DEPENDENT SCALAR QUANTIZATION FOR NEURAL NETWORK COMPRESSION

Paul Haase[1], *Heiko Schwarz*[1,2], *Heiner Kirchhoffer*[1], *Simon Wiedemann*[1], *Talmaj Marinc*[1], *Arturo Marban*[1], *Karsten Müller*[1], *Wojciech Samek*[1], *Detlev Marpe*[1], *Thomas Wiegand*[1,3]

[1] Fraunhofer Heinrich-Hertz-Institute, Berlin, Germany
[2] Institute of Computer Science, Free University of Berlin, Germany
[3] Department of Telecommunication Systems, Technical University of Berlin, Germany

ABSTRACT

Recent approaches to compression of deep neural networks, like the emerging standard on compression of neural networks for multimedia content description and analysis (MPEG-7 part 17) , apply scalar quantization and entropy coding of the quantization indexes. In this paper we present an advanced method for quantization of neural network parameters, which applies dependent scalar quantization (DQ) or trellis-coded quantization (TCQ), and an improved context modeling for the entropy coding of the quantization indexes. We show that the proposed method achieves 5.778% bitrate reduction and virtually no loss (0.37%) of network performance in average, compared to the baseline methods of the second test model (NCTM) of MPEG-7 part 17 for relevant working points.

***Index Terms*—** dependent scalar quantization, trellis-coded quantization, entropy coding, arithmetic coding, neural network compression

1. INTRODUCTION

Deep neural networks have become a major element in the field of machine learning. Recent advances, especially due to the availability of large amounts of data and growing computational resources, as well as novel algorithms and model architectures [1], enabled more and more complex machine learning tasks but also lead to an exponential growth in the number of parameters over the past years [2]. Implicitly, the models are also becoming more complex with respect to memory requirements [3]. Particularly, this can be problematic for different scenarios with constrained memory resources or bandwidth-limited communication channels, e.g., on mobile devices or in distributed learning applications [4].

However, this problem can be addressed by applying model compression techniques, such as DeepCABAC [5], an entropy coding method based on Context-Based Binary Arithmetic Coding (CABAC) [6] originally developed for state-of-the-art video coding standards like H.265/HEVC (High Efficiency Video Coding) [7] or the emerging H.266/VVC (Versatile Video Coding) [8]. Most recently, DeepCABAC was adopted as part of the upcoming standard on compression of neural networks for multimedia content description and analysis (MPEG-7 part 17) [9], currently developed within ISO-MPEG (Moving Picture Experts Group).

DeepCABAC (as in [9]) specifies scalar quantization with a uniform reconstruction quantizer (URQ), for which the admissible reconstruction values are completely determined by the quantization step size Δ and the mapping of quantization indexes q to reconstructed weight parameters $w' = q \cdot \Delta$. For an increased compression efficiency, the quantization index selection at the encoder aims at minimizing a Lagrangian function $D + \lambda R$ with distortion D and number of bits R that are required for representing each quantization index [10].

Applying some form of vector quantization instead, can further improve the coding efficiency, since the admissible reconstruction vectors are denser packed in the high dimensional signal space. Due to its complexity, unconstrained vector quantization is not suitable, but there are variants with reasonable encoder and decoder complexity like dependent scalar quantization (DQ) also referred to as trellis-coded quantization (TCQ) [11]-[14]. DQ has been investigated in image (see [15]) and video coding and is currently part of the emerging video coding standard VVC [8]. In the following, a neural network parameter coding design with dependent scalar quantization and DeepCABAC is described. The coding efficiency is evaluated by integrating the approach into the neural network compression test model (NCTM) [10] of MPEG-7 part 17.

2. DEPENDENT SCALAR QUANTIZATION

Dependent scalar quantization (DQ, or trellis-coded quantization (TCQ)), first described in [11], can be considered as a vector quantizer for very large dimensions, but with a restricted set of values for the vector components. From a decoder point of view, it specifies two quantizers and a procedure for switching between them. Commonly, the admissible reconstruction values are denoted by integer multiples of a quantization step size Δ for both scalar quantizers. The switching process can be represented by a state machine with

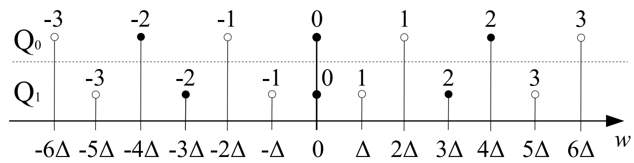

Fig. 1. The scalar quantizers Q_0 and Q_1 used in the chosen DQ design. All reconstruction values represent integer multiples of the quantization step size Δ. The labels above the circles show the associated quantization indexes.

2^K states ($K \geq 2$), where each state is associated with one of the scalar quantizers. The current state, and hence the applied quantizer, is uniquely determined by the previous state and the value of the previous quantization index.

For encoding, the potential transitions between the two scalar quantizers can be illustrated by a trellis with 2^K states per sample. Thus, selecting the optimal sequence of quantization indexes is equivalent to finding the trellis path with minimum rate-distortion cost. Principally, this problem can be solved optimally, using the well-known Viterbi algorithm [16]. The packing density in the high-dimensional signal space, and thus the coding efficiency, for long sample sequences is maximized by choosing the state transitions (state machine) properly. Furthermore, the achievable packing density, but also the encoding complexity, increases with the number of states.

2.1. Quantizer Design

The structure of the two scalar quantizers Q_0 and Q_1, used in the proposed method, is depicted in Fig. 1. As it can be seen, Q_0 contains all even multiples of the quantization step size Δ and Q_1 contains all odd multiples of the quantization step size Δ. Additionally, both quantizers include the value of zero which generally improves the low rate performance of DQ [12, 17]. Analogous to [14] (and thus different from [12, 17]), symmetric quantizers are chosen due to higher coding efficiency in the conducted experiments. However, integer quantization indexes q indicate the reconstruction values, whereas the quantization index equal to zero is associated with reconstruction value equal to zero.

Typically, layers of deep neural networks contain a large number of parameters, but depending on the layer or network type, a significant amount of the parameters is equal to zero. In order to address these properties adequately, a DQ design with 8 states is chosen, which provides a good trade-off between complexity and coding gain. This DQ design and the quantizer selection process are illustrated in Table 1. Since a current state is determined by the previous state and the value of the previous quantizer index, the neural network parameters of a layer have to be reconstructed in a pre-defined order, which shall be indicated by the indexes $k = 0, 1, \ldots$. The initial state s_0 is set to zero. Then, given a current state s_k and the value of the current quantization index q_k, the next state s_{k+1} is uniquely determined by the current state s_k and the parity p_k of the current quantization index q_k.

Table 1. State transitions for quantizer selection. p_k represents the parity of the current quantizer index q_k

state s_k		0	1	2	3	4	5	6	7
quantizer used		Q_0	Q_0	Q_1	Q_1	Q_0	Q_0	Q_1	Q_1
next state s_{k+1}	$p_k = 0$	0	4	5	1	6	2	3	7
	$p_k = 1$	4	0	1	5	2	6	7	3

As shown in table 1, for neural network parameters, that are associated with the states s_k equal to 0, 1, 4 and 5, Q_0 is used, and for the parameters, associated with the states s_k equal to 2, 3 , 6 and 7, quantizer Q_1 is used. As a consequence of the state transition process, the quantization indexes for each quantizer are partitioned into two subsets, one with parity 0 and the other with parity 1, indicated by filled and hollow circles in Fig. 1.

2.2. Reconstruction Process for Network Parameters

The proposed DQ design requires that the neural network parameters of a layer are reconstructed sequentially in a predefined order, chosen equal to the coding order of the quantization indexes, since the knowledge about the selected quantizer can be exploited in the entropy coding (sec. 3). Here, the reconstruction is processed in row-first order (left to right, top to bottom). Then, the algorithm for obtaining the N reconstructed parameters w' of a layer, given the quantization indexes q_k, where k indicates the reconstruction order, is given with the pseudo code:

$s_0 = 0$
for $k = 0$ to $N - 1$ **do**
  $w'_k(q_k, s_k) = \big(2 \cdot q_k - ((s_k \gg 1)\ \&\ 1) \cdot \mathrm{sgn}(q_k)\big) \cdot \Delta$
  $s_{k+1} = \mathrm{sttab}[\,s_k\,][\,q_k\ \&\ 1\,]$
end for

Note that, Δ is the quantization step size, $\mathrm{sgn}(\cdot)$ denotes the signum function, and the 2-D array sttab[·][·] represents the state transition given in Table 1. The parity p_k of the quantization index q_k can be obtained by applying the bit-wise "and" operator $\&$ (two's complement arithmetic) according to $q_k \& 1$. The operator "$\gg$" represents a bit shift to the right.

3. ENTROPY CODING

In order to achieve a higher coding efficiency, the entropy coding of the DeepCABAC approach in [10] is modified. Specifically, an improved context modeling is applied, such that it better exploits the conditional statistics of the parameters, induced by the DQ method. Prior to encoding, each quantization index is decomposed into a series of binary decisions, so-called bins, according to the binarization procedure of DeepCABAC (see [5], [9]), which enables adaption to typical distributions of neural network parameters, where usually most of parameters are close to the value 0. Since the binarization is not changed for DQ, only a short summary is given.

First a bin, *SigFlag* (Significance Flag), is encoded which determines whether a quantization index is non-zero, followed by another bin (*SignFlag*), which denotes the value of the sign, and subsequently a series of bins (*AbsGr(n_i)Flags*, with $n_i = 0, 1, \ldots, 10$) indicating if the current quantization index is greater than n_i. Whenever the *SigFlag* or a *AbsGr(n_i)Flags* is equal to zero, encoding for a current quantization index is terminated. Finally, if there is a remainder, it is encoded using an Exponential Golomb code [18]. Usually the bins, i.e. *SigFlag*, *SignFlag*, *AbsGr(n_i)Flags* and the unary part of Exponential Golomb code (*ExpGoUnary(l_a)*, $a = 0, 1, \ldots$), are coded using probability models (regular mode), also denoted as context models, which enable the entropy coder to adapt to the source statistics. All other bins use a bypass mode without probability models.

For bins in regular mode, a probability model is selected from a set of possible candidates, based on a so-called context (e.g. previously coded bins in a local neighborhood), in order to better exploit local statistics. The selected context models for *SigFlag* and *SignFlag*, where each has three possible candidates, depend on the value of the quantization index directly preceding the current quantization index within the same row. For the *AbsGr(n_i)Flags* and *ExpGoUnary(l_a)*, the selection is based on the value of the preceding *SignFlag*, where each flag has a set of two possible models.

However, due to the quantizer design in DQ, the distances between zero and the first non-zero reconstruction values are different for Q_0 and Q_1. For the *SigFlag*, this suggests to use distinct sets of context models for each quantizer. Here, experiments indicated that assigning an own set of context models for each state value yields even higher coding efficiency. Consequently, the number of context models for the *SigFlag* is increased from 3 up to 24. There are no changes in the context modeling for other bins. For further details on context modeling and binarization of DeepCABAC, refer to [9], [5].

4. QUANTIZER INDEX SELECTION

In order to maximize the coding efficiency, the selection of quantization indexes q_k in the encoder aims at minimizing a Langrangian cost function $J = D + \lambda R$ of the distortion D, measuring the mean-squared error (MSE), and the number of bits R required for transmitting the quantization indexes. The Lagrange multiplier λ depends on the quantization step size Δ and scaling parameter between 0 and 1. The distortion D_k for an individual neural network parameter w_k is given by:

$$D_k(q_k, s_k) = (w_k - w'_k(q_k, s_k))^2 \tag{1}$$

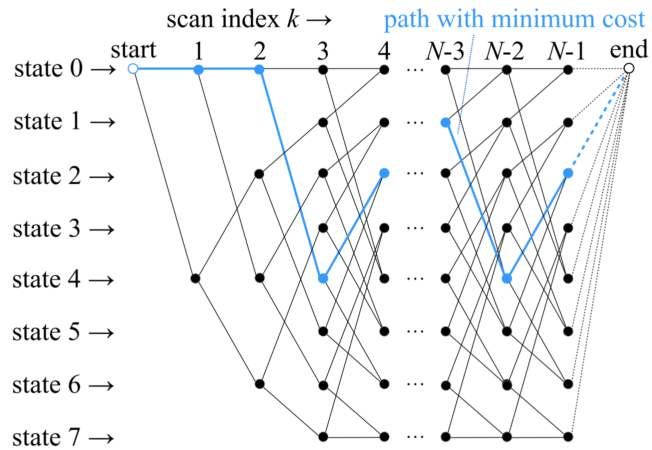

Fig. 2. Trellis structure used for determining the quantization indexes in the encoder. An exemplary path with minimum cost is highlighted as blue line.

where w_k represents the original neural network parameter, and $w'_k(q_k, s_k)$ the reconstructed neural network parameter as given in Sec. 2.2. Note that, due to the complex dependencies, the impact, of quantizing an individual parameter, on the corresponding performance measure can not be evaluated during the quantizer selection procedure. Hence, although it does not yield optimal results, the MSE is used in our approach.

As mentioned above, the solution to the optimal quantization problem is then equivalent to finding the path through the trellis graph, depicted in Fig. 2, with the minimum associated rate-distortion cost, as given by the Viterbi algorithm [16]. Strictly, the Viterbi algorithm does not achieve the global optimal solution, due to the complicated dependencies (between the quantization indexes) in the entropy coding, but provides a good trade-off between coding efficiency and implementation complexity. In fact, the trellis (Fig. 2), and thus the state transitions (Tab. 1), are identical to those in JPEG2000 [15].

The trellis is processed in coding order, indicated by the index $k = 0, \ldots, N-1$ (where N is the number of neural network parameters in the layer), and the rate-distortion cost for the initial state (state 0) is $J_i = 0$. All possible transitions (connections of the trellis nodes) between a scan index $k-1$ and a next scan index k are associated with rate-distortion costs J_k according to:

$$J_k = K_{k-1} + D_k(q_k, s_k) + \lambda \cdot R_k(q_k | \cdots) \tag{2}$$

J_k denotes the rate-distortion cost of the source node and $R_k(q_k|\cdots)$ represents the estimated number of bits required for transmitting a quantization index q_k given the preceding quantization indexes $q_{k-1}, q_{k-2}, \ldots$ on the path from start to source node. For a current neural network parameter at stage k the costs for all connections are computed. Then, for each destination node, only the connection with the minimum rate-distortion cost is kept and this cost J_k is assigned to the des-

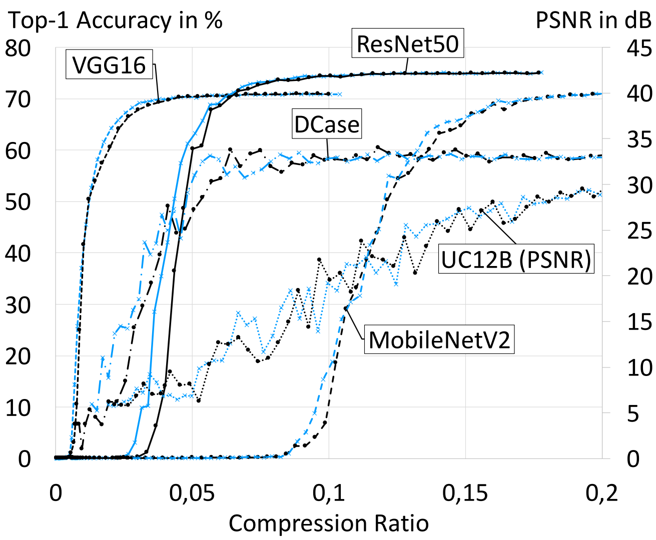

Fig. 3. Simulation results for the test models, where the performance measures, Top-1 Accuracies and PSNR (only for UC12B) are presented with respect to the compression ratios. The blue lines indicate results for the proposed DQ method and black lines the corresponding baseline results.

tination node. Continuing this until the last neural network parameter of the layer is processed ($k = N-1$), yields 8 surviving paths through the trellis. Finally, the sequence of selected quantization indexes $q_0, q_1, \ldots, q_{N-1}$ can be obtained by choosing the path with the minimum cost J_{N-1}.

5. EXPERIMENTAL RESULTS

For evaluation, the dependent scalar quantization method and the changes to the entropy coding are integrated in the Test Model (NCTM) [10] of MPEG-7 part 17 [9]. The experiments are conducted on five test model neural networks (specified in [19]), where three are for image classification (VGG16, ResNet50 and MobileNetV2), one for audio classification (DCase) and another one is an image autoencoder (UC12B). The networks for image and audio classification use Top-1 Accuracy in % as performance measure, whereas the image autoencoder UC12B applies the PSNR (Peak Signal-to-Noise-Ratio) of the reconstructed image. The coding performance is presented with respect to the baseline methods of NCTM for quantization and encoding.

Fig. 3 shows the achieved performance measure values and corresponding compression ratios C for different step size parameters Δ in the range from 2^{-15} to 1, where C is defined as $C = R_C/R_O$, with R_C being the number of compressed bits and R_O the number of uncompressed bits. Additionally, some working points (and related encoding and decoding runtimes) are presented in Tab. 2, which are selected such that, the highest compression with a maximum loss of 1% of performance measure is achieved with respect to the

Table 2. Results for relevant working points. DQ denotes results for the proposed method and B for the baseline method. Compression ratios C, performance measures Perf. and encoder/decoder runtimes $t_{Enc/Dec}$ are presented.

Model		C	Perf.	t_{Enc}	t_{Dec}
VGG16	B	0.04503	70.284%	4.314s	3.714s
	DQ	0.04296	70.224%	16.53s	3.952s
Result		-4.597%	-0.085%	383%	106%
Res-Net50	B	0.09675	74.398%	1.357s	1.220s
	DQ	0.09109	74.382%	3.551s	1.246s
Result		-5.850%	-0.022%	262%	102%
Mobile-NetV2	B	0.18855	70.78%	0.426s	0.391s
	DQ	0.18765	70.928%	0.727s	0.393s
Result		-0.477%	0.209%	171%	110%
DCase	B	0.06406	60.00%	0.020s	0.010s
	DQ	0.05656	58.89%	0.031s	0.010s
Result		-11.71%	-1.85%	155%	100%
UC12B	B	0.22566	29.91dB	0.027s	0.018s
	DQ	0.21153	29.88dB	0.029s	0.017s
Result		-6.262%	-0.100%	107%	94.4%
Average		**-5.778%**	**-0.370%**	**216%**	**102%**

unquantized models VGG16 (70.93%), ResNet50 (74.98%), MobileNetV2 (71.47%), DCase (58.27%) and UC12B (30.13 dB). This working point was selected as the most relevant, since the aim of neural network compression is to achieve high compression but virtually no loss in model performance. For these working points the results show an average bitrate reduction of 5.778%, whereas the performance measure is virtually the same (average loss of 0.37%). The encoder runtime is 216% and the decoder runtime 102%, in average, with respect to the baseline methods of NCTM. For all tests the Langrangian multiplier is set 0, which yields the best results.

6. CONCLUSION

Dependent scalar quantization shows improved coding efficiency for compression of deep neural networks when compared to conventional scalar quantization as applied as baseline method in the current draft of MPEG-7 part 17 [9]. It can be interpreted as a constraint vector quantization method which provides part of the space-filling advantage, but also has an acceptable trade-off regarding encoder complexity and coding efficiency. In fact, the decoder complexity is virtually the same compared to the scalar quantization method of [9]. Additionally, minor changes to the context modeling in the entropy coding stage of DeepCABAC provide further improvements of the coding efficiency. Consequently, the method is proposed for adoption into the standard on compression of neural networks for multimedia content description and analysis MPEG-7 part 17.

7. REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Xiaowei Xu, Yukun Ding, Sharon Xiaobo Hu, Michael Niemier, Jason Cong, Yu Hu, and Yiyu Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, pp. 216–222, 2018.

[3] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.

[4] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2019.

[5] Simon Wiedemann, Heiner Kirchhoffer, Stefan Matlage, Paul Haase, Arturo Marbán, Talmaj Marinc, David Neumann, Tung Nguyen, Ahmed Osman, Detlev Marpe, Heiko Schwarz, Thomas Wiegand, and Wojciech Samek, "Deepcabac: A universal compression algorithm for deep neural networks," *CoRR*, vol. abs/1907.11900, 2019.

[6] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620–636, July 2003.

[7] ITU-T and ISO/IEC, "High efficiency video coding," ITU-T Rec. H.265 and ISO/IEC 23008-10, vers. 1, 2013.

[8] B. Bross, J. Chen, S. Liu, and Y.-K. Wang, "Versatile video coding (draft 8)," Joint Video Experts Team (JVET), doc. JVET-Q2001, Brussels, Jan. 2020.

[9] MPEG, "Working draft 2 of compression of neural networks for multimedia content description and analysis," Tech. Rep. w18784, ISO/IEC JTC1/SC29/WG11, Geneva, October 2019.

[10] MPEG, "Test model 2 of compression of neural networks for multimedia content description and analysis," Tech. Rep. w18785, ISO/IEC JTC1/SC29/WG11, Geneva, October 2019.

[11] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and gauss-markov sources," *IEEE Transactions on Communications*, vol. 38, no. 1, pp. 82–93, Jan 1990.

[12] J. H. Kasner, M. W. Marcellin, and B. R. Hunt, "Universal trellis coded quantization," *IEEE Transactions on Image Processing*, vol. 8, no. 12, pp. 1677–1687, Dec 1999.

[13] T. R. Fischer and M. Wang, "Entropy-constrained trellis-coded quantization," *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 415–426, March 1992.

[14] H. Schwarz, T. Nguyen, D. Marpe, and T. Wiegand, "Hybrid video coding with trellis-coded quantization," in *2019 Data Compression Conference (DCC)*, March 2019, pp. 182–191.

[15] David Taubman and Michael Marcellin, *JPEG2000 Image Compression Fundamentals, Standards and Practice*, Springer Publishing Company, Incorporated, 2013.

[16] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, March 1973.

[17] R. L. Joshi, V. J. Crump, and T. R. Fischer, "Image subband coding using arithmetic coded trellis coded quantization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 515–523, Dec 1995.

[18] Jukka Teuhola, "A compression method for clustered bit-vectors.," *Inf. Process. Lett.*, vol. 7, pp. 308–311, 10 1978.

[19] MPEG, "Description of core experiments on compression of neural networks for multimedia content description and analysis," Tech. Rep. w18782, ISO/IEC JTC1/SC29/WG11, Geneva, October 2019.