}(x,0,c)\cdot\frac{|U|}{c})\times\frac{c}{|U|} \\ g_{|U|} &\overset{\text{!STE}}{=} g_{\tilde{\mathtt{x}}_{\mathtt{q}}}\cdot\left(\frac{-c}{|U|^2}\cdot Round(\mathtt{Clip}(x,0,c)\cdot\frac{|U|}{c})+\frac{\mathtt{Clip}(x,0,c)}{|U|}\right) \end{aligned} \tag{2}$$

We avoid using another gradient approximation by allowing the cardinality $|U|$ to be a real-valued trainable parameter. Note that our approach in Eq. 2 differs from Eq. 1 by not restricting the values to an integer number b in 2^b, allowing $|U|$ to have an arbitrary number of unique elements to represent the trainable parameters. We introduce a hyperparameter $E_{\text{Quant, Start}}$ which represents the epoch at which the learning process for the bit-widths is started. This allows smooth, float-point values for the size of $|U|$, progressively lowering the *projected* cardinality of U for each layer's datatypes, until $E_{\text{Quant, Stop}}$ is reached. The number of unique values $|U|$ is updated by cross-entropy loss $\mathcal{L}_{ce}$ based on the gradient $g_{|U|}$ as shown in Eq. 2. The unique values in U required to represent x increases based on the rounding error $(\frac{x\cdot|U|}{c} - Round(\frac{x\cdot|U|}{c}))$. The HW loss objective $\mathcal{L}_{HW}$ is captured by fractional bit-widths (e.g. 3.5-bits) during the initial stages of training (i.e. between $E_{\text{Quant, Start}}$ and $E_{\text{Quant, Stop}}$ epochs). This also leads to progressive quantization which lowers the gradient approximation at the initial stages of the training, thereby improving the learning capability of the neural network. We gradually determine the optimal bit-widths based on $\mathcal{L}_{ce}$ and $\mathcal{L}_{HW}$ objectives as we approach the end of the training. We round the number of unique values $|U|$ to the nearest power-of-two in the middle of training process ($E_{\text{Quant, Stop}}$), deriving the optimal bit-width b as $Round(\log_2|U|)$.

We define the constrained loss function $\mathcal{L}_{HW}$ as shown in Eq. 3, to account for HW-specific compression objectives. b^* and b^{max} represent constrained and maximum supported bit-widths, respectively. The inference complexity of the CNN depends on the number of bits assigned to the weights b_w and activations b_a for each layer $l \in [1,...,L]$. We represent the workload shape of layer l as l_{shape} and HW inference complexity as a function of l_{shape}, b_w^l and b_a^l given as $\varphi_l(l_{\text{shape}}, b_w^l, b_a^l)$. We use the scaling factor v to control the convergence speed for the optimal quantization strategies $\{(b_w^1,b_a^1),...,(b_w^L,b_a^L)\}$ during the training process. We conduct an ablation study in Sec. 4.1 to determine the optimal scaling v.

$$\begin{aligned} \mathcal{L}_{\text{total}} &= \mathcal{L}_{ce} + \mathcal{L}_{reg} + v\times\mathcal{L}_{HW} \\ \mathcal{L}_{HW} &= max\Big(\frac{\sum_{l=1}^{L}(\varphi_l(l_{\text{shape}},b_w^i,b_a^i) - \sum_{l=1}^{L}\varphi_l(l_{\text{shape}},b_w^*,b_a^*))}{\sum_{l=1}^{L}(\varphi_l(l_{\text{shape}},b_w^{max},b_a^{max}))},0\Big) \end{aligned} \tag{3}$$

Decreasing the number of bits b_w^l and b_a^l leads to a lower number of BOPs for a given layer l, as defined in Eq. 4.

$$\varphi_{\text{BOPs}}^l = X_o^l \times Y_o^l \times K_x^l \times K_y^l \times C_i^l \times C_o^l \times b_w^l \times b_a^l \tag{4}$$

3.2 Differentiable Hardware-Awareness

Many HW-aware compression works use hardware-in-the-loop setups and/or hardware-measurement look-up tables to integrate the actual HW performance into the optimization loop [13, 25, 26, 29]. Although this approach is valid for agents which require a simple reward value for their NN optimization decisions, it cannot be extended to the gradient-descent optimization used in this work for in-train quantization. For the training optimizer to work seamlessly with HW-based loss minimization, the HW measurements need to be provided through a differentiable function. By nature of a differentiable function providing the HW-estimates, intermediate values for quantization can also be supported during the in-train quantization's progressive bit-width reduction. For example, 3.5-bits does not reflect any executable computation bit-width on real HW. During smooth, progressive in-train quantization, such bit-widths may appear as described in Sec. 3.1, which need to have a sensible loss value associated with them to guide the gradient-descent-based training optimizer.

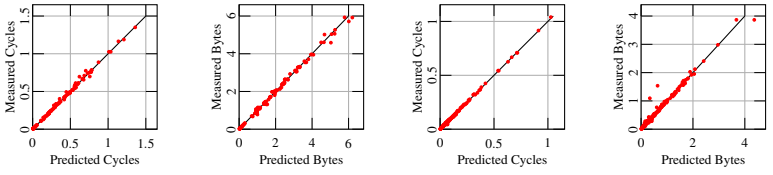
Gaussian process (GP) regression provides the means to construct a differentiable HW estimator. This injects HW-awareness into the chain-rule for the SGD, allowing it to set the layer-wise quantization values $(b_w^l, b_a^l)\ \forall\ l$. A GP prior is trained on measurements φ_{HW} collected on real HW, with respect to different computation workloads ρ. Starting with the covariance matrix K shown in Eq. 5, we use a squared exponential kernel, inspired by the approach in [4]. σ and ℓ represent the amplitude and lengthscale of the GP's kernel, respectively. ρ indicates the workload features, i.e. the convolutional layer's dimensions and bit-widths. Considering a general matrix multiplication (GEMM) execution of a convolutional layer, ρ is the vector of features representing rows, columns and inner product (depth) of the matricies, as well as the bit-widths of weights and activations.

$$K(\rho,\rho') = \sigma^2 exp(-\frac{||\rho-\rho'||^2}{2\ell^2}),\ \text{where}\ \rho = (row, depth, col, b_w, b_a)$$
$$\varphi_{HW} \sim \mathcal{GP}(m(\rho), K(\rho,\rho')) \tag{5}$$

Based on the GP prior in Eq. 5, a predictive function can also be described by a mean and a covariance matrix. To guarantee the GP regressor is differentiable, we must assert that the covariance function K is differentiable. This condition is fulfilled by our squared exponential kernel, as shown in Eq. 6.

$$\frac{\partial K(\rho,\rho')}{\partial\rho\partial\rho'} = \frac{\sigma^2}{\ell^4}(\ell^2 - (\rho-\rho')^2) exp(-\frac{||\rho-\rho'||^2}{2\ell^2}) \tag{6}$$

The GP regressor's HW predictions φ_{HW} can be used in the HW loss formulation $\mathcal{L}_{HW}$, presented in Eq. 3. The differentiable GP regressor provides $\frac{\partial\varphi}{\partial\rho}$ during backpropagation, which links in the chain-rule, allowing the in-train quantization SGD to manipulate the $(b_w^l, b_a^l)\ \forall\ l$ through the ρ gradients, thereby minimizing the latency and/or DRAM accesses of the inference execution on HW. In Fig. 2, we present the performance of the GP regressor on unseen validation workloads from ResNet20-CIFAR and ResNet18-ImageNet, with varying b_w^l, b_a^l. The high-accuracy of the HW measurement predictions can clearly guide the in-train quantization algorithm to make decisions on minimizing the HW loss $\mathcal{L}_{HW}$, which reflect real reductions in latency and DRAM accesses on the final HW accelerator.



(a) ResNet-CIFAR Latency (b) ResNet-CIFAR DRAM (c) ResNet-ImageNet Latency (d) ResNet-ImageNet DRAM

Figure 2: Validating the performance of the GP regressor on unseen CNN workloads from ResNet20-CIFAR and ResNet18-ImageNet for prediction of HW latency and DRAM accesses.

4 Experiments

We evaluate the proposed in-train quantization technique on CIFAR-10, CIFAR-100 [15], and ImageNet [16] datasets. We use ResNet20 and ResNet56 as baseline models for the CIFAR-10, CIFAR-100 datasets, and ResNet18 as a baseline model for the ImageNet dataset. In all the experiments, we set the first and last layer as 8-bit quantization. If not otherwise mentioned, all hyper-parameters specifying the task-related training were adopted from ResNet's base implementation [12]. To train the GP regressor presented in Sec. 3.2, latency and DRAM accesses measurements were collected from two BISMO [22] bit-serial accelerators synthesized on an FPGA board with a Z7020 SoC. For CIFAR-10 and CIFAR-100 experiments, an array of 8×8 processing elements (PEs), with 256 lanes was synthesized. For ImageNet workloads, we synthesize HW with 6×6 PEs and 256 lanes each. The collected measurements include all layer shapes in the considered CNNs, and all possible bit-width combinations for weights and activations. For Tab. 3 and 4, we report the DRAM accesses and latency by executing the quantized CNN workloads directly on the BISMO accelerator.

4.1 Mixed-Precision Quantization

We investigate the effectiveness of our in-train quantization approach in Tab. 1, based on different constraints φ^* on the number of target BOPs. We perform in-train quantization by constraining the number of BOPs to the complexity equivalent of uniform 4-bit (see row 3, 4 in Tab. 1). By introducing trainable bit-widths for both weights and activations, we produce 0.1 pp and 1.1 pp better prediction accuracy than uniform 4-bit quantization for ResNet20 and ResNet56, respectively. We highlight the importance of the scaling factor v (Eq. 3) for lower BOPs constraints $\varphi^* = 333, 1024$. We observe that the target constraint φ^* is only met when the scaling factor is increased. We observe a reduction in BOPs by 1.8× and 1.5× with negligible accuracy degradation compared to uniform 4-bit quantization for ResNet20 and ResNet56, respectively. Furthermore, we report the training cost required to obtain the baseline and mixed precision models. Our in-train quantization scheme learns optimal bit-widths with minimal overhead in training time, i.e. 6%, 3% extra cost compared to uniformly quantized ResNet20 and ResNet56 respectively. We demonstrate the training curves and weight distributions indicating the effectiveness of progressive quantization in the supplementary material Sec.S1.

In Tab. 2, we compare our approach with state-of-the-art uniform quantization approaches, such as PACT [3] and ABC-Net [17]. We also compare with works which produce variable bit-widths for weights and activations such as HAQ [25], DNAS [28], and LBS [18]. HAQ [25] determines layer-wise bit-widths using reinforcement learning. Such methods have a high

Table 1: Influence of scaling factor ν in $\mathcal{L}_{HW}$ for BOPs-constrained in-train quantization.

Model/ Dataset	Mixed Precision Weight	Mixed Precision Activation	Scaling Factor (ν)	Avg. Bitwidth W_{bit}	Avg. Bitwidth A_{bit}	Constraint φ^* BOPs (M)	Actual φ BOPs (M)	Top-1 (%)	Training Cost (GPU hours)**
ResNet20 CIFAR-10	✗	✗	-	8	8	-	2592	92.4	2hr 43min
	✗	✗	-	4	4	-	666	92.2	
	✓	✗	1.0	4.0	4	666	679	91.0	2hr 53min
	✓	✓	1.0	4.0	4.0	666	651	**92.3**	
	✓	✓	0.1	7.1	7.1	333	2028	92.3	
	✓	✓	0.5	2.9	4.8	333	575	92.3	
	✓	✓	1.0	2.3	3.7	333	**376**	**91.5**	
ResNet56 CIFAR-100	✗	✗	-	8	8	-	8025	71.1	7hr 29min
	✗	✗	-	4	4	-	2029	70.5	
	✓	✗	1.0	4.0	4	2029	2019	**72.2**	7hr 46min
	✓	✓	1.0	3.7	5.1	2029	2123	71.6	
	✓	✓	0.5	3.7	4.8	1024	2201	71.9	
	✓	✓	1.0	3.7	4.5	1024	1739	71.3	
	✓	✓	2.0	2.6	3.2	1024	**1311**	**69.9**	

** Training cost is measured on a NVIDIA TITAN-X GPU

Table 2: Comparison of our in-train quantization approach with state-of-the-art methods. $*$ indicates that the accuracy and BOPs measurements are reported from [18].

Model/ Dataset	Method	Mixed Precision Weight	Mixed Precision Activation	BOPs (M)	Top-1 (%)
ResNet20 CIFAR-100	PACT-8 [8]	fixed	fixed	2592	68.3
	PACT-4 [8]	fixed	fixed	666	67.0
	PACT-2 [8]	fixed	fixed	189	61.6
	ABCNet-3x3 [17]	fixed	fixed	390	61.0
	HAQ (RL)* [25]	learned	learned	653	67.7
	DNAS* [23]	learned	learned	660	67.8
	LBS* [18]	learned	learned	630	68.1
	Ours	learned	learned	646	**68.3**
ResNet56 CIFAR-100	PACT-8 [8]	fixed	fixed	8025	71.1
	PACT-4 [8]	fixed	fixed	2029	70.4
	PACT-2 [8]	fixed	fixed	528	67.8
	ABCNet-3x3 [17]	fixed	fixed	1153	68.4
	HAQ (RL)* [25]	learned	learned	2015	71.2
	DNAS* [23]	learned	learned	2035	71.2
	LBS* [18]	learned	learned	1918	71.6
	Ours	learned	learned	**1739**	71.3

computational search cost as the bit-width policy must be learned, involving iterative fine-tuning/evaluation for different bit-width combinations at each episode. DNAS [23] and LBS [18] determine the quantization strategy using gradient based optimization. DNAS constructs a super net consisting of several parallel edges representing search convolution operations with different quantization levels. LBS reduces the search complexity compared to the multi-path DNAS and also exploits filter pruning to further extract compression benefits. However, all the three approaches retrain the sampled quantized strategies obtained from the search phase indicating higher GPU hours compared to our approach, which requires only the regular training time of the CNN as shown in Tab. 1. Our quantization scheme produces dominating solutions in the number of BOPs and prediction accuracy compared to HAQ and DNAS. Compared to LBS, our ResNet20 model has slightly higher accuracy (0.2 pp), with a slight increase in BOPs. We further benchmark our approach on a semantic segmentation model, DeepLabV3+, in the supplementary material Sec. S2.

4.2 In-train HW-aware Quantization

We investigate the effectiveness of the proposed in-train optimization scheme in Tab. 3, based on pseudo-HW-aware constraints (BOPs), as well as real HW constraints, i.e. inference latency. Using the GP regressor introduced in Sec. 3.2, our approach produces bit-widths based on the target metric and target HW. We observe that constraining the number of BOPs does not necessarily produce optimal latency benefits in all the three networks, making it a pseudo-HW-aware metric. Our approach directly reduces the latency by 1.3× with respect to all 4-bit CNNs, with negligible degradation in prediction accuracy (<1 pp). In case of ResNet56 based latency constrained model, we obtain lower BOPs and prediction accuracy than BOPs based optimization. This can be attributed to the strict latency constraint imposed by the HW model demanding high compression ratios across several layers.

Table 3: Pseudo-HW-aware constraints and real HW constraints for various CNN models on CIFAR-10, CIFAR-100, and ImageNet datasets.

Model/ Dataset	Constraint	BOPs (M)	Latency (KCycles)	Top-1 (%)
ResNet20 CIFAR-10	PACT-4 [8]	666	1135	**92.2**
	PACT-2 [8]	189	769	89.5
	Ours (BOPs)	**448**	951	91.3
	Ours (Latency)	530	**875**	91.2
ResNet56 CIFAR-100	PACT-4 [8]	2029	3134	70.4
	PACT-2 [8]	528	2025	67.8
	Ours (BOPs)	1739	2753	**71.3**
	Ours (Latency)	**1498**	**2374**	70.7
ResNet18 ImageNet	PACT-4 [8]	34714	39637	**65.4**
	PACT-2 [8]	14984	28939	60.4
	Ours (BOPs)	**27424**	36930	64.6
	Ours (Latency)	35356	**31112**	64.5

To further demonstrate the effectiveness of the differentiable HW-awareness, we learn optimal bit-width strategies which capture different scheduling schemes on the inference HW. For the target BISMO bit-serial accelerator [22], we derive two scheduling schemes namely, activation-reuse schedule (ARS) and weight-reuse schedule (WRS). We provide more details on the formulation of these scheduling schemes in the supplementary material Sec.S3. In Tab. 4, we observe that our quantization approach assigns higher bit-widths to the favorable datatype being reused by the HW, thus *learning* to exploit the inherent efficiency of the chosen schedule, without knowing its details. The average bit-width of weights in the WRS-based mixed-precision strategy remains at the highest value (8-bits), while the activations are quantized more aggressively, as they are costly and not reused by the WRS schedule. Conversely, we observe higher average bit-width for activations in ARS-based mixed-precision strategy. We also observe 1.33× and 1.01× reduction in DRAM accesses for ARS and WRS-based quantization strategies in ResNet56, with 0.4 pp better accuracy.

4.3 Adversarially Robust Mixed-Precision CNNs

We demonstrate our proposed mixed-precision approach's ability to achieve compressed models with a balanced trade-off between natural accuracy and adversarial robustness. As a baseline for adversarial training, we implement FastAT [27] with uniform quantization. We augment our trainable quantization parameters in the FastAT defense method and report the natural accuracy and PGD robustness [20] for our mixed precision strategies in Tab. 5.

Table 4: Influence of quantization strategies based on the ARS and WRS compiler schedules.

Model/ Dataset	Training Scheme	Avg. bit-width Weight	Avg. bit-width Activation	ARS Mem (MB)	WRS Mem (MB)	Top-1 (%)
ResNet20 CIFAR-10	PACT-4 [8]	4	4	6.6	5.4	**92.2**
	PACT-2 [8]	2	2	4.1	3.3	89.7
	Ours (ARS-Opt)	2.3	5.3	**5.0**	6.3	91.8
	Ours (WRS-Opt)	8.0	3.3	10.3	**4.7**	91.6
ResNet56 CIFAR-100	PACT-4 [8]	4	4	18.7	15.3	70.4
	PACT-2 [8]	2	2	10.8	8.9	67.8
	Ours (ARS-Opt)	2.1	4.2	**14.0**	19.2	**70.8**
	Ours (WRS-Opt)	8.0	3.7	30.3	**15.1**	**70.8**

The PGD attack, referred as the "ultimate" first-order adversary [20], generates perturbations using iterative multi-step optimization method. By considering random uniform initialization, arbitrary starting points on the corresponding loss surface are ensured, thus resulting in worst-case adversaries for the given image with respect to an underlying CNN model. For PGD evaluation, we use a strength of $8/255$, step size $2/255$ for 20 iterations. We elaborate the implementation details of in-train robust quantization in the supplementary material Sec.S4. Existing work shows that low-precision models exhibit higher adversarial robustness due to the discrete nature of the quantization operations [11]. Thus, we observe an increase in adversarial robustness as the bit-width is reduced for the uniform PACT quantization. Our in-train quantization approach improves the trade-off between the three objectives, namely prediction accuracy, adversarial robustness, and BOPs reduction. The achieved robustness is increased by 0.9 pp and 1.4 pp, while reducing the number of BOPs by 1.6× and 1.9×, compared to adversarially trained uniform 4-bit ResNet20 and ResNet56, respectively.

Table 5: Adversarial Robustness of uniformly quantized and mixed precision CNNs.

Model/ Dataset	Method	Bitwidth W_{bit}	Bitwidth A_{bit}	BOPs (M)	Top-1 (%)	PGD-20 (%)
ResNet20 CIFAR-10	PACT [8]	4	4	666	81.9 ± 0.04	40.6± 0.27
	PACT [8]	2	2	189	76.0 ± 0.06	41.5 ± 0.32
	Ours	3.5	2.9	**427**	**81.7± 0.08**	**41.5± 0.31**
ResNet56 CIFAR-10	PACT [8]	4	4	2029	85.3 ± 0.25	41.5± 0.72
	PACT [8]	2	2	529	82.3± 0.58	47.3± 2.28
	Ours	2.9	2.7	**1049**	**84.7 ± 0.91**	**42.9± 0.21**

5 Conclusion

In this work, we propose an *in-train* quantization technique, which eliminates the need for computationally expensive model exploration time, typically required in post-training compression methods. We directly optimize our quantization strategy by formulating a HW-aware differentiable loss object using Gaussian process regression. Compared to state-of-the-art mixed-precision approaches, we reduce the number of BOPs while improving the prediction accuracy, producing dominating solutions. We show the applicability of our mixed-precision quantization scheme to an adversarial training method, by improving the trade-off between prediction accuracy, robustness, and the reduction in number of BOPs.

References

[1] Yoshua Bengio, N. Léonard, and Aaron C. Courville. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *ArXiv*, abs/1308.3432, 2013. URL https://arxiv.org/abs/1308.3432.

[2] Michaela Blott, Thomas B Preußer, Nicholas J Fraser, Giulio Gambardella, Kenneth O'brien, Yaman Umuroglu, Miriam Leeser, and Kees Vissers. FINN-R: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2018.

[3] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I.-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. PACT: Parameterized Clipping Activation for Quantized Neural Networks. *ArXiv*, abs/1805.06085, 2018. URL http://arxiv.org/abs/1805.06085.

[4] Xiaoliang Dai, Peizhao Zhang, B. Wu, Hongxu Yin, Fei Sun, Y. Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Y. Jia, Péter Vajda, M. Uyttendaele, and N. Jha. ChamNet: Towards Efficient Network Design Through Platform-Aware Model Adaptation. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemehsadat Mireshghallah, Tarek Elgindi, Charles-Alban Deledalle, and Hadi Esmaeilzadeh. WaveQ: Gradient-based deep quantization of neural networks through sinusoidal adaptive regularization. *ArXiv*, abs/2003.00146, 2020. URL https://arxiv.org/abs/2003.00146.

[6] Ahmed T. Elthakeb, Prannoy Pilligundla, Fatemehsadat Mireshghallah, Amir Yazdanbakhsh, and Hadi Esmaeilzadeh. ReleQ : A reinforcement learning approach for automatic deep quantization of neural networks. *IEEE Micro*, 40(5):37–45, 2020. doi: 10.1109/MM.2020.3009475.

[7] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=rkgO66VKDS.

[8] Nael Fasfous, Manoj Rohit Vemparala, Alexander Frickenstein, Emanuele Valpreda, Driton Salihu, Nguyen Anh Vu Doan, Christian Unger, Naveen Shankar Nagaraja, Maurizio Martina, and Walter Stechele. HW-FlowQ: A Multi-Abstraction Level HW-CNN Co-Design Quantization Methodology. *ACM Trans. Embed. Comput. Syst.*, 20, 2021. URL https://doi.org/10.1145/3476997.

[9] Alexander Frickenstein, Manoj-Rohit Vemparala, Christian Unger, Fatih Ayar, and Walter Stechele. DSC: Dense-Sparse Convolution for Vectorized Inference of Convolutional Neural Networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2019.

[10] Alexander Frickenstein, Manoj Rohit Vemparala, Jakob Mayr, Naveen Shankar Nagaraja, Christian Unger, Federico Tombari, and Walter Stechele. Binary dad-net: Binarized driveable area detection network for autonomous driving. *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[11] Angus Galloway, Graham W. Taylor, and Medhat Moussa. Attacking Binarized Neural Networks. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=HkTEFfZRb.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. AMC: AutoML for Model Compression and Acceleration on Mobile Devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the Knowledge in a Neural Network. In *NIPS Deep Learning and Representation Learning Workshop*, volume abs/1308.3432, 2015. URL http://arxiv.org/abs/1503.02531.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-100 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.

[17] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards Accurate Binary Convolutional Neural Network. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[18] Jing Liu, Bohan Zhuang, Peng Chen, Yong Guo, Chunhua Shen, Jianfei Cai, and Mingkui Tan. ABS: Automatic Bit Sharing for Model Compression. *ArXiv*, abs/2101.04935, 2021. URL https://arxiv.org/abs/2101.04935.

[19] Qian Lou, Feng Guo, Minje Kim, Lantao Liu, and Lei Jiang. AutoQ: Automated Kernel-Wise Neural Network Quantization. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=rygfnn4twS.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[21] Stefan Uhlich, Lukas Mauch, Fabien Cardinaux, Kazuki Yoshiyama, Javier Alonso Garcia, Stephen Tiedemann, Thomas Kemp, and Akira Nakamura. Mixed Precision DNNs: All you need is a good parametrization. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=Hyx0slrFvH.

[22] Yaman Umuroglu, Lahiru Rasnayake, and Magnus Sjalander. Bismo: A scalable bit-serial matrix multiplication overlay for reconfigurable computing. In *Field Programmable Logic and Applications (FPL)*, 2018.

[23] Manoj-Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Mhd Ali Moraly, Aquib Jamal, Lukas Frickenstein, Christian Unger, Naveen-Shankar Nagaraja, and Walter Stechele. L2pf - learning to prune faster. In *Computer Vision and Image Processing*, 2021.

[24] Manoj Rohit Vemparala, Nael Fasfous, Alexander Frickenstein, Sreetama Sarkar, Qi Zhao, Sabine Kuhn, Lukas Frickenstein, Anmol Singh, Christian Unger, Naveen Shankar Nagaraja, Christian Wressnegger, and Walter Stechele. Adversarial robust model compression using in-train pruning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2021.

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

[26] Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. APQ: Joint Search for Network Architecture, Pruning and Quantization Policy. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[27] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=BJx040EFvH.

[28] B. Wu, Y. Wang, P. Zhang, Yuandong Tian, Péter Vajda, and K. Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *ArXiv*, abs/1812.00090, 2018. URL https://arxiv.org/abs/1812.00090.

[29] Tien-Ju Yang, Andrew Howard, Bo Chen, Xiao Zhang, Alec Go, Mark Sandler, Vivienne Sze, and Hartwig Adam. Netadapt: Platform-aware neural network adaptation for mobile applications. In *The European Conference on Computer Vision (ECCV)*, 2018.

[30] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2016. URL http://arxiv.org/abs/1606.06160.