DeepCABAC: Context-adaptive binary arithmetic coding for deep neural network compression

Simon Wiedemann [*1] **Heiner Kirchhoffer** [*1] **Stefan Matlage** [*1] **Paul Haase** [*1] **Arturo Marban** [1] **Talmaj Marinc** [1] **David Neumann** [1] **Ahmed Osman** [1] **Detlev Marpe** [1] **Heiko Schwarz** [1] **Thomas Wiegand** [1] **Wojciech Samek** [1]

Abstract

We present DeepCABAC, a novel context-adaptive binary arithmetic coder for compressing deep neural networks. It quantizes each weight parameter by minimizing a weighted rate-distortion function, which implicitly takes the impact of quantization on to the accuracy of the network into account. Subsequently, it compresses the quantized values into a bitstream representation with minimal redundancies. We show that DeepCABAC is able to reach very high compression ratios across a wide set of different network architectures and datasets. For instance, we are able to compress by x63.6 the VGG16 ImageNet model with no loss of accuracy, thus being able to represent the entire network with merely 8.7MB.

1. Introduction

Inspite of their state-of-the-art performance across a wide spectrum of problems (LeCun et al., 2015), deep neural networks have the well-known caveat that most often they have high memory complexity. This does not only imply high storage capacities as a requirement, but also high energy resources and slower runtimes for execution (Horowitz, 2014; Sze et al., 2017; Wang et al., 2019). This greatly limits their adoption in industrial applications or their deployment into resource constrained devices. Moreover, this also difficults their transmission into communication channels with limited capacity, which becomes an obstacle for distributed training scenarios such as in federated learning (McMahan et al., 2016; Sattler et al., 2018; 2019).

As a reaction, a plethora of work has been published on the topic of deep neural network compression (Cheng et al., 2017; Cheng et al., 2018). From all different proposed methods, sparsification followed by weight quantization and entropy coding arguably belong to the set of most popular approaches, since very high compression ratios can be achieved under such paradigm (Han et al., 2015a; Louizos et al., 2017; Wiedemann et al., 2018a;b). Whereas much of research has focused on the sparsification part, a substantially less amount have focused on improving the later two steps. In fact, most of the proposed (post-sparsity) compression algorithms come with at least one of the following caveats: 1) they decouple the quantization procedure from the subsequent lossless compression algorithm, 2) ignore correlations between the parameters and 3) apply a lossless compression algorithm that produce a bitstream with more redundancies than principally needed (e.g. scalar Huffman coding). Moreover, some of the proposed compression algorithms do also not take the impact of quantization on to the accuracy of the network into account.

In this work we present DeepCABAC, a compression algorithm that overcomes all of the above limitations. It is based on applying a context-adaptive binary arithmetic coder (CABAC) on to the quantized parameters, which is the state-of-the-art for lossless compression. It also couples the quantization procedure with CABAC by minimizing a rate-distortion cost function where the rate explicitly measures the bit-size of the network parameters as determined by CABAC. Moreover, it implicitly takes the impact of quantization on to the networks accuracy into account by weighting the distortion with a term that measures the "robustness" of the networks parameter. In our experiments we show that we can significantly boost the compression performance of a wide set of pre-sparsified network architectures, consequently achieving new state-of-the-art results for the VGG16 model.

2. CABAC

Context-adaptive binary arithmetic coding (CABAC) is a form of lossless coding which was originally designed for the video compression standard H.264/AVC (Marpe

[*]Equal contribution [1]Department of Video Coding & Analytics, Fraunhofer Heinrich-Hertz Institut, Berlin, Germany. Correspondence to: Wojciech Samek <wojciech.samek@hhi.fraunhofer.de>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

et al., 2003), but it is also an integral part of its successor H.265/HEVC. CABAC does not only offer high flexibility of adaptation, but also a highly efficient implementation, thus attaining higher compression performance as well as throughputs compared to other entropy coding methods (Marpe & Wiegand, 2003). In short, it applies three powerful coding techniques: 1) Firstly, it binarizes the data to be encoded. That is, it predefines a series of binary decisions (also called bins) under which each data element (or symbol) will be uniquely identified. 2) 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 with a high degree of adaptation to different data distributions. 3) Finally, it employs an arithmetic coder in order to optimally and efficiently code each bin, based on the respective context model. To recall, arithmetic coding is a form of entropy coding which encodes entire strings of symbols into a single integer value. It is well-known to outperform other coding techniques such as the Huffman code (Huffman, 1952) with regards to both, compactness of the data representation and coding efficiency (Witten et al., 1987).

Due to the above reasons, we chose CABAC as our lossless compression method and adapted it for the task of neural network compression.

2.1. Binarization on deep neural networks

Inspired by a prior analysis on the empirical weight distribution of different neural network architectures, we adopted the following bianrization procedure. Given a quantized weight tensor in its matrix form[1], DeepCABAC scans the weight elements in row-major order[2] and encodes each quantized weight element value by: 1) firstly determining if the weight element is a significant element or not. That is, each weight element is assigned with a bit which determines if the element is 0 or not. This bit is then encoded using a binary arithmetic coder, according to its respective context model. 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 bit is analogously encoded, according to its respective context model. 3) Subsequently, a series of bits are analogously encoded, which determine if the element is greater than $1, 2, ..., n \in \mathbb{N}$. The number n becomes a hyperparameter for the encoder. 4) Finally, the reminder is encoded using a fixed-length binary code.

The decoding process is performed analogously. An example scheme of the binarization procedure is depicted in figure 1.

[1]For fully-connected layers this is trivial. For convolutional layers we converted them into their respective matrix form according to (Chetlur et al., 2014).

[2]From left to right, up to down.

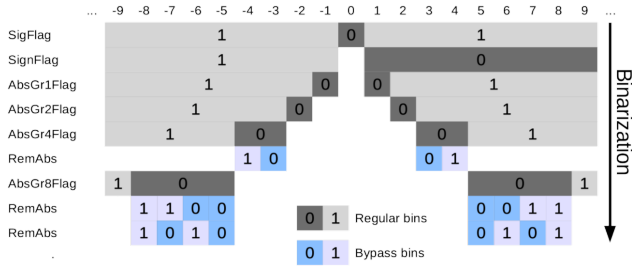

Figure 1. DeepCABAC binarization of neural networks. It encodes each weight element by performing the following steps: 1) encodes a bit named *sigflag* which determines if the weight is a significant element or not (in other words, if its 0 or not). 2) If its not 0, then the sign bit, *signflag*, is encoded. 3) Subsequently, a series of bits are encoded, which indicate if the weight value is greater equal than $1, 2, ..., n \in \mathbb{N}$ (the so called *AbsGr(n)Flag*). 4) Finally, the reminder is encoded. The grey bits (also named regular bins) represent bits that are encoded using an arithmetic coder according to a context model. The other bits, the so called bypass bins, are encoded in fixed-point form. The decoder is analogous.

3. Weighted rate-distortion function

Before we apply CABAC, we have to firstly quantize the weight parameters of the network. We do this by minimizing a generalised form of a rate-distortion function. Namely, we quantize each weight parameter w_i to the quantization point q_{k^*} that minimizes the cost function

$$w_i \rightarrow q_{k^*} = \min_k \eta_i (w_i - q_k)^2 + \lambda R_{ik} \qquad (1)$$

where R_{ik} is the bit-size of the quantization point q_k as determined by DeepCABAC, and λ is the lagrangian multiplier that specifies the desired trade-offs between the bit-size and distortion incurred by the quantization. Notice, how the bit-size R_{ik} now also depends on the index i of the weight to be encoded. This is due to the context-adaptive models which update their probabilities as new elements are being encoded, thus being different for each weight w_i and consequently the bit-size of each quantization point q_k.

Moreover, (1) introduces a weight-specific parameter η_i which takes into account the relative impact that the distortion of a particular weight inccurs on to the accuracy of the network. In this work we take a bayesian approach in order to estimate this parameter. Namely, we assume a gaussian prior for each weight parameter and apply scalable bayesian techniques in order to estimate their sufficient statistics (Kingma et al., 2015; Molchanov et al., 2017; Louizos et al., 2017). As a result, we attain a mean and standard deviation value for each weight parameter of the

Table 1. Compression ratios achieved when combining DeepCABAC with a sparsification method. In parenthesis are the results from previous work, where [1](Han et al., 2015a) and [2](Louizos et al., 2017).

Models	Dataset	Org.acc. (top1) [%]	Org. size	Spars. $\frac{\lvert w\neq 0\rvert}{\lvert w\rvert}$ [%]	Comp. ratio [%]	Acc. (top1) [%]
VGG16	ImageNet	69.43	553.43 MB	9.85 (7.5[1])	**1.57** (2.05[1])	**69.43** (68.83[1])
ResNet50	ImageNet	76.13	102.23 MB	25.40 (29.0[1])	**5.95** (5.95[1])	**74.12** (76.15[1])
Mobile-Net-v1	ImageNet	70.69	16.93 MB	50.73	**12.7**	**66.18**
Small-VGG16	CIFAR10	91.35	59.9 MB	7.57 (5.5[2])	**1.6** (0.86[2])	**91.00** (90.8[2])
LeNet5	MNIST	99.22	1722 KB	1.90 (8.0[1]) (0.6[2])	**0.72** (2.55[1]) (0.13[2])	**99.16** (99.26[1]) (99.00[2])
LeNet-300-100	MNIST	98.29	1066 KB	9.05 (8.0[1]) (2.2[2])	**1.82** (2.49[1]) (0.88[2])	**98.08** (98.42[1]) (98.20[2])
FCAE	CIFAR10	30.14 PSNR	304.72 KB	55.69	**16.15**	**30.09** PSNR

network, where the former can be interpreted as its (new) value and the later as a measure of its "robustness". Thus, when quantizing each w_i, we set $\eta_i = 1/\sigma_i^2$ in (1), with σ_i being the respective standard deviation. This is also theoretically motivated, since (Achille et al., 2017) established a connection between the variances and the diagonal elements of the fisher information matrix.

In order to minimize (1), we also need to define a set of quantization points q_k. We chose them to be equidistant to each other with a particular distance $\Delta \in \mathbb{R}$, namely,

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

where $\sigma_{\min}$ is the smallest standard deviation and $w_{\max}$ the parameter with highest magnitude value. S is then a hyperparameter, which controls the "coarseness" of the quantization points. By selecting Δ in such a manner we ensure that the quantisation points lie within the range of the standard deviation of each weight parameter, in particular for values $S \geq 0$. Moreover, by constraining them to be equidistant to each other we encourage fixed-point representations, which can be exploited in order to perform inference with lower complexity (QNN; TFl).

4. Experiments

We applied DeepCABAC on the set of models described in the evaluation framework (MPEG Requirements, 2019a) of the MPEG call on neural network representations (MPEG Requirements, 2019b). This includes the VGG16, ResNet50 and MobileNet-v1 models and a fully-convolutional autoencoder pretrained on a task of end-to-end image compression (which we named *FCAE*). In addition, we also applied DeepCABAC on the LeNet-300-100 and LeNet5 models and on a smaller version of VGG16[3] model (which we named *Small-VGG16*).

We aplied the variational sparsification method introduced in (Molchanov et al., 2017) on to the LeNet-300-100, LeNet5, Small-VGG16, FCAE and MobileNet-v1 models. However, due to the high training complexity that this method requires, we adopted a slightly different approach for the VGG16 and ResNet50. Namely, we firstly sparsified them by applying the iterative algorithm (Han et al., 2015b), and subsequently applied method (Molchanov et al., 2017) but only for estimating the variances of the distributions (thus, fixing the mean values during training). After sparsification, we applied DeepCABAC on to the weight parameters of each layer separately, excluding biases and normalization parameters. Since the compression result can be sensitive to the parameter S in (2), we probed the compression performance for all $S \in \{0, 1, ..., 256\}$ and selected the best performing model.

The resulting sparsities as well as the compression ratios are displayed in table 1. Notice that for most networks we are not able to reproduce the sparsity ratios reported in the literature. In addition, we did not perform any fine-tuning after compression, thus having a particularly challenging setup for achieving good post-sparsity compression ratios. Nevertheless, in-spite of these two disadvantages, DeepCABAC is able to significantly compress further the models, boosting the compression by 74% ($\pm$ 8%) on average and consequently achieving compression results comparable to the current state-of-the-art. Moreover, we are able to compress by x63.6 (1.57%) the VGG16 model without loss of accuracy, thus reaching a new state-of-the-art benchmark.

5. Conclusion

We show that one can boost significantly the compression gains if one applies state-of-the-art coding techniques on to pre-sparsified deep neural networks. In particular, our proposed coding scheme, DeepCABAC, is able to increase the compression rates of pre-sparsified networks by 74% on average, attaining as such compression ratios comparable (or sometimes higher) to the current state-of-the-art. In future work we will benchmark DeepCABAC also on non-sparsified networks, as well as apply it in the context of distributed learning scenarios where memory complexity is critical (e.g. in federated learning).

[2]https://github.com/slychief/ismir2018_tutorial/tree/master/metadata

[3]http://torch.ch/blog/2015/07/30/cifar.html

References

QNNPACK open source library for optimized mobile deep learning. https://github.com/pytorch/QNNPACK. Accessed: 28.02.2019.

TensorFlow Lite. https://www.tensorflow.org/lite. Accessed: 28.02.2019.

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

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

Cheng, Y., Wang, D., Zhou, P., and Zhang, T. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, Jan 2018.

Chetlur, S., Woolley, C., Vandermersch, P., Cohen, J., Tran, J., Catanzaro, B., and Shelhamer, E. cudnn: Efficient primitives for deep learning. *arXiv:1410.0759*, 2014.

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

Han, S., Pool, J., Tran, J., and Dally, W. J. Learning both weights and connections for efficient neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1135–1143, 2015b.

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

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, Sep. 1952.

Kingma, D. P., Salimans, T., and Welling, M. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2575–2583, 2015.

LeCun, Y., Bengio, Y., and Hinton, G. E. Deep learning. *Nature*, 521:436–444, 2015.

Louizos, C., Ullrich, K., and Welling, M. Bayesian Compression for Deep Learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 3288–3298, 2017.

Marpe, D. and Wiegand, T. A highly efficient multiplication-free binary arithmetic coder and its application in video coding. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 2, pp. 263–266, Sep. 2003.

Marpe, D., Schwarz, H., and Wiegand, T. Context-based adaptive binary arithmetic coding in the h.264/avc video compression standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, July 2003.

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

Molchanov, D., Ashukha, A., and Vetrov, D. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning (ICML)*, pp. 2498–2507, 2017.

MPEG Requirements. Updated evaluation framework for compressed representation of neural networks. N18162. Technical report, Moving Picture Experts Group (MPEG), Marrakech, MA, Jan. 2019a.

MPEG Requirements. Updated call for proposals on neural network compression. N18129. Cfp, Moving Picture Experts Group (MPEG), Marrakech, MA, Jan. 2019b.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Sparse binary compression: Towards distributed deep learning with minimal communication. *arXiv:1805.08768*, 2018.

Sattler, F., Wiedemann, S., Müller, K.-R., and Samek, W. Robust and communication-efficient federated learning from non-iid data. *arXiv:1903.02891*, 2019.

Sze, V., Chen, Y.-H., Yang, T.-J., and Emer, J. S. Efficient processing of deep neural networks: A tutorial and survey. *arXiv:1703.09039*, 2017.

Wang, E., Davis, J. J., Zhao, R., Ng, H.-C., Niu, X., Luk, W., Cheung, P. Y. K., and Constantinides, G. A. Deep neural network approximation for custom hardware: Where we've been, where we're going. *arXiv:1901.06955*, 2019.

Wiedemann, S., Marbán, A., Müller, K.-R., and Samek, W. Entropy-constrained training of deep neural networks. *arXiv:1812.07520*, 2018a.

Wiedemann, S., Müller, K.-R., and Samek, W. Compact and computationally efficient representation of deep neural networks. *arXiv:1805.10692*, 2018b.

Witten, I., H, I., , N., M, R., , C., and G, J. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, Jun 1987.