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 inference, which can be particularly beneficial on resource-constrained edge devices. To achieve high accuracy, existing methods that quantize activations rely on expensive mathematical computations or perform extensive online searches for the best hyper-parameters. However, these expensive operations are impractical on edge 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, we propose DQA (Deep Quantization of DNN Activations), a new method that focuses on sub-6-bit quantization of activations. DQA leverages simple shifting-based operations and Huffman coding to achieve high accuracy. We evaluate DQA with 3, 4, and 5-bit quantization levels and three different DNN models on two different datasets for two different tasks, image classification and image segmentation. DQA shows significantly better accuracy (up to 29.28%) compared to the direct method and NoisyQuant for sub-6-bit quantization.

1 Introduction

Quantization is a popular compression method 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 techniques are used. For example, NoisyQuant [12] injects noise in the activations before quantization and removes the noise after the de-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 like sub-6-bit quantization (i.e., 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 shift-out errors using Huffman coding. Then, the shift-out 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 the activations, i.e., it simply applies the mathematical definition of quantization/de-quantization [5; 10]. By learning the importance of channels offline, DQA avoids the need to separate the channels online, saving significant computational resources; and by using

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

an m more bits quantization approach, DQA can quantize the important channels differently like mixed-precision quantization [1; 14], and can also reduce the rounding error exponentially and can efficiently shift back to the target bits without expensive mathematical computations. It is 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 [11]; ii) by applying Huffman coding to the shift-out 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 [9], and CityScapes [2]) for three different DNN models (ResNet-32 [6], MobileNetV2 [7] and U-Net [15]) 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/de-quantization and NoisyQuant [12].

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 shift-out 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/de-quantization and NoisyQuant [12], clearly outperforming them on accuracy.

2 Background and Related Work

Quantization is a well-known compression method to lower the precision format of parameters in DNNs, which reduces memory storage and computational requirements [5; 8]. The initial 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, 8 bits). The process normally includes clipping and rounding operations. Clipping restricts the minimum and maximum values of the quantization, whereas rounding rounds the values into the closest integers, which causes rounding errors that cannot be recovered during de-quantization. De-quantization is the opposite operation to quantization that approximately recovers the quantized values back to floating point.

Quantization is 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 it is the same as the direct quantization method for unimportant channels described 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 [14]. To improve that, 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; 14]. However, due to the differences in bit length among the quantized values, mixed-precision quantization can cause hardware inefficiencies such as the wasted storage [11]. Our method adapts mixed-precision quantization in an effective way.

Furthermore, previous works on DNN quantization apply it to the model weights and/or activations. For weights, AWQ [10] 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 [12] 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 methods do not focus on deep quantization like sub-6-bit quantization, thus being less suitable for resource-constrained devices.

3 Proposed Method: DQA

In this section, we present our DQA method in detail. The main goal of DQA is to minimize the computation and memory overheads for efficient quantization of DNN activations. Figure 1 gives an overview of DQA.

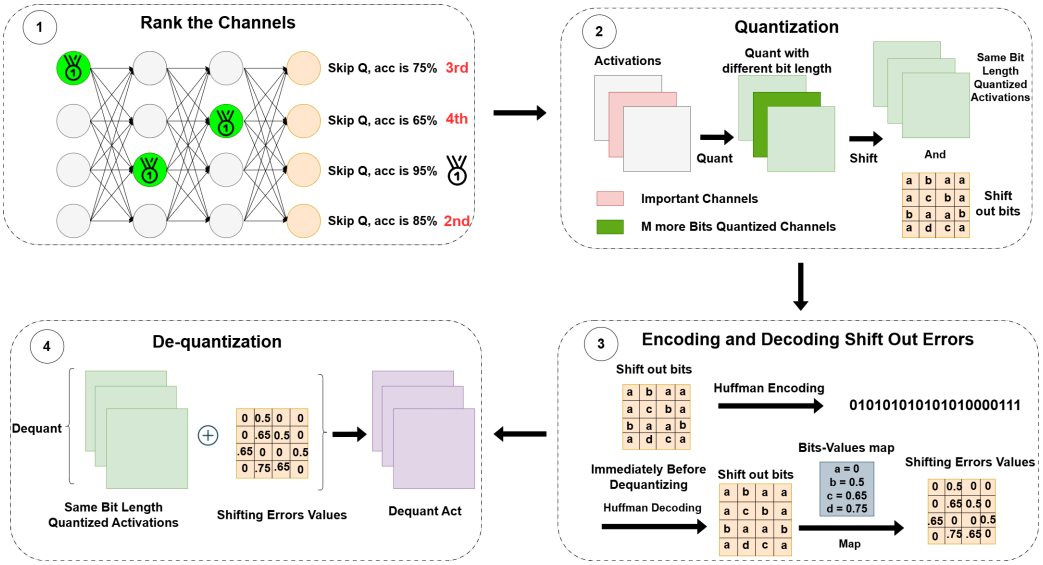


Figure 1: DQA overview. ① check the activation importance ranks based on offline training or calibration data (the green circles indicate important channels for which quantization is skipped); ② during inference, quantize important channels with m extra bits and right shift while bookkeeping the shift-out errors; ③ the errors are Huffman encoded to reduce memory usage. Before de-quantizing, decode the Huffman-encoded shift-out errors and add them back to the quantized channels; ④ de-quantize important channels. For non-important channels, use the direct method to quantize and de-quantize.

Algorithm 1 DQA Layer Activation Quant

Input:
(1) Activation Channels A, (2) Target Bits n, (3) Extra Bits m, (4) Important Channels I, (5) Huffman Encoder $H(.)$
Output: Quantized A, encoded shift-errors e
$e \leftarrow empty$
for $a \in A$ **do**
  if $a \in I$ **then**
    $\Delta_{n+m} \leftarrow \frac{|max(A_{layer})|}{2^{n+m-1}}$
    $a_{more} \leftarrow Round(\frac{a}{\Delta_{n+m}})$
    $a \leftarrow RightShift(a_{more}, m)$
    $e \leftarrow e + H(a_{more} - a)$
  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 Activation De-quant

Input:
(1) Quantized Activation Channels A_q, (2) Target Bits n, (3) Encoded Shifting Errors e, (4) Important Channels I, (5) 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

Given a target of quantization bits n, DQA starts by quantizing the target activations to n bits. As shown in Figure 1, ① DQA first uses the training or calibration data to find the rank (i.e., the channels importance) of the activation channels by an offline greedy search (as described in Section 3.1); ② it then uses the knowledge of the channels' rank to determine the activation channels. During quantization, for a pre-selected amount of important activation channels in each layer, DQA quantizes them using m more (extra) bits than the rest of channels, which are only quantized to the target n

bits. Generally, the higher the accuracy required, the more important channels are needed. Then DQA right-shifts the important channels by m bits while keeping the shifted-out m bits of each value. For non-important channels, DQA uses the direct quantization method; ③ DQA encodes the shift-out bits for important channels using Huffman coding [16] (Section 3.2). Immediately before the de-quantization operation, DQA decodes the encoded shift-out bits and converts them to decimal floating point error values; ④ finally, during the de-quantization operation, DQA compensates the information loss of the corresponding channels by adding the error values to them.

Algorithms 1 and 2 describe the steps in more detail. Note that in practice, the *Important Channels I* are the top channels get from each layer's channel rank by the important channel ratios.

3.1 Channel Importance Ranking Based on Greedy-Search

We hypothesize that there are important activation channels that can be treated differently during quantization, similar to how AWQ [10] processes weights. We define important activation channels as those for which skipping quantization yields significantly better accuracy than skipping others. Since activations are dynamically generated, it is expensive to find the ranks of the activation channels online. As a result, we leverage the offline activation ranks from training or calibration data. Assuming the training/calibration data and the inference data are drawn from sufficiently similar distributions, the ranks can be reused online during inference. Based on the ranks, we can select important channels, which are fixed before inference. As a result, the storage requirement is constant and thus can be accurately known when designing the hardware.

To find the ranks of the activation channels, we avoid brute force and dynamic programming approaches due to their impractical time complexity. Instead, we use greedy search which only requires $O(LN)$, where N is the number of channels and L is the number of layers. In greedy search, we iterate the channels from the first layer. At each iteration, we skip the current channel and only quantize the remaining channels in the layer. Then we evaluate the model on evaluation data. The skipped channels that give the highest accuracies are considered as the most important channels for this layer. We repeat this process for every layer, first quantizing the activation channels of the previous layers and skipping their most important channels.

3.2 Round-Shift quantization

To preserve more information from important activation channels, we quantize them with m more bits, where m should to be less than or equal to the target number of bits n. Then, we right-shift the activations by m bits to reach the target bit length. Note that this operation allows the devices to store the quantized values with same bit length, which helps DQA to avoid hardware inefficiencies like wasted storage [11]. Also note that right-shifting is computationally much cheaper than multiplication [4]. Since right-shifting will lead to information loss for the values, we store the shift-out errors (the shifted-out lower m bits) after the operations and add them back during the de-quantization phase to compensate the information loss. Since there are only 2^m possible arrangements of the shifted-out m bits that correspond to the shift-out errors, they can be easily converted to decimal floating point values by a pre-computed map.

3.2.1 Quantization Error

We now formally analyze the expected Round-Shift quantization error for the important channels. We denote by A the important activation channels. The most straightforward quantization/de-quantization approach to the important activation channels can be expressed as [10]:

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

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 Rounding Error is 0.25 [10]. With DQA, A will be quantized to m more bits first, and then do m bits right-shifting. Then, the de-quantized important activation channels can be expressed

as (SE is the Shifting Error):

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

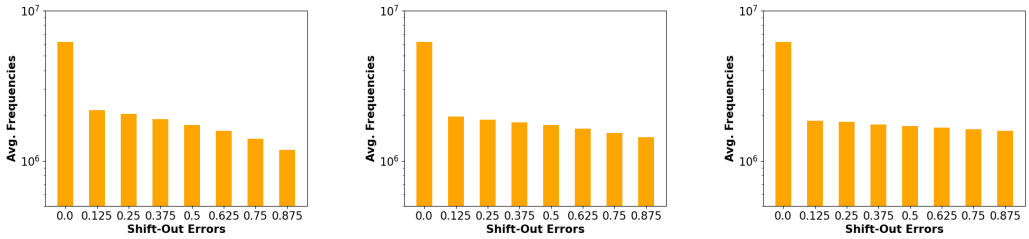
Then the quantization error becomes:

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

From equations 2 and 4, it is clear that Round-Shifting 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 storing the shift-out errors, however this involves some memory overhead. To reduce this memory overhead, we empirically explore the pattern of the shift-out errors. As an example, we study the frequency distribution of the shift-out errors for ResNet-32 and the CIFAR-10 dataset with $m = 3$. If a given activation is quantized, all its channels use Round-Shift quantization.



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

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

In Figure 2 we observe that the distributions of the shift-out errors are not uniform. This indicates that they can be compressed by Huffman coding [16], which is able to encode the more frequent errors with fewer bits and less frequent errors with more bits. In general, it could generate shorter encoding bits compared to directly storing m bits. This justifies our choice of using Huffman coding to compress the shift-out errors. In our experiments in Section 4.2.1, the average compression ratios ($\frac{\mathrm{SizeofOriginalShiftoutErrors}}{\mathrm{SizeofHuffmanCode}}$) for Huffman coding with the CIFAR-10 dataset are up to 1.12.

4 Evaluation

4.1 Experimental Setup

For the image classification task, we use the ResNet-32 [6] and MobileNetV2 [7] models with the CIFAR-10 dataset [9]. For the image segmentation task, we use the U-Net [15] 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 stored for a relatively long term and need to be compressed to have a manageable peak memory usage (e.g., when targeting resource-constrained devices).

We implement our experiments using PyTorch [13] and run them on an Nvidia RTX 3090 GPU. For each layer, we quantize/de-quantize the activations channel-wise batch by batch. Note that for simplicity, we only apply the operations on activations that are outputted from the main branch of main blocks (i.e., not the blocks on the shortcut branches). Every experiment is run 5 times for each number of quantization bits, and we take the average value in each case. For each model in each task, we first created 5 rank maps (every map includes the ranks of the activation channels for each main block of the whole model) with 5 random subsets of the training data using greedy search (see Section 3.1).

Table 1: Image Classification task and Image Segmentation task results (Acc means accuracy)

(a) Classify by 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) Classify by MobileNetV2
Original Acc: 91.42%

Method	Bits	Acc (%)
Direct	3	82.75
NoisyQuant	3	87.67
DQA(m = 3)	3	**88.76**
Direct	4	90.01
NoisyQuant	4	90.56
DQA(m = 3)	4	**90.85**
Direct	5	91.19
NoisyQuant	5	91.33
DQA(m = 3)	5	91.31

(c) Segment by 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

4.2 Results

We compare DQA with two methods: i) the direct quantization method defined in Section 2; and NoisyQuant [12] which is state-of-the-art. Even though NoisyQuant is designed for Vision Transformers [3], it should still work for other types of DNNs. Note that all experiments are only for activations, i.e., without considering weights.

4.2.1 Image Classification Task

The size of each random training data subset for creating each rank map is 5000. We select important channels with ratios of 40% for all channels in all blocks. The batch size is 128 for all the experiments, including the direct method and NoisyQuant.

In Tables 1a and 1b we observe that DQA with $m = 3$ can achieve up to 29.28% higher accuracy than the direct method and NoisyQuant. We also see that the lower the quantization bits, the more accuracy advantages DQA can provide.

4.2.2 Image Segmentation Task

The size of each random training data subset for creating each rank map is 50. We select important channels with ratios of 50% for all channels in all blocks. The batch size is 4 for all the experiments including the direct method and NoisyQuant.

From Table 1c, we observe that DQA with $m = 3$ can provide higher accuracy (up to 0.9%) than the direct method and NoisyQuant after quantization when the quantization bits are equal to 4, and similar accuracy when the quantization bits are equal to 5.

5 Conclusion

In this paper we proposed DQA, an efficient method that applies deep quantization to DNN activations using cheap operations such as right-shifting while providing high accuracy, thus being suitable for resource-constrained devices. We evaluate DQA on 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. In future work, we will co-design the hardware accelerator that can further exploit the benefits of DQA [17].

Acknowledgments

This work was partially supported by the EU Project dAIEDGE (GA Nr 101120726).

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] 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)

[7] 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 [cs] (Apr 2017)

[8] 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. pp. 2704–2713 (2018)

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

[10] 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)

[11] 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). https://doi.org/10.1109/JSSC.2024.3377292

[12] 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)

[13] 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)

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

[15] 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)

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

[17] Wolf, W.H.: Hardware-software co-design of embedded systems. Proceedings of the IEEE **82**(7), 967–989 (1994)