EAST: Encoding-Aware Sparse Training for Deep Memory Compression of ConvNets

EAST: Encoding-Aware Sparse Training for Deep Memory Compression of ConvNets

Matteo Grimaldi, Valentino Peluso, Andrea Calimera
Politecnico di Torino, 10129 Torino, Italy

***Abstract*—The implementation of Deep Convolutional Neural Networks (ConvNets) on tiny end-nodes with limited non-volatile memory space calls for smart compression strategies capable of shrinking the footprint yet preserving predictive accuracy. There exist a number of strategies for this purpose, from those that play with the topology of the model or the arithmetic precision, e.g. pruning and quantization, to those that operate a model agnostic compression, e.g. weight encoding. The tighter the memory constraint, the higher the probability that these techniques alone cannot meet the requirement, hence more awareness and cooperation across different optimizations become mandatory. This work addresses the issue by introducing EAST, *Encoding-Aware Sparse Training*, a novel memory-constrained training procedure that leads quantized ConvNets towards deep memory compression. EAST implements an adaptive group pruning designed to maximize the compression rate of the weight encoding scheme (the LZ4 algorithm in this work). If compared to existing methods, EAST meets the memory constraint with lower sparsity, hence ensuring higher accuracy. Results conducted on a state-of-the-art ConvNet (ResNet-9) deployed on a low-power microcontroller (ARM Cortex-M4) validate the proposal.**

I. Introduction & Motivations

The deployment of Convolutional Neural Networks (ConvNets) on the edge of the Internet-of-Things (IoT) is a challenge as tiny sensor nodes are powered by low-cost microcontroller units (MCU) with limited memory and computational resources. The lack of memory space is of particular relevance here. Indeed, even the most optimized ConvNets may require tens of MBs of pre-trained parameters, while for off-the-shelf MCUs the non-volatile memories range from 32 KB to 2 MB. Moreover, flash memories are also used to host other routines to drive the sensors and/or to pre-process the data, reducing, even more, the available space. Not less important, a ConvNet is trained to process a single specific task, while multi-sensor applications (e.g. [1]) have to process different sources of information, thus needing multiple ConvNets on board.

Several recent works investigated aggressive optimization techniques for memory compression. Among them, quantization via fixed-point representation is now considered a mandatory step. The use of 8-bit integers is a common choice for general-purpose MCUs [2] as it reduces the memory footprint up to 4× w.r.t. 32-bit floating-point with no, or negligible, accuracy loss. However, quantization alone might be not enough to fit stringent memory requirements. Sparse training via *weight pruning* [3] [4] is an additional strategy that can improve the compression if combined with some encoding scheme and/or when quantization is jointly applied [5] [6].

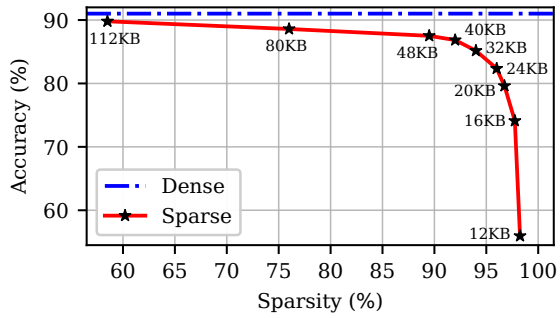


Figure 1: Sparsity vs. Accuracy of a compressed 9-layer ResNet under different memory constraints (the labeled numbers). The net is trained on CIFAR-10, then compressed via weight pruning and encoding. The blue dash-dotted line marks the accuracy of the original dense version (140 KB).

Sparse trained and quantized ConvNets get easier to be compressed by lossless encoding schemes as their weight tensors have long chains of zeros or repeated values.

As a rule of thumb, the higher the sparsity, the larger the compression rate. However, under stringent constraints, i.e. a few tens of KBs, the level of sparsity which will let encoding schemes meet the constraint is extremely high, much higher than what ConvNets may tolerate. Fig. 1 illustrates such an important aspect for a 9-layer ResNet trained on CIFAR-10. Above 90% of sparsity, the value needed to achieve a memory footprint ≤40KB, the accuracy curve shows a sudden drop.

This poses a new challenge: *Is there a way to achieve higher compression rates with lower sparsity to preserve accuracy?* Yes indeed, and this work aims at providing a first viable optimization strategy named *EAST* (Encoding-Aware Sparse Training). The rationale is simple, yet effective: find groups of adjacent weights to be pruned rather than pruning single connections. EAST implements a sparse training procedure based on *adaptive group pruning* where the group size *adapts* to the memory constraint minimizing the amount of sparsity needed. EAST operates the *LZ4* [7] encoding scheme, which (i) can be implemented with a lightweight routine of few bytes of memory and (ii) guarantees fast decompression and negligible impact on the inference latency; other encoding schemes can be seamlessly applied. The validation is conducted on a state-of-the-art 9-layer ResNet trained on the CIFAR-10 dataset and ported on an Arm Cortex-M4 MCU. A comparison against the same network compressed with a classical weight pruning and encoded with the same LZ4 scheme proves EAST can achieve higher accuracy (up to 8.73%) when the available memory is very limited (12KB of flash).

II. Related Works

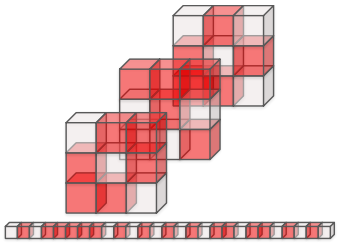
Weight pruning is a common technique to generate sparse ConvNets. During training, less important connections are gradually removed until a target level of compression is reached. The weight magnitude is the most popular proxy to drive weight selection, with the intuition that collapsing low-value weights to zero affects the prediction accuracy less. Pioneering works in this field [3] demonstrated that most of the parameters can be removed (up to 90%), still keeping the accuracy of the dense model. Combined with aggressive quantization (down to 2 bits) and weight encoding, sparse ConvNets may achieve impressive compression ratios, up to 49× depending on the network topology [5]. Also, other recent works investigated the bond between weight pruning and quantization [6] [8] [9], suggesting a joint optimization to determine the best combination of pruning rate and bit-width. While those kinds of strategies are effective with arithmetic units supporting arbitrary bit-widths, general-purpose cores (the focus of this work) provide a limited instruction-set which restricts the choice of the bit-width to few options, i.e., 8-bit for the Cortex-M core used in this work.

The authors of [4] showed that *large-sparse* models outperforms *small-dense* models. However, above a certain threshold of sparsity (>90%), the ConvNet may experience dramatic accuracy degradation. This work addresses this issue with an encoding-aware pruning that meets the same memory constraint with fewer weights pruned.

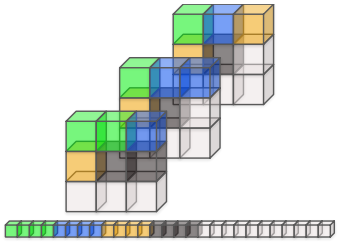
An interesting study conducted in [10] empirically demonstrated that a ConvNet contains an iso-accuracy sub-network that can be discovered via sparse training. This sub-network may represent the smallest achievable implementation that guarantees the highest accuracy, therefore it could be the best starting point for any encoding scheme. However, how to find it under extremely high levels of sparsity still remains an open issue. This work explores a complementary strategy, namely, to maximize the efficiency of the encoding scheme forcing a certain degree of proximity for zeroed weights.

Also, other works experimented pruning at higher-granularity, namely groups of adjacent weights. An example is [11], in which the group size is fixed to match the parallelism of single-instruction multiple-data (SIMD) units to reduce the inference latency. Rather than improving performance, in this work, we aim at decreasing the memory footprint. Specifically, we adopt group pruning to accelerate the compression rate. As a distinctive feature, both the group size and the physical place of the pruned groups adapt during training according to the target memory. Moreover, our strategy is effective also for cores without a SIMD unit (e.g. Cortex-M0 and M3).

Sparse networks can be compressed using different encoding techniques, like *Huffman Coding* [5], *Compressed Sparse Column* [12], *Zero Run Length* [13], to name the most known. In this work we focus on the *LZ4* algorithm which proved faster than others during decompression (in the order of several GB/s on high-end CPUs). For LZ4 to be effective, and in general for any compression algorithm, to have big chunks of adjacent *zeros* is paramount. That is why an efficient compression strategy should take the sparsity distribution (and not just its absolute value) as a first order variable.



(a) weight pruning



(b) group pruning ($GS = 4$)

Figure 2: Weight Pruning (a) vs Group Pruning (b). Colored weights denotes zero-values

III. Methodology

A. Flow Overview

The optimization flow encompasses three stages: *(i) sparse training*, *(ii) quantization*, and *(iii) encoding*. The first, *Sparse training*, is to train the sparse network under a user-defined memory constraint. Then, *Quantization* reduces the arithmetic precision of the model to a given bit-width (8-bit in this work). Finally, *Encoding* compresses the model size leveraging favorable patterns made available through the sparse training. The EAST strategy implements the first stage.

B. EAST

Encoding-Aware Pruning. As already introduced in Section I, accuracy-driven weight pruning algorithms return tensors with sequences of zeros much longer than in the original dense model. Although this helps to increase the compression rate of the encoding scheme, there is no direct control on the position of the zeros, which is ruled by accuracy. The EAST technique is based on the assumption that a weight pruning which is aware of the encoding scheme could make better use of sparsity. A pictorial view of this principle is given in Fig. 2, which illustrates how multi-dimensional tensors are transformed into a 1-D array that can be processed by standard general-purpose cores. Fig. (a) shows a standard weight pruning, while Fig. (b) is for a group pruning with group size (GS) of 4. In both cases, the picture refers to a *channel-last* layout organization (the same scheme used by the inference library adopted in this work). The colored items represent the pruned weights. While for the standard method the pruned weights are often placed far away as the selection is just accuracy-driven (smaller weights pruned first), with a

group pruning the proximity of the pruned weights is forced by the size of the group itself. This brings to clear advantages. Indeed, even if both cases show the same sparsity (59%), group pruning gets 55% higher compression ratio (using LZ4, see Sec. III-C). The savings get larger when considering tensors of higher dimensionality.

It is thereby intuitive that the group size serves as a control knob to reach the best trade-off between accuracy, sparsity, and compression rate. When the available memory is extremely small, groups of larger size may help to achieve higher compression with lower sparsity, hence preserving accuracy more. With a too-small group size, e.g. 1 as for standard weight pruning, the amount of sparsity needed by the encoding algorithm to meet the memory constraint would be too large, with negative effects on the accuracy. The EAST strategy implements a memory-driven adaptive group sizing during the sparse training procedure.

Sparse Training. In EAST, both sparsity and group size are gradually increased during the training loop until the memory constraint is met. In the beginning, the sparsity is low and the group size is set to one, hence EAST behaves like a standard weight pruning. If the memory constraint is not satisfied, sparsity and group size are updated following a pre-defined schedule. The sparse training iterates for a new bunch of epochs and if the memory constraint is still not met, sparsity and group size are newly updated. The larger the group size, the faster the memory reduction. Therefore, group pruning helps to converge faster attaining the target memory with a lower sparsity. The group selection is driven by the ℓ_2-norm: groups with lower norm are removed first. However, they can be restored later during the training steps that follow. Once the target memory is met, the sparsity and group size updates are stopped, the pruned weights are frozen, and the training iterates for the last set of epochs adjusting the remaining weights in order to maximize accuracy.

Hyperparamters. Group pruning is applied at the end of each epoch, namely after a complete iteration over the entire training set. The initial target sparsity is 30% with an increased step of 1% every epoch; the step is halved at epochs 20 and 50. The initial group size is set to one; starting from epoch 20, it increases with a step of 1 every 10 epochs.

C. Quantization & Encoding

After the sparse training, the 32-bit floating-point ConvNet is quantized to 8-bit. The effect of the quantization is (i) to reduce the memory footprint ensuring marginal accuracy loss, (ii) to increase the frequency of repeated weights, (iii) to accelerate the inference time. We opted for a binary-point quantization scheme which is fully compliant with the inference library used for on-board deployment (CMSIS-NN [2]), therefore tailored for the target MCU (the Cortex-M by ARM). As the very last stage, the quantized model is compressed. EAST can operate different encoding algorithms, but we found the LZ4 algorithm is a good choice for resource-constrained MCUs due to its lightweight routine that ensures high decoding speed. On-board measurements validated this qualitative analysis. The implemented compression strategy is layer-wise, namely, layers are compressed as separate blocks. This solution allows more efficient management of the available SRAM as it avoids one-shot full model decoding. In fact, layers are processed in sequence during inference, therefore each layer block can be decoded independently and temporarily stored in the SRAM using time-shared buffers.

IV. Experimental Results

A. Benchmarks, Datasets, and Training

We used as benchmark a 9-layer ResNet [14] (ResNet-9) for image classification on the CIFAR-10 dataset. ResNet-9 currently holds the first position in the DawnBench Competition [15]. In our implementation, we removed 75% of the filter from each convolutional layer. As it is already optimized for fast training and inference, this ConvNet represents a challenging test-case to assess the efficiency of different compression pipelines.

The dataset is split in training (45K images), validation (5K) and test (10K) set. The model with the highest accuracy on the validation set is selected for evaluation. For data augmentation, we applied padding with random crop, random horizontal flip, and cutout. The same setting is used for both dense and sparse training. The training is driven by SGD for 200 epochs with batch-size 128. The learning rate follows a cosine annealing schedule with an initial value of 0.1. All the experiments have been run in Pytorch 1.2.

For what concerns quantization, the fixed-point position is determined by a heuristic that minimizes the mean squared error between the floating-point and the 8-bit values. For the intermediate activations, the statistics have been collected on a sub-set of the validation set (size 100 samples).

Table I reports the top-1 accuracy on the test set and the memory size of the network. The reported values refer to a standard training (i.e. EAST off). Results confirm the efficiency of quantization (column *Q8*) that gets $4\times$ memory reduction with negligible accuracy losses (0.09%) w.r.t. the floating-point ConvNet (column *FP32*). Applying the LZ4 compression to the quantized model does not show significant savings: just a few bytes of memory reduction (column *Q8+LZ4*).

Table I: Top-1 accuracy on CIFAR-10 and weight memory of the dense ResNet-9 after 32-bit floating-point training (FP32), after quantization (Q8), and after LZ4 compression (Q8+LZ4).

	FP32	**Q8**	**Q8+LZ4**
Top-1	91.10%	91.01%	91.01%
Memory	558 KB	140 KB	140 KB

B. Results

EAST opens the deep memory space. Table II reports the comparison between a standard sparse training via weight pruning (WP) and the proposed flow built upon EAST. The two are compared for different target memories (M_t). The WP is trained using the same sparsity schedule of EAST (see Sec. III-B). For each M_t, the table collects the compression

Table II: Sparsity (S) and Top-1 Accuracy (A) of weight pruning (WP) and EAST on ResNet-9 at different memory constraint M_t (KB). CR is the compression ratio w.r.t. the floating-point ConvNet.

M_t	CR	S_{WP}	S_{EAST}	A_{WP}	A_{EAST}	ΔA
112	5.0×	58.5%	49.5%	89.80%	89.46%	-0.34%
80	7.0×	76.0%	60.5%	88.67%	88.61%	-0.06%
48	11.6×	89.5%	74.8%	87.51%	87.44%	-0.07%
40	14.0×	92.0%	79.0%	86.80%	86.82%	**0.02%**
32	17.4×	94.0%	83.3%	85.30%	86.11%	**0.81%**
24	23.3×	96.0%	87.8%	82.33%	83.65%	**1.32%**
20	27.9×	96.8%	90.0%	79.63%	81.11%	**1.48%**
16	34.9×	97.5%	91.8%	74.16%	78.45%	**4.29%**
12	46.5×	98.3%	94.0%	55.59%	64.32%	**8.73%**

ratio (CR) achieved after quantization and encoding, the sparsity reached after training (columns S_{WP} and S_{EAST}), the top-1 accuracy measured on the test-set (A_{WP} and A_{EAST}) and the relative accuracy distance (ΔA) between EAST and WP. As demonstrated by previous works, when the memory constraint is met with low sparsity, weight pruning guarantees marginal accuracy losses. For instance, at $M_t = 112\,\text{KB}$ the accuracy loss is only 1.21% lower than the dense 8-bit ConvNet (89.80% vs 91.01%). In this region of memory, EAST reaches similar accuracy levels than weight pruning, 0.34% lower in the worst case ($M_t = 112\,\text{KB}$). However, in the deep memory space ($M_t \leq 40\,\text{KB}$) weight pruning starts experiencing dramatic accuracy degradation. The reason is that very high sparsity ($> 90\%$) is needed to reach the desired memory constraint, therefore the model loses its expressive power as only a few weights remain up. In this region EAST outperforms WP; the encoding-aware pruning enables better control of the sparsity indeed ($S_{EAST} < S_{WP}$), preserving the same amount of information within the same amount of memory. On the extreme corner, $M_t = 12\,\text{KB}$, EAST is 8.73% more accurate than WP due to a lower sparsity (94% vs 98.3%). To emphasize the role of EAST, one can consider that with the same amount of sparsity (e.g. 94%) the model optimized with EAST is 2.7× smaller (row 12 KB vs 32 KB).

EAST accelerates the memory compression. Fig. 3 shows the evolution of the memory footprint during the training epochs for both WP (blue line) and EAST (red line) under the same memory constraint $M_t = 32\,\text{KB}$. During the first 20 epochs, when the group size is one (as set by the training schedule, see Sec. III-B), EAST follows the same trend of WP. Every time the group size gets increased (events indicated with black dots), the memory compression accelerates quickly. As a result, EAST reaches the target memory (indicated with the dashed black line) 43 epochs sooner than WP. These findings suggest that the group size works as an effective knob to boost the compression rate without seeking additional sparsity.

Efficient deployment of sparse ConvNets. We validated the optimization flow on a STM32 NUCLEO-F412ZG [16] board powered with an Arm Cortex-M4 core running at 100 MHz, 1 MB of flash memory, and 512 KB of SRAM. As the inference engine, we adopted the CMSIS-NN library. The original dense ConvNet takes 28 KB of SRAM to store intermediate activations and classifies a single image in 492 ms. The sparse ConvNets needs 884 B of flash for the LZ4 routine, which

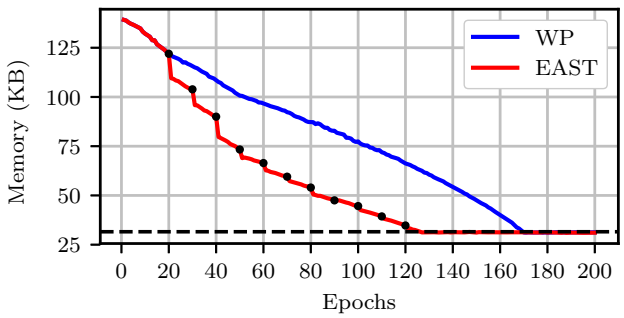


Figure 3: Epochs vs. Memory in weight pruning (blue line) and EAST (red line) for $M_t = 32\,\text{KB}$ (dashed line). The dots indicates when the group size increases.

thereby has a negligible impact on the compression rates achieved. Furthermore, an additional SRAM buffer of 36 KB is needed to store the decompressed weights. Since this buffer is time-shared among different layers, its size is given by the biggest layer. However, the buffer can be dynamically allocated just before the execution of the ConvNet.

The total execution time is function of the memory constraint M_t: the larger the M_t, the longer the decompression stage. For ResNet-9 generated with EAST, the execution time ranges from 482 ms at $M_t = 12\,\text{KB}$ to 497 ms at $M_t = 112\,\text{KB}$. At the lowest memory, the decompression only accounts for 6 ms; in all cases, the network layers execute faster than the dense counterpart as the weights resides in the SRAM instead of flash.

C. Final Remark

This work opens new paths towards the optimization of ConvNets in memory-bounded cores. EAST is particularly suited for the deep memory space, where it outperforms state-of-the-art sparse training. Nevertheless, further investigation is needed to bridge the accuracy gap with dense nets at extreme constraints. First, we plan to consider other proxies than the ℓ_2-norm to drive the group selection. Second, group size and sparsity follow a relative straightforward scheduling during the training; in order to achieve better trade-offs between sparsity, group size, and the position of the pruned groups, future works will explore the adoption of smarter hyper-parameter tuning techniques (e.g. Bayesian optimization or reinforcement learning) that might help EAST to reach global optima in the sparsity-memory-accuracy space.

V. Conclusion

EAST is a novel strategy for the training of memory-bounded sparse ConvNets. Leveraging the working principle upon which the encoding algorithms are built, it trains sparse networks that are more amenable to compression, yet less sparse and thus more accurate. The collected results are promising; EAST reaches up to 46.5× compression w.r.t. the original floating-point model achieving 8.73% higher accuracy than state-of-the-art sparse training. Also, we presented an efficient per-layer compression strategy, which exploits the proposed sparse training on commercial off-the-shelf IoT cores.

REFERENCES

[1] L. Seidenari, C. Baecchi, T. Uricchio, A. Ferracani, M. Bertini, and A. D. Bimbo, "Deep artwork detection and retrieval for automatic context-aware audio guides," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 13, no. 3s, p. 35, 2017.

[2] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," *arXiv preprint arXiv:1801.06601*, 2018.

[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*, 2015, pp. 1135–1143.

[4] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.

[5] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th International Conference on Learning Representations, ICLR 2016*, 2016.

[6] F. Tung and G. Mori, "Clip-q: Deep network compression learning by in-parallel pruning-quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7873–7882.

[7] Lz4. [Online]. Available: https://lz4.github.io/lz4/

[8] M. Grimaldi, V. Tenace, and A. Calimera, "Layer-wise compressive training for convolutional neural networks," *Future Internet*, vol. 11, no. 1, p. 7, 2019.

[9] M. Grimaldi, V. Peluso, and A. Calimera, "Optimality assessment of memory-bounded convnets deployed on resource-constrained risc cores," *IEEE Access*, vol. 7, pp. 152 599–152 611, 2019.

[10] J. Frankle, G. K. Dziugaite, D. M. Roy, and M. Carbin, "The lottery ticket hypothesis at scale," *arXiv preprint arXiv:1903.01611*, 2019.

[11] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing dnn pruning to the underlying hardware parallelism," in *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2. ACM, 2017, pp. 548–560.

[12] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: efficient inference engine on compressed deep neural network," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 243–254.

[13] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 27–40.

[14] Torchskeleton. [Online]. Available: https://github.com/wbaek/torchskeleton

[15] C. Coleman, D. Kang, D. Narayanan, L. Nardi, T. Zhao, J. Zhang, P. Bailis, K. Olukotun, C. Ré, and M. Zaharia, "Analysis of dawnbench, a time-to-accuracy machine learning performance benchmark," *ACM SIGOPS Operating Systems Review*, vol. 53, no. 1, pp. 14–25, 2019.

[16] Nucleo-f412zg. [Online]. Available: https://www.st.com/en/evaluation-tools/nucleo-f412zg.html