ENCODER OPTIMIZATIONS FOR THE NNR STANDARD ON NEURAL NETWORK COMPRESSION

Paul Haase[1], Daniel Becking[1], Heiner Kirchhoffer[1], Karsten Müller[1], Heiko Schwarz[1,2], 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

The novel Neural Network Compression and Representation Standard (NNR), recently issued by ISO/IEC MPEG, achieves very high coding gains, compressing neural networks to 5% in size without accuracy loss. The underlying NNR encoder technology includes parameter quantization, followed by efficient arithmetic coding, namely DeepCABAC. In addition, NNR also allows very flexible adaptations, such as signaling specific local scaling values, setting quantization parameters per tensor rather than per network and supporting specific parameter fusion operations.

This paper presents our new approach for optimally deriving these parameters, namely the derivation of parameters for local scaling adaptation (LSA), inference-optimized quantization (IOQ), and batch-norm folding (BN). By allowing inference and fine tuning within the encoding process, the quantization errors are reduced and the NNR coding efficiency is further improved to a compressed bitstream size of only 3% in comparison to the original model size.

Index Terms— MPEG, NNR, DeepCABAC, neural network compression, encoder optimization

1. INTRODUCTION

Neural Networks (NNs) have demonstrated remarkable breakthrough's in a wide range of machine learning tasks [1], such as image classification, speech recognition, object detection, natural language understanding, etc. For this, current NNs are equipped with several millions of neuron connections [2], sometimes even in the order of billions, making them very resource-intensive. Furthermore, NN applications become distributed across many devices, e.g., in federated learning and training [3][4], imposing very large network traffic, when the NN weights and bias data (representing the NN neuron connections) are transmitted across devices.

To address the demand for efficient NN compression and transmission, the ISO/IEC Moving Picture Experts Group (MPEG) has issued the first international standard for compression and representation of neural networks (NNR) [5]. With efficient tools for preprocessing, quantization and entropy coding, NNR achieves already very high compression rates for neural network models without degrading performance measures. However, the choice of actual tool parameters is usually not within the scope of the standard, but rather up to specific encoder implementations. Accordingly, this paper presents three specific encoder tools, namely local scaling adaptation, batch-norm folding, and inference-optimized quantization together with specific methods to optimize their parameters in order to increase the compression efficiency. These methods are implemented and evaluated using the NNR reference software.

2. NNR CODING STRUCTURE

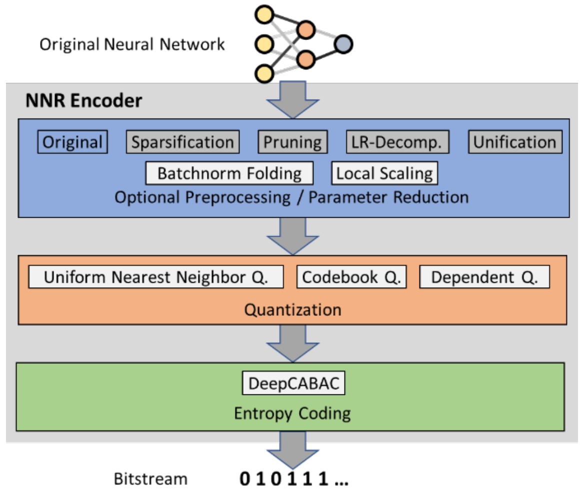

Fig. 1: Coding Structure of the NNR Standard (from an encoder perspective).

A typical encoder compliant to the NNR standard [5] includes three stages for neural network compression: Preprocessing and parameter reduction, quantization, and entropy coding. These are depicted for the NNR encoder in Fig. 1. The first tool set contains optional preprocessing and parameter reduction tools. Instead of coding and compressing the original model, different methods allow to modify the model in a way that is beneficial for the succeeding coding and compression steps. In NNR, there are several parameter reduction tools, i.e., sparsification, unification, pruning, and low-rank decomposition [5]. Sparsification and unification aim at setting as many values as possible to zero or at reducing the number of unique values within a tensor, respectively. The methods for low-rank decomposition and pruning transform tensors or remove values, resulting in tensors of reduced size. In fact, these methods change the model's values or its architecture which may affect the inference or classification performance. Further preprocessing tools, namely batch-norm folding and local scaling adaption, remove redundancies in NN tensors and use scaling factors respectively in order to partly compensate quantization errors introduced by the quantization stage.

Generally, the values are quantized to integers in order to further compress the tensor and to output integer indices that can be handled by the succeeding entropy coding stage. For this, NNR specifies quantization tools which include the use of integer codebooks, uniform scalar quantization, and a form of vector quantization called dependent scalar quantization [6], also known as trellis-coded quantization [7]. For all tools a quantization step size is derived from an (integer) quantization parameter (QP) that provides a mechanism for controlling the rate-performance trade-off. In general, coarser quantization reduces the bitrate, however at the cost of a loss in inference performance. Therefore, a careful selection of QP values is necessary for a good neural network compression.

The integer indices output by the quantization stage are then encoded using a binary arithmetic coding scheme, called DeepCABAC [8]. This tool is based on the context-based adaptive binary arithmetic coding (CABAC) [9], which is part of several video coding standards [10]-[12]. Generally, arithmetic coders achieve compression close to the entropy of the source signal, if the statistics are known. DeepCABAC employs a set of probability estimators, called context models, which try to adapt to the source statistics [13]. This enables high compression for a large variety of different models without any prior knowledge of the statistics.

The respective NNR decoder, which is actually specified in the corresponding NNR standard [5], operates in reverse order, i.e., arithmetically decodes a received NNR bitstream, using DeepCABAC, applies tensor value reconstruction, and finally inverts preprocessing methods, if necessary.

3. NNR ENCODER OPTIMIZATION

The NNR standard enables the use of several tools in order to achieve a good rate-performance trade-off. Usually, this trade-off highly depends on the decisions in the encoder. In NNR, as well as for other media coding standards, the specific decision process is not in the scope of the standard, which gives a high degree of freedom to encoder development and tuning. In the following, three tools are presented that allow the encoder to significantly improve the rate-performance trade-off and that are presented together with methods to find optimized parameters and further improve coding efficiency.

3.1. Local scaling derivation

Local scaling adaptation (LSA) enables the encoder to partially compensate errors introduced by quantizing neural network weight tensors. Usually, the sensitivity of different neurons of deep neural networks to quantization with respect to the inference performance varies. Considering this, local scaling adaptation multiplies an additional scaling factor to each row vector of the weight tensor **W**, which is shaped as a 2-D matrix such that a row corresponds to an output neuron, as shown in eq. (1). More specifically, the scaling factors are given by a vector **s** and the weight tensor is represented by a 2-D matrix, such that each row contains the weights of the same neuron.

$$\widetilde{\mathbf{W}} \coloneqq \mathbf{s} \circ \mathbf{W} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} \circ \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \\ w_{2,1} & w_{2,2} & w_{2,3} \\ w_{3,1} & w_{3,2} & w_{3,3} \end{bmatrix} \tag{1}$$

Here, the operator $\circ$ denotes element-wise scaling of each row vector of a matrix with the corresponding element of a transposed vector.

The NNR standard only specifies the compression of local scales in **s**, while the optimal derivation and thus the specific values of these parameters is not part of the standard and up to the specific encoder implementation. Accordingly, we introduce a method for deriving **s** for improving the rate-performance trade-off. If an encoder has the ability to fine-tune the model, this can be done by fine-tuning the local scaling parameters after quantization of the weight tensors. For this, the scaling factors can be initialized with a value of 1 and then adapted, by using backpropagation, such that the inference performance of the model is increased. The resulting scaling factors are then quantized and encoded so that they can be derived at the decoder.

3.2. Batch-norm folding

Batch-normalization is a concept often used in deep neural networks in order to normalize the inputs so that they have zero mean and unit variance [14]. For this, the models employ so-called batch-norm layers with several parameters. Batch-norm folding (BN) aims at reducing redundancy of the parameters by exploiting interdependencies which are known by encoder and decoder. It assumes that the combination of a convolutional or a fully-connected layer with a batch-norm layer of the following form can be expressed as

$$\mathbf{BN}(\mathbf{X}) = \frac{\mathbf{W} * \mathbf{X} + \mathbf{b} - \boldsymbol{\mu}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} \circ \boldsymbol{\gamma} + \boldsymbol{\beta}, \tag{2}$$

where $\mathbf{BN}(\mathbf{X})$ is the output, $\mathbf{X}$ is the input, $\mathbf{W}$ is the weight tensor, $\mathbf{b}$ is a bias parameter, and the remaining parameters are batch-normalization parameters (see [14]). Note that $\mathbf{b}$, $\boldsymbol{\mu}$, $\boldsymbol{\sigma}^2$, $\boldsymbol{\gamma}$, and $\boldsymbol{\beta}$ have the same shape as $\mathbf{X}$ which is shaped as a transposed vector. Similar to the approach in sec. 3.1, $\mathbf{W}$ is represented as 2D matrix, such that each row corresponds to an output neuron. Parameter ϵ is a scalar close to zero.

Usually, each parameter is encoded individually. Without changing any of the parameters the bitrate can be reduced by combining several parameters. For this, the following transformation is applied prior to encoding:

$$\mathbf{BN}(\mathbf{X}) = \boldsymbol{\alpha} \circ \mathbf{W} * \mathbf{X} + \boldsymbol{\delta} \tag{3}$$

with $\boldsymbol{\alpha} = \frac{\boldsymbol{\gamma}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}}$ and $\boldsymbol{\delta} = \frac{(\mathbf{b} - \boldsymbol{\mu}) \circ \boldsymbol{\gamma}}{\sqrt{\boldsymbol{\sigma}^2 + \epsilon}} + \boldsymbol{\beta}$.

Consequently, it is now sufficient to encode the parameters $\mathbf{W}$, $\boldsymbol{\alpha}$, and $\boldsymbol{\delta}$ in order to obtain the same output. Furthermore, whenever batch-norm folding and local scaling adaptation (as described in sec. 3.1) are combined, the scaling parameter vector $\mathbf{s}$ can be merged with parameter $\boldsymbol{\alpha}$ to obtain a single vector $\tilde{\boldsymbol{\alpha}} \coloneqq \boldsymbol{\alpha} \circ \mathbf{s}$. In this case, the scaling factors practically do not affect the bitrate (except for a few bits induced by the changes of the values).

3.3. Inference-optimized quantization

In the NNR standard the quantization step size is controlled by a quantization parameter (QP) which is signaled in the bitstream. Furthermore, NNR allows to apply specific QP values to each tensor. In NNR, a QP value of 0 corresponds to a step size of 1 and increasing or decreasing the QP value by four corresponds to doubling or halving the step size, respectively. E.g., a QP value of -4 yields a step size of 0.5.

Selection of the QP values at the encoder is again outside the scope of the standard, however, is a key parameter for controlling the rate-performance trade-off. Proper selection of the quantization parameter is a challenging task due to the complex dependencies between the tensors. In this specific trade-off, a coarser quantization decreases the bitrate in general, however coarse quantization of individual NN tensors can cause a significant decrease in the overall inference performance of the entire network. Thus, tensors can have very different sensitivities towards quantization, but deriving the impact of the quantization error of a single tensor on the performance seems not to be feasible without evaluation of the whole model. For targeting the quantization optimization, we started with the following two (empirical) observations: First, quantizing large weight tensors highly affects both the bitrate and performance, whereas quantizing small tensors may significantly affect the performance, while only minimally influencing the bitrate. And second, tensors with a high standard deviation stronger degrade the network performance, when coarser quantized.

Starting from this, we introduce inference-optimized quantization (IOQ) as a method to select optimized QP values considering the model's performance on a parameter tuning set (which is different from the verification dataset), as well as its bitrate. For this, an algorithm is applied that iteratively quantizes a tensor and also encodes and evaluates the model.

In a first step a QP value, denoted as $baseQP$, is selected and the QP values (QP_i) for each weight tensor $\mathbf{W}_i$ are assigned based on the tensor's statistics according to:

$$QP_i = \begin{cases} rnd\big(baseQP \cdot (1-\eta_i)\big), & \text{if } \eta_i \leq 0.5 \\ rnd(baseQP \cdot 0.85), & \text{if } \eta_i > 0.5 \end{cases} \tag{4}$$

where $\eta_i = \varepsilon_i + \vartheta_i$ and:

$$\varepsilon_i = \frac{\#\mathbf{W}_i}{\sum_i(\#\mathbf{W}_i)} \tag{5}$$

$$\vartheta_i = \left(1 - \frac{std(\mathbf{W}_i)}{max_{\forall i}\big(std(\mathbf{W_i})\big)}\right) \tag{6}$$

The operators $\#$, $rnd(\cdot)$ and $std(\cdot)$, denote number of elements, rounding to the next integer and standard deviation, respectively. With this approach, the above two observations are exploited, such that a finer quantization is applied for tensors which have a small size and high standard deviation, i.e., where both ε_i and ϑ_i are small. This first step already provides a better coding gain compared to simply assigning the same QP value to all network tensors.

In the next step, a local QP optimization is applied as follows: First, all tensors are sorted by the number of elements in a descending order which is also the processing order for quantization of the tensors. Thus, the weight tensors which have the largest impact on the bitrate, are quantized first. This enables good control of the accuracy measure while quantizing them as coarse as possible in order to achieve low bitrates. Then, starting with the second tensor in the list, for each tensor QP offsets in the range from -4 to 4 are tested. Note that, the first tensor $\mathbf{W}_0$ always applies the value QP_0 computed in the previous step. This avoids obtaining identical working points for different $baseQPs$.

For a current tensor $\mathbf{W}_i$ to be processed, the QP offset is added to the QP value (QP_i) and the whole model is quantized, encoded and evaluated by inference on the parameter tuning set. The QP values of the other tensors remain unchanged. Then, a QP offset ΔQP that minimizes a Lagrangian cost function $D + \lambda R$ is added to the QP value and this new value is assigned to the current tensor. Here, D is the change of performance measure (negative for an increase), R is the change of bitrate and λ is the Lagrange multiplier which depends on the QP value and can be derived from the rate-performance curve.

The procedure is repeated for each tensor in the sorted list. In that way, the method yields optimized QP values which significantly improve the rate-performance trade-off.

4. EXPERIMENTAL RESULTS

The compression performance for the tools presented in this paper is evaluated using the standard reference software NCTM (Neural network Compression Test Model version 6.0, [15]) on a verification dataset of different neural networks defined in [16][17]. An overview of the models can be found in [17] as well as corresponding use cases, performance measures, application data, and number of parameters. The dataset includes five models, three for image classification (VGG16, ResNet50, MobileNetV2), one for audio classification (DCase), and an image autoencoder (UC12B). The classification performance of the image- and audio classification models is measured as Top-1/ Top-5 accuracies, while for UC12B Peak Signal-to-Noise Ratio (PSNR) / Structural Similarity Index Measure (SSIM) is applied.

The baseline (BL) software configuration [16] uses dependent scalar quantization with a constant QP value for all weight tensors (and a smaller QP value for all non-weight tensors) and entropy coding with DeepCABAC. On top, individual coding performance of LSA, BN, and IOQ as well as the combined performance are evaluated. Coding results for the baseline configuration as well as the overall combination of LSA+BN+IOQ at working points with same classification quality are given in TABLE 1. Note that batch-norm folding has no effect on VGG16 and UC12B, since these models do not contain batch-norm layers.

TABLE 1 shows that an overall compression ratio c_r of less than 3% can be achieved without loss of classification performance for VGG16, reducing its original size of 550MB down to a compressed bitstream of 16MB. When comparing the compression ratios between the proposed method "All" and the baseline "BL", an overall compression gain between 13% for DCase and 46% for VGG16 can be achieved at nearly identical accuracies. Comparable compression methods, such as those mentioned in [8], are clearly outperformed.

TABLE 1
NNR Transparent Coding Results

Model	Method	c_r in %	Top-1 / Top-5 Acc. reconstr.	Top-1 / Top-5 Acc. original
VGG16	BL	5.50	70.55 / 89.61	70.93 / 89.85
	All	2.98	70.51 / 89.54	
	Result	-46%	-0.06 / -0.08 %	
ResNet50	BL	9.68	74.45 / 91.93	74.98 / 92.15
	All	6.54	74.42 / 91.80	
	Result	-32%	-0.04 / -0.14 %	
MobileNetV2	BL	19.18	71.15 / 90.06	71.47 / 90.27
	All	12.18	71.13 / 90.06	
	Result	-36%	-0.03 / 0.00 %	
DCase	BL	4.71	59.26 / 91.36	58.27 / 91.85
	ALL	4.12	58.15 / 92.35	
	Result	-13%	-1.87 / 1.08 %	
Model	Method	c_r in %	PSNR / SSIM reconstructed	PSNR / SSIM original
UC12B	BL	22.23	29.98 / 0.955	30.13 / 0.956
	All	17.34	29.98 / 0.954	
	Result	-22%	0.00 / -0.10 %	

Additional results for the presented methods from sec. 3, i.e. local scaling adaptation (LSA), batch-norm folding (BN), inference-optimized quantization (IOQ), all methods combined (LSA+BN+IOQ) and the baseline configuration as reference, are depicted in Fig. 2 (a) to (e), respectively. Each figure shows compression ratios c_r with respect to the performance measure (Top-1 accuracies for all classification models and PSNR for the image autoencoder). The results show that the presented methods achieve high compression even for near lossless performance measures and even higher compression by combining the methods, while significantly outperforming the baseline configuration.

5. CONCLUSIONS

In this paper, we presented specific NNR encoder optimization methods that all together achieved an increase in overall compression performance by up to 46%. In particular we applied the optimal derivation of local scaling parameters, a batch-norm folding, and a specific inference-optimized QP derivation. For assessing the impact on each presented method, we showed individual coding results for each tool as well as overall coding results, when combining all encoder optimization tools.

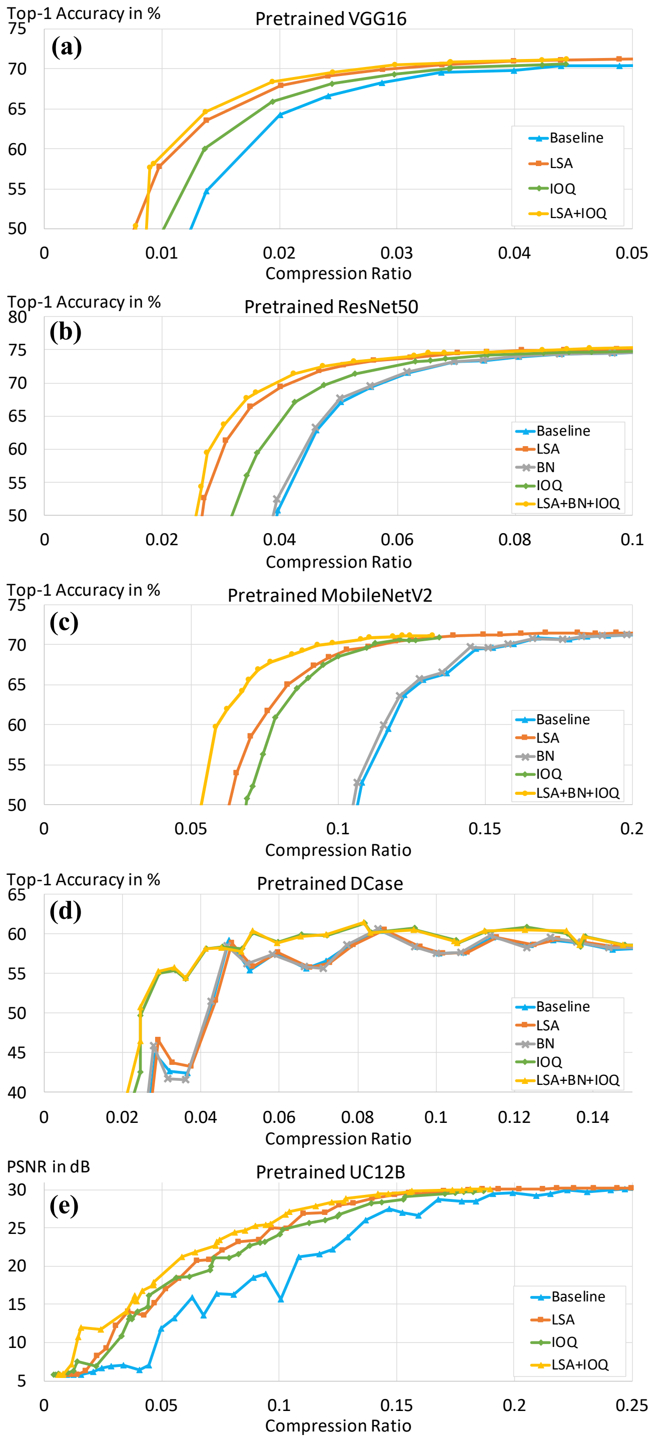

Fig. 2: Compression Ratio-Performance curves for (a) VGG16, (b) ResNet50, (c) MobileNetV2, (d) DCase showing Top-1 Accuracies and for (e) UC12B showing PSNR.

6. REFERENCES

[1] K. Ota, M. S. Dao, V. Mezaris, and F. G. B. D. Natale, “Deep learning for mobile multimedia: A survey”, *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 13, no. 3s, pp.34:1–34:22, 2017.

[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 Elec- tronics*, vol. 1, no. 4, pp. 216–222, 2018.

[3] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek “Robust and Communication-Efficient Federated Learning from Non-IID Data”, IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 9, pp. 3400-3413, September 2020, doi: 10.1109/TNNLS.2019.2944481.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-efficient learning of deep networks from decentralized data,” 2016, arXiv:1602.05629. [Online]. Available: http://arxiv.org/abs/1602.05629.

[5] „Text of ISO/IEC DIS 15938-17 Compression of Neural Networks for Multimedia Content Description and Analysis“, MPEG document N00016, ISO/IEC JTC1/ SC29/ WG 04, Oct. 2020

[6] P. Haase et al., "Dependent Scalar Quantization For Neural Network Compression," 2020 IEEE International Conference on Image Processing (ICIP), Abu Dhabi, United Arab Emirates, 2020, pp. 36-40, doi: 10.1109/ICIP40778.2020.9190955.

[7] M. W. Marcellin and T. R. Fischer, "Trellis coded quantization of memoryless and Gauss-Markov sources," in IEEE Transactions on Communications, vol. 38, no. 1, pp. 82-93, Jan. 1990, doi: 10.1109/26.46532.

[8] S. Wiedemann et al., "DeepCABAC: A Universal Compression Algorithm for Deep Neural Networks," in IEEE Journal of Selected Topics in Signal Processing, vol. 14, no. 4, pp. 700-714, May 2020, doi: 10.1109/JSTSP.2020.2969554.

[9] D. Marpe, H. Schwarz and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 620-636, July 2003, doi: 10.1109/TCSVT.2003.815173.

[10] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, "Overview of the H.264/AVC video coding standard," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003, doi: 10.1109/TCSVT.2003.815165.

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

[12] 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.

[13] P. Haase et al., "State-Based Multi-parameter Probability Estimation for Context-Based Adaptive Binary Arithmetic Coding," 2020 Data Compression Conference (DCC), Snowbird, UT, USA, 2020, pp. 163-172, doi: 10.1109/DCC47342.2020.00024.

[14] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, “Dive into Deep Learning,” Release 0.16.0, Jan. 2021, pp. 277-285, https://d2l.ai/d2l-en.pdf

[15] “Test Model 6 of Compression of Neural Networks for Multimedia Content Description and Analysis”, MPEG document N00017, ISO/IEC JTC 1/SC 29/ WG 04, Oct. 2020

[16] “Description of Core Experiments on Compression of neural networks for multimedia content description and analysis”, MPEG document N18574, ISO/IEC JTC 1/SC 29/WG 11, Jul. 2019.

[17] “Evaluation Framework for Compressed Representation of Neural Networks”, MPEG document N17929, ISO/IEC JTC 1/SC 29/WG 11, Oct. 2018