DQA: An Efficient Method for Deep Quantization of Deep Neural Network Activations

Wenhao Hu, Paul Henderson, José Cano
School of Computing Science, University of Glasgow, Scotland, UK

Abstract

Quantization of Deep Neural Network (DNN) activations is a commonly used technique to reduce compute and memory demands during DNN inference, which can be particularly beneficial on resource-constrained devices. To achieve high accuracy, existing methods for quantizing activations rely on complex mathematical computations or perform extensive searches for the best hyper-parameters. However, these expensive operations are impractical on devices with limited computation capabilities, memory capacities, and energy budgets. Furthermore, many existing methods do not focus on sub-6-bit (or deep) quantization.

To fill these gaps, in this paper we propose DQA (Deep Quantization of DNN Activations), a new method that focuses on sub-6-bit quantization of activations and leverages simple shifting-based operations and Huffman coding to be efficient and achieve high accuracy. We evaluate DQA with 3, 4, and 5-bit quantization levels and three different DNN models for two different tasks, image classification and image segmentation, on two different datasets. DQA shows significantly better accuracy (up to 29.28%) compared to the direct quantization method and the state-of-the-art NoisyQuant for sub-6-bit quantization.

1 Introduction

Quantization is a popular compression technique to reduce the compute and memory demands of Deep Neural Networks (DNNs). During inference, DNN weights are typically fixed, so they can be quantized offline. However, activations are dynamically generated, which means that effective quantization methods must dynamically quantize activations online.

There are many methods for quantizing activations. The most straightforward one is to directly use the mathematical definition of quantization [5], but it provides limited accuracy. To get higher accuracy, more sophisticated approaches are used. For example, NoisyQuant [13] injects noise in the activations before quantization and removes the noise after de-quantization (the opposite operation to quantization). However, these methods are not specifically designed for resource-constrained devices, where the computational cost of expensive mathematical operations (e.g. matrix multiplication) or large online search spaces is impractical. Furthermore, many of these methods do not focus on deep quantization (e.g., quantizing to less than 6 bits), which is preferred for devices with limited memory.

To fill these gaps, in this paper we propose DQA (Deep Quantization of DNN Activations), a new method to deeply quantize DNN activations suited for resource-constrained devices that provides high accuracy. DQA first determines the *importance* of each activation channel offline using training or calibration datasets. For important channels, DQA quantizes their values with m extra bits first and then right-shifts the m bits to achieve the target number of bits while storing the shifting errors using Huffman coding. Then, the shifting errors are decoded and added back to the corresponding channels during the de-quantization phase. For unimportant channels, DQA uses a direct method to quantize and de-quantize activations, i.e., it simply applies the mathematical definition of quantization/de-quantization [5; 11]. By learning important channels offline, DQA saves significant computational

38th Second Workshop on Machine Learning with New Compute Paradigms at NeurIPS 2024 (MLNCP 2024).

resources during DNN inference. In addition, by using m more bits, DQA can quantize important channels with a different number of bits (i.e., mixed-precision quantization [1; 15]), reduce the quantization error exponentially, and efficiently shift back to the target bits without expensive mathematical computations. It is also important to note that: i) by right-shifting, all the quantized values can ultimately be stored using the same bit length, thus avoiding wasted storage [12]; ii) by applying Huffman coding to the shifting errors, DQA can reduce the extra memory overhead.

We evaluated DQA with three sub-6-bit quantization levels (3, 4, and 5 bits) on two different datasets (CIFAR-10 [10], and CityScapes [2]) for three different DNN models (ResNet-32 [7], MobileNetV2 [8] and U-Net [16]) for two different tasks (image classification and image segmentation). Overall, DQA shows significantly better accuracy (up to 29.28%) than two existing methods: direct quantization and the state-of-the-art NoisyQuant [13].

The main contributions of this paper are as follows:

- We propose DQA, a deep (sub-6-bit) quantization method for DNN activations that is especially relevant for resource-constrained devices. DQA deals with important activation channels separately and leverages Huffman coding to optimize memory usage.
- We explore the patterns of shifting errors in DQA to justify our choice of Huffman coding.
- We conduct a detailed evaluation of DQA and compare it with two existing methods, direct quantization and NoisyQuant [13], clearly outperforming them on accuracy.

2 Background and Related Work

Quantization is a widely used compression method to lower the precision format of parameters in DNNs, which reduces memory storage and computational requirements [5; 9]. The initial DNN parameters, typically using a floating-point format (e.g., 32 bits), are converted into fixed-point or integer values that require fewer bits (e.g., 16 or 8 bits). The process normally includes clipping and rounding operations. Clipping restricts the minimum and maximum values of the quantization, whereas rounding approximates values to the nearest integers, which causes rounding errors that cannot be recovered during de-quantization. De-quantization is the opposite operation of quantization that approximately recovers the quantized values back to floating point.

Quantization methods are typically divided into uniform and non-uniform. In uniform quantization, the quantized values are evenly spaced, while in non-uniform quantization they are not [5]. Uniform quantization can be further divided into symmetric and asymmetric. In symmetric quantization, the clipping range is symmetric with respect to the origin, while in asymmetric quantization it is not [5]. In this paper, we use uniform symmetric quantization due to its popularity and relatively low cost of computation [5], and we define the direct uniform symmetric quantization and de-quantization as the **Direct** method (note that this is the same as the direct quantization method for unimportant channels mentioned in Section 1).

It is important to note that keeping the same bit length for the quantized values across the whole DNN model is straightforward but can be sub-optimal for accuracy [15]. To improve this, many methods apply mixed-precision quantization which assigns different bit lengths to different parts of a DNN model, such as layers or channels, to satisfy their different precision sensitivities [1; 15]. However, due to the differences in bit length among the quantized values, mixed-precision quantization can cause inefficiencies such as wasted storage [12]. Our DQA method adapts mixed-precision quantization in an effective way (see Section 3).

Furthermore, previous works on DNN quantization apply it to the model weights and/or activations. For weights, AWQ [11] checks the importance of the weight channels for each layer and scales up important channels before quantization to reduce the rounding errors. For activations, quantization is difficult due to its dynamic nature, i.e., the activation values are typically not known before inference. Therefore, to achieve high accuracy, expensive mathematical operations (e.g., matrix multiplication) or large online search spaces to find the best hyper-parameters are required by many methods. For example, NoisyQuant [13] injects noise, obtained by online search within the given search spaces, to the activations before quantization and removes the noise after de-quantization. However, these previous methods do not focus on deep quantization like sub-6-bit quantization, thus being less suitable for resource-constrained devices.

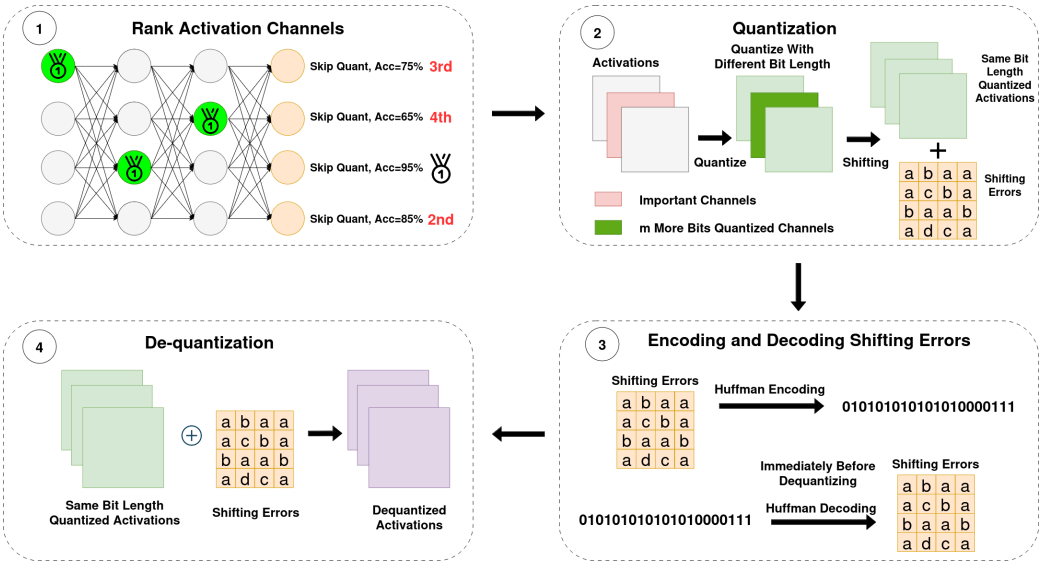


Figure 1: DQA overview. ① offline, rank the activation channels based on importance using training/calibration data and a greedy search algorithm (green circles represent the most important channels for which we skip quantization); ② during inference, quantize important channels with m extra bits and right-shift them while saving the shifting errors; ③ the shifting errors are Huffman-encoded to reduce memory usage. Before de-quantization, decode the Huffman-encoded shifting errors and add them back to the quantized activation channel values; ④ de-quantize important activation channels. For non-important channels, use the direct method to quantize and de-quantize.

3 Proposed Method: DQA

In this section, we present our DQA method in detail. The main goal is to efficiently quantize DNN activations, providing high accuracy while minimizing computation and memory requirements, which is especially important on resource-constrained devices. Figure 1 gives an overview of DQA.

Given a target of quantization bits n, DQA starts by quantizing all activations to n bits. Then, as shown in Figure 1: ① DQA uses the training/calibration data to rank the activation channels (i.e., their importance) offline using a greedy search algorithm (described in Section 3.1); ② During quantization in inference, for a tunable pre-selected ratio of important activation channels in each layer, DQA quantizes them using m extra bits (i.e., $n + m$ bits in total) while the rest of channels are quantized using n bits. Generally, the higher the accuracy required, the more important channels are needed. Then DQA right-shifts the important activation channels by m bits and saves the shifting errors. For non-important channels, DQA uses the direct quantization method; ③ DQA encodes the shifting errors for important channels using Huffman coding [17] (Section 3.2). Immediately before the de-quantization operation, DQA decodes the encoded shifting errors. ④ finally, during the de-quantization operation, DQA compensates the information loss of the important activation channels by adding back the shifting error values to them.

Algorithms 1 and 2 describe steps ②, ③ and ④ of DQA in more detail. Note that important channels are quantized/de-quantized one by one, and that *se* refers to the shifting error. Also note that in practice, the *Important Channels (I)* are obtained from the rank of each layer using the importance ratios. Finally, the shifting errors can be obtained by reading the lower m bits of each activation value before shifting. Since there are only 2^m possible arrangements of lower m bits that correspond to the shifting errors, they can be converted to decimal floating point values using a pre-computed map (the map of lower m bits to shifting errors).

3.1 Ranking Important Activation Channels Using Greedy-Search

We hypothesize that there are important activation channels that can be treated differently during quantization, similar to how AWQ [11] processes weights. We define important activation channels

Algorithm 1 DQA Layer Quantization

Input: Activation Channels A, Target Bits n, Extra Bits m, Important Channels I, Huffman Encoder $H(.)$
Output: Quantized A, encoded shift-errors e
$e \leftarrow empty$
for $a \in A$ **do**
  if $a \in I$ **then**
    $\Delta_{n+m} \leftarrow \frac{|max(A_{layer})|}{2^{n+m-1}}$
    $a_{more} \leftarrow Round(\frac{a}{\Delta_{n+m}})$
    $a, se \leftarrow RightShift(a_{more}, m)$
    $e \leftarrow e + H(se)$
  else
    $\Delta_n \leftarrow \frac{|max(A_{layer})|}{2^{n-1}}$
    $a \leftarrow Round(\frac{a}{\Delta_n})$
return A, e

Algorithm 2 DQA Layer De-quantization

Input: Quantized Activation Channels A_q, Target Bits n, Encoded Shifting Errors e, Important Channels I, Huffman Dencoder with Values Converting $H(.)$
Output: De-Quantized activation channels A
for $a \in A_q$ **do**
  $\Delta_n = \frac{|max(A_{layer})|}{2^{n-1}}$
  if $a \in I$ **then**
    $a = \Delta_n \cdot (a + H(e))$
  else
    $a = \Delta_n \cdot a$
return A

as those for which skipping quantization to them can yield better DNN inference accuracy. Since activations are dynamically generated, it would be expensive to rank the activation channels online. As a result, we compute the ranks offline from training/calibration data. Assuming that both training/calibration and inference data are drawn from sufficiently similar distributions, the ranks computed offline can be reused during inference. Therefore, we can select important channels (which are fixed base on a tunable pre-selected ratio of the total channels in each layer) before inference. As a result, the memory requirement can be accurately calculated offline before inference.

To calculate the rank of the activation channels, we avoid brute force and dynamic programming approaches due to their impractical time complexity. Instead, we use greedy search that only requires $O(LN)$ operations, where N is the number of channels and L is the number of layers. In greedy search, we iterate the activation channels from the first layer. In each iteration, we skip the current activation channel and only quantize the remaining activation channels in the layer. We then run inference on evaluation data to measure the impact of skipping the current activation channel on accuracy. The skipped activation channels that provide the highest accuracy are considered the most important ones for the layer. We repeat this process for every layer, first quantizing the activation channels of the previous layers while skipping their most important ones.

3.2 Quantizing Important Activation Channels

To preserve more information for important activation channels, we quantize them with m extra bits, where m is less than or equal to the target number of bits n, so they are quantized using $n + m$ bits. Then, we right-shift the activations of important channels by m bits to reach the target bit length n. Note that this operation allows storing all quantized values with same bit length n, which helps avoid inefficiencies such us wasted storage [12]. Also note that right-shifting is a computationally cheap operation [4]. Since right-shifting will lead to information loss for the important activation values, we save the shifting errors and add them back during the de-quantization phase to compensate the information loss.

3.2.1 Saving Quantization Error

We now formally analyze the expected shifting errors for the important channels. We denote by A the important activation channels. The most straightforward quantization/de-quantization approach for the important activation channels can be expressed as [11]:

$$\mathrm{Q(A)} = \Delta_N \cdot Round(\frac{A}{\Delta_N}), \quad where \quad \Delta_N = \frac{|max(A)|}{2^{N-1}} \tag{1}$$

The quantization error of this method is (RE is the Rounding Error):

$$Error = |A - Q(A)| = |A - \Delta_N \cdot (\frac{A}{\Delta_N} + RE)| = \Delta_N \cdot RE \tag{2}$$

The average RE is 0.25 [11]. With DQA, A will be quantized with m extra bits first, and then right-shifted also with m bits. Then, the de-quantized important activation channels can be expressed as (SE is the Shifting Error):

$$\mathrm{Q(A)} = \Delta_N \cdot (RightShift(Round(\frac{A}{\Delta_{N+m}})) + SE) \tag{3}$$

Finally, the quantization error becomes:

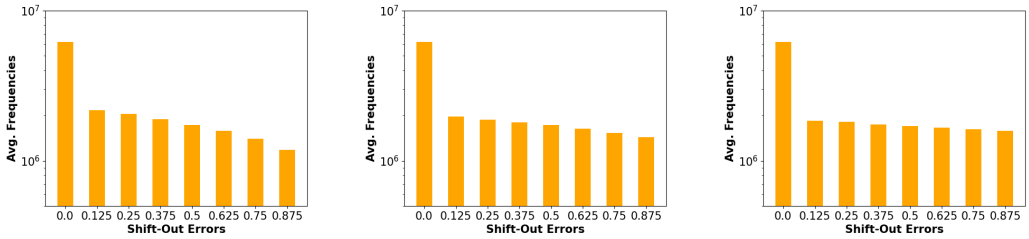
$$Error = |A - \Delta_N \cdot (\frac{1}{2^m} \cdot (\frac{A}{\Delta_{N+m}} + RE) - SE + SE)| = \Delta_N \cdot \frac{1}{2^m} \cdot RE \tag{4}$$

From equations 2 and 4, it is clear that this way of quantization exponentially reduces the quantization error. This is expected since we retain m more bits of information.

3.2.2 Memory Overhead

DQA achieves higher accuracy by saving the shifting errors, which involves some memory overhead. To reduce this memory overhead, we empirically explore the pattern of the shifting errors. As an example, we study the frequency distribution of the shifting errors for ResNet-32 and the CIFAR-10 dataset with $m = 3$. If a given layer is quantized, all its activations channels use the quantization described in Section 3.2.

In Figure 2 we observe that the distributions of shifting errors are not uniform. This indicates that they can be compressed by Huffman coding [17], which is able to encode the more frequent errors with fewer bits and the less frequent errors with more bits. In general, Huffman coding could generate shorter encoding information compared to directly storing m bits. This justifies our choice of using Huffman coding to compress the shifting errors. In our experiments in Section 4, the average compression ratios in experiment ($\frac{\text{Size_of_Original_Shifting_Errors}}{\text{Size_of_Huffman_Code}}$) for Huffman coding with the CIFAR-10 dataset are up to 1.12.



(a) 3 bits (b) 4 bits (c) 5 bits

Figure 2: 3, 4, 5 bits quantization with $m = 3$ shifting errors' average frequencies distribution.

4 Evaluation

4.1 Experimental Setup

For the image classification task, we use the ResNet-32 [7] and MobileNetV2 [8] models with the CIFAR-10 dataset [10]. For the image segmentation task, we use the U-Net [16] with the CityScapes [2] dataset. Note that U-Net stores the activations of encoder layers before concatenating them with activations of decoder layers. U-Net is a typical example where activations need to be i) stored for a relatively long term and ii) compressed to have a manageable peak memory usage (e.g., when targeting resource-constrained devices). Since DQA is designed for deep activation quantization, we set the target quantization levels to 3 different bits, 3 bits, 4 bits and 5 bits.

We implement our experiments using PyTorch [14] and run them on an Nvidia RTX 3090 GPU for we only focus on accuracy comparisons in the experiments (in the future work, we will also evaluate the performances such latency in real edge devices). Every experiment is run 5 times for each quantization level, and we take the average value in each case. For each DNN model of each

Table 1: Accuracy (Acc) results for the three DNN models under study.

(a) Classification: ResNet-32
Original Acc: 92.49%

Method	Bits	Acc (%)
Direct	3	52.85
NoisyQuant	3	71.22
DQA(m = 3)	3	**82.13**
Direct	4	87.74
NoisyQuant	4	89.66
DQA(m = 3)	4	**90.55**
Direct	5	91.33
NoisyQuant	5	91.86
DQA(m = 3)	5	**92.12**

(b) Classification: MobileNetV2
Original Acc: 91.42%

Method	Bits	Acc (%)
Direct	3	82.75
NoisyQuant	3	87.67
DQA(m = 3)	3	**88.63**
Direct	4	90.01
NoisyQuant	4	90.56
DQA(m = 3)	4	**90.84**
Direct	5	91.19
NoisyQuant	5	91.33
DQA(m = 3)	5	91.28

(c) Segmentation: U-Net
Original Acc: 92.5%

Method	Bits	Acc (%)
Direct	3	84.41
NoisyQuant	3	**92.16**
DQA(m = 3)	3	90.38
Direct	4	91.54
NoisyQuant	4	91.13
DQA(m = 3)	4	**92.03**
Direct	5	**91.48**
NoisyQuant	5	91.41
DQA(m = 3)	5	91.46

task, we first created 5 ranks with 5 random subsets of the training data using greedy search (see Section 3.1).

For fast evaluation, we select different training data subset for the different tasks. The size of the training data subset for creating each rank is 5000 for image classification and 50 for image segmentation for the model we use is relatively large and smaller amount of data can save the evaluation time. We select important channels with ratios of 40% for image classification and 50% for image segmentation, for all the target activations as empirical numbers. The batch size is 128 for image classification and 4 for image segmentation in all the corresponding experiments, including the direct method and NoisyQuant. For showing the impact of DQA to different quantization levels, we control the m value in all experiments. Also, to store as more information as possible for the important activation channels, we set $m = 3$ as it is the largest value we can use (the smallest quantization level is 3).

4.2 Results

We compare DQA with two methods: i) the direct quantization defined in Section 2; and the state-of-the-art NoisyQuant [13]. Even though NoisyQuant was designed for Vision Transformers [3], our experiments show that it also works well for other types of DNNs. Note that all experiments only quantize activations, i.e., without considering weights.

Table 1 shows the accuracy results for the models under study for the two tasks. As we can see, DQA achieves up to 29.28% higher accuracy than the direct method and NoisyQuant for image classification models. In the image classification task, DQA achieves more significant accuracy improvement than the comparisons on ResNet-32. We also see that the lower the quantization bits, the more accuracy advantages DQA can provide. For U-Net, we observe that DQA provides higher accuracy (up to 0.9%) than the direct method and NoisyQuant when quantizing at 4 bits, and similar accuracy when quantizing at 5 bits. For 3 bits, NoisyQuant provides the best accuracy value. Overall, DQA works better for classification tasks than segmentation task in our experiments.

5 Conclusion

In this paper we proposed DQA, an efficient method that applies deep quantization (using less than 6 bits) to DNN activations and uses cheap operations such as right-shifting to providing high accuracy while being suitable for resource-constrained devices. We evaluate DQA on three DNN models for both image classification and image segmentation tasks, showing up to 29.28% accuracy improvement compared to a direct quantization method and the state-of-the-art method NoisyQuant. In future work, we will co-design [6] DQA and a hardware accelerator to further exploit the benefits of DQA.

References

[1] Chen, Z., Xie, B., Li, J., Shen, C.: Channel-wise mixed-precision quantization for large language models. arXiv preprint arXiv:2410.13056 (2024)

[2] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)

[4] Elhoushi, M., Chen, Z., Shafiq, F., Tian, Y.H., Li, J.Y.: Deepshift: Towards multiplication-less neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2359–2368 (2021)

[5] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision, pp. 291–326. Chapman and Hall/CRC (2022)

[6] Gibson, P., Cano, J., Crowley, E.J., Storkey, A., O'Boyle, M.: DLAS: A Conceptual Model for Across-Stack Deep Learning Acceleration. ACM Transactions on Architecture and Code Optimization (TACO) (2024)

[7] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2016)

[8] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 (Apr 2017)

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

[10] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[11] Lin, J., Tang, J., Tang, H., Yang, S., Dang, X., Han, S.: Awq: Activation-aware weight quantization for llm compression and acceleration. arXiv preprint arXiv:2306.00978 (2023)

[12] Liu, R., Wei, C., Yang, Y., Wang, W., Yuan, B., Yang, H., Liu, Y.: A dynamic execution neural network processor for fine-grained mixed-precision model training based on online quantization sensitivity analysis. IEEE Journal of Solid-State Circuits **59**(9), 3082–3093 (2024)

[13] Liu, Y., Yang, H., Dong, Z., Keutzer, K., Du, L., Zhang, S.: Noisyquant: Noisy bias-enhanced post-training activation quantization for vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20321–20330 (2023)

[14] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: an imperative style, high-performance deep learning library. Curran Associates Inc. (2019)

[15] Rakka, M., Fouda, M.E., Khargonekar, P., Kurdahi, F.: Mixed-precision neural networks: A survey. arXiv preprint arXiv:2208.06064 (2022)

[16] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18. pp. 234–241. Springer (2015)

[17] Van Leeuwen, J.: On the construction of huffman trees. In: ICALP. pp. 382–410 (1976)