Adaptive Quantization and Pruning of Deep Neural Networks via Layer Importance Estimation

Tushar Shinde
School of Engineering and Science, Indian Institute of Technology Madras Zanzibar, Tanzania
shinde@iitmz.ac.in

Abstract

Deep neural networks (DNNs) have achieved remarkable performance in various applications, but their deployment on edge devices is hindered by significant computational and storage requirements. To mitigate these challenges, quantization has proven effective in reducing model size while maintaining accuracy, with pruning further enhancing model compression. However, achieving an optimal balance between compression and performance, particularly in mixed-precision strategies that allocate different bit widths to individual layers, remains a challenge. In this paper, we present a method that ranks layers based on their statistical importance and adaptively selects bit-width precision and pruning for each layer, ensuring minimal accuracy loss. Our approach dynamically determines layer-specific thresholds, optimizing compression without the need for complex tuning or costly optimization. We validate our interpretable and efficient method through image classification tasks, demonstrating its effectiveness across multiple DNN architectures. Experimental results show that our method maintains classification accuracy levels of 91.16% for VGG19, 86.06% for ResNet18, and 86.13% for ResNet34 on CIFAR-10 dataset while achieving average bit-width reductions to 1.08, 2.66, and 2.42 bits, respectively.

1 Introduction and Related Works

Deep neural networks (DNNs) have demonstrated remarkable performance across various domains such as computer vision and natural language processing (1). However, deploying these models on resource-constrained edge devices presents significant challenges due to their substantial computational and storage demands (2; 3). Edge devices typically have stringent limitations in terms of power, memory, and processing capabilities, creating a need to compress and optimize DNNs while maintaining high accuracy. Several approaches have been proposed to tackle these challenges, including model pruning (4; 5), knowledge distillation (6), low-rank factorization (7), and quantization (8; 9). Model pruning focuses on inducing sparsity by reducing the number of nonzero parameters in the network, typically by mapping parameters close to zero to zero (4). On the other hand, quantization reduces the bit precision of weights and activations, transitioning from 32-bit floating-point representations to lower-bit formats, which is a widely used technique for model compression (10).

Most existing quantization techniques employ uniform bit-width precision across all layers. While effective for model size reduction and faster inference, these methods can result in accuracy degradation, particularly for more complex models and tasks (11). This is because uniform precision does not account for the varying importance of different layers in contributing to model performance (12). Non-uniform quantization allows for different bit-widths across layers, adapting precision based on the sensitivity of each layer (13). While uniform 8-bit quantization can reduce model size by a factor of four without significant accuracy loss, more advanced mixed-precision quantization approaches are necessary for achieving further reductions. Mixed-precision quantization methods

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

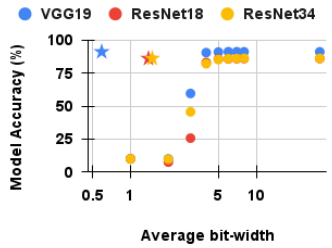


Figure 1: Model accuracy across DNN architectures. The star markers highlight our method.

(14) have been developed to address these challenges, but determining the optimal bit precision for each layer requires complex and computationally expensive optimization processes (15). For instance, value-aware quantization (16) adjusts precision based on weight and activation distributions. Similarly, existing pruning methods often use a fixed threshold across all layers (5). While uniform pruning can reduce model size significantly with minimal accuracy impact, this approach is not scalable due to the different parameter distributions across layers. To achieve even more compact models without sacrificing accuracy, adaptive pruning strategies are needed.

Our proposed method addresses these limitations by introducing a novel layer-wise optimization algorithm and an efficient method for computing layer importance. The algorithm iteratively adjusts the bit-width of each layer based on its effect on model accuracy, optimizing the trade-off between model size and performance. Additionally, we incorporate layer sparsity into the importance computation and use an adaptive threshold for zeroing out parameters during pruning. Fig. 1 presents a comparison of model accuracy across various DNN architectures against the average bit-width. Our approach achieves comparable accuracy with adaptive quantization and pruning, while reducing the average bit-width per parameter more effectively.

The remainder of this paper is organized as follows: Section 2 provides details on the proposed method. Section 3 describes the experimental setup, and the results are discussed in Section 4. Finally, the paper is concluded in Section 5.

2 Methodology

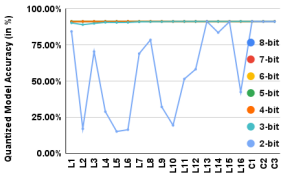
This section outlines our adaptive layer-wise quantization and pruning strategy for deep neural networks, focusing on minimizing accuracy loss while significantly reducing model size. The approach involves a quantization and pruning strategy and an iterative optimization algorithm, aiming to make deep neural networks more feasible for edge devices with limited computational resources. The method optimizes the bit-precision and pruning of each layer based on its importance, balancing model size and accuracy.

The motivation comes from observing that different layers affect final model accuracy differently (17). Experiments with varying low bit-precisions for individual layers (keeping others at 8-bit quantization) show different performance impacts, as illustrated in Fig. 2a. For instance, quantizing the first layer to 2-bit yields 84.28% accuracy, while quantizing the last layer to 2-bit yields 91.07%. Similarly, varying pruning percentages for individual layers (without pruning other layers) show different performance impacts, as illustrated in Fig. 2b. For instance, pruning the first layer by 90% yields 71.05% accuracy, while pruning the last layer by 90% yields 90.99%.

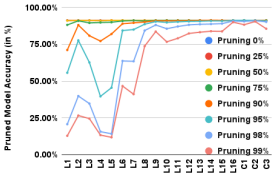
To leverage layer characteristics, each layer is quantized with different bit-precisions and pruned with different thresholds. Deciding the order of quantization and pruning thresholds is challenging, hence, we rank each layer based on its importance.

2.1 Layer Importance Computation

We compute the importance of each layer using multiple statistical measures relevant to the trade-off between bit-precision, pruning percentage, and model accuracy. Firstly, the number of parameters in each layer affects the overall model size. Layers with more parameters are prioritized for quantization



(a) Model Quantization



(b) Model Pruning

Figure 2: Comparison of model accuracy for VGG19.

to reduce size without compromising accuracy. Hence, we consider the normalized parameter proportion, which is computed as:

$$N_P(l) = \frac{\text{Number of parameters in layer } l}{\text{Total number of parameters in the model}} \tag{1}$$

Secondly, zero-order entropy of parameters impacts bit requirements. Higher entropy indicates more information and hence the requirement of more bits. Thus we consider the normalized zero-order entropy, which is computed as:

$$N_E(l) = \frac{\text{Zero-order entropy for parameters in layer } l}{\text{Bit-precision of quantized model}} \tag{2}$$

Thirdly, it is well established that the parameter distribution drives the bit-precision requirement. Most layers in the DNN model follow a Gaussian-like distribution, with variance indicating compactness. Hence, we consider the normalized variance, which is computed as:

$$N_V(l) = \log\left(e - 1 + \frac{\text{Variance of parameters in layer } l}{\max_k\left(\text{Variance of parameters in layer } k\right)}\right) \tag{3}$$

Lastly, activation sparsity indicates layer criticality. Layers with higher zero or near-zero activations might be less critical, so we consider the normalized activation sparsity, which is computed as:

$$S(l) = \frac{\text{Number of zero or near-zero activations in layer } l}{\text{Total number of activations in layer } l} \tag{4}$$

The final layer importance combines these components:

$$Importance(l) = w_P \cdot N_P(l) + w_E \cdot N_E(l) + w_V \cdot N_V(l) + w_S \cdot S(l) \tag{5}$$

where w_P, w_E, w_V, and w_S are weight multiplying factors used to control the impact on the overall importance score. These weights can be tuned to prioritize certain aspects of a layer, such as the number of parameters or entropy, based on the optimization objective (e.g., reducing model size or preserving accuracy).

2.2 Layer-wise Pruning Scheme

We present a method where weights close to zero are pruned based on an adaptive threshold, replacing them with zero. The pruning process involves multiple steps. Firstly, a zero threshold $Z_T(l)$ is calculated for each layer based on standard deviation $\sigma(l)$ of weights:

$$Z_T(l) = k(l) \times \sigma(l) \tag{6}$$

where $k(l)$ is a pruning multiplying factor that controls the pruning percentage for layer l. By adjusting $k(l)$, one can influence the percentage of weights that are pruned. A larger value of $k(l)$ results in a higher threshold, leading to more weights being pruned, whereas a smaller value of $k(l)$ results in a lower threshold, which reduces the amount of pruning. We propose to find an optimal $k(l)$ for each layer through a search process, ensuring maximum pruning while maintaining high accuracy. Then, the weights within the threshold are pruned using:

$$\hat{W}_{l,ij} = \begin{cases} W_{l,ij} & \text{if } |W_{l,ij}| > Z_T(l) \\ 0 & \text{if } |W_{l,ij}| \leq Z_T(l) \end{cases} \tag{7}$$

Lastly, the pruned model is updated for subsequent steps. In our framework, the sparsity is measured by the proportion of pruned weights:

$$S(l) = \frac{\text{Number of pruned weights in layer } l}{\text{Total number of weights in layer } l} \tag{8}$$

Overall sparsity is calculated as weighted average across all layers:

$$S_{overall} = \sum_{l=1}^{L} S(l) \cdot N_P(l) \tag{9}$$

2.3 Layer-wise Bit-width Selection Algorithm

Optimal bit-precision for each layer is determined through a search process, ensuring minimal bit-width while maintaining high accuracy. Layers are ranked by importance and search process starting with the layer with highest importance. Bit-width search begins from the lowest possible bit-width with adaptive margin $T_{margin}(l)$:

$$T_{margin}(l) = Tmargin \times Importance(l) \tag{10}$$

where T_{margin} is a threshold representing the accuracy loss margin, which can be empirically chosen for best performance. The bit-width $b(l)$ and pruning parameter $k(l)$ are adjusted iteratively, optimizing sequentially based on layer importance. This ensures overall model accuracy remains within an acceptable margin of the original model's accuracy, starting with the most significant layers and gradually decreasing bit precision while validating performance after each change.

3 Experimental setup

Our experiments were conducted on the Kaggle platform, utilizing NVIDIA Tesla P100 and G4 GPUs, which provided significant acceleration for training our neural networks.

Dataset: The CIFAR-10 dataset (18) is considered in our experiments. This dataset comprises 60,000 32x32 color images divided into 10 classes, with 50,000 images for training and 10,000 for testing. The images were normalized to the range [0, 1] and augmented using techniques such as horizontal flips and random crops.

Implementation Details: We validated our method using various architectures, including VGG19 (19), ResNet18 (20), and ResNet34 (20). Models were initialized from scratch without pre-trained weights. The training was performed with a batch size of 128 for 100 epochs. The initial learning rate was set to 0.02, with a decay factor of 0.5 applied every 20 epochs. The Stochastic Gradient Descent (SGD) optimizer was used with a momentum of 0.9 and a weight decay of 5e-4, along with the cross-entropy loss function. Note that our primary focus was on post-training quantization, not quantization-aware training for faster deployment.

Hyper-parameter Settings: The weights w_P, w_E, w_V, and w_S used in the layer importance computation were set equally, summing to 1. The threshold margin T_{margin} is empirically set to 0.1%. Given the impact of the first and last layers on overall performance, T_{margin} was adjusted to 1/2 of the predefined threshold (21). We primarily focused on weight quantization with low bit precision starting from 1-bit to 8-bit quantization. The 8-bit quantization serves as the baseline. Moreover, we considered a pruning multiplying factor $k(l)$ for layer l, starting from 3, representing 99.7% pruning, and decreasing to 0, which represents no pruning.

Evaluation Metrics: We used accuracy and average bit-width to evaluate the effectiveness of model compression techniques. Accuracy measures the proportion of correctly classified instances out of the total predictions made by the model. Whereas, an average bit-width measure the overall bit-precision used across all layers of the model, calculated as the weighted average bit-width:

Model	VGG19	ResNet18	ResNet34
Original (32-bit)	91.16%	86.06%	86.22%
Quantization only approach			
Fixed (8-bit) Quantization	91.17%	86.01%	86.25%
Fixed (7-bit) Quantization	91.23%	85.89%	86.12%
Fixed (6-bit) Quantization	91.25%	85.79%	85.97%
Fixed (5-bit) Quantization	91.00%	85.65%	85.28%
Fixed (4-bit) Quantization	90.54%	83.04%	82.10%
Fixed (3-bit) Quantization	59.57%	25.72%	45.63%
Fixed (2-bit) Quantization	10.00%	7.53%	9.75%
Fixed (1-bit) Quantization	10.00%	10.00%	10.00%
Ours Adaptive Quantization	91.09%	86.00%	86.18%
(Avg. Bit-width $\bar{b}$)	(2.24)	(3.41)	(4.18)
($\bar{b}$ with Huffman Enc.)	(1.49)	(2.08)	(2.79)
Pruning only approach			
Pruned (25%)	91.09%	85.8%	86.09%
Pruned (50%)	90.21%	81.17%	82.23%
Pruned (75%)	64.96%	43.70%	45.02%
Pruned (90%)	10.00%	19.91%	10.29%
Ours Adaptive Pruned	91.10%	85.94%	86.19%
Pruned $S_{overall}$ (in %)	(81%)	(63%)	(60%)
(Avg. Bit-width $\bar{b}$)	(6.20)	(11.85)	(12.87)
Adaptive quantization and pruning (AQP) approach			
Proposed AQP	91.16%	86.06%	86.13%
(Avg. Bit-width $\bar{b}$)	(1.08)	(2.66)	(2.42)
($\bar{b}$ with Huffman Enc.)	(0.59)	(1.39)	(1.49)

Table 1: Comparison of model accuracy and average bit-width across different DNN architectures for adaptive quantization, adaptive pruning, and the combined adaptive quantization and pruning approaches.

$$\bar{b} = \sum_{l=1}^{L} b(l) \cdot S(l) \cdot N_P(l) \tag{11}$$

where, $b(l)$ is the bit-width of the parameters in layer l, $S(l)$ is the sparsity in layer l, and $N_P(l)$ is the normalized parameter proportion in layer l. $\bar{b}$ refers to the weighted average bit-width across all layers of the model.

4 Results and Discussion

We present results for different quantization and pruning strategies, comparing them against our proposed approaches. This includes the performance of adaptive quantization, adaptive pruning, and their combined application.

4.1 Adaptive Quantization Results

We applied our adaptive quantization method to VGG19, ResNet18, and ResNet34 trained on the CIFAR-10 dataset. Layers were ranked by importance, and quantization was performed sequentially, using the lowest bit-precision that maintained model accuracy within a margin T_{margin}. Table 1 shows model accuracy and average bit-width across various quantization levels. The baseline accuracy for full-precision (32-bit) models was 91.16% for VGG19, 86.06% for ResNet18, and 86.22% for ResNet34. Accuracy generally decreased as bit-width was reduced, with fixed 2-bit quantization causing a drastic drop to about 10%. In contrast, our adaptive quantization method preserved competitive accuracy (91.09% for VGG19, 86.00% for ResNet18, and 86.18% for ResNet34) while achieving an average bit-width of 2.24, 3.41, and 4.18, respectively. Incorporating Huffman encoding further reduced the average bit-widths to 1.49, 2.08, and 2.79, demonstrating that adaptive quantization effectively balances model accuracy and bit-width.

4.2 Adaptive Pruning Results

Our adaptive pruning method was similarly applied to the same model architectures. Layers were pruned based on their computed importance, allowing us to select the highest pruning percentage that maintained accuracy within T_{margin}. Table 1 also summarizes model accuracy and average

Method	Model	#Parameters (M) $\times$ Avg. bit-width $\bar{b}$	Accuracy difference (in %)
Proposed AQP	VGG19	20.04 $\times$ 1.08	**0.00**%
Proposed AQP	ResNet18	11.69 $\times$ 2.66	0.00%
Proposed AQP	ResNet34	21.80 $\times$ 2.42	-0.09%
APoT (22)	ResNet18	11.69 $\times$ 4.00	-0.40%
APoT (22)	ResNet18	11.69 $\times$ 3.00	-0.84%
APoT (22)	ResNet18	11.69 $\times$ 2.00	-1.75%
LIEI-NNQ (21)	ResNet18	11.69 $\times$ 1.96	-1.55%
APoT (22)	MobileNetV2	3.47 $\times$ 4.00	-4.25%
APoT (22)	MobileNetV2	3.47 $\times$ 3.00	-10.39%
APoT (22)	MobileNetV2	3.47 $\times$ 2.00	-24.45%
LIEI-NNQ (21)	MobileNetV2	3.47 $\times$ 3.32	-9.42%

Table 2: Comparison of accuracy difference (%) relative to the 32-bit baseline model, average bit-width, and number of parameters between the proposed approach and existing methods across various DNN architectures.

sparsity. As the pruning percentage increased, accuracy typically decreased. Fixed thresholds resulted in significant accuracy drops, whereas our adaptive pruning method achieved competitive accuracy (91.10% for VGG19, 85.94% for ResNet18, and 86.19% for ResNet34) with overall sparsity of 81%, 63%, and 60%, respectively. This highlights the effectiveness of our adaptive layer importance-guided pruning technique in preserving accuracy while reducing model size.

4.3 Adaptive Quantization and Pruning Results

Table 1 compares the performance of different architectures under various quantization and pruning techniques. The original 32-bit models and fixed 8-bit quantized models maintained comparable accuracy, with our adaptive methods achieving accuracies of 91.16% for VGG19, 86.06% for ResNet18, and 86.13% for ResNet34. By employing adaptive quantization and pruning, we reduced average bit-widths to 1.08, 2.66, and 2.42, respectively, with minimal accuracy loss.

4.4 Performance Comparison with Existing Methods

Table 2 presents a comparative analysis between our approach and existing methods, focusing on parameters, average bit-width, and accuracy differences. The proposed method maintained accuracy for VGG19 and ResNet18 at significantly lower bit-widths (1.08 and 2.66, respectively), with minimal accuracy drop (0.09%) for ResNet34 at an average bit-width of 2.42. In contrast, existing methods like APoT (22) and LIEI-NNQ (21) experienced higher accuracy losses, especially at lower bit-widths. These results highlight the potential of our adaptive layer-wise quantization and pruning approach for deploying DNNs on resource-constrained devices. By optimizing bit-widths adaptively, our method ensures efficient models that retain high accuracy, making it a valuable solution for applications with limited computational resources.

Our work has following limitations. Although the adaptive quantization and pruning techniques effectively reduce model size on disk, they can lead to fragmented unstructured sparsity, which may not improve computation costs or latency. In the worst case, loading additional computation operators for different precision levels can increase memory consumption, potentially offsetting the benefits of model compression.

5 Conclusion

In this paper, we introduced an adaptive layer-wise quantization and pruning method for deep neural networks (DNNs) aimed at enhancing model efficiency while preserving accuracy. By tailoring the quantization and pruning processes to the sensitivity of each layer, we achieved substantial reductions in average bit-width with minimal loss of accuracy. Experimental results indicated that our approach outperforms uniform quantization and pruning techniques, effectively maintaining accuracy across varying bit-widths, which is particularly beneficial for edge devices with constrained computational resources. Our analysis underscores the robustness of the proposed adaptive method, striking an optimal balance between performance and resource efficiency. Future research will focus on extending this method to additional datasets and architectures to further validate its generalizability.

References

[1] LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. nature, 521(7553), pp.436-444.

[2] Li, Z., Li, H. and Meng, L., 2023. Model compression for deep neural networks: A survey. Computers, 12(3), p.60.

[3] Zhou, A., Ma, Y., Zhu, J., Liu, J., Zhang, Z., Yuan, K., Sun, W. and Li, H., 2021. Learning n: m fine-grained structured sparse neural networks from scratch. arXiv preprint arXiv:2102.04010.

[4] Liu, Z., Sun, M., Zhou, T., Huang, G. and Darrell, T., 2018. Rethinking the value of network pruning. arXiv preprint arXiv:1810.05270.

[5] Li, H., Kadav, A., Durdanovic, I., Samet, H. and Graf, H.P., 2016. Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710.

[6] Sarfraz, F., Arani, E. and Zonooz, B., 2021, January. Knowledge distillation beyond model compression. In 2020 25th International Conference on Pattern Recognition (ICPR) (pp. 6136-6143). IEEE.

[7] Yin, M., Sui, Y., Liao, S. and Yuan, B., 2021. Towards efficient tensor decomposition-based dnn model compression with optimization framework. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10674-10683).

[8] Kim, N., Shin, D., Choi, W., Kim, G. and Park, J., 2020. Exploiting retraining-based mixed-precision quantization for low-cost DNN accelerator design. IEEE Transactions on Neural Networks and Learning Systems, 32(7), pp.2925-2938.

[9] Deng, L., Li, G., Han, S., Shi, L. and Xie, Y., 2020. Model compression and hardware acceleration for neural networks: A comprehensive survey. Proceedings of the IEEE, 108(4), pp.485-532.

[10] Rokh, B., Azarpeyvand, A. and Khanteymoori, A., 2023. A comprehensive survey on model quantization for deep neural networks in image classification. ACM Transactions on Intelligent Systems and Technology, 14(6), pp.1-50.

[11] Menghani, G., 2023. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. ACM Computing Surveys, 55(12), pp.1-37.

[12] Yang, G., Yu, S., Yang, H., Nie, Z. and Wang, J., 2023. HMC: Hybrid model compression method based on layer sensitivity grouping. Plos one, 18(10), p.e0292517.

[13] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H. and Kalenichenko, D., 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2704-2713).

[14] Koryakovskiy, I., Yakovleva, A., Buchnev, V., Isaev, T. and Odinokikh, G., 2023. One-shot model for mixed-precision quantization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7939-7949).

[15] Chen, W., Wang, P. and Cheng, J., 2021. Towards mixed-precision quantization of neural networks via constrained optimization. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 5350-5359).

[16] Park, E., Yoo, S. and Vajda, P., 2018. Value-aware quantization for training and inference of neural networks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 580-595).

[17] Elkerdawy, S., Elhoushi, M., Singh, A., Zhang, H. and Ray, N., 2020. To filter prune, or to layer prune, that is the question. In proceedings of the Asian conference on computer vision.

[18] Krizhevsky, A. and Hinton, G., 2009. Learning multiple layers of features from tiny images.

[19] Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[20] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[21] Liu, H., Elkerdawy, S., Ray, N. and Elhoushi, M., 2021. Layer importance estimation with imprinting for neural network quantization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2408-2417).

[22] Li, Y., Dong, X. and Wang, W., 2019. Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks. arXiv preprint arXiv:1909.13144.