DeepCABAC: A Universal Compression Algorithm for Deep Neural Networks

Simon Wiedemann, Heiner Kirchhoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinč, David Neumann, Tung Nguyen, Heiko Schwarz, Thomas Wiegand, *Fellow, IEEE*, Detlev Marpe, *Fellow, IEEE*, and Wojciech Samek, *Member, IEEE*

***Abstract*—In the past decade deep neural networks (DNNs) have shown state-of-the-art performance on a wide range of complex machine learning tasks. Many of these results have been achieved while growing the size of DNNs, creating a demand for efficient compression and transmission of them. In this work we present DeepCABAC, a universal compression algorithm for DNNs that is based on applying Context-based Adaptive Binary Arithmetic Coder (CABAC) to the DNN parameters. CABAC was originally designed for the H.264/AVC video coding standard and became the state-of-the-art for the lossless compression part of video compression. DeepCABAC applies a novel quantization scheme that minimizes a rate-distortion function while simultaneously taking the impact of quantization to the DNN performance into account. Experimental results show that DeepCABAC consistently attains higher compression rates than previously proposed coding techniques for DNN compression. For instance, it is able to compress the VGG16 ImageNet model by x63.6 with no loss of accuracy, thus being able to represent the entire network with merely 9 MB. The source code for encoding and decoding can be found at https://github.com/fraunhoferhhi/DeepCABAC.**

***Index Terms*—Deep learning, neural network compression, efficient representation, source coding, rate-distortion quantization, arithmetic coding.**

I. Introduction

IT HAS been well established that deep neural networks excel at solving many complex machine learning tasks [1]. Their relatively recent success can be attributed to three phenomena: 1) access to large amounts of data, 2) researchers having designed novel optimization algorithms and model architectures that allow to train very deep neural networks, 3) the increasing availability of compute resources [1]. In particular, the latter two allowed machine learning practitioners to equip neural networks with an ever growing corpora of layers and, consequently, to consistently attain state-of-the-art results on a wide spectrum of complex machine learning tasks.

However, this has triggered an exponential growth in the number of parameters these models entail over the past years [2]. Trivially, this implies that the models are becoming more and more complex in terms of memory. This can become very problematic since it does not only imply higher memory requirements, but also slower runtimes and high energy consumption. In fact, IO operations can be up to three orders of magnitude more expensive than arithmetic operations in terms of energy consumption [3]. Moreover, [2] show that the memory-energy efficiency trend of most common hardware platforms are not able to keep up with the exponential growth of neural networks size, thus expecting them to be more and more power hungry over time.

In addition, there has also been an increasing demand on deploying deep models to resource constrained devices such as mobile or wearable devices [4], [5], and on distributed learning scenarios such as in federated learning [6]–[8]. These approaches have direct advantages with regards to privacy, latency and efficiency issues. However, high memory complexity greatly complicates the applicability of neural networks in those use cases, in particular in federated learning since the parameters of the networks are transmitted through bandwidth-limited communication channels.

Model compression is one possible paradigm to solve this problem. Namely, by attempting to maximally compress the information contained in the networks parameters we automatically leave only the bits that are necessary for solving the task. In addition, model compression can have direct practical advantages such as reduced communication and compute cost [9]–[11]. In fact, the Moving Picture Expert Group (MPEG) of the International Organization of Standards (ISO) has recently issued a call on neural network compression [12], which stresses the relevance of the problem and the broad interest by the industry to find practical solutions.

A. Entropy Coding in Video Compression

The topic of signal compression has been long studied and highly practical and efficient algorithms have been

Manuscript received July 1, 2019; revised November 19, 2019 and January 15, 2020; accepted January 16, 2020. Date of publication January 27, 2020; date of current version August 10, 2020. This work was supported in part by the German Ministry for Education through the Berlin Big Data Center under Grant 01IS14013A and in part by the Berlin Center for Machine Learning under Grant 01IS18037I. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Diana Marculescu. *(Corresponding author: Wojciech Samek.)*

The authors are with the Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany (e-mail: simon.wiedemann@hhi.fraunhofer.de; heiner.kirchhoffer@hhi.fraunhofer.de; stefan.matlage@hhi.fraunhofer.de; paul.haase@hhi.fraunhofer.de; arturo.marban@hhi-extern.fraunhofer.de; talmaj.marinc@hhi.fraunhofer.de; david.neumann@hhi.fraunhofer.de; tung.nguyen@hhi.fraunhofer.de; heiko.schwarz@hhi.fraunhofer.de; Thomas.Wiegand@hhi.fraunhofer.de; detlev.marpe@hhi.fraunhofer.de; wojciech.samek@hhi.fraunhofer.de).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/JSTSP.2020.2969554

designed. State-of-the-art video compression schemes like H.265/HEVC [13] employ efficient entropy coding techniques that can also be used for compressing deep neural networks. Namely, the context-based adaptive binary arithmetic coding (CABAC) scheme [14] provides a very flexible interface for entropy coding that can be adapted to a wide range of applications. It is optimized to allow high throughput and a high compression ratio at the same time. In particular, the transform coefficient coding part of H.265/HEVC contains many interesting aspects that might be suitable for compressing deep neural network. Hence, it appears only natural to try to apply current state-of-the-art compression techniques on deep neural networks and assess their compression gains.

B. Contributions

Our contributions can be summarized as follows:

1) We empirically identify a set of priors that are common across all studied neural networks.
2) We redefine CABAC's core scheme such that it captures those priors, thus, adapting it to the task of neural network compression. To the best of our knowledge, we are the first in applying state-of-the-art coding techniques from video compression to neural networks.
3) We quantize the parameters of the networks by minimizing a generalized form of a rate-distortion function which takes the impact of quantization on the accuracy of the network into account.
4) In our experiments we show that DeepCABAC is able to attain very high compression ratios and that it consistently attains a higher compression performance than previously proposed coders.

C. Outline

In Section II we start by reviewing some basic concepts from information theory, in particular from source coding theory. We also highlight the main difference between the classical source coding and the model compression paradigms in subsection II-D. Subsequently, we propose DeepCABAC in Section III. In Section IV we provide a comprehensive review of the related work on neural network compression. Finally, we provide experimental results and a respective discussion in Section V.

II. Source Coding

Source coding is a subfield of information theory that studies the properties of so called *codes*. These are mappings that assign a binary representation and a reconstruction value to a given input element. Figure 1 depicts their most common structure. They are comprised of two parts, an *encoder* and a *decoder*. The encoder is a mapping that assigns a binary string of finite length b to an input element w. In contrast, the decoder assigns a reconstruction value q to the corresponding binary representation. We will also sometimes refer to q as a *quantization point*. Furthermore, it is assumed that the output elements b and q of the code C are elements of finite countable sets, and that there is a one-to-one correspondence between them. Therefore, without loss of generality, we can decompose the encoder into a *quantizer* and a *binarizer*, where the former maps the input to an integer value $Q(w) = i \in \mathbb{Z}$, and the latter maps the integers to their corresponding binary representation $B(i) = b$. Analogously for the decoder. Naturally, it follows that the binarizer is always a bijective map, thus $(B^{-1} \circ B)(i) = i$.

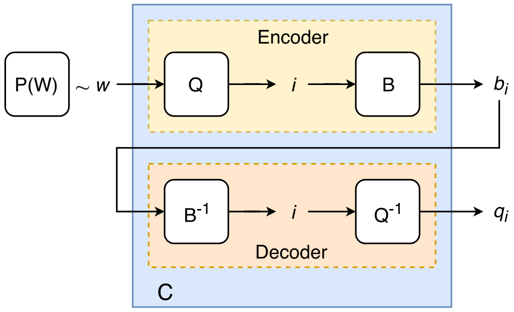


Fig. 1. The general structure of *codes*. Firstly, the encoder maps an input sample w from a probability source $P(w)$ to a binary representation b by a two-step process. It quantizes the input by mapping it to an integer $i = Q(w)$. Then, the integer is mapped to its corresponding binary representation $b = B(i)$ by applying a binarization process. The decoder functions analogously maps the binary representation back to its integer value by applying the inverse $B^{-1}(b) = i$ and assigns a reconstruction value (or quantization point) $Q^{-1}(i) = q$ to it. We stress that Q^{-1} does not have to be the inverse of Q.

We also distinguish between two types of codes, the so called *lossless codes* and *lossy codes*. They respectively correspond to the cases where Q is either bijective or not, thus, the latter implies that information is lost in the coding process. Therefore, we stress that the map Q^{-1} does **not** necessarily have to be the **inverse** of Q!

After establishing the basic definition of codes we will now formalize the source coding problem. In simple terms, source coding studies the problem of

> *finding the code that maximally compresses a set of input samples, while maintaining the error between the input and reconstruction values under an error tolerance constraint.*

Or more precisely: let $\mathbb{W} \subset \mathbb{R}^n$ be a given input set and let $P(w)$ be the probability of an element $w \in \mathbb{W}$ being sampled. Then, find a code C^* that

$$C^* = \arg\min_{C} \mathbb{E}_{P(w)} \left[D(w, q) + \lambda L_C(b) \right] \tag{1}$$

where $b = (B \circ Q)(w)$ and $q = (Q^{-1} \circ Q)(w)$. D is some distance measure and L_C is the length of the binary representation b. We will refer to $L_C(\cdot)$ as the *code-length* of a sample, and to D as the *distortion* between w and q. $\mathbb{E}_P[\cdot]$ denotes expectations as taken by the probability distribution P. $\lambda \in \mathbb{R}$ is the Lagrange multiplier that controls the trade-off between the compression strength and the error incurred by it.

Minimization objectives of the form (1) are called *rate-distortion objectives* in the source coding literature. However, solving the rate-distortion objective for a given input source is most often NP-hard, since it involves finding optimal quantizers Q, binarizers B and reconstruction values Q^{-1} from the space of all possible maps. However, concrete solutions can be found for special cases, in particular in the lossless case. In the following we will review some of the fundamental theorems of source coding theory and introduce state-of-the-art coding algorithms that produce binary representations with minimal redundancy.

A. Lossless Coding: Producing Binary Representations With Minimal Redundancy

Lossless coding implies that the quantizer Q is bijective and therefore $q = (Q^{-1} \circ Q)(w) = w \; \forall w$. Thus, $D(w, q) = 0 \; \forall w \in \mathbb{W}$ in (1) and the rate-distortion objective simplifies into finding a binarizer B^* that maximally compresses the input samples. Hence, throughout this section we will equate the general code C with the binarizer B and refer to it accordingly.

Information theory already makes concrete statements regarding the minimum information contained in a probability source. Namely, Shannon in its influential work [15] stated that the minimum information required to fully represent a sample w that has probability $P(w)$ is of $-\log_2 P(w)$ bits. Consequently, the entropy $H_P(\mathbb{W}) = \sum_{w \in \mathbb{W}} -P(w) \log_2 P(w)$ states the minimum average number of bits required to represent any element $w \in \mathbb{W} \subset \mathbb{R}^n$. This implies that

$$H_P(\mathbb{W}) \leq \bar{L}_C(\mathbb{W}), \quad \forall C \tag{2}$$

where $\bar{L}_C(\mathbb{W}) = \sum_{w \in \mathbb{W}} P(w) L_C(w)$ is the average code-length that any code C assigns to each element $w \in \mathbb{W}$. Eq. (2) is also referred as the *fundamental theorem of lossless coding*.

Fortunately, from the source coding literature [16] we know of the existence of codes that are able to reach the average code-length, up to only 1 bit of redundancy to the theoretical minimum. That is,

$$\exists C : H_P(\mathbb{W}) \leq \bar{L}_C(\mathbb{W}) < H_P(\mathbb{W}) + 1 \tag{3}$$

Moreover, we even know how to build them.

Before we start discussing in more detail some of these codes, we want to recall an important property of joint probability distributions. Namely, due to their sequential decomposition property, we can express the minimal information entailed in an output sample $w \in \mathbb{R}^n$ as

$$-\log_2 P(w) = -\sum_{j=0}^{n-1} \log_2 P(w_j | w_{j-1}, \ldots, w_0)$$

That is, we can always interpret a given input vector as an n-long random process and encode its outputs sequentially. As long as we know the respective conditional probability distributions, we can optimally encode the entire sequence. Respectively, we denote with w_j the scalar value of the j-th dimension of w (or equivalently j-th output of the random process). Also, we denote with $\mathbb{W}_s$ the set of possible scalar inputs, where $w_j \in \mathbb{W}_s, \forall j$.

1) (Scalar) Huffman Coding: One optimal code is the well known Huffman code [17]. However, Huffman codes can be very inefficient in practice since the Huffman-tree grows very quickly for large input dimensions n. Therefore, most often *scalar Huffman codes* are used instead. These codes consider 1-dimensional inputs only and are therefore suboptimal, i.e., they produce redundant binary representations. Concretely, they produce average code-lengths of

$$H_P(\mathbb{W}) \leq \bar{L}_{\text{SH}}(\mathbb{W}) < H_P(\mathbb{W}) + n$$

where $\bar{L}_{\text{SH}}(\cdot)$ is the average code-length produced by the scalar Huffman code on an entire input sequence $w \in \mathbb{W} \subset \mathbb{R}^n$. Hence, if n is large (e.g., in the order of hundreds of millions of values as in the case of deep neural networks parameters), then the resulting binary representation of the entire sequence of values may have a high number of redundant bits.

We provide a pseudocode of the encoding and decoding process of scalar Huffman codes in the appendix.

2) Arithmetic Coding: A concept that approaches the joint entropy $H(\mathbb{W})$ of eq. (3) in a practical and efficient manner is arithmetic coding. Concretely, an ideal arithmetic coder produces only up to two bits more than the minimum possible code length of an n-long random process. It therefore became the state-of-the-art for lossless coding and is widely applied across different compression algorithms. We would like to refer to the appendix for a more detailed description of the arithmetic coding method.

B. Universal Coding

In the previous subsection we learned that there exist codes that are able to produce binary representations of (almost) minimal redundancy (e.g. arithmetic codes). However, we implicitly assumed that the decoder knows the joint probability distribution of the input source. This is not the case in many real world scenarios. Hence, in such cases one usually relies on so called *universal codes*. They basically apply the following principle: 1) start with a general, prior probability model P_{Dec}, 2) update the model upon seeing data, 3) encode the input samples with regards to the updated probability model.

Thus, the theoretical minimum of universal codes are lower bounded by the decoder's probability estimate. Concretely, let P_{Dec} be the decoder's estimate of the inputs probability model, then the minimum average code-length that can be achieved is

$$\bar{L}_C(\mathbb{W}) \geq H_{P,P_{\text{Dec}}}(\mathbb{W}) = H_P(\mathbb{W}) + D_{KL}(P \,||\, P_{\text{Dec}})$$

with $H_{P,P_{\text{Dec}}}(\mathbb{W}) = -\sum_{w \in \mathbb{W}} P(w) \log_2 P_{\text{Dec}}(w)$ being the cross-entropy and D_{KL} the Kullback-Leibler divergence. Hence, a lossless code can only create binary representations with minimal redundancies if and only if its decoder‘s probability model matches the input source's. In other words, the better its estimate is, the more compact it can encode the input samples.

In general, a universal lossless code should have the following properties:

- **Universality:** The code should have a mechanism that allows it to adapt its probability model to a wide range of different types of input distributions, in a sample-efficient manner.
- **Minimal redundancy:** The code should produce binary representations of minimal redundancy with regards to its probability estimate.
- **High efficiency:** The code should have high coding efficiency, meaning that encoding/decoding should have high throughput.

1) CABAC: Context-based Adaptive Binary Arithmetic Coding is a form of universal lossless coding that fulfils all the above properties. It offers a high degree of adaptation,

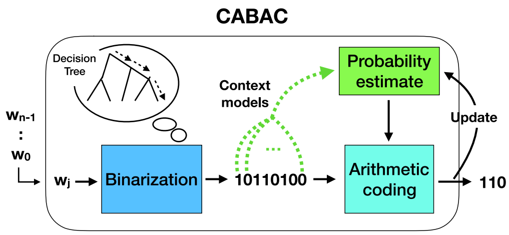


Fig. 2. Context-based Adaptive Binary Arithmetic Coding (CABAC) is a universal lossless codec that encodes an n-long sequence of 1-dimensional values by: 1) representing each unique value by a binary string that corresponds to traversing a particular path on a predefined decision tree, 2) assigning to each decision (or bin) a probability model (context model) and updating these upon encoding/decoding data, and 3) applying a binary arithmetic coder in order to encode/decode each bin.

optimal code-lengths, and a highly efficient implementation. It was originally designed for the video compression standard H.264/AVC [14], but it is also an integral part of its successor H.265/HEVC. It is well known to attain higher compression performance as well as higher throughput than many other entropy coding methods [18]. In short, it encodes each input sample by applying the following three stages:

1) **Binarization:** Firstly, it predefines a series of binary decisions (also called bins) under which each unique input sample element (or symbol) will be uniquely identified. Thus, it builds a predefined binary decision tree where each leaf identifies a unique input value.
2) **Context-modeling:** Then, it assigns a binary probability model to each bin (also named context model) which is updated on-the-fly by the local statistics of the data. This enables CABAC to model a high degree of different source distributions.
3) **Arithmetic coding:** Finally, it employs an arithmetic coder in order to optimally and efficiently code each bin, based on the respective context model.

Notice that, in contrast to Huffman codes, CABAC's encoder does **not need to encode its probability estimates**, since the decoder is able to analogously update its context models upon sequentially decoding the input samples. Codes that have this property are called *backward-adaptive* codes. Moreover, it is able to take **local correlations** into account, since the context models are updated in an autoregressive manner by the local statistics of the data.

C. Lossy Coding

In contrast to lossless coding, information is lost in the lossy coding process. This implies that the quantizer Q is non-invertible, and therefore $\exists w : D(w, q) \neq 0$. An example of a distortion measure may be the mean-squared error $D(w, q) = ||w - q||_2^2$, but we stress that other measures can be considered as well (which will become apparent in Section III).

The infimum of the rate-distortion objective (1) $\forall\lambda$ is referred to as the *rate-distortion function* in the source coding literature [16]. It represents the fundamental bound on the performance of lossy source coding algorithms. However, as we have already discussed above, finding the most optimal code that follows the rate-distortion function is most often NP-hard, and can be calculated only for very few types and/or special cases of input sources. Therefore, in practice, we usually relax the problem until we formalize an objective that can be solved in a feasible manner.

Firstly, we fix the binarization map B by selecting a particular (universal) lossless code and condition the minimization of (1) on it. That is, now we only ask for the quantizer Q, along with its reconstruction values Q^{-1}, that minimize the respective rate-distortion objective. Secondly, we will always assume that we encode an n-long 1-dimensional random process. Then, objective (1) simplifies to:

Given a lossless code (B, B^{-1}), find $(Q, Q^{-1})^*$ that

$$(Q, Q^{-1})^* = \underset{(Q,Q^{-1})}{\arg\min}\, \mathbb{E}_{P(w_j)}\left[D(w_j, q_j) + \lambda L_Q(b_j)\right], \quad (4)$$

$\forall j \in \{0, \ldots, n-1\}$, where $q_i \in \mathbb{Q}_s := \{q_0, q_1, \ldots, q_{K-1}\} \subset \mathbb{R}$ and $K = |\mathbb{Q}_s| < |\mathbb{W}_s| = n$.

For instance, if we choose B such that it assigns a binary representation of fixed-length to all w_j, then the minimizer of (4) can be found by applying the k-Means algorithm.

The minimizers of (4) are called *scalar quantizers*, since they measure the distortion independently for each input sample. In contrast, *vector quantizers* measure the distortion in the respective vector space by grouping a sequence of input samples together. It is well known that the infimum of scalar codes are fundamentally more redundant than vector quantizers, however, due to the associated complexity of vector quantizers it is more common to apply scalar quantizers in practice. Moreover, the inherent redundancy of scalar quantizers is negligible for most practical applications [16].

We stress that although the distortion in (4) is measured independently for each sample w_j, the binarization b_j (and consequently the respective code-length) can still depend on the other samples by taking correlations into account.

1) Scalar Lloyd Algorithm: An example of an algorithm that finds a local optimum is the Lloyd algorithm. It approximates the average code-length of the quantized samples $q_j = (Q^{-1} \circ Q)(w_j)$ with the entropy of their empirical probability mass distribution (EPMD). Thus, it substitutes the code-length in (4) by $L_C(b_j) = -\log_2 P_{\text{EPMD}}(q_j)$ and applies a greedy algorithm in order to find the most optimal quantizer Q and quantization points Q^{-1} that minimize the respective objective. A pseudocode can be seen in the appendix.

2) CABAC-Based RD-Quantization: If we are given a set of quantization points $\mathbb{Q}_s$ and select CABAC as our universal lossless code, then we can trivially minimize (4) by sequentially quantizing the input samples. In the video coding standards the set of quantization points are predefined by the particular choice of quantization strength λ [13]. However, in the context of neural network compression we do not know of a good relationship between the quantization strength and the set of quantization points. In the next Section III we describe how we tackled this problem.

D. Model Compression vs. Source Coding

So far we have reviewed some fundamental results of source coding theory. However, in this work, we are rather interested in the general topic of *model compression*. There is a fundamental difference between both paradigms. Namely, now we are more interested in the predictive performance of the resulting quantized model rather than the distance between the quantized and original parameters. We will formalize the general model compression paradigm for the supervised learning setting. However, the problem can be analogously formulated for other learning tasks.

Firstly, we assume that we are given only one model sample with n real-valued weight parameters (thus, here the data to be coded is equivalent to the ones discussed above). In addition, we assume a universal coding setting, where the decoder has no prior knowledge regarding the distribution of the parameter values. We argue that this simulates most real world scenarios.

Let now $(x, y) \in \mathbb{D}$ be a set of data samples related to the learning task. Let further $y' \sim P(y'|x, w)$ denote the approximate posterior of the data, parameterized by $w \in \mathbb{R}^n$. For instance, $P(y'|x, w)$ may be a trained neural network model with parameters w. Finally, let B be a chosen and fixed universal lossless code. Then, we aim to find a quantizer Q^* that minimizes

$$(Q, Q^{-1})^* = \underset{(Q,Q^{-1})}{\arg\min} \sum_{(x,y)\in\mathbb{D}} \mathcal{L}(y, y'') + \lambda L_Q(b) \qquad (5)$$

with $y'' \sim P(y''|x, q)$ being outputs of the quantized model $q = (Q^{-1} \circ Q)(w)$ and $b = (B \circ Q)(w)$.

The first term in (5) expresses the minimization of the usual learning task of interest, whereas the second term explicitly expresses the code-length of the model. This minimization objective is well motivated from the Minimum Description Principle (MDL) [19]. However, finding the minimum of (5) is also most often NP-hard. This motivates further approximations where, as a result, one can directly apply techniques from the source coding literature in order to minimize the desired objective.

1) Relaxation of the model compression problem into a source coding problem: We now may further assume that the given unquantized model has been pre-trained on the desired task and that it reaches satisfactory accuracies. Then, it is reasonable to replace the first term in (5) by the KL-Divergence between the unquantized model $P(y'|x, w)$ and the respective quantized model $P(y''|x, q)$. That is, now we aim to quantize our model such that its output distribution does not differ too much from its original version.

Furthermore, if we assume that the output distributions do not differ too much from each other, then we can approximate the KL-Divergence with the Fisher Information Matrix (FIM). Concretely,

$$\mathbb{E}_{P_{\mathbb{D}}}\left[D_{KL}(y'' \,||\, y')\right] = \delta w F \delta w^T + \mathcal{O}(\delta w^2) \qquad (6)$$

with $\delta w = q - w$ and

$$F := \mathbb{E}_{P_{\mathbb{D}}} \mathbb{E}_{P(y'|x,w)}\left[\partial_w \log P(y'|x, w)(\partial_w \log P(y'|x, w))^T\right]$$

Then, by substituting (6) in (5) we get the following minimization objective

$$(Q, Q^{-1})^* = \min_{(Q,Q^{-1})} (q - w)F(q - w)^T + \lambda L_Q(b) \qquad (7)$$

Objective (7) now follows the same paradigm as the usual source coding problem. However, with the peculiarity that now $D(w, q)$ (approximately) measures the distortion of w and q in the space of output distributions instead the euclidian space. The advantage of the rate-distortion objective (7) is that, after the FIM has been calculated, it can be solved by applying common techniques from the source coding literature, such as the scalar Lloyd algorithm.

However, minimizing (7) as well as estimating the FIM for deep neural networks usually requires considerable computational resources, and is most often infeasible in practical scenarios. Therefore, we usually consider only the diagonal elements of the FIM (FIM-diagonals), which can be efficiently estimated (see appendix). As a result, (7) simplifies to

$$(Q, Q^{-1})^* = \underset{(Q,Q^{-1})}{\arg\min}\ F_i(q_i - w_i)^2 + \lambda L_Q(b) \qquad (8)$$

$\forall i \in \{0, \ldots, n-1\}$, which can be feasibly solved.

In the next section we will give a thorough description of our proposed coder. Its design complies with all desired properties that a coder for neural network compression should have.

III. DeepCABAC

In light of the discussion introduced in the previous section, we can highlight a set of desiderata that a coder for neural network compression should have.

- **Minimal redundancy**: State-of-the-art deep neural networks usually contain millions of parameters. Thus any type of redundancy in the weight parameters may imply several additional MB being stored. Hence, the code should output a binary representation with minimal redundancy per weight element.
- **Universality:** The code should be applicable to any type of incoming neural networks, without having to know their distribution a priori. Hence, the code should entail a mechanism that allows it to adapt to a rich family of possible parameter distributions.
- **High coding efficiency:** The computational complexity of encoding/decoding should be minimal. In particular, the throughput of the decoder should be very high if performing inference on the compressed representation is desired.
- **Configurable error vs. compression strength:** The coder should have a hyperparameter that controls the trade-off between the compression strength and the incurred prediction error.
- **High data efficiency:** Minimizing (5) implies access to data. Hence, it is desirable that the coder finds a (local) solution with the least amount of data samples possible.

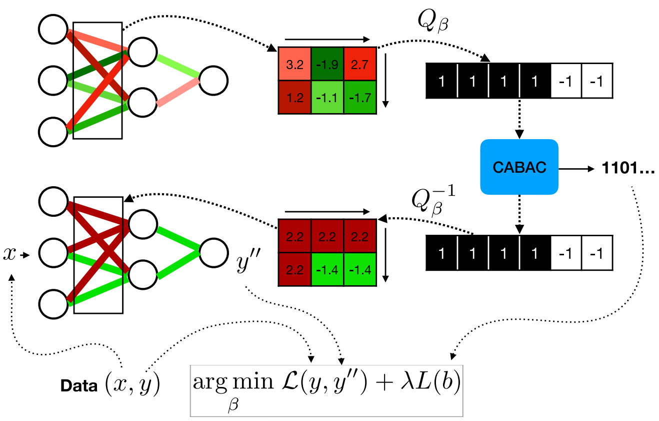


Fig. 3. Sketch of the DeepCABAC compression procedure. Firstly, DeepCABAC scans the weight parameters of each layer of the network in row-major order. Then, it selects a particular hyperparameter β that will define the set of quantization points. Subsequently, it applies a quantizer on to the weight values that minimizes the respective weighted rate-distortion function (9). Then, it compresses the quantized parameters by applying our adapted version of CABAC. Finally, it reconstructs the network and measures the respective accuracy of it. The process is repeated for different hyperparameters β until the desired trade-off between accuracy and size of the network is achieved.

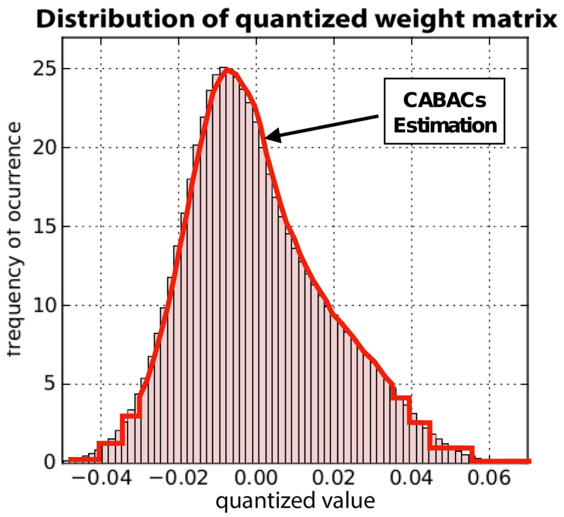


Fig. 4. Distribution of the weight matrix of the last layer of VGG16 (trained on ImageNet) after uniform quantization over the range of values. In red is CABAC's (possible) estimation of the distribution. The first $n + 2$ bits allow to adapt to any type of shape around 0 since they are encoded with regards to a context model. The remainder can only approximate the shape by a step-like distribution, since they are encoded with an Exponential-Golomb where the fixed-length parts are encoded without a context model.

A. DeepCABAC's Coding Procedure

We propose a coding algorithm that satisfies all the above properties. We named it *DeepCABAC*, since its based on applying CABAC on the networks quantized weight parameters. Figure 3 shows the respective compression scheme. It performs the following steps:

1) It extracts the weight parameters of the neural networks layer-by-layer in row-major order.[1]
2) Then, it selects a particular value β which defines the set of quantization points.
3) Subsequently, it quantizes the weight values by minimizing a weighted rate-distortion function of the form (8), which implicitly takes the impact of quantization on the accuracy of the network into account.
4) Then, it compresses them by applying our adapted version of CABAC.
5) Finally, it reconstructs the network and evaluates the prediction performance of the quantized model.
6) The process is repeated for a set of hyperparameters β, until the desired accuracy-vs.-size trade-off is achieved.

This approach has several advantageous properties. Firstly, it applies CABAC to the quantized parameters and therefore we ensure that the code satisfies the desiderata 1–3. Secondly, by conducting the compression for a set of hyperparameters for the quantizer we can select the desired pareto-optimal solutions of the accuracy vs. bit-size plane, thus satisfying property 4. Finally, since only evaluation of the model is required in the process, a significantly lower amount of data samples are required for the compression process than usually employed for training.

In the following we will explain in more detail the different components of DeepCABAC. The source code for encoding and decoding can be found at https://github.com/fraunhoferhhi/DeepCABAC.

B. Lossless Coder of DeepCABAC

Consider the weight distribution of the last fully-connected layer of the trained VGG16 model displayed in Figure 4. As we can see, there is **one peak near 0** and the distribution is **asymmetric** and **monotonically decreasing** on both sides. In our experience, all layers we have studied so far have weight distributions with similar properties. Hence, in order to accommodate to this type of distributions, we adopted the following binarization procedure.

Given a quantized weight tensor in its matrix form,[2] DeepCABAC scans the weight elements in row-major order and binarizes them as follows:

			Exp.-Golomb	
SigFlag	SignFlag	AbsGr(n)Flags	Unary	FL

1) The first bit (or bin), *SigFlag*, determines if the weight element is a significant element or not. That is, it indicates if the weight value is 0 or not. This bin is then encoded using a binary arithmetic coder, according to its respective context model (color-coded in grey). The context model is initially set to 0.5 (thus, 50% probability that a weight element is 0 or not), but will automatically be adapted to the local statistics of the weight parameters as DeepCABAC encodes more elements.
2) Then, if the element is not 0, the sign bin or *SignFlag* is analogously encoded, according to its respective context model.

[1] Thus, it assumes a matrix form where the parameters are scanned from left-to-right, top-to-bottom.

[2] For fully-connected layers this is trivial. For convolutional layers we converted them into their respective matrix form according to [20]

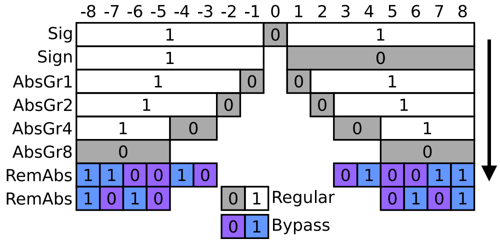


Fig. 5. DeepCABAC binarization of neural networks. It encodes each weight element by performing the following steps: 1) encodes a bit (or bin) named *SigFlag* which determines if the weight is a significant element or not (if its 0 or not). 2) If its not 0, then the sign bin, *SignFlag*, is encoded. 3) Subsequently, a series of bins are encoded, which indicate if the weight value is greater equal than $1, 2, \ldots, n \in \mathbb{N}$ (the so called *AbsGr(n)Flag*). 4) Finally, the remainder is encoded. The gray bins (also named regular bins) represent bits that are encoded using an arithmetic coder according to a context model. The other bins, the so called bypass bins, are encoded in fixed-point form. For instance, in the above diagram $n = 1$, and thus $1 \rightarrow 100$, $-4 \rightarrow 111101$ or $7 \rightarrow 10111010$.

3) Subsequently, a series of bins are analogously encoded, which determine if the element is greater than $1, 2, \ldots, n \in \mathbb{N}$ (hence *AbsGr(n)Flags*). The number n becomes a hyperparameter for the encoder.
4) Finally, the remainder is encoded using an Exponential-Golomb[3] code [21], where each bin of the unary part are also encoded relative to their context-models. Only the fixed-length part of the code are not encoded using a context-model (color-coded in blue).

For instance, assume that $n = 1$, then the integer -4 would be represented as 111101, or the 7 as 10111010. Figure 5 depicts an example scheme of the binarization procedure.

The first three parts of the binarization scheme empower CABAC to adapt its probability estimates to any shape distribution around the value 0 and, therefore, to encode the most frequent values with minimal redundancy. For the remainder values, we opted for the Exponential-Golomb code since it automatically assigns smaller code-lengths to smaller integer values. However, in order to further enhance its adaptability, we also encode its unary part with the help of context models. We left the fixed-length part of the Golomb code without context models, meaning that we approximate the distribution of those values by a uniform distribution (see Fig. 4). We argue that this is reasonable since usually the distribution of large numbers become more and more flat and it comes with the direct benefit of increasing the efficiency of the coder.

C. Lossy Coder of DeepCABAC

After establishing CABAC as our choice of universal lossless code, now we aim to find the optimal quantizer that minimizes the objective stated in (5) (Section II-D). To recall, this involves the optimization of two components:

- **Assignment:** finding the quantizer Q that optimally assigns the quantization point (or cluster centre) to each weight parameter,
- **Quant. points:** finding the optimal quantization point values $q_j = (Q^{-1} \circ Q)(w_j)$.

Since neural networks usually rely on scalable, gradient-based minimization techniques in order to optimize their loss function, finding the quantizers that solve (5) becomes infeasible in most cases since Q is a non-differentiable map. Therefore, we opted for a simpler approach.

Firstly, we decouple the assignment map Q and the quantization points Q^{-1} from each other and optimize them independently. The quantization points then become hyperparameters for the quantizer, and their values are selected such that they minimize the loss function directly. This separation between Q and Q^{-1} was empirically motivated, since we discovered that the networks performance is significantly more sensitive to the choice of Q^{-1} than to the assignment Q. We discuss this in more detail in the experimental section.

1) The Quantization Points: Since finding the correct map Q^{-1} for a large number of points can be very complex, we constrain them to be equidistant to each other with a specific step-size Δ. That is, each point q_k can be rewritten as to be $q_k = \Delta I_k$ with $I_k \in \mathbb{Z}$. This does not only considerably simplify the problem, but it does also encourage fixed-point representations which can be exploited in order to perform inference with lower complexity.[4]

2) The Assignment: Hence, the quantizer has two configurable hyperparameters $\beta = (\Delta, \lambda)$, the former defining the set of quantization points and the latter the quantization strength. Once a particular tuple is given, the quantizer Q_β will then assign each weight parameter w_i to its corresponding quantization point q_k by minimizing the weighted rate-distortion function

$$Q_\beta(w_i) = k^* = \arg\min_k \; F_i(w_i - q_k)^2 + \lambda L_{ik} \tag{9}$$

$\forall i \in \{0, \ldots, n-1\}$, where L_{ik} is the code-length of the quantization point q_k at the weight w_i as estimated by CABAC.

As previously mentioned, we perform a grid-search algorithm over the hyperparameters Δ and λ in order to find the quantizer configuration that achieves the desired accuracy vs. bit-size trade-off. However, for that we need to define a predefined set of candidate hyperparameters to look for. In this work we considered two approaches for finding the set of step-sizes, which we denote as *DeepCABAC-version1* (DC-v1) and *DeepCABAC-version2* (DC-v2).

3) DeepCABAC-Version1 (DC-v1): In DC-v1 we firstly estimate the diagonals of the FIM by applying scalable bayesian techniques. Concretely, we parametrize the network with a fully-factorized gaussian posterior and minimize the variational objective proposed in [22]. As a result, we obtain a mean μ_j and a standard deviation σ_j for each parameter, where the former can be interpreted as its (new) value (thus $w_i \rightarrow \mu_i$) and the latter as a measure of their "robustness" against perturbations. Therefore, we simply replaced $F_i = 1/\sigma_i^2$ in (9). This is also well motivated theoretically since [23] showed that the variance of the parameters approximate the diagonals of the FIM for a similar variational objective. We also provide a more

[3] To recall, the Exponential-Golomb code encodes a positive integer $2^k < i \leq 2^{k+1}$ by firstly encoding the exponent k using an unary code and subsequently the remainder $r = i - 2^k$ in fixed-point representation.

[4] [Online]. Available: https://www.tensorflow.org/lite

thorough discussion and a precise connection between them in the appendix.

After the FIM-diagonals have been estimated, we define the set of considered step-sizes as follows:

$$q_k = \Delta I_k, \quad \Delta = \frac{2|w_{\max}|}{\frac{2|w_{\max}|}{\sigma_{\min}} + S}, \quad S, I_k \in \mathbb{Z} \tag{10}$$

where $\sigma_{\min}$ is the smallest standard deviation and $w_{\max}$ the parameter with highest magnitude value. S is then the quantizers hyperparameter, which controls the "coarseness" of the quantization points. By selecting Δ in such a manner we ensure that the quantization points lie within the range of the standard deviation of each weight parameter, in particular for values $S \geq 0$. Hence, we selected S to be $S \in \{0, 1, \ldots, 256\}$.

One advantage of this approach is that we can have one global hyperparameter S for the entire network, but each layer will automatically attain a different value for its step-size if we select one $\sigma_{\min}$ per layer. Thus, with this approach we can adapt the step-size to the layers sensitivity to perturbations. Moreover, the quantization will also take the sensitivity of each single parameter into account.

4) DeepCABAC-Version2 (DC-v2): Estimating the diagonals of the FIM can still be computationally expensive since it requires the application of the backpropagation algorithm for several iterations in order to minimize the variational objective. Moreover, it only offers an approximation of the robustness of each parameter, and can therefore sometimes be misleading and limit the potential compression gains that can be achieved. Therefore, due to simplicity and complexity reasons, we also considered to directly try to find a good set of candidates $\Delta \in \{\Delta_0, \ldots, \Delta_{m-1}\}$ for the entire network. We do so by applying a first round of the grid-search algorithm while applying a nearest-neighbor quantization scheme (that is, for $\lambda = 0$). This allows us to identify the range of step-sizes that do not considerably harm the networks accuracy when applying the simplest quantization procedure. Then, we quantize the parameters as in eq. (9), but without the diagonals of the FIM (thus, $F_j = 1 \; \forall j$).

Under a limited computational budget, this approach has the advantage that we can directly search for a more optimal set of step-sizes Δ since we spare the computational complexity of having to estimate the FIM-diagonals. However, since DC-v2 considers only one global step-size for the entire network, it cannot adapt to the different sensitivities of the layers.

IV. Related Work

There has been a plethora of work focusing on the topic of neural network compression [9], [10]. When discussing about different compression techniques it is important to also consider their respective complexity, since different real-world scenarios may constrain their applicability. Again, we are not only interested in attaining high compression performance, but also in finding practical viable solutions. This becomes very relevant at the time of benchmarking these algorithms, since only methods that belong to the same category ought to be compared against each other. Following this line of thought, [24] proposed a four-level categorisation paradigm in which factors such as whether or not a method requires data, whether or not a method requires error backpropagation on the quantized model, and whether or not a method is generally applicable for any architecture (universality) are taken into account. In a similar spirit, we also considered low data as well as low coding complexity as desired properties in the previous Section III. Nevertheless, in the following we will classify different compression techniques based not only on their complexity, but also on important inherent properties that the methods share. Concretely, we classify methods into two main groups, lossy vs. lossless methods. Inside the lossy group we further distinguish between methods that require training vs. methods that do not, whereas in the lossless group we distinguish between methods that require decoding for performing inference vs. methods that do not. Notice, that the MPEG group on neural network compression classifies compression techniques in a similar manner [25], [26].

A. Lossy Neural Network Compression

1) Trained Scalar Quantization (Require Training): These are methods that aim to minimize the model compression objective (5) by applying training algorithms that learn the optimal quantization map and reconstruction values, most often in a simultaneous fashion. Inter alia, this includes sparsification methods [22], [27]–[29] which try to minimize the L_0-norm of the networks parameters. Others attempted to find optimal binary or ternary weighted networks [30], [31], or a more general set of (locally) optimal quantizers [32]–[34].

These methods are able to attain very high compression ratios since they are allowed to entirely change the underlying distribution of the weight parameters during the compression procedure. However, it comes at the cost of requiring access to a large training data set and high computational resources in order to apply the method.

2) Non-Trained Scalar Quantization Methods (do not Require Training): Another line of work have focused on implicitly minimizing (5). They also rely on distance measures for quantizing the networks parameters [35], [36]. In fact, these methods can be seen as special cases of (8), in that they either use different approximations of the FIM-diagonals or apply other minimization algorithms. To the best of our knowledge, mainly two quantizers are widely applied by the community, either the scalar uniform quantizers or the weighted scalar Lloyd algorithm. The former basically consist on spreading $K \in \mathbb{N}$ quantization points separated equidistantly over the range of parameter values, and then applying nearest-neighbor quantization [32], [33], [35]. The latter consists of applying the scalar Lloyd algorithm in order to find the most optimal quantizer that minimizes a weighted rate-distortion objective (8). In particular, [35] considers the diagonal elements of the empirical average of the Hessian of the loss function, which has a close connections to the FIM-diagonals (see appendix for a comprehensive discussion).

Applying quantization methods that do not rely on retraining during quantization are significantly **less computationally expensive** and do **not require access to training data** at the time of quantization.[5] Therefore, these methods can be applied

[5] Although sometimes they may require access to some amount of data in order to estimate the FIM-diagonals or the Hessian-diagonals.

to a much broader set of real-world scenarios, including cases where access to training data is, e.g., prohibited (for instance, in medical applications). But their compression gains are heavily limited by the networks underlying parameter distribution, since they rely on a distance measure for quantization. Moreover, as already mentioned, most of these methods do only implicitly take the impact on to the accuracy of the network into account, thus at least a one-time evaluation is required in order to assess the resulting prediction performance of the model after quantization.

B. Lossless Neural Network Compression

In the field of lossless network compression we are given an already quantized model and we want to apply a universal lossless code to its parameters in order to maximally compress it. In the following we describe some of the most popular lossless compression algorithms that are applied to the networks parameters.

1) Fixed-Length Numerical Representations and Compressed Matrix Formats (do not Require Decoding for Inference): One line of work try to reduce the bit-length representation of the parameter values after quantization [30], [31], [37]–[39]. They usually have the advantage of directly reducing the complexity for performing inference. However, it results in a highly redundant network representation, greatly limiting the attainable compression ratios.

Another line of work apply compressed matrix representations, e.g., *Compressed Sparse Row* (CSR) representation. These matrix data structures do not only offer higher compression gains, but also an efficient execution of the associated dot product algorithm. Similarly, [11] proposed two novel matrix representations, the *Compressed Entropy Row* (CER) and *Compressed Shared Elements Row* (CSER) representations, that are provably more optimal than the CSR with regards to both, compression and execution efficiency when the networks parameters have low entropy statistics.

However, these matrix representations are also redundant in that they do not approach the reachable entropy limit (3) (Section II-A). [32] attempted to extract some of the redundancies entailed in the CSR representations by applying a scalar Huffman code to its numerical arrays. However, this has again the same fundamental limitations that come by applying the scalar Huffman code.

2) Entropy Coders (Require Decoding for Inference): Even though decoding is necessary for inference, applying entropy coders to the networks parameters may still be beneficial in many real-world applications (e.g., in order to reduce parameter size on disk or in federated learning scenarios where parameters are send through a communication channel with limited capacity). Moreover, in inference complexity may still be reduced even if decoding is performed during run-time (see appendix for a more thorough discussion).

Although lossless compression of neural networks parameters is entirely equivalent to the usual lossless source coding setting discussed in Sections II-A and II-B, there has not been as much work studying the impact of applying state-of-the-art entropy coding techniques to the networks parameters. In fact, to the best of our knowledge, only the scalar Huffman code and the bzip2 entropy coder have been applied so far in the literature [35], [36]. However, as we have already discussed in Section II-A, these codes have several disadvantages compared to other state-of-the-art lossless codes such as arithmetic codes. Probably the most prominent one is that the scalar Huffman code is suboptimal in that it incurs up to 1 bit of redundancy per parameter being encoded. This can be quite significant for large networks with millions of parameters. For instance, VGG16 [40] contains 138 million parameters, meaning that the binary representation of any quantized version of it may have up to **17 MB** of redundancy if we encode it using the scalar Huffman code.

C. Compression Pipelines

Among all different proposed approaches for deep neural network compression there is one paradigm that stands out in that very high compression gain can be achieved with it [22], [32]–[34]. Namely, it consist on applying four different compression stages:

1) **Sparsification:** Firstly, the networks are maximally sparsified by applying a trained sparsification technique.
2) **Quantization:** Then, the non-zero elements are quantized by applying one of the non-trained quantization techniques.
3) **Fine-tuning:** Subsequently, the quantization points are fine-tuned in order to recover the accuracy loss incurred by the quantization procedure.
4) **Lossless compression:** Finally, the quantized values are encoded using a lossless coding algorithm.

Hence, DeepCABAC is designed to enhance points 2 and 4. As we will see in the next section, DeepCABAC is able to considerably boost the attainable compression gains, surpassing previously proposed methods for steps 2 and 4.

V. Experiments

In this section we benchmark DeepCABAC and compare it to other compression algorithms. We also perform further ablation studies, with the purpose to shed light into how different components involved in DeepCABAC impact the final compression performance.

A. General Compression Benchmark

Here we benchmark the end-to-end compression gains attained by applying DeepCABAC. In order to assess its universality, we applied it to a wide set of pretrained network architectures, trained on different data sets. Concretely, we used the VGG16, ResNet50 and MobileNet-v1 architectures, trained on the ImageNet dataset, a smaller version of the VGG16 architecture trained on the CIFAR10 dataset,[6] which we denote as *Small-VGG16*, and the LeNet-300-100 and LeNet5 trained on MNIST.

We compare the two versions of DeepCABAC, DC-v1 and DC-v2, against previously proposed compression schemes. Again, since DeepCABAC is based on quantization and entropy

[6] [Online]. Available: http://torch.ch/blog/2015/07/30/cifar.html

coding and does therefore not rely on retraining, it is important that we do only compare it against compression techniques that fall into the same category. These correspond to methods described in the related work Sections IV-A2 and IV-B2. That is, we benchmark DeepCABAC against the nearest-neighbor quantization scheme with uniformly separated quantization points (a.k.a. uniform quantization) and the weighted-Lloyd algorithm, as they are the two most widely applied quantization techniques. Furthermore, we apply the scalar Huffman code, the entropy coder proposed by [32] which we denote *CSR-Huffman*, and the bzip2 entropy coder, as they are as well some of the most commonly applied lossless compression techniques. See appendix for a more detailed explanation of the respective implementations.

Since the attainable compression gains are highly conditioned by the underlying distribution of the neural networks parameters, we also applied these methods to pre-sparsified versions of the above mentioned pretrained models. For that, we employed the variational sparsification algorithm [22] to all networks, except for the VGG16 and ResNet50 due to the high computational complexity demanded by the method. For the latter two, we applied the iterative-pruning approach proposed by [28]. The advantage of employing [22] is that we obtain an estimation of the FIM-diagonals as a byproduct of the methods output, thus being able to directly apply DC-v1 and the weighted Lloyd algorithm after the sparsification process finished. In the former cases (pretrained but non-sparse models), we estimated the FIM-diagonals by minimizing the same variational objective proposed in [22], however, while fixing the parameter values (thus without inducing any sparsity in the process). We would like to refer to the appendix for a more comprehensive explanation on the respective sparsification method as well as a thorough discussion on the estimation of the FIM-diagonals.[7]

Table I shows the results. It reports the attained compression ratios as calculated by

$$\mathrm{CR} = \frac{\sum_{i=0}^{p} C_s(W_i)}{32n}(\times 100\%)$$

where W_i is a particular parameter tensor of the network (e.g., 2 d convolutional weight tensor), $C_s(W_i)$ denotes its size in bits (e.g., if W_i is uncompressed then $C_s(W_i) = 32n_i$, where n_i is its total number of elements), p the total number of parameter tensors present in the network and n is the total number of parameter elements of the network. Thus, we calculate the ratio between the size of the compressed network (which involves the sum of the compressed and uncompressed parameter tensors) and the original neural network.

TABLE I

Compression Ratios Achieved Within an Accuracy Loss of ± 0.5% From the Original Accuracy When Applying Different Coding Methods. *DC-v1* and *DC-v2* Denote the Two Versions of Deepcabac, Whereas *Lloyd* Denotes the Weighted Lloyd Algorithm and *Uniform* the Nearest-Neighbor Quantization Scheme With Equidistant Quantization Points. For the Latter Two, We Report the Best Compression Results Attained After Applying Scalar Huffman, CSR-Huffman [32] and the bzip2 Lossless Coding Algorithms on to the Quantized Networks. In Parenthesis Are the Resulting Top-1 Accuracies and in Brackets the Sparsity Ratios Achieved as Measured by the Number of Non-Zero Parameters Divided by the Total Number of Parameters

Models [Spars. %]	Org. acc. top1 %	Org. size MB	DC-v1 (Acc. top1 %)	DC-v2 (Acc. top1 %)	Lloyd (Acc. top1 %)	Uniform (Acc. top1 %)
			PRETRAINED			
VGG16	69.94	553.43	5.84 (69.44)	**3.96** (69.54)	7.74 (69.50)	17.37 (69.90)
ResNet50	74.98	102.23	**10.14** (74.40)	**10.14** (74.51)	13.04 (74.74)	15.58 (74.64)
MobileNet -v1	70.69	17.02	**21.40** (70.21)	22.08 (70.21)	15.00[7] (68.10)	24.23 (70.10)
Small- VGG16	91.54	60.01	6.35 (91.11)	**5.88** (91.13)	9.98 (91.59)	16.18 (91.53)
LeNet5	99.46	1.722	3.77 (99.23)	**2.52** (99.12)	3.96 (98.96)	20.60 (99.45)
LeNet-300 -100	98.32	1.066	8.61 (98.04)	**5.87** (98.00)	8.07 (97.92)	15.01 (98.30)
			SPARSIFIED			
VGG16 [9.85]	69.43	553.43	**1.58** (69.43)	1.67 (69.04)	1.72 (69.01)	2.77 (69.42)
ResNet50 [25.40]	74.09	102.23	5.45 (73.73)	**5.14** (73.65)	5.61 (73.73)	6.68 (73.98)
MobileNet -v1 [50.73]	66.18	17.02	13.29 (66.01)	12.89 (66.02)	**11.16** (65.63)	14.78 (65.71)
Small- VGG16 [7.57]	91.35	60.01	**1.90** (91.03)	1.95 (91.06)	2.08 (91.10)	2.84 (91.20)
LeNet5 [1.90]	99.22	1.722	0.88 (99.14)	**0.87** (99.02)	1.09 (99.25)	3.01 (99.22)
LeNet-300 -100 [9.05]	98.29	1.066	2.26 (98.00)	2.20 (98.00)	**1.69** (97.76)	4.17 (98.36)

As one can see, DeepCABAC is able to attain higher compression gains across all networks (with the exception of the last model) as compared to the previously proposed coders. It is able to compress the pretrained by **x18.9** and the sparsified models by **x50.6** on average. In contrast, the Lloyd algorithm compresses the models by x13.6 and x47.3 on average, whereas uniform quantization only achieves x5.7 and x25.0 compression gains. In addition, notice how DeepCABAC attains higher compression ratios than state-of-the-art 8-bit quantization methods of MobileNet-v1 ($21.40\% < 25\%$). In contrast to [24], we attain similar accuracies at higher compression ratios *without applying any a posteriori optimization such as bias correction*. However, we stress that the later technique is orthogonal to DeepCABAC, and it can also be applied in order to further reduce the accuracy loss. Similarly, notice that [32] reports a compression ratio of 2.05% at a top-1 accuracy of 68.83% of the VGG16 model, whereas we were able to attain a compression ratio of 1.58% at an accuracy of 69.43%. To recall, Deep Compression consists on applying the sparsification technique [28], then the k-Means algorithm followed by the CSR-Huffman entropy coder and, finally, fine-tuning the cluster centers to the loss function. In contrast, we were able to attain higher compression performance at higher accuracies by simply applying [28] + DeepCABAC, *without having to perform any a posterirori fine-tuning of the quantization points*.

[7] Although a better compression ratio was attained, we were not able to get an accuracy in the ±0.5 percentage point range of the original accuracy. Therefore, this result shall not be considered as the best result.

TABLE II
Comparison of Attained Compression Ratios With Related Literature. VGG16 Results Are From [32], MobileNet-v1 From [24], MixNet From 1st Place Submission to Micronet-Challenge-19, BERT From [43] and YOLO-v3 From [41]. DeepCABAC Attains Similar (or Better) Results Without the Necessity of Applying Expensive Optimizations Such as Distillation, Bias Correction or Fine-Tuning. The Top-1 Accuracies, F1- and MAP-Scores Are Respectively in Parenthesis, and the Compression Ratios Otherwise. All Values Are Measured in Percentage

Model	Compression ratios	
	Related literature	DeepCABAC
VGG16 [553 MB]	2.05 (68.83)	**1.58** (69.43)
MobileNet-v1 [17.02 MB]	25.00 (70.50)	**21.40** (70.21)
MixNet [16.57 MB]	100.00 (75.03)	**7.9** (75.03)
BERT [433 MB]	11.00 (89.00)	**9.00** (86.00)
YOLO-v3 [248 MB]	100.00 (55.30)	**8.65** (53.45)

TABLE III
Average Bit-Sizes Per Parameter for the Small-VGG16 Network After Applying Different Quantizers. *DC-v1* and *DC-v2* Denote the Two Versions of DeepCABAC, Whereas *Lloyd* Denotes the Weighted Lloyd Algorithm and *Uniform* Corresponds to the Nearest-Neighbor Quantization. We Chose the Networks That Resided Within the ± 0.1 Percentage Point Range From the Accuracy Attained After Applying a Uniform Quantizer. In the Case of the Lloyd and Uniform Quantizers, the Size of the Quantized Networks Were Measured With Regards to the Entropy of Their Empirical Probability Mass Distribution. In Contrast, We Measured the Explicit Average Bit-Size Per Parameter in DC-v1 and DC-v2

step-sizes (top1 acc.)	DC-v1	DC-v2	Lloyd	Uniform
		PRETRAINED		
0.032 (90.35)	**1.48**	**1.48**	1.79	1.60
0.016 (91.13)	2.21	**2.20**	2.29	2.40
0.001 (91.55)	4.27	4.80	**2.34**	5.61
		SPARSIFIED [7.57%]		
0.032 (90.22)	**0.47**	**0.47**	0.52	0.48
0.016 (91.06)	0.59	**0.58**	0.62	0.60
0.001 (91.17)	0.91	1.00	**0.74**	1.00

Furthermore, we also applied DeepCABAC to YOLO-v3 [41] and BERT [42] which are state-of-the-art models for object detection and natural language processing tasks respectively (see Table II and the supplement for an extensive report of the results). In short, these models can be compressed down to 10% of their original size while negligibly affecting their prediction performance. In particular, we were able to attain a compression ratio of 9% at a F1-score of 86% on the BERT model, which is competitive with the reported results from the literature [43] (compression ratio of 11% at a F1-score of 89%). However, we remark that [43] attained these results after applying expensive retraining/fine-tuning procedures, plus a compression technique known as distilling [44] which modifies the network architecture. In contrast, DeepCABAC achieved the above results by simply applying quantization plus entropy coding techniques.

Finally, we also compressed a sparsified version of MixNet[8] [45], which is a state-of-the-art model for ImageNet classification designed for efficient use on embedded devices. We were able to compress the network by x12.7 from its original size, attaining as such a network that achieves **75.03%** top-1 accuracy and needs only **1.31 MB** of storage requirements.

We summarise all above results and their comparison with the related literature in Table II. They corroborate that DeepCABAC serves as a powerful universal quantizer + entropy coder for neural networks.

B. Ablation Study: Assignment vs. Quantization Points

To recall, lossy quantization involves two types of mappings, the quantization map Q where parameter values are assigned to quantization points (or equivalently integers), and the reconstruction map Q^{-1} which assigns a value to each quantization point. In DeepCABAC the former is influenced by the Lagrangian multiplier λ which controls the trade-off between the bit-size and the distortion incurred by the quantization, and the latter by the quantization step-size Δ which controls the "coarseness" of the quantization points. Both hyperparameters influence the resulting accuracy and compression ratio of the network. Hence, the following experiment aims to assess their respective impact.

Table III shows the average bitsizes attained after applying different quantizers to the Small-VGG16 network at different step-sizes but fixed accuracies (within a range of $\pm 0.1\%$ from each other). Moreover, the average bitsizes that result from applying the Lloyd and the uniform quantizers are measured in terms of first-order entropy values. It corresponds to the entropy of the empirical probability mass distribution, which marks the theoretical compression limit for all entropy coders that were considered in conjunction with these quantizers (e.g., scalar Huffman code).[9] Thus, the difference in compression performance at a fixed step-size do now only reflect the difference in "assignment-decisions" made by the respective quantizer. In other words, each row in Table III assess the clustering performance of each quantizer.

There are two main insights we attain from Table II, namely in the regime of strong quantizations (or big step-sizes):

1) The distortion measures do not seem to reflect the actual accuracy loss that will result from quantization.
2) The simple, nearest-neighbour quantization scheme seems to make (almost) as good assignment decisions as rate-distortion based quantizers.

[8] [Online]. Available: https://github.com/wps712/MicroNetChallenge

[9] Other entropy coders that take correlations between the parameters into account, such as CABAC, may attain lower values (as can be seen in Tables II and III).

The first insight may be due to the approximation nature of the distortion measures. Namely, since they do only represent local approximations to the incurred accuracy, they may very well be inaccurate at strong perturbations. Indeed, from Table III we can also see that as the step-sizes become smaller, rate-distortion based quantizers make better assignment decisions than the uniform quantizers.[10] In particular, we see that DeepCABAC-v1 outperforms DeepCABAC-v2 at smallest step-size, which indicates that the Fisher-weighted rate-distortion cost function serves as a better approximation of the actual MDL-loss function than the non-weighted rate-distortion cost function. Therefore, in the high rate regime (low step-sizes), we can see that the lossy and lossless components are closely coupled together.

The second insight may be considered as a more intriguing result. Namely, we see that in the strong quantization regime (high step-size values) the entropy of the uniform quantization scheme is *lower* than of the Lloyd quantizer. Moreover, it comes closer to DeepCABAC's compression performance as we increase the step-size. Thus, the simple nearest-neighbour quantization scheme seems to be a surprisingly good quantizer in terms of compression-vs.-accuracy performance for large quantization step-sizes. Notice, that [35] also reports a similar phenomena. This may be due to rate-distortion quantizers inducing a biased error that affects stronger the accuracy of the network. However, it is not entirely trivial why this is the case, and as of today it still remains an open question.

In conclusion, it seems that the compression performance is more strongly affected by the particular choice of the quantization step-size Δ. These insights motivated the design of DC-v2 in the first place, since it is able to explore a larger set of step-sizes for the best accuracy vs. bit-size trade-offs. Indeed, as Table I from the previous experiment shows, DC-v2 attains similar or even higher compression gains than DC-v1, in particular in the case of pretrained networks.

C. Ablation Study: Lossless Coding

In our last experiment we aimed to assess the efficiency of the considered universal lossless coders. To recall, there are two essential components that influence its compression performance of a universal entropy coder:

1) A probability estimate of the data to be coded
2) A mapper that assigns a binary string to the data with minimal redundancy.

For this experiment we quantized the Small-VGG network using three different quantizers, and subsequently compressed each of them using different universal lossless coders. More concretely, we quantized the model by applying DC-v2, the weighted Lloyd algorithm and the nearest-neighbor quantizer. We then applied the scalar Huffman code, the CSR-Huffman code [32], the bzip2 algorithm, and the CABAC-component of DeepCABAC. Moreover, we also calculated the first-order entropy of the quantized networks, which measures the entropy of the empirical probability mass distribution (EPMD). The resulting average bit-sizes are reported in Table IV.

TABLE IV

Compression Ratios Achieved From Lossless Compressing Different Quantized Versions of the Small-VGG16 Network (and Its Sparse Version). The Network Was Quantized in Three Different Manners, One by Applying DC-v2, Another With the Weighted Lloyd Algorithm, and Finally With the Uniform Quantization (Nearest-Neighbor Quantization). The Top1 Accuracy of Each Quantized Model Lies Within the ± 0.1 Percentage Point Range From the Original Accuracy of the Model, Which Is 91.54% and 91.35% Respectively. Subsequently, Each of Them Was Compressed by Applying the Scalar Huffman Code, the CSR-Huffman Code [32], the bzip2 Coder, and by CABAC. The Second Last Row Denotes the Entropy of the EPMD

Quantizers→ Lossless codes ↓	Uniform	Lloyd	DC-v2
PRETRAINED			
scalar-Huffman	5.18	3.19	2.33
bzip2	5.22	3.22	2.42
CABAC	**4.77**	**2.74**	**2.07**
H	5.09	2.91	2.20
SPARSIFIED [7.57%]			
scalar-Huffman	1.35	1.71	1.33
CSR-Huffman	0.91	0.67	0.65
bzip2	0.73	0.72	0.71
CABAC	**0.63**	**0.63**	**0.61**
H	0.84	0.60	0.58

As one can see, CABAC is able to attain higher compression gains across all quantized versions of the Small-VGG16 network. The benefits from using CABAC come from its inherent flexibility in that it can be accommodated to capture the prior statistics of the weight parameters. Namely, by defining the binarization procedure as in Section III-B, DeepCABAC is able to quickly capture the statistics of unimodal and asymmetric distributions, with their maxima close to 0. In addition, we decided for a row-major scanning order so that CABAC is able to capture correlations between elements in a row (which are indeed present as demonstrated in [24]). This is also crucial since CABAC's estimate is updated in an autoregressive manner and therefore, its compression performance also depends on the scanning order. Indeed, as Table IV shows, CABAC is able to capture correlations between the weight parameters and consequently compress them beyond the first-order entropy of the parameter distribution. This particular property highlights its superiority as compared to the previously proposed universal entropy coders, e.g., scalar Huffman and CSR-Huffman, since their average code-lengths are bounded by the first-order entropy and therefore it would be **impossible for them to attain lower code-lengths than CABAC.**

Finally, since CABAC applies arithmetic coding, it automatically fulfils point 2) and therefore produces binary representations with minimal redundancies. This allows him to attain average bit-lengths of less than 1 bit per parameter element, which usually reflects the information content in sparse models. In contrast, the average bit-lengths of e.g. the scalar Huffman code is lower bounded by 1 bit (since it incurs at least 1 bit of redundancy per element), thus limiting the attainable compression ratios of the model.

[10] The fact that DeepCABAC performs worse than the Lloyd algorithm for the smaller step-size lies in the algorithmic design choice rather than the rate-distortion decision.

VI. Conclusion

In this work we proposed a novel compression algorithm for deep neural networks called *DeepCABAC*. It is based on applying a Context-based Adaptive Binary Arithmetic Coder (CABAC) to the networks parameters, which is the state-of-the-art universal lossless coder employed in the H.264/HEVC and H.265/HEVC video coding standards. DeepCABAC also incorporates a novel quantization scheme that explicitly minimizes the accuracy vs. bit-size trade-off without relying on expensive retraining or access to large amounts of data. Experiments showed that it can compress pretrained neural networks by x18.9 on average, and their sparsified versions by x50.6, consistently attaining higher compression performance than previously proposed coding techniques with similar characteristics. Moreover, DeepCABAC is able to capture correlations between the networks parameters, as such being able to compress the networks parameters beyond the entropy limit of codes that assume a stationary distribution. DeepCABAC has recently been adopted as baseline quantization and entropy coding method for the upcoming MPEG-7 standard for compression of neural networks for multimedia content description and analysis (ISO/IEC 15938 part 17) [46].

As future work we will investigate the impact of compression on more recent and accurate edge-level neural networks such as FBNet and ChamNet [47], [48] to neural network's problem solving strategies [49] and apply DeepCABAC in distributed training scenarios, where the precision of the communicated update parameters may be critical for identifying similar clients [50]. We will also further investigate the achievable compression limits by applying different state-of-the-art sparsification techniques such as [51].

References

[1] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015, [Online]. Available: https://doi.org/10.1038/nature14539.

[2] X. Xiaowei *et al.*, "Scaling for edge inference of deep neural networks," *Nature Electron.*, vol. 1, pp. 216–222, 2018.

[3] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2014, pp. 10–14.

[4] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[5] K. Ota, M. S. Dao, V. Mezaris, and F. G. B. D. Natale, "Deep learning for mobile multimedia: A survey," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 13, no. 3s, pp. 34:1–34:22, 2017.

[6] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," 2016, *arXiv:1602.05629*.

[7] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2019, pp. 1–8.

[8] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2019.2944481.

[9] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," 2017, *arXiv:1710.09282*.

[10] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 126–136, Jan. 2018.

[11] S. Wiedemann, K.-R. Müller, and W. Samek, "Compact and computationally efficient representation of deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2019.2910073.

[12] MPEG Requirements, "Updated call for proposals on neural network compression (n18129)," Moving Picture Experts Group (MPEG), Marrakech, MA, CfP, Jan. 2019.

[13] V. Sze, M. Budagavi, and G. J. Sullivan, *High Efficiency Video Coding: Algorithms and Architectures*. Berlin, Germany: Springer, 2014.

[14] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, Jul. 2003.

[15] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, 1948.

[16] T. Wiegand and H. Schwarz, "Source coding: Part 1 of fundamentals of source and video coding," *Found. Trends Signal Process.*, vol. 4, no. 1–2, pp. 1–222, 2011.

[17] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[18] D. Marpe and T. Wiegand, "A highly efficient multiplication-free binary arithmetic coder and its application in video coding," in *Proc. Int. Conf. Image Process.*, 2003, vol. 2, pp. 263–266.

[19] P. Grünwald and J. Rissanen, *The Minimum Description Length Principle*, (Adaptive computation and machine learning). Cambridge, MA, USA: MIT Press, 2007.

[20] S. Chetlur *et al.*, "cuDNN: Efficient primitives for deep learning," 2014, *arXiv:1410.0759*.

[21] J. Teuhola, "A compression method for clustered bit-vectors," *Inf. Process. Lett.*, vol. 7, no. 6, pp. 308–311, 1978.

[22] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2498–2507.

[23] A. Achille, M. Rovere, and S. Soatto, "Critical learning periods in deep neural networks," 2017, *arXiv:1711.08856*.

[24] M. Nagel, M. van Baalen, T. Blankevoort, and M. Welling, "Data-free quantization through weight equalization and bias correction," 2019, *arXiv:1906.04721*.

[25] Video subgroup, "Description of core experiments on compression of neural networks for multimedia content description and analysis (n18782)," Moving Picture Experts Group (MPEG), Geneva, CH, Tech. Rep., Oct. 2019.

[26] Video subgroup, "Test model 2 of compression of neural networks for multimedia content description and analysis (n18785)," Moving Picture Experts Group (MPEG), Geneva, CH, Tech. Rep., Oct. 2019.

[27] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Proc. IEEE Int. Conf. Neural Netw.*, 1993, pp. 293–299.

[28] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[29] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," 2016, *arXiv:1608.04493*.

[30] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," 2016, *arXiv:1603.05279*.

[31] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," in *Proc. Int. Conf. Learn. Representations*, 2017.

[32] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," 2015, *arXiv:1510.00149*.

[33] C. Louizos, K. Ullrich, and M. Welling, "Bayesian Compression for Deep Learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3288–3298.

[34] S. Wiedemann, A. Marban, K.-R. Müller, and W. Samek, "Entropy-constrained training of deep neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[35] Y. Choi, M. El-Khamy, and J. Lee, "Towards the limit of network quantization," 2016, *arXiv:1612.01543*.

[36] Y. Choi, M. El-Khamy, and J. Lee, "Universal deep neural network compression," 2018, *arXiv:1802.02271*.

[37] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*.

[38] X. Chen, X. Hu, N. Xu, H. Zhou, and and,"Fxpnet: Training deep convolutional neural network in fixed-point representation," in *Proc. Int. Joint Conf. Neural Netw.*, 2017, pp. 2494–2501.

[39] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless CNNs with low-precision weights," 2017, *arXiv:1702.03044*.

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

[41] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018, *arXiv:1804.02767*.

[42] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[43] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "Albert: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[44] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[45] M. Tan and Q. V. Le, "Mixconv: Mixed depthwise convolutional kernels," *CoRR*, vol. abs/1907.09595, 2019. [Online]. Available: http://arxiv.org/abs/1907.09595

[46] Video subgroup, "Working draft 2 of compression of neural networks for multimedia content description and analysis (n18784)," Moving Picture Experts Group (MPEG), Geneva, Switzerland, Tech. Rep., Oct. 2019.

[47] B. Wu *et al.*, "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 10 734–10 742.

[48] X. Dai *et al.*, "ChamNet: Towards efficient network design through platform-aware model adaptation," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2019, pp. 11 398–11 407.

[49] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Commun.*, vol. 10, 2019, Art. no. 1096.

[50] F. Sattler, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints," 2019, *arXiv:1910.01991*.

[51] X. Dai, H. Yin, and N. K. Jha, "NeST: A neural network synthesis tool based on a Grow-and-Prune Paradigm," *IEEE Trans. Comput.*, vol. 68, no. 10, pp. 1487–1497, Oct. 2019.

Simon Wiedemann received the M.Sc. degree in applied mathematics from Technische Universität Berlin, Berlin, Germany. He is currently with the Machine Learning Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany. His major research interests are in machine learning, neural networks, and information theory.

Heiner Kirchhoffer received the Dipl.-Ing. (FH) degree in television technology and electronic media from the Wiesbaden University of Applied Sciences, Wiesbaden, Germany, in 2005 and the Dr.-Ing. degree from the University of Rostock, Rostock, Germany, in 2016. In 2004, he joined the Fraunhofer Institute for Telecommunications-Heinrich Hertz Institute, Berlin, Germany, where he is currently Research Associate in the Video Coding and Analytics Department. He has contributed successfully to the H.265/High Efficiency Video Coding (HEVC) standardization activity of the ITU-T Video Coding Experts Group and the ISO/IEC Moving Pictures Experts Group. His research interests include image and video compression, entropy coding, and efficient representations of deep neural networks.

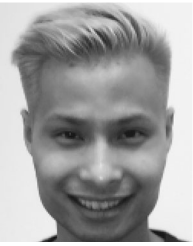
Stefan Matlage received the B.Eng. degree in communications engineering from the University of Applied Sciences Berlin, Berlin, Germany, in 2014 and the M.Sc. degree in electrical engineering from Technische Universität Berlin, Berlin, Germany, in 2018. Since 2013, he has been with the Video Coding Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany. His major research interests are in source coding, prediction and neural networks.

Paul Haase received the B.Sc. and M.Sc. degrees in electrical engineering from Technische Universität Berlin, Berlin, Germany, in 2012 and 2015, respectively. Since 2013, he has been with the Fraunhofer Heinrich Hertz Institute, Berlin, Germany, and joined the Image and Video Coding Group in 2016. His major research interests are in data compression and video coding, specifically entropy coding and compression of neural networks.

Arturo Marban received the B.Eng. degree in mechatronics and the M.Sc. degree in manufacturing systems from the Monterrey Institute of Technology and Higher Education, Monterrey, Mexico, in 2007 and 2010, respectively, and the Ph.D. degree in automatic control, robotics, and computer vision from the Polytechnic University of Catalonia, Catalonia, Spain, in 2018. Afterward, in the same year, he joined the Machine Learning Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany. His major research interests are in machine learning and neural networks, specifically in computer vision and efficient deep learning.

Talmaj Marinč received the B.Sc. degree in financial mathematics from the University of Ljubljana, Ljubljana, Slovenia, in 2014, and the M.Sc. degree in computer science from Technische Universität Berlin, Berlin, Germany, in 2018. Since 2017, he has been with Fraunhofer Heinrich Hertz Institute, Berlin, Germany, and joined the Machine Learning Group in 2019. His major research interests are in machine learning and neural networks, specifically compression and acceleration of neural networks.

David Neumann received the M.Sc. degree in computer science from the Technische Universität Berlin, Berlin, Germany, in 2019. He has been with the Machine Learning Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany since 2016. His research interests include machine learning, neural networks, and efficient deep learning.



Tung Nguyen received the Diploma in computer science (Dipl.-Inf.) from the Technical University of Berlin (TUB), Berlin, Germany, in 2008. He joined the Fraunhofer Institute for Telecommunications, Heinrich Hertz Institute (HHI), Berlin, Germany, in 2009. Since that time, he has worked as a Research Associate, and his current association within the HHI is the Image and Video Coding Group of the Video Coding and Analytics Department. From 2009 to 2015, he actively participated in the standardization activity of the Joint Collaborative Team on Video Coding (JCT-VC). He successfully contributed as a member of the JCT-VC on the topic of entropy coding for the development of HEVC, Version 1. He is also one of the main contributors of the Cross-Component Prediction scheme for the Range Extensions in Version 2 of HEVC. His current research interests include image and video processing and their efficient implementation.

Heiko Schwarz received the Dipl.-Ing. in electrical engineering and the Dr.-Ing. degree from the University of Rostock, Rostock, Germany, in 1996 and 2000, respectively. In 1999, he joined the Image and Video Coding Group, Fraunhofer Institute for Telecommunications–Heinrich Hertz Institute, Berlin, Germany. He has contributed successfully to the standardization activities of the ITU-T Video Coding Experts Group (ITU-T SG16/Q.6-VCEG) and the ISO/IEC Moving Pictures Experts Group (ISO/IEC JTC 1/SC 29/WG 11- MPEG). He has been appointed as a Co-Editor of ITU-T H.264 and ISO/IEC 14496-10 and as a Software Coordinator for the SVC reference software. During the development of the scalable video coding extension of H.264/AVC, he co-chaired several ad hoc groups of the Joint Video Team of ITU-T VCEG and ISO/IEC MPEG, investigating the particular aspects of the scalable video coding design.

Thomas Wiegand (Fellow, IEEE) received the Dipl.-Ing. degree in electrical engineering from the Technical University of Hamburg, Harburg, Germany, in 1995 and the Dr.-Ing. degree from the University of Erlangen, Nuremberg, Germany, in 2000. As a student, he was a Visiting Researcher with Kobe University, Japan, the University of California at Santa Barbara, and Stanford University, USA, where he also returned as a Visiting Professor. He is a Professor with the Department of Electrical Engineering and Computer Science, Technical University of Berlin and is jointly heading the Fraunhofer Heinrich Hertz Institute, Berlin, Germany. He was a Consultant with Skyfire, Inc., Mountain View, CA, USA, and with Vidyo, Inc., Hackensack, NJ, USA. Since 1995, he has been an active participant in standardization for multimedia with many successful submissions to ITU-T and ISO/IEC. In particular, he made significant technical contributions to H.263, H.264/AVC, H.265/HEVC, and the future H.266/VVC standard. In 2000, he was appointed as the Associated Rapporteur of ITU-T VCEG and from 2005 to 2009, he was a Co-Chair of ISO/IEC MPEG Video. The projects that he co-chaired for the development of the H.264/AVC standard have been recognized by an ATAS Primetime Emmy Engineering Award and a pair of NATAS Technology & Engineering Emmy Awards. He was also a recipient of an ATAS Primetime Emmy Engineering Award for the development of H.265/HEVC. For his research in video coding and transmission, he received numerous awards including the Vodafone Innovations Award, the EURASIP Group Technical Achievement Award, the Eduard Rhein Technology Award, the Karl Heinz Beckurts Award, the IEEE Masaru Ibuka Technical Field Award, and the IMTC Leadership Award. He received multiple best paper awards for his publications. Since 2014, Thomson Reuters named him in their list of "The World's Most Influential Scientific Minds" as one of the most cited researchers in his field. He is the recipient of the ITU150 Award. He has been elected to the German National Academy of Engineering (Acatech) and the National Academy of Science (Leopoldina). Since 2018, he has been appointed the Chair of the ITU/WHO Focus Group on Artificial Intelligence for Health.

Detlev Marpe (Fellow, IEEE) received the Dipl.-Math. (with Hons.) from the Technical University of Berlin, Berlin, Germany, in 1990 and the Dr.-Ing. degree from the University of Rostock, Rostock, Germany, in 2004. He joined the Fraunhofer Institute for Telecommunications–Heinrich Hertz Institute, Berlin, Germany, in 1999, where he is currently the Head of the Video Coding and Analytics Department and the Head of the Image and Video Coding Research Group. He was a major technical contributor to the entire process of the development of the H.264/MPEG-4 Advanced Video Coding (AVC) standard and the H.265/MPEG High Efficiency Video Coding (HEVC) standard, including several generations of major enhancement extensions. In addition to the CABAC contributions for both standards, he particularly contributed to the Fidelity Range Extensions (which include the High Profile that received the Emmy Award in 2008) and the Scalable Video Coding Extensions of H.264/MPEG-4 AVC. During the recent development of its successor H.265/MPEG-HEVC, he also successfully contributed to the first model of the corresponding standardization project and further refinements. He also made successful proposals to the standardization of its Range Extensions and 3-D Extensions. His academic work includes more than 200 publications in image and video coding. He holds more than 250 internationally issued patents and numerous patent applications in this field. His research interests include still image and video coding, signal processing for communications and computer vision, and information theory. He is a member of the Informationstechnische Gesellschaft of the Verband der Elektrotechnik Elektronik Informationstechnik e.V. He was the co-recipient of two Technical Emmy Awards as a Key Contributor and a Co-Editor of the H.264/MPEG-4 AVC standard in 2008 and 2009, respectively. He received the IEEE Best Paper Award at the 2013 IEEE International Conference on Consumer Electronics, Berlin, and the SMPTE Journal Certificate of Merit in 2014. He was nominated for the German Future Prize in 2012. He was the recipient of the Karl Heinz Beckurts Award in 2011, the Best Paper Award of the IEEE Circuits and Systems Society in 2009, the Joseph von Fraunhofer Prize in 2004, and the Best Paper Award of the Informationstechnische Gesellschaft in 2004. As a Co-Founder of the Berlin-based daviko GmbH, he received the Prime Prize of the Multimedia Start-Up Competition of the German Federal Ministry of Economics and Technology in 2001. Since 2014, he has been an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.

Wojciech Samek (Member, IEEE) received the Dipl.-Inf. in computer science from the Humboldt University of Berlin, Berlin, Germany, in 2010, and the Dr.rer.nat. degree from the Technical University of Berlin, Berlin, Germany, in 2014. During his studies, he was Scholar of the German National Academic Foundation, a Ph.D. Fellow with the Bernstein Center for Computational Neuroscience Berlin and a Visiting Researcher with NASA Ames Research Center, Mountain View, USA. He is the Head of the Machine Learning Group, Fraunhofer Heinrich Hertz Institute, Berlin, Germany. Since 2014, he has been associated with the Berlin Big Data Center, and in 2018, he became an Principle Investigator of the Berlin Center of Machine Learning. He serves as an Associate Editor for several journals, including IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Digital Signal Processing*, and *PLoS One*, and is a member of the Machine Learning for Signal Processing Technical Committee of the IEEE Signal Processing Society. He is part of various international standardization initiatives, including the MPEG AHG on Compression of Neural Networks for Multimedia Content Description and Analysis, and was organizer of special sessions, workshops and tutorials at top-tier machine learning and signal processing conferences (NIPS, CVPR, ICASSP, MICCAI). He has coauthored more than 100 peer-reviewed journal and conference papers, predominantly in the areas interpretable deep learning, neural network compression, robust signal processing, and federated learning.