Learning Sparse & Ternary Neural Networks with Entropy-Constrained Trained Ternarization (EC2T)

Arturo Marban, Daniel Becking, Simon Wiedemann, Wojciech Samek

Department of Video Coding & Analytics, Fraunhofer Heinrich-Hertz Institut
Berlin, Germany

{arturo.marbangonzalez,daniel.becking,simon.wiedemann,wojciech.samek}@hhi.fraunhofer.de

Abstract

Deep neural networks (DNN) have shown remarkable success in a variety of machine learning applications. The capacity of these models (i.e., number of parameters), endows them with expressive power and allows them to reach the desired performance. In recent years, there is an increasing interest in deploying DNNs to resource-constrained devices (i.e., mobile devices) with limited energy, memory, and computational budget. To address this problem, we propose Entropy-Constrained Trained Ternarization (EC2T), a general framework to create sparse and ternary neural networks which are efficient in terms of storage (e.g., at most two binary-masks and two full-precision values are required to save a weight matrix) and computation (e.g., MAC operations are reduced to a few accumulations plus two multiplications). This approach consists of two steps. First, a super-network is created by scaling the dimensions of a pre-trained model (i.e., its width and depth). Subsequently, this super-network is simultaneously pruned (using an entropy constraint) and quantized (that is, ternary values are assigned layer-wise) in a training process, resulting in a sparse and ternary network representation. We validate the proposed approach in CIFAR-10, CIFAR-100, and ImageNet datasets, showing its effectiveness in image classification tasks.

1. Introduction

Convolutional neural networks (CNN) have excelled in numerous computer vision applications. Their performance is attributed to their design. That is, deeper (i.e., designed with many layers) and high-capacity (i.e., equipped with many parameters) CNNs achieve better performance in a given task, at the cost of sacrificing computational and memory efficiency. This general trend has been disrupted by the need to deploy neural networks in resource-constrained devices (e.g., autonomous vehicles, robots, smartphones, wearable, and IoT devices) with limited energy, memory, and computational budget, as well as low-latency and/or low-communication cost requirements. Thus, driven by both the industry and the scientific community, the design of efficient CNNs has become an active area of research. Moreover, the Moving Picture Expert Group (MPEG) of the International Organization of Standards (ISO) joined this endeavor, and recently issued a call on neural network compression techniques [1].

Recent studies have shown that most CNNs are over-parameterized for the given task [2]. Such models can be interpreted as super-networks, designed with millions of parameters to reach a target performance (e.g., high classification accuracy), while being memory and computational inefficient. However, from these models, it is possible to find a small and efficient sub-network with comparable performance. This hypothesis has been validated with simple methods, i.e., by pruning neural network connections based on the weights' magnitude [3], resulting in little accuracy degradation. Moreover, the recently proposed lottery-ticket hypothesis [4], supports the existence of an optimal sub-network inside a super-network, and has shown to generalize across different datasets and optimizers [5].

Among existing network compression techniques, pruning and quantization are two popular and effective techniques to reduce the redundancy of deep neural networks [6]. Pruning entails systematically removing network connections in a structured (i.e., by removing groups of parameters) or unstructured fashion (i.e., by removing individual parameter elements) [7]. In contrast, quantization minimizes the bit-width of the network parameter values (and thus, the number of distinct values) [8, 9]. From another perspective, efficient neural networks can be designed by finding the right balance between its dimensions, i.e., the networks' width, depth, and input resolution. In this regard, compound model scaling [10] allows scaling the di-

mensions of a baseline-network according to some heuristic rules grounded on computational efficiency.

In this work, we propose Entropy-Constrained Trained Ternarization (EC2T), a method that leverages on compound model scaling [10] and ternary quantization techniques [9], to design a sparse and ternary neural network. The motivations behind such network representation are based on efficiency. Specifically, in terms of storage, at most two binary-masks and two full-precision values are required to represent and save each layer's weight matrix. Regarding mathematical operations, multiply-accumulate operations (MACs) are reduced to a few accumulations plus two multiplications. The EC2T approach is illustrated in Figure 1 and consists of two stages. In the first stage, a super-network is created by scaling the dimensions of a baseline-network (its width and depth). Subsequently, during a training stage, a sparse and ternary sub-network is found by simultaneously pruning (enforced by introducing an entropy constraint in the assignment cost function) and quantizing (ternary values are assigned layer-wise) the super-network. Specifically, our contributions are:

- We propose an approach to design sparse and ternary neural networks, that relies on compound model scaling [10] and quantization techniques. For the latter, we extend the approach described in [9] by introducing an assignment cost function in terms of distance and entropy constraints. The entropy constraint allows adjusting the trade-off between sparsity and accuracy in the quantized model. Therefore, quantized models with different levels of sparsity can be rendered, according to the compression and application requirements.

- Our approach allows simultaneous quantization and sparsification in a single training stage.

- In the context of image classification, the proposed approach finds sparse and ternary networks across different datasets (CIFAR-10, CIFAR-100, and ImageNet), whose performance is competitive with efficient state-of-the-art models.

This paper is organized as follows. First, in section 2, a literature review of techniques to design efficient neural networks is provided, emphasizing those that are related to our approach. Subsequently, in section 3, the proposed EC2T approach is detailed. Afterward, in section 4, we present experimental evidence and results, validating the proposed method across different networks and datasets. Finally, in section 5, we discuss the insights of the EC2T approach, its advantages and downsides, and future work.

2. Related Works

In recent years, various techniques have been proposed in the literature to design efficient neural networks, e.g., pruning, quantization, distillation, and low-rank factorization [6]. In particular, pruning and quantization provide unique benefits to DNNs in terms of hardware efficiency and acceleration.

Pruning removes non-essential neural network connections, according to different criteria, either in groups (structured pruning) or individual parameters (unstructured pruning). Specifically, the second approach is achieved by maximizing the sparsity [1] of the network parameters. Consequently, the computational complexity of the network is reduced, since arithmetic operations can be skipped for those parameter elements which are zero [11]. Early works on sparsity use second-order derivatives (Hessian) to compute the saliency of parameters, suppressing those with the smallest value [12, 13]. Current state-of-the-art techniques to promote sparsity in DNNs rely either on magnitude-based pruning or Bayesian approaches [14]. Magnitude-based pruning is the simplest and most effective way to induce sparsity in neural networks, [7]. In contrast, Bayesian approaches although computationally expensive, represent an elegant solution to the problem. Moreover, they establish connections with information theory. In this context, variational dropout [15] and l_0-regularization [16] are two representative techniques.

Regarding quantization, it reduces the redundancy of deep neural networks by minimizing the bit-width of the full-precision parameters. Therefore, quantized networks require fewer bits to represent each full-precision weight, and demand less mathematical operations than their full-precision counterparts. Binary networks [17, 18] represent an extreme case of quantization where both, weights and activations are binarized. Thus, arithmetic operations are reduced to bit-wise operations. By introducing three distinct elements per layer, ternary networks achieve more expressive power and higher performance than binary networks. Moreover, sparsity can be induced in the network by including zero as a quantized value, while the remaining values are modeled with scaling factors per layer. Following this approach, [19] proposed to minimize the Euclidean distance between full-precision and quantized parameters (e.g., w_q), where the latter are symmetrically constrained (e.g., $w_q \in \{-a, 0, a\}$, with $a > 0$). In contrast, [9] used asymmetric constraints (e.g., $w_q \in \{-a, 0, b\}$, with $a > 0$ and $b > 0$), improving the modeling capabilities of ternary networks. Several variants of ternary network quantization exist, e.g., based on Truncated Gaussian Approximation (TGA) [20], Alternating Direction

[1] Percentage of zero-valued parameter elements in the whole neural network.

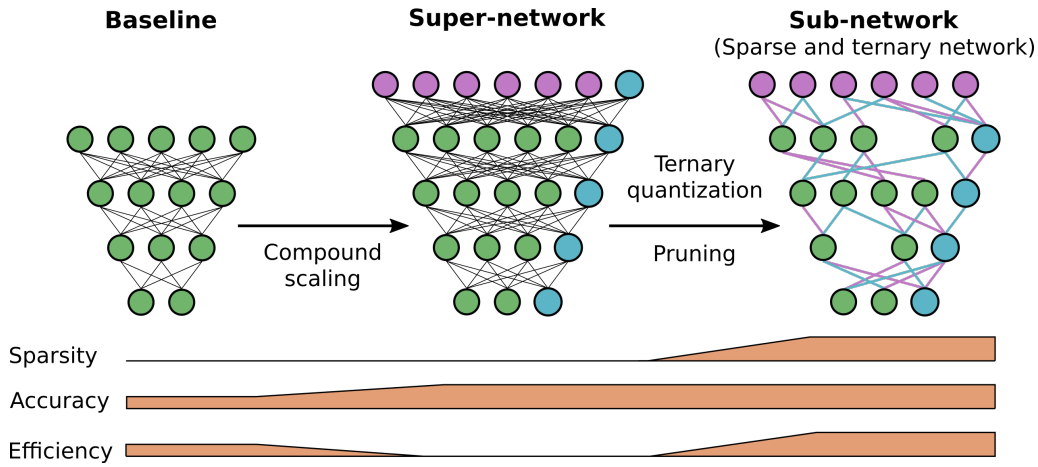

Figure 1. In the EC2T approach, model compound scaling is used to create a super-network from a baseline-network. Afterward, in a ternary quantization stage, this super-network is simultaneously pruned and quantized, rendering a sparse and ternary sub-network with comparable performance.

Method of Multipliers (ADMM)) [21], and Multiple-Level-Quantization (MLQ) [22], among others. With regards to hardware efficiency, ternary networks represent a trade-off between binary networks (extremely hardware-friendly, but with limited modeling capabilities) and their full-precision counterparts (with higher modeling capabilities, but expensive in terms of storage and computational resources), [19].

Usually, highly efficient network representations are the result of combining multiple techniques. For instance, pruning followed by quantization [23, 24], in addition to entropy coding [25, 26, 27]. From a different perspective, progress in designing efficient neural networks has been fueled by advances in hand-crafted architectures (e.g., Mobilenet [28], Mobilenet-V2 [29], and ShuffleNet [30]) as well as neural architecture search techniques (e.g., Mnasnet [31], EfficientNet [10], and MobileNet-V3 [32]). Moreover, simpler methods such as model scaling, allows increasing the performance of a baseline network by scaling one or more dimensions (i.e., its depth, width, and input resolution) independently [31, 32]. In [10], this approach is improved with the introduction of compound model scaling, where the network dimensions are treated as dependent variables, constrained by a limited number of resources, measured in terms of floating-point operations (FLOPs).

In this research work, we advocate for compound model scaling, ternary quantization, and information theory techniques, as the core building blocks to design a CNN with optimal dimensions (i.e., the right balance between the networks' width and depth) and efficient parameter representation (i.e., three distinct values per layer and maximal sparsity).

3. Learning Sparse & Ternary Networks

The entropy-constrained trained ternarization (EC2T) approach (see Figure 1), consists of two stages, namely compound model scaling followed by ternary quantization, both described in sections 3.1 and 3.2, respectively.

3.1. Compound model scaling

In this stage, a super-network is created by scaling the dimensions of a pre-trained model, resulting in an over-parameterized network. Specifically, the pre-trained network's depth, width, and input image resolution, are modified with the scaling factors d, w, and r, respectively, according to Equation (1). In this equation, a, b and c, are constants determined by grid search, and ϕ is an user specified parameter. For small-scale datasets (CIFAR-10 and CIFAR-100) the input image resolution was fixed in the pre-trained model. Thus, Equation (1) was solved with $r = 1$. On the other hand, for large-scale datasets (ImageNet), the EfficientNet-B1 network was adopted using the scaling factors suggested in [10].

$$d = a^{\phi},\ w = b^{\phi},\ r = c^{\phi} \qquad (1)$$
$$\text{s.t.} \quad a \cdot b^2 \cdot c^2 \approx 2 \quad \text{and} \quad a \geq 1, b \geq 1, c \geq 1$$

3.2. Ternary quantization

In this stage, a sparse and ternary sub-network is obtained by simultaneously pruning and quantizing a super-network. To this end, we extend the approach described in [9], where a ternary network is obtained by the interplay between quantized and full-precision models. That is, gradients from the quantized model are used to update both, its parameters and those of the full-precision model. Therefore, the first parameter update enables the learning

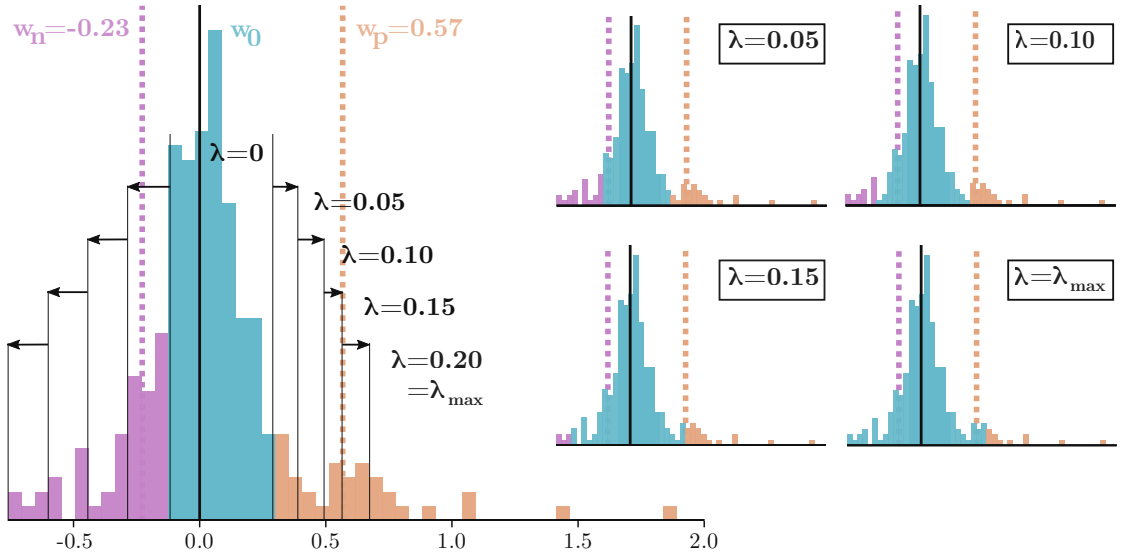

Figure 2. Histograms of the parameters in the projection-convolution layer, in the first block (MBConv1) of the EfficientNet-B1 network. The centroid values w_n (negative scalar), w_0 (zero), and w_p (positive scalar), are shown in magenta, blue and orange colors, respectively. The hyper-parameter λ controls the intensity of the network sparsification, i.e., how many full-precision weight elements are assigned with the value w_0. When λ=0, the weights are quantized to their nearest neighbor centroids. Using small values for λ (see the histogram with λ=0.05) results in quantized parameters with low sparsity (i.e., few parameters are set to zero). As λ is increased (see histograms with λ=0.10 and 0.15), the sparsity of the quantized parameters is promoted (i.e., most parameters are set to zero). Eventually, as this process continues, there is a value $\lambda = \lambda_{max}$, at which the network parameters are binarized. In this special case, one of the two clusters of values (represented by w_n and w_p) is completely assigned to w_0 (see the histogram with λ=λ_{max}).

of ternary values (i.e., only two scalar values per layer are learned, while the third quantized value, which is zero, is excluded from the learning process). On the other hand, the latter parameter update promotes the learning of ternary assignments (i.e., by adapting the full-precision parameters to the quantization process). Nonetheless, this approach does not allow explicit control of the sparsification process. To overcome this limitation, we introduce the assignment cost function shown in Equation (2), which guides the assignment (with centroid indices) of ternary values (or centroid values) in the quantized network, in terms of distance and entropy constraints.

$$\mathbf{C}_c^{(l)} = d(\mathbf{W}^{(l)}, w_c^{(l)}) - \lambda^{(l)} \ \log_2(P_c^{(l)}) \tag{2}$$

$$d_{W_{ij},w_c} = (W_{ij} - w_c)^2 \tag{3}$$

In Equation (2), $\mathbf{C}_c^{(l)}$ stands for the assignment cost for the full-precision weights $\mathbf{W}^{(l)}$ at layer l, given the centroid values $w_c^{(l)}$, indexed by c. Therefore, if $\mathbf{W}^{(l)}$ has $m \times n$ dimensions and there are n_c centroid values in that layer, then $\mathbf{C}_c^{(l)} \in \Re^{n_c \times m \times n}$. The first term in Equation (2) measures the distance between every full-precision weight element $W_{ij}^{(l)} \in \mathbf{W}^{(l)}$ (where i and$j$ are indices along the dimensions of $\mathbf{W}^{(l)}$) and the centroid values $w_c^{(l)} \in \Re$, according to Equation (3). The second term in Equation (2), weighted by the scalar $\lambda^{(l)} \in \Re$, is an entropy constraint which promotes sparsity in the quantized model. This is achieved by measuring the information content of the quantized weights, i.e., $I = -\log_2(P_c^{(l)}) \in \Re$, where the probability $P_c^{(l)} \in [0, 1]$ defines how likely a weight element $W_{ij}^{(l)} \in \mathbf{W}^{(l)}$ is going to be assigned to the centroid value $w_c^{(l)}$. This probability is calculated for each layer l as $P_c^{(l)} = N_{w_c}^{(l)}/N_{\mathbf{W}}^{(l)}$, with $N_{w_c}^{(l)}$ being the number of full-precision weight elements assigned to the centroid value $w_c^{(l)}$, and $N_{\mathbf{W}}^{(l)}$ the total number of parameters in $\mathbf{W}^{(l)}$.

After computing Equation (2) (for all layers and centroid values), the quantized model is updated at layer l, by assigning the current centroid values ($w_c^{(l)}$), using the new centroid indices (c) obtained from Equation (4). In this equation, the assignment matrix $\mathbf{A}^{(l)}$ has the dimensions of the full-precision weights $\mathbf{W}^{(l)}$. For ternary networks, we define the centroid values as $w_c^{(l)} \in \{w_n, w_0, w_p\}$, and their assignments with the indices $c \in \{n, 0, p\}$. In this notation, the indices n, 0, and p, correspond to negative, zero, and positive values, respectively.

$$\mathbf{A}^{(l)} = \underset{c}{\text{argmin}} \ \mathbf{C}_c^{(l)} \tag{4}$$

During the ternary quantization process, the strength of the sparsification (at layer l) is modulated by the scalar $\lambda^{(l)}$

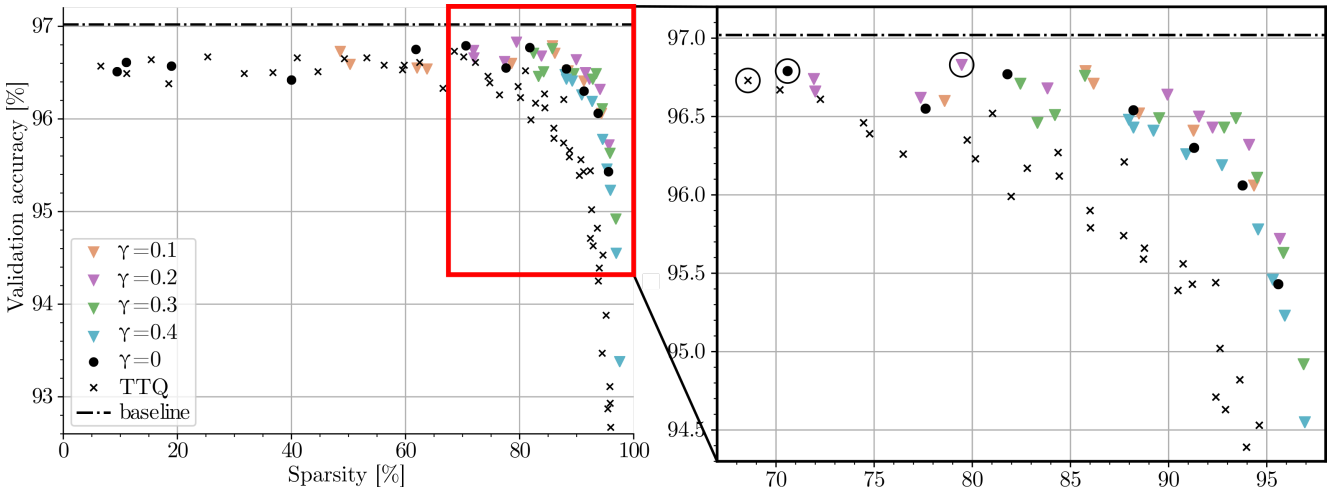

Figure 3. Performance of the C10-MicroNet network evaluated in the CIFAR-10 dataset, using TTQ vs our proposal (EC2T). Every data point in this plot represents a quantized model, trained with a specific level of sparsity, and initialized with different centroid values. In the TTQ approach, the sparsity is controlled via simple thresholding as described in [9], whereas in the EC2T approach, it is modulated by γ, which was increased from 0.0 (low sparsity) to 0.4 (high sparsity), in steps of 0.1. Notice that beyond 70% sparsity, the accuracy of the quantized models degrades quickly. However, this effect is more evident when using TTQ than EC2T.

(shown in Equation (2)). As a concrete example, Figure 2 illustrates the effect of using different values for $\lambda^{(l)}$ during the quantization of the parameters (in the first block) of the EfficientNet-B1 network. In practice, $\lambda^{(l)}$ is computed as $\lambda^{(l)} = \gamma\, \delta^{(l)}\, \lambda^{(l)}_{\max}$. In this expression, γ is a global hyper-parameter that controls the intensity of the sparsification, while $\delta^{(l)}$ and $\lambda^{(l)}_{\max}$ are scalars computed layer-wise. The scaling factor $\delta^{(l)}$, renders higher values for layers with lots of parameters. Analogously, it renders lower values for layers with few parameters. Finally, $\lambda_{\max}$ is updated during training and avoids a binary quantization process (see the histogram with λ=λ_{max} in Figure 2).

4. Experiments & Results

The experiments were conducted in a variety of networks across different datasets (i.e., CIFAR-10, CIFAR-100, and ImageNet), using multiple GPUs (NVIDIA Titan-V and Tesla-V100).

First, to reveal the advantages of our proposal (EC2T) over Trained-Ternary-Quantization (TTQ) [9], an image classification network was designed for the CIFAR-10 dataset, by introducing the building blocks of PyramidNet [33] in the ResNet-44 architecture [34]. This neural network, termed C10-MicroNet, was derived from models designed for the 2019 MicroNet Challenge [2] competition. For a detailed description of the network architecture, see Appendix A. The experimental results contrasting the two mentioned approaches are depicted in Figure 3. In this illustration, notice that as the sparsity of the quantized networks increases, EC2T shows less accuracy degradation than TTQ.

Subsequently, Table 1 provides a comparison of the EC2T approach vs state-of-the-art ternary quantization techniques, by applying them to ResNet-20 and ResNet-18 networks, in CIFAR-10 and ImageNet datasets, respectively. From these results, we have two main conclusions. First, they suggest that disabling the entropy constraint in Equation (2) (i.e., setting $\lambda = 0$), renders ternary models with low sparsity. Nonetheless, they are more efficient than their full-precision counterparts and show little accuracy degradation. These ternary networks are referred to as EC2T-1 in Table 1. Specifically, in the ImageNet dataset, the EC2T-1 model reduces the parameter count in 92.25% and the FLOPs in 79.73%, while in the CIFAR-10 dataset, the reductions are 95.02% and 86.35% in parameter count and FLOPs, respectively. In contrast, by enabling the entropy constraint in Equation (2) (i.e., setting $\lambda > 0$), it results in ternary models with increased sparsity, and thus, they are more efficient in terms of parameter size and mathematical operations. For instance, in the ImageNet dataset, the model with the highest sparsity is EC2T-4, which reduces the number of parameters by 93.88% and the number of FLOPs by 86.61%, while its accuracy is degraded only by 2.73%. Likewise, in the CIFAR-10 dataset, the model with the highest sparsity is EC2T-3, with an accuracy degradation of 0.91%, while the parameter count and FLOPs are reduced by 95.91% and 91.88%, respectively. The second conclusion is that the EC2T approach renders accurate ternary models, which are competitive with state-of-the-art techniques. Regarding sparsity, only [9] provides an es-

[2] https://micronet-challenge.github.io

Table 1. Comparison of the EC2T approach vs state-of-the-art ternary network quantization techniques, applied to ResNet-20 and ResNet-18 networks, in CIFAR-10 and ImageNet datasets, respectively.

Model	**Top-1 Acc.** (%)	$\frac{\lvert W=0\rvert}{\lvert W\rvert}$ (%)‡	**#Params.**	#+	#×	**#FLOPs**
			ImageNet			
ResNet-18^{a}	69.75	0.00	11M	1795M	1797M	3592M
EC2T-1 ($\lambda = 0$)^{b}	67.30	26.80	852K	669M	59M	728M
EC2T-2 ($\lambda > 0$)^{c}	**67.58**	59.00	734K	560M	61M	622M
EC2T-3 ($\lambda > 0$)^{c}	67.26	72.09	686K	528M	57M	585M
EC2T-4 ($\lambda > 0$)^{c}	67.02	**75.62**	**673K**	**424M**	**57M**	**481M**
TTQ [9]	66.60	30-50	⊘	⊘	⊘	⊘
ADMM [21]	67.00	⊘	⊘	⊘	⊘	⊘
TGA [20]	66.00	⊘	⊘	⊘	⊘	⊘
			CIFAR-10			
ResNet-20^{a}	91.67	0.00	269K	40.6M	40.7M	81.3M
EC2T-1 ($\lambda = 0$)^{b}	**91.16**	45.17	13.4K	10.6M	0.5M	11.1M
EC2T-2 ($\lambda > 0$)^{c}	91.01	63.90	11.8K	8.0M	0.5M	8.5M
EC2T-3 ($\lambda > 0$)^{c}	90.76	**73.26**	**11.0K**	**6.1M**	**0.5M**	**6.6M**
TTQ [9]	91.13	30-50	⊘	⊘	⊘	⊘
TGA [20]	90.39	⊘	⊘	⊘	⊘	⊘
MLQ [22]	90.02	⊘	⊘	⊘	⊘	⊘

^{a} Baseline model. ^{b} EC2T approach with the entropy constraint disabled ($\lambda = 0$).
^{c} EC2T approach with the entropy constraint enabled ($\lambda > 0$).
‡ Sparsity, measured as the percentage of zero-valued parameters in the whole neural network.
⊘: Not reported by the authors.

timated value for the ternary models after applying TTQ (30%-50%). For the remaining techniques (ADMM [21], TGA [20], and MLQ [22]), only the quantized model accuracy is reported.

Finally, Table 2 contrasts efficient state-of-the-art neural networks vs sparse and ternary networks rendered with our proposal, in three distinct datasets (CIFAR-10, CIFAR-100, and ImageNet). The former models include CondenseNet [35], Mobilenet-V2 [29], and Mobilenet-V3 [32]. The latter models result from applying the EC2T approach to the pre-trained networks, C10-MicroNet, C100-MicroNet, and EfficientNet-B1 [10]. In particular, the C10-MicroNet and C100-MicroNet networks were designed and improved based on our submissions to the 2019-MicroNet Challenge. Both share the same topology, except in the last layer (i.e., the softmax layer), which is adapted to the number of output classes (see Appendix A). From the results in Table 2, we highlight two points. First, the ternary networks found by our proposed technique (see models indicated with EC2T), are more efficient in terms of parameter size and FLOPs than their respective baselines (C10-MicroNet, C100-MicroNet, and EfficientNet-B1). Moreover, using the tree adder [36] and efficient matrix representations (including Compressed-Entropy-Row (CER)/Compressed-Sparse-Row (CSR) formats [11] and the method described in Appendix B), leads to further savings in mathematical operations and storage (see models referred with Improvements). Second, these ternary models are competitive with current state-of-the-art efficient neural networks (i.e., CondenseNet, Mobilenet-V2, and Mobilenet-V3), offering similar advantages in terms of memory and computational resources.

5. Conclusions

In this work, we presented Entropy-Constrained Trained Ternarization, an approach that relies on compound model scaling and ternary quantization to design efficient neural networks. By incorporating an entropy constraint during the network quantization process, a sparse and ternary model is rendered, which is efficient in terms of storage and mathematical operations. The proposed approach has shown to be effective in image classification tasks in both, small and large-scale datasets. As future work, this method will be investigated in other tasks and scenarios, e.g., federated-learning [37]. Moreover, interpretability techniques [38] will help us to understand how these models make predictions given their constrained parameter space.

Table 2. Ternary models rendered with the EC2T approach vs efficient state-of-the-art neural networks, in CIFAR-10, CIFAR-100, and ImageNet datasets.

Model	Top-1 Acc. (%)	$\frac{\lvert W=0\rvert}{\lvert W\rvert}$ (%)‡	#Params.	#+	#×	#FLOPs
			ImageNet			
EfficientNet-B1[a]	78.43	0.00	7.72M	654M	670M	1324M
+EC2T ($\lambda = 0$)[b]	75.05	60.73	1.07M	338M	50M	387M
+Improvements[c]	-	-	**972K**	**212M**	**50M**	**261M**
MobileNet-V2 (d=1.4)	74.70	⊘	6.90M	⊘	⊘	585M⋆
MobileNet-V3 (Large)	**75.20**	⊘	5.40M	⊘	⊘	219M⋆
			CIFAR-100			
C100-MicroNet[a]	81.47	0.00	8.03M	1243M	1243M	2487M
+EC2T ($\lambda = 0$)[b]	80.13	90.49	412K	126M	3M	129M
+Improvements[c]	-	-	**226K**	**67M**	**3M**	**71M**
CondenseNet-86	76.36	⊘	520K	⊘	⊘	65M⋆
CondenseNet-182	**81.50**	⊘	4.20M	⊘	⊘	513M⋆
			CIFAR-10			
C10-MicroNet[a]	97.02	0.00	8.02M	1243M	1243M	2487M
+EC2T ($\lambda = 0$)[b]	95.87	95.64	295K	72M	3M	75M
+Improvements[c]	-	-	**133K**	**39M**	**3M**	**42M**
CondenseNet-86	95.00	⊘	520K	⊘	⊘	65M⋆
CondenseNet-182	**96.24**	⊘	4.20M	⊘	⊘	513M⋆

[a] Baseline model. [b] EC2T approach with the entropy constraint enabled ($\lambda > 0$).
[c] Improved representation of the neural network parameters by applying the tree adder, the Compressed-Entropy-Row (CER)/Compressed-Sparse-Row (CSR) formats, and the method described Appendix B.
‡ Sparsity, measured as the percentage of zero-valued parameters in the whole neural network.
⋆ Reported as Multiply-Additions (MAdds). The number of FLOPs is approximately twice this value.
⊘: Not reported by the authors.

Acknowledgement

This work was funded by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref 01IS18037I).

References

[1] MPEG-Requirements. Updated call for proposals on neural network compression. Call for proposals, Moving Picture Experts Group (MPEG), Marrakech, MA, Jan 2019. 1

[2] M. Denil, B. Shakibi, L. Dinh, M.A. Ranzato, and N. De Freitas. Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156, 2013. 1

[3] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1

[4] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. 1

[5] A. Morcos, H. Yu, M. Paganini, and Y. Tian. One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In *Advances in Neural Information Processing Systems*, pages 4933–4943, 2019. 1

[6] Y. Cheng, D. Wang, P. Zhou, and T. Zhang. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, Jan 2018. 1, 2

[7] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag. What is the state of neural network pruning?, 2020. 1, 2

[8] Y. Choi, M. El-Khamy, and J. Lee. Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*, 2016. 1

[9] C. Zhu, S. Han, H. Mao, and W. Dally. Trained ternary quantization. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. 1, 2, 3, 5, 6

[10] M. Tan and Q.V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 6105–6114, 2019b. 1, 2, 3, 6

[11] S. Wiedemann, K. R. Müller, and W. Samek. Compact and computationally efficient representation of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):772–785, March 2020. 2, 6

[12] Y. LeCun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990. 2

[13] B. Hassibi, D. G. Stork, and G. J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks*, pages 293–299 vol.1, March 1993. 2

[14] T. Gale, E. Elsen, and S. Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019. 2

[15] D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017. 2

[16] C. Louizos, M. Welling, and D.P. Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017b. 2

[17] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 2

[18] T. Simons and D.J. Lee. A review of binarized neural networks. *Electronics*, 8(6), 2019. 2

[19] Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016. 2, 3

[20] Z. He and D. Fan. Simultaneously optimizing weight and quantizer of ternary neural network using truncated gaussian approximation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11438–11446, 2019. 2, 6

[21] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3, 6

[22] Y. Xu, Y. Wang, A. Zhou, W. Lin, and H. Xiong. Deep neural network compression with single and multiple level quantization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 3, 6

[23] M. Federici, K. Ullrich, and M. Welling. Improved bayesian compression. *arXiv preprint arXiv:1711.06494*, 2017. 3

[24] S. Wiedemann, A. Marban, K. R. Müller, and W. Samek. Entropy-constrained training of deep neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2019. 3

[25] S. Han, H. Mao, and W. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. 3

[26] C. Louizos, K. Ullrich, and M. Welling. Bayesian compression for deep learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS17, page 32903300, Red Hook, NY, USA, 2017a. Curran Associates Inc. 3

[27] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marinc, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, and W. Samek. DeepCABAC: A universal compression algorithm for deep neural networks. *IEEE Journal of Selected Topics in Signal Processing*, pages 1–1, 2020. 3

[28] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, June 2018. 3, 6

[30] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, June 2018. 3

[31] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q.V. Le. Mnasnet: Platform-aware neural architecture search for mobile. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019a. 3

[32] A. Howard, M. Sandler, G. Chu, L.C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q.V. Le, and H. Adam. Searching for mobilenetv3. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 3, 6

[33] D. Han, J. Kim, and J. Kim. Deep pyramidal residual networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6307–6315, July 2017. 5, 10

[34] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, June 2016. 5, 10

[35] G. Huang, S. Liu, L. Van der Maaten, and K.Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2752–2761, 2018. 6

[36] J. Cheng. A Comprehensive Study of Network Compression for Image Classification. In *MicroNet: Large-Scale Model Compression Competition*, 2019. 6

[37] F. Sattler, S. Wiedemann, K. R. Müller, and W. Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. 6

[38] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):e0130140, 2015. 6

A. MicroNet-C10 & MicroNet-C100 Networks

The MicroNet-C10 and MicroNet-C100 networks were designed for the CIFAR-10 and CIFAR-100 datasets, respectively. They share the same architecture described in Table A.1, which consists of three sections of layers. The first section is represented by the input layer or "Stem Convolution". The next section has three stages, each one containing identical building blocks, whose elements are depicted in Figure A.1. This block was designed by introducing the building blocks PyramidNet [33] in the ResNet-44 architecture [34]. The third section consists of a global average-pooling layer followed by a fully-connected layer. Finally, as an important remark, when applying the Entropy-Constrained Trained Ternarization (EC2T) approach, the first and last layers are not quantized.

Table A.1. Architecture of MicroNet-C10 and MicroNet-C100 networks, where d and w are scaling factors for the networks' depth and width, respectively. For the baseline neworks (i.e., before applying compound-model-scaling), $d = w = 1$. The number of classes, $n_{classes}$, corresponds to 10 for CIFAR-10 and 100 for CIFAR-100.

Stage	**Operation**	**Resolution**	**Output Channels**	**Repetitions**
	Stem Convolution (3×3) + BN & ReLU	32×32	$16 \times w$	1
1	Building Block	32×32	$16 \times w$	$7 \times d$
2	Building Block	16×16	$32 \times w$	$7 \times d$
3	Building Block	8×8	$64 \times w$	$7 \times d$
	ReLU & Global Avg. Pooling	8×8	$64 \times w$	1
	Fully-Connected	1×1	$n_{classes}$	1

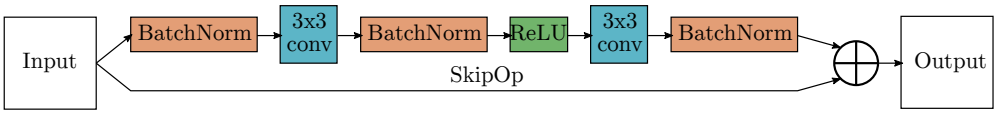

Figure A.1. Building block for the baseline models, MicroNet-C10 and MicroNet-C100.

B. Efficient Storage of Sparse & Ternary Weight Matrices

In addition to the trainable network parameters, we count those values that are needed to reconstruct the model from sparse matrix formats, i.e., binary masks or indices. Specifically, full-precision parameters (32-bits) count as one, while quantized parameters (with less than 32-bits) as a fraction of a parameter. For instance, a binary mask element counts as 1/32 with respect to a full-precision (32-bit) parameter.

If Compressed-Entropy-Row(CER)/Compressed-Sparse-Row (CSR) formats are not applied, a ternary convolution layer of size $\mathcal{N}K^2\mathcal{M}$ consists of two binary masks as illustrated in Figure B.1. One mask indicates the location of the centroid values (see Figure B.1b), while the other describes the sign of those values (see Figure B.1c). Thus, the parameter count for these masks is $1/32 \times \mathcal{N}K^2\mathcal{M}$ and $1/32 \times \sigma\mathcal{N}K^2\mathcal{M}$, respectively. In this notation, $\mathcal{N}$ is the number of effective input channels, K the kernel size, $\mathcal{M}$ the number of effective output channels, and $\sigma = 1 - \text{sparsity}$, with $\sigma \in [0, 1]$. The effective number of channels is computed as the original number of channels minus the number of channels pruned by the Entropy-Constrained Trained Ternarization (EC2T) approach. To calculate the layers' sparsity, we exclude the pruned channels. The third matrix in Figure B.1, uses two 16-bit numbers to represent the centroid values. Thus, they count as a single full-precision (32-bit) parameter (Figure B.1d). For the batch normalization layers, we add a 16-bit value (bias) per effective output channel. Therefore, their corresponding parameter count is $\mathcal{M}/2$.

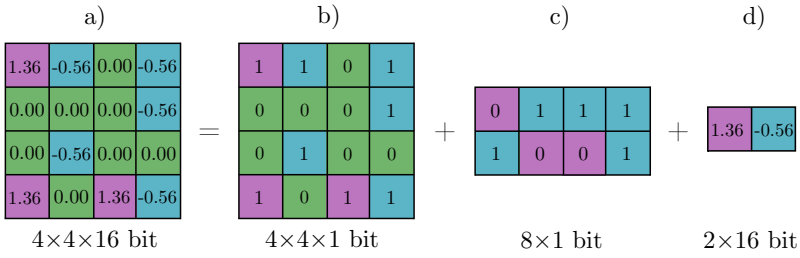

Figure B.1. Efficient storage of sparse and ternary weight matrices.