HEMP: High-order entropy minimization for neural network compression

HEMP: High-order Entropy Minimization for neural network comPression

Enzo Tartaglione[a,1], Stéphane Lathuilière[b], Attilio Fiandrotti[a,b], Marco Cagnazzo[b], Marco Grangetto[a]

[a] *University of Torino, Torino, Italy*
[b] *Télécom Paris, Paris, France*

Abstract

We formulate the entropy of a quantized artificial neural network as a differentiable function that can be plugged as a regularization term into the cost function minimized by gradient descent. Our formulation scales efficiently beyond the first order and is agnostic of the quantization scheme. The network can then be trained to minimize the entropy of the quantized parameters, so that they can be optimally compressed via entropy coding. We experiment with our entropy formulation at quantizing and compressing well-known network architectures over multiple datasets. Our approach compares favorably over similar methods, enjoying the benefits of higher order entropy estimate, showing *flexibility* towards non-uniform quantization (we use Lloyd-max quantization), *scalability* towards any entropy order to be minimized and *efficiency* in terms of compression. We show that HEMP is able to work in synergy with other approaches aiming at pruning or quantizing the model itself, delivering significant benefits in terms of storage size compressibility without harming the model's performance.

Keywords: deep learning, compression, entropy, neural networks, regularization

1. Introduction

Artificial Neural Networks (ANNs) achieve state-of-the-art performance in several tasks via complex architectures with millions of parameters. Deploying such architectures over resource-constrained devices such as mobiles or autonomous vehicles entails tackling a number of practical issues. Such issues include tight bandwidth and storage caps for delivering and memorizing the trained networks and limited memory for its deployment. Let us assume a neural network has to be deployed to a device such as a smartphone or an autonomous car over a wireless link. Downloading the network may exhaust the subscriber's traffic plan, plus the downloaded network will take storage on the device that will be unavailable to other applications. In an autonomous driving context, safety-critical updates may be delayed due to the limited bandwidth available

*Corresponding author
Email address: enzo.tartaglione@unito.it (Enzo Tartaglione)

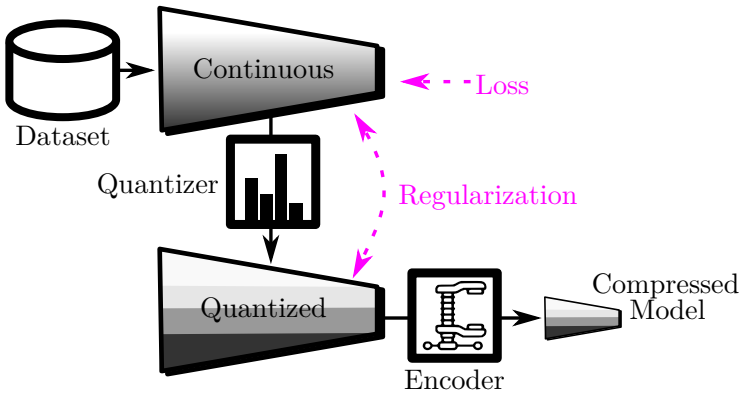


Figure 1: Proposed approach for neural network compression. At training time, we employ two parametrizations of the same neural network: continuous parameters are used for loss minimization, while quantized parameters are used for high-order entropy estimation. A regularization term enforces consistency between the neuron activations of the networks and a low entropy of the quantized network. The final model is obtained using any entropy-based encoder.

over the wireless channel [1]. Such examples show the importance of efficiently compressing neural networks for transmission and storage purposes.

Multiple approaches have been proposed to compress neural networks. A first approach consists is designing the network topology from the ground up to encompass fewer parameters [2, 3]. Needless to say, this approach requires designing novel topologies from scratch. A second approach consists in pruning some parameters from the network, i.e. removing some connections between neurons [4, 5, 6], yielding a sparse topology. Pruning might reduce the memory footprint [7, 8, 9], however it does not necessarily minimize storage or bandwidth requirements. A third approach consists in quantizing the network parameters [10, 11, 12], possibly followed by entropy-coding the quantized parameters. Similar approaches achieve promising results, however most quantization schemes just aim at learning a compressible representation of the parameters [12, 13, 8] rather than properly minimizing the compressed parameters entropy. Indeed, the entropy of the quantized parameters is not differentiable and cannot be easily minimized in standard gradient descent-based frameworks. In this work, we tackle the problem of compressing a neural network by minimizing the entropy of the compressed parameters at learning time. Enhancing model's compressibility we are able to reduce required bandwidth for bit streaming as well as storage required. Deep models are redundant [14, 15]: hence, there is a overhead in the deep model's representation, which can be hereby compressed.

This work introduces HEMP, a method that relies on high-order entropy minimization to allow for efficiently compression of the parameters of a neural network. The proposed method is illustrated in Fig. 1. The main contribution of HEMP is the differentiable formulation of the quantized parameters' entropy, which can be extended beyond first-order with finite computational and memory complexity. Namely, HEMP relies on a twin parametrization of the neural network: continuous parameters and the corresponding quantized parameters, where the entropy of the latters is estimated from

the entropy of the former. We design a regularization term around our entropy formulation that can be plugged into gradient descent frameworks to train a network to minimize the entropy of the quantized parameters No assumptions are made on the quantization scheme (including non-uniform quantization), nor entropy coding scheme, that are not part of the proposed method and towards which our method is totally agnostic. Other techniques like ℓ_1 norm or rank-based ones aim at removing parameters: this has a different effect on the distribution of the parameters. Indeed, while these other approaches maximize the frequency of the pruned parameters, which are still encoded with zeros, HEMP is more general as it is able to enhance the compressibility of any quantized representation for the values.

We experiment with different quantization and entropy coding scheme showing that training a network to minimize the 2-nd order entropy of the quantized parameters is already sufficient to outperform state-of-the-art competing schemes.

The rest of the paper is organized as follows. Sec. 2 reviews state-of-the-art approaches in network compression, Sec. 3 introduces the proposed high-order entropy regularizer, and the overall training procedure is described in Sec. 4. Experimental results are discussed in Sec. 5 and finally, in Sec. 6 conclusions are drawn.

2. Related works

A lot of work has been done around neural network size reduction. In general, we can group them into three large categories, according to their primary goal.

Minimizing the architecture. Focusing from the architectural point of view, it is possible to design some memory-efficient deep networks, typically relying on strategies like channel shuffling, point-wise convolutional filters, weight sharing or a combination of them. Some examples of customized deep networks towards memory footprint reduction are SqueezeNet [2], ShuffleNet [16] and MobileNet-v2 [3]. Recently, a huge interest in automatically reducing the shape of the deep networks has gained interest, with works on neural network sparsification [4, 5, 6, 17] boosted by the recent lottery ticket hypothesis by Frankle and Carbin [18]. These approaches address the problem of improving inference efficiency with limited memory footprint, but do not directly tackle the problem of reducing stored model size.

Minimizing the computation. Recently, this topic is collecting ever-increasing interest. While deepening its roots in statistical physics, some works exploited low-precision training in artificial neural networks [7, 19, 20]. A large number of works attempts to also use low-precision back-propagation signals and low-precision activations, as it leads to lower power consumption at inference time [21, 9, 8]. These techniques, however, do not explicitly address the problem of minimizing the storage size of the entire model.

Minimizing the stored model's memory. Here the main goal is not to modify the architecture of a deep model, but to merely compress it, to reduce its stored size: while the other two approaches focused on somehow changing the architecture of the deep model to simplify it and/or to reduce its memory footprint, here the objective is to compress a stored model with no architectural change. Towards this end, many approaches have been proposed: context-adaptive binary arithmetic coding [12], learning the quantized parameters using the local reparametrization trick [22], cluster similar parameters between different layers [11], using matrix factorization followed by Tucker decomposition [10], training adversarial neural networks towards compression [23] or employing a Huffman encoding scheme [24] are just some of them. Recently, Oktay *et al.* proposed an entropy penalized reparametrizations to the parameters of a deep model, which leads to competitive compression values sacrificing a little bit the deep model's performance [13]. However, their approach has some training overhead, like the fact they require to train a decoder, they make their formulation differentiable through the use of straight-through estimators (STE). The big advantage provided by their approach relies more in the re-parametrization leading to the quantization strategy, but the compressibility of their quantized parameters is limited to arithmetic coding.
Deep learning based compression schemes proposing a direct high entropy-based regularizer are difficult to design because of the non-differentiability of the entropy and its computational heaviness. All the discussed methods do not explicitly minimize the final compressed file size but they are limited to rigid quantization and compression schemes [12] or they build dictionaries on-purpose [24], losing generality. In the next section, we introduce our efficient and differentiable n-th order entropy proxy, to be used in the HEMP framework: it can be freely associated to any quantization strategy and any entropic compression algorithm. Differently to the work by Wiedemann *et al.* [12], HEMP is not bound to a particular quantization scheme, and provides a direct, scalable and differentiable entropy estimator on the continuous parameters.

3. Entropy-based regularization

In this section, we describe our entropy-based framework for quantization. We introduce a regularization formulation that uses a differentiable entropy proxy, evaluated on the continuous parameters of the model, to indirectly reduce the compressed size of the quantized network. We will show that this term easily scales-up to any entropy orders, thus improving the compression efficiency of actual algorithms such as dictionary-based compression.

3.1. Preliminaries

Here, we introduce preliminaries and notations. Let a feed-forward, multi-layer artificial neural network be composed of L layers. Let $w_{l,i} \in \mathbb{R}$ be the i-th parameter of the l-th layer. Let us assume all ANN parameters are quantized onto N discrete levels, with:

- quantization index $q_{l,i} \in [1, N]$ for every parameter $w_{l,i}$;
- reconstruction (or representation) levels $r_l(k),\ k = 1 \dots N$; as shown in the following, every layer of the ANN model gets its own optimized set of reconstruction levels.

Table 1: Overview on the notation used in this work.

Symbol	Meaning
$w_{l,i}$	i-th (continuous) parameter in the l-th layer
$\widehat{w}_{l,i}$	i-th (quantized) parameter in the l-th layer
$q_{l,i}$	quantization index corresponding to the i-th parameter in the l-th layer
N	quantization levels
ξ	generic quantization index in range $[1; N]$
$p(\widehat{w}_{l,i} \to \xi)$	probability that the quantized representation of $w_{l,i}$ will have ξ as quantization index
$\widehat{H}_n$	n-th order entropy on the quantization indices
H_n	differentiable proxy of $\widehat{H}_n$ proposed within HEMP

From these, we get the quantized parameters $\widehat{w}_{l,i}$ according to

$$\widehat{w}_{l,i} = r_l(q_{l,i}). \tag{1}$$

Table 1 collects the most recurring symbols of this section. Please notice that multi-dimensional versions of the symbols are in bold.
Now, let us consider $\widehat{\mathbf{w}}^n$ as the n-uples of the quantized parameters, where $\widehat{\mathbf{w}}^n_j$ is the j-th n-uple of quantized parameters. In general, the n-th order entropy on the quantized model is

$$\widehat{H}_n = -\sum_{\boldsymbol{\xi}} p(\widehat{\mathbf{w}}^n \to \boldsymbol{\xi}) \log_2 p(\widehat{\mathbf{w}}^n \to \boldsymbol{\xi}), \tag{2}$$

where $\|W\|_0$ (L0-norm) is the total number of parameters, $\boldsymbol{\xi} \in [1; N]^n$ and, using the chain rule, we can express $p(\widehat{\mathbf{w}}^n \to \boldsymbol{\xi})$ as

$$p(\widehat{\mathbf{w}}^n \to \boldsymbol{\xi}) = \frac{n}{\|W\|_0} \sum_j \prod_{m=1}^{n} p\left[\widehat{w}^n_{j,m} \to \xi_m \,\middle|\, \bigcap_{s=1}^{m-1} \left(\widehat{w}^n_{j,s} \to \xi_s\right)\right]. \tag{3}$$

In (3), the "probability" of the event $\widehat{w}^n_{j,m} \to \xi_m$ is

$$p(\widehat{w}^n_{j,m} \to \xi_m) = \mathbb{1}_{\xi_m}(q_{l,i}) \tag{4}$$

where $\mathbb{1}_{\xi}(\cdot)$ is the indicator function. Minimizing (2) results in maximizing the final compression for the quantized model when using an entropic compression algorithm ([25, 26]). Unfortunately, the problem in minimizing (2) within a gradient descent based optimization framework lies in the non-differentiability of (4). In the next section we introduce a differentiable proxy for (4) which directly optimizes the continuous parameters $w_{l,i}$ such that their quantization is highly compressible.

3.2. Differentiable n-th order entropy regularization

In the previous section we have stated the impossibility of directly optimizing (2) using gradient descent-based techniques because of the non-differentiability of (3). We

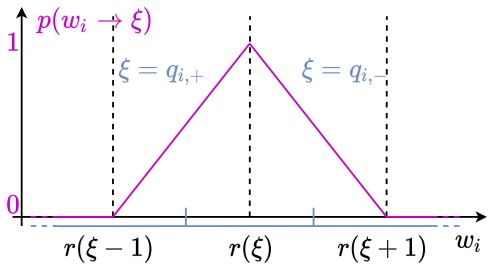


Figure 2: Visual representation of (7).

are going to overcome this obstacle providing a formulation of (3) based on the distance between the continuous parameter $w_{l,i}$ and its quantized reconstruction $\widehat{w}_{l,i}$. Here on, we will drop the subscript l, but in general all the layers have different reconstruction levels.
Let us define first the distance between a parameter w_i and the reconstruction level $r(\xi)$:

$$d[w_i, r(\xi)] = |w_i - r(\xi)| \tag{5}$$

From (5), we can estimate the probability of binning w_i to ξ using the softmax function:

$$p(w_i \to \xi) = \frac{e^{-d[w_i, r(\xi)]}}{\sum_j e^{-d[w_i, r(j)]}} \tag{6}$$

Such general formulation is computationally expensive, so we propose an efficient approximation thereof exploiting a "bin locality" principle, for which we say that a parameter w_i can be binned to the two closest bins only. Under the assumption of quasi-static process, indeed, locally the probability of binning the continuous parameter w_i in other bins than the two closest between two iteration steps can be neglected. We refer to these bins as $q_{i,-}$ and $q_{i,+}$. In this case, we know $w_i \in [r(q_{i,-}); r(q_{i,+})]$. Here we can design a relative distance linearly-scaling probability:

$$p(w_i \to \xi) = \begin{cases} 1 - \frac{w_i - r(q_{i,-})}{\Delta_i} & \xi = q_{i,-} \\ 1 - \frac{r(q_{i,+}) - w_i}{\Delta_i} & \xi = q_{i,+} \\ 0 & otherwise \end{cases} \tag{7}$$

where $\Delta_i = r(q_{i,+}) - r(q_{i,-})$. Figure 2 displays the behavior of (7). Hence, the binning probability in (7) scales as the relative distance from the center of the bin. If we combine (7) with (4) and, finally, with (2), we obtain

$$H_n = \frac{n}{\|W\|_0} \sum_{\boldsymbol{\xi}} \left\{ \left[\sum_j p(\mathbf{w}_j^n \to \boldsymbol{\xi}) \right] \cdot \right. \\ \left. \cdot \left[\log_2(\|W\|_0) - \log_2(n) - \log_2 \sum_j p(\mathbf{w}_j^n \to \boldsymbol{\xi}) \right] \right\} \tag{8}$$

where

$$p(\mathbf{w}_j^n \to \boldsymbol{\xi}) = \prod_{m=1}^{n} p\left[w_{j,m}^n \to \xi_m \middle| \bigcap_{s=1}^{m-1} \left(w_{j,s}^n \to \xi_s \right) \right] \tag{9}$$

3.3. Study of the entropy regularization term

In this section we are detailing the derivations of the proposed entropy regularization. Obtaining an explicit formulation for the update terms allows to efficiently implement the update rule explicitly (when using gradient-based optimizers, without relying on the automatic differentiation packages) and to study both the stationary points of the regularization term and bounds of the gradient respectively.

3.3.1. Explicit derivation of the entropy regularization term's gradient

Let us consider here the first order entropy proxy:

$$H_1 = -\sum_{\xi} p(\mathbf{w} \to \xi) \log_2 \left[p(\mathbf{w} \to \xi) \right] \tag{10}$$

with

$$p(\mathbf{w} \to \xi) = \frac{1}{\|W\|_0} \sum_i p(w_i \to \xi) \tag{11}$$

Let us differentiate (10) with respect to w_i:

$$\begin{aligned}
\frac{\partial H_1}{\partial w_i} &= -\frac{\partial}{\partial w_i} \left\{ \sum_{\xi} p(\mathbf{w} \to \xi) \log_2 \left[p(\mathbf{w} \to \xi) \right] \right\} \\
&= -\sum_{\xi} \left\{ \frac{\partial}{\partial w_i} \left[p(\mathbf{w} \to \xi) \right] \cdot \log_2 \left[p(\mathbf{w} \to \xi) \right] + p(\mathbf{w} \to \xi) \cdot \frac{\partial}{\partial w_i} \log_2 \left[p(\mathbf{w} \to \xi) \right] \right\} \\
&= -\left\{ \sum_{\xi} \frac{\partial}{\partial w_i} \left[p(\mathbf{w} \to \xi) \right] \cdot \log_2 \left[p(\mathbf{w} \to \xi) \right] + \right. \\
&\qquad \left. + \sum_{\xi} p(\mathbf{w} \to \xi) \cdot \frac{\partial}{\partial w_i} \log_2 \left[p(\mathbf{w} \to \xi) \right] \right\}.
\end{aligned} \tag{12}$$

According to (7), we can write

$$\begin{aligned}
\frac{\partial H_1}{\partial w_i} = -\left\{ \sum_{\xi} \frac{\partial}{\partial w_i} \left[p(\mathbf{w} \to \xi) \right] \cdot \log_2 \left[p(\mathbf{w} \to \xi) \right] + \right. \\
\left. + \sum_{\xi = \{q_{i,-}, q_{i,+}\}} p(\mathbf{w} \to \xi) \cdot \frac{\partial}{\partial w_i} \log_2 \left[p(\mathbf{w} \to \xi) \right] \right\},
\end{aligned} \tag{13}$$

and considering that

$$\frac{\partial}{\partial w_i} p(\mathbf{w} \to \xi) = \begin{cases} - \quad \frac{1}{\|W\|_0 \Delta_i} & \xi = q_{i,-} \\ \\ \quad \frac{1}{\|W\|_0 \Delta_i} & \xi = q_{i,+} \\ \\ \quad 0 & otherwise \end{cases} \tag{14}$$

where

$$\Delta_i = r(q_{i,+}) - r(q_{i,-}), \tag{15}$$

we have

$$\begin{aligned} \frac{\partial H_1}{\partial w_i} &= - \sum_{\xi = \{q_{i,-}, q_{i,+}\}} \left\{ \frac{\partial}{\partial w_i} \left[p(\mathbf{w} \to \xi) \right] \cdot \log_2 \left[p(\mathbf{w} \to \xi) \right] \right\} \\ &= \frac{1}{\|W\|_0 \Delta_i} \log_2 \frac{p(\mathbf{w} \to q_{i,-})}{p(\mathbf{w} \to q_{i,+})} \,. \end{aligned} \tag{16}$$

Using a similar approach, we can explicitly write the gradient for the n-th order entropy term in (8):

$$\frac{\partial H_n}{\partial w^n_{j,m}} = \frac{n}{\Delta^n_{j,m} \|W\|_0} \log_2 \left\{ \frac{\prod\limits_{\{\boldsymbol{\xi}\}_j,\, \xi_m = \xi^n_{j,m,-}} p(\mathbf{w}^n_j \to \boldsymbol{\xi})}{\prod\limits_{\{\boldsymbol{\xi}\}_j,\, \xi_m = \xi^n_{j,m,+}} p(\mathbf{w}^n_j \to \boldsymbol{\xi})} \right\} \tag{17}$$

where $\{\boldsymbol{\xi}\}_j$ indicates the set of $\boldsymbol{\xi}$ whose binning probability for $\mathbf{w}^n_j$ is non-zero.
Having (17) explicit enables efficient gradient computation: indeed, given the designer choice in (7), every n-uple of parameters has 2^n possible quantization indices n-uples $\{\boldsymbol{\xi}\}$ only, which is independent on the number of quantization levels N. On the contrary, using (6) would result in N^n possible quantization indices n-uples $\forall\ \mathbf{w}^n_j$. Hence, out proposed approach allows us to save $(N/2)^n\times$ memory at computation time.
For sake of simplicity, the following analysis on stationary points and boundaries will be performed on the first order entropy, but similar conclusions can be equivalently drawn for any n-th order.

3.3.2. Stationary points for H_1

In this section we are looking for H_1 stationary points (or in other words, when gradient vanishes). From (16) we observe that

$$\frac{\partial H_1}{\partial w_i} = 0 \Leftrightarrow p(\mathbf{w} \to q_{i,-}) = p(\mathbf{w} \to q_{i,+}), \tag{18}$$

assuming Δ_i, $\|W\|_0$ finite positive numbers. We can make w_i explicit in the condition (18):

$$[p(w_i \to q_{i,+}) - p(w_i \to q_{i,-})] = K_{i,+} - K_{i,-} \tag{19}$$

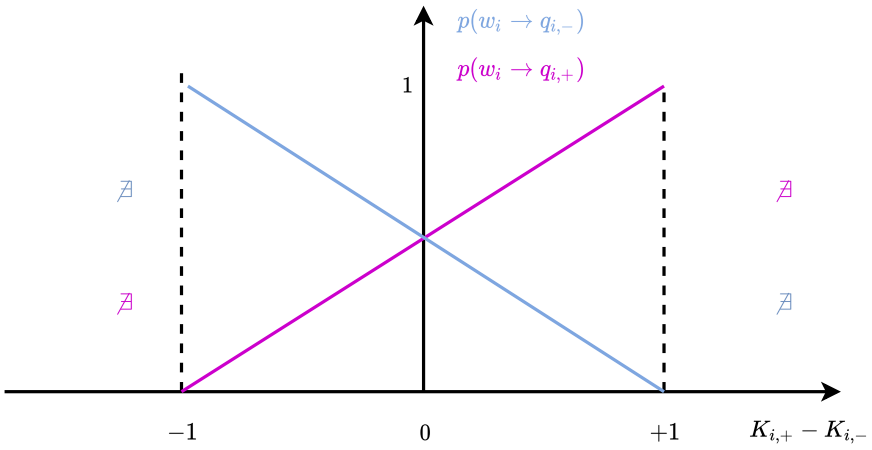


Figure 3: Gradient vanishing condition for both $p(w_i \rightarrow q_{i,-})$ (in cyan) and $p(w_i \rightarrow q_{i,+})$ (in violet).

where

$$K_{i,+} = \sum_{j \neq i} p(w_j \rightarrow q_{i,+})$$
$$K_{i,-} = \sum_{j \neq i} p(w_j \rightarrow q_{i,-}) \, .$$

According to (7), we can rewrite (19) as

$$p(w_i \rightarrow q_{i,+}) = \begin{cases} \frac{1}{2} \left(K_{i,+} - K_{i,-} + 1 \right) & if \ (K_{i,+} - K_{i,-}) \in [-1; +1] \\ \\ \nexists & otherwise \end{cases} \tag{20}$$

because $p(w_i \rightarrow q_{i,+}) \in [0; 1]$ by definition. As we expect, if $q_{i,+}$ and $q_{i,-}$ are evenly populated, $K_{i,+} = K_{i,-}$ and the stationary point of $\frac{\partial H_1}{\partial w_i}$ is

$$p(w_i \rightarrow q_{i,+}) = p(w_i \rightarrow q_{i,-}) = \frac{1}{2}$$

which results in

$$w_i = \frac{1}{2} \left[r(q_{i,+}) + r(q_{i,-}) \right] , \tag{21}$$

exactly between the centre of the two bins. From the entropy point of view, this is essentially what we expect, since we have two equi-populated bins; however, this is not what we like to have when we quantize a deep network, considering that it leads to an high quantization error. For this reason, favoring solutions in which $p(w_i \rightarrow \xi) \neq \frac{1}{2}$ is a good strategy and this is also the reason we included a reconstruction error in the overall regularization function.

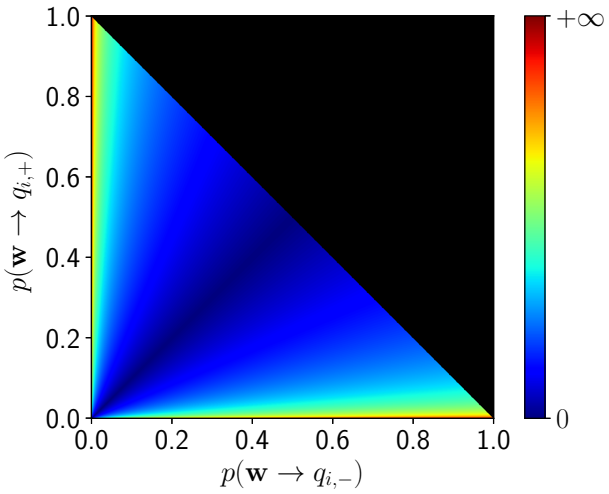


Figure 4: Plot of the absolute upper bound (22) for $\frac{\partial H_1}{\partial w_i}$ as a function of $p(\mathbf{w} \to q_{i,-})$ and $p(\mathbf{w} \to q_{i,+})$. In black we represent regions out of the considered domain.

3.3.3. Bound for $\frac{\partial H_1}{\partial w_i}$

In this section we are looking for an upper bound of $\frac{\partial H_1}{\partial w_i}$ and we study the cases in which such quantity explodes, in order to assess conditions to avoid gradient explosion. We can set the bound for the gradient magnitude as:

$$\frac{\partial H}{\partial w_i} \leq \left| \frac{1}{\Delta_i \|W\|_0} \log_2 \frac{p(\mathbf{w} \to q_{i,-})}{p(\mathbf{w} \to q_{i,+})} \right| . \tag{22}$$

Considering that $\|W\|_0 \in \mathbb{N}$ and that $\Delta_i > 0$ (so, both are finite, real-valued quantities), we are interested to guarantee

$$\frac{\partial H}{\partial w_i} \leq \frac{1}{\Delta_i \|W\|_0} \left| \log_2 \frac{p(\mathbf{w} \to q_{i,-})}{p(\mathbf{w} \to q_{i,+})} \right| < K \tag{23}$$

where K is a positive real-value finite number. Given $p(\mathbf{w} \to \xi) \in [0; 1]$, let us study the cases in which such quantity explodes.

- Case $\frac{p(\mathbf{w} \to q_{i,-})}{p(\mathbf{w} \to q_{i,+})} \to 0^+$. In this case we have $p(\mathbf{w} \to q_{i,-}) \to 0^+$ and $p(\mathbf{w} \to q_{i,+}) \neq 0$. According to (16) $w_i \in [r(q_{i,-}); r(q_{i,+}))$; so at least one parameter lies in the considered interval and the condition is impossible by construction.

- Case $\frac{p(\mathbf{w} \to q_{i,-})}{p(\mathbf{w} \to q_{i,+})} \to +\infty$. In this case we have $p(\mathbf{w} \to q_{i,+}) \to 0^+$ and $p(\mathbf{w} \to q_{i,-}) \neq 0$. Similarly to the previous case, $w_i \in [r(q_{i,-}); r(q_{i,+}))$; so at least one parameter lies in the considered interval and the condition is impossible by construction.

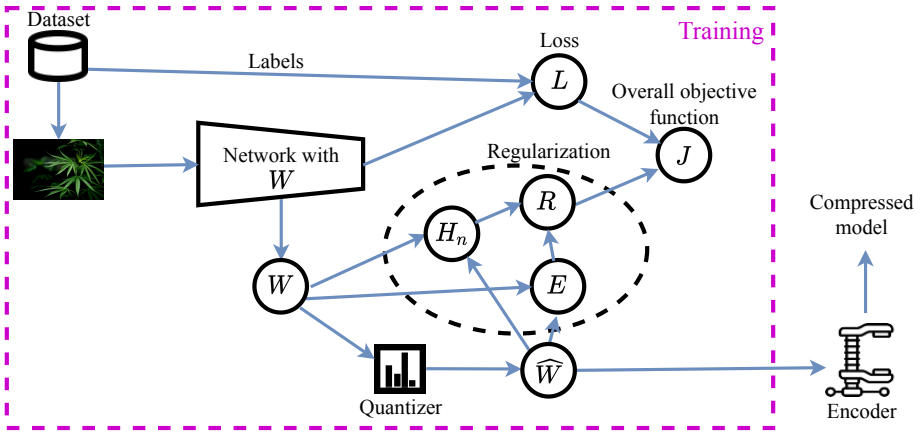


Figure 5: Schematic representation of HEMP.

- Case $\frac{p(\mathbf{w}\to q_{i,-})}{p(\mathbf{w}\to q_{i,+})} \to \frac{0^+}{0^+}$. By construction, $w_i \in [r(q_{i,-}); r(q_{i,+}))$ so this case is impossible.

In the next section we will describe the overall HEMP framework.

4. Training scheme

The overall training scheme is summarized in Fig. 5 and includes a quantizer and an entropy encoder. The quantizer generates the discrete-valued representation $\widehat{W}$ of the network parameters at training time. The encoder produces the final compressed file embedding the deep model once the training is over. Our scheme does not make any assumption about the quantization or entropy coding scheme, contrarily to other strategies tailored for, e.g., specific quantization schemes ([24, 12]). Therefore, in the following we will assume a very general, non-uniform Lloyd-max quantizer, while we will not make any assumption about the entropy encoder for the moment as it is external to the training process. Our learning problem can be formulated as follow: given a dataset and a network architecture, we want to compress the network parameters W, while preserving the network performance as measured by some loss function L. Towards this end, we introduce the following regularization function:

$$R = \lambda_H H_n + \lambda_E E \tag{24}$$

where λ_H and λ_E are two positive hyper-parameters and

$$E = \sqrt{\frac{1}{\|W\|_0} \sum_l \sum_i \left[w_{l,i} - r_l(q_{l,i})\right]^2} \tag{25}$$

is a reconstruction error estimator. Minimizing E makes $w_{l,i} \to \widehat{w}_{l,i}$ and, for instance, loss evaluation on the continuous parameters network approaches the loss estimated on

the quantized network. Overall, we minimize the objective function:

$$J = L + R \tag{26}$$

Minimizing J requires finding the right balance between L and R: towards this end, we propose to dynamically re-weight R according insensitivity of each parameter ([5]). The key idea here is to re-weight the regularization gradient $\frac{\partial R}{\partial w_{l,i}}$ at every parameter update depending on the sensitivity of the loss L with respect to every parameter. We say that the larger the magnitude of the gradient of the loss with respect to w_i, the smaller the perturbation from the minimization induced by R we desire. Hence, in the update of the parameter $w_{l,i}$, we re-weight the gradient of R by the insensitivity:

$$\bar{S}_{l,i} = 1 - \frac{\left|\frac{\partial L}{\partial w_{l,i}}\right|}{\max_j \left\{\left|\frac{\partial L}{\partial w_{l,j}}\right|\right\}} \tag{27}$$

The HEMP framework allows to solve the learning problem using standard optimization strategies, where the gradient of (26) is descended.

5. Experiments

In this section we evaluate the effectiveness of HEMP. Towards this end, we propose experiments on several widely used datasets with different architectures.

Datasets and architectures. We experiment with LeNet-5 on MNIST, ResNet-32 and MobileNet-v2 on CIFAR-10, ResNet-18 and ResNet-50 on ImageNet. We always train from scratch except for ImageNet experiments where we rely on pre-trained models.[1]

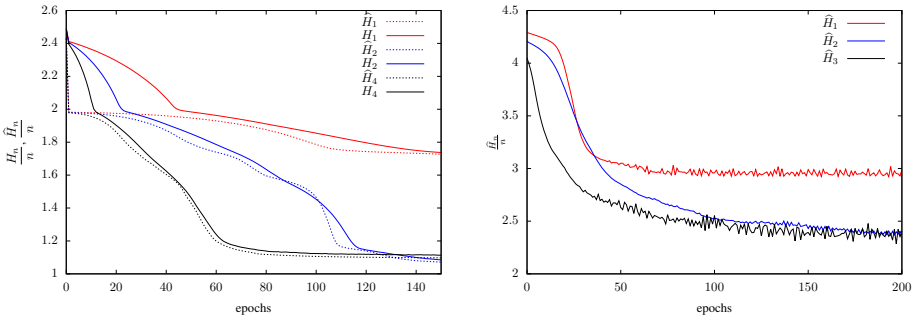
Setup. We experiment on a Nvidia RTX 2080 Ti GPU. Our algorithm is implemented using PyTorch 1.5.[2] For all our simulations we use SGD optimization with momentum 0.9, $\lambda_H = 1$ and $\lambda_E = 0.1$. Learning rate and batch-size depend on the dataset and the architecture: for all the datasets except for ImageNet the learning rate used is 10^{-2} and batch-size 100, for ResNet-18 trained on ImageNet the learning rate is 10^{-3} with batch-size 128 while for ResNet-50 learning rate is 10^{-4} with batch-size 32. The file containing the quantized parameters is entropy-coded using LZMA [27], a popular dictionary-based compression algorithm well-suited to exploit high-order entropy.

Metrics. The goal of the present work is to compress a neural network without jeopardizing its accuracy, so we rely on two distinct, largely used, performance metrics:

- the compressed model size as the size of the file containing the entropy-encoded network,
- the classification accuracy of the compressed network (indicated as Top-1 in the following).

[1] `https://pytorch.org/docs/stable/torchvision/models.html`

[2] The source code will be made available on GitHub upon acceptance of the work.



(a) LeNet-5 trained on MNIST

(b) ResNet-32 trained on CIFAR-10

Figure 6: Different entropy order minimization for LeNet-5 trained on MNIST (a) and on ResNet-32 trained on CIFAR-10 (b): in red first order is minimized, in blue the second one and in black the fourth (a) / third (b). Continuous lines represent the differentiable quantity introduced in (8) while dashed lines are the actual entropies directly computed on the quantized architecture (2).

5.1. Preliminary experiments

As preliminary experiments, we evaluate if the regularizer function (8) is a good estimator of (2). Towards this end, we train the LeNet-5 architecture on MNIST minimizing H_n while logging the entropy on the quantized parameters $\widehat{H}_n$. Fig. 6a shows the normalized $\widehat{H}_n$ and its approximation H_n: three findings are noteworthy.

First, H_n accurately estimates $\widehat{H}_n$, i.e. minimizing H_n yields to minimizing $\widehat{H}_n$. Under the assumption the quantized parameters are entropy-coded, minimizing $\widehat{H}_n$ shall minimize the size of a file where the encoded parameters are stored.

Second, when $n = 1$, the training converges to a higher entropy, while minimizing higher entropy orders enables access to lower entropy embeddings. Higher entropy reflects on the final size of the model: while for $n = 1$ we could get a final network size of 61kB, for $n = \{2,\ 4\}$ the final size drops to approximately 27.5kB, having a top-1 accuracy of 99.27%. This better performance can be explained by the fact that higher order entropy can catch repeated sequences of parameters' binnings which can lead to a significant compression boost.

Third, the higher n, the fewer the epochs required to converge to low entropy values. However, in terms of actual training time, the available GPU memory limits the parallelism degree for computing the derivative term in (17). In the following, we will stick to $n = 2$ as it enables both reasonably low entropy embeddings and training times.

As a further verification, we have run the same experiments on the ResNet-32 architecture trained on the CIFAR-10 dataset: also here, we minimize H_n while logging the entropy on the quantized parameters $\widehat{H}_n$ at different values of n. Fig. 6b shows the normalized $\widehat{H}_n$. Similarly to what observed in the main paper, second-order entropy minimization results to be a good trade-off between complexity and final performance, considering that the reached entropic rate of $n = 2$ is comparable to $n = 3$. Please notice also that the entropy estimated on the quantized model, and reported Fig. 6b, is proportional to the final file sizes.

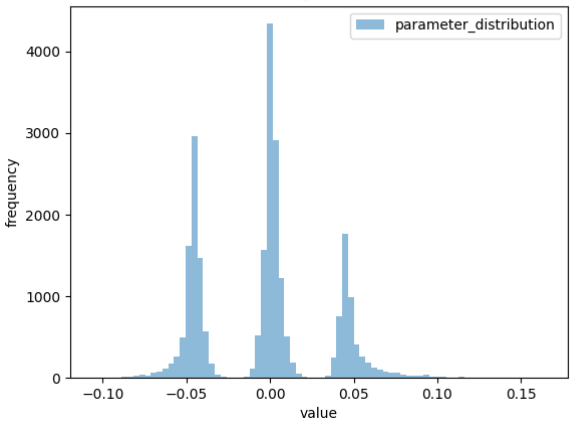


Figure 7: Typical distribution of w values during HEMP optimization for LeNet-5 trained on MNIST (second convolutional layer), with $N = 3$.

Table 2: Results on the MNIST dataset using LeNet-5 architecture.

Model	**Method**	**Top-1 [%]**	**Size**
LeNet-5	Baseline	99.30	1.7MB
	LOBSTER [28]	99.10	19kB
	Han *et al.* [24]	99.26	44kB
	Wiedemann *et al.* [12]	99.12	43.4kB
	HEMP	99.27	27.5kB
	Wiedemann *et al.*(+pruning) [12]	99.02	11.9kB
	HEMP+LOBSTER [28]	99.05	2.00kB

As a further analysis of HEMP's effect of the parameter, in Fig. 7 we show the distribution of the optimized parameters on the second convolutional layer in LeNet-5 trained on MNIST (the other layers follow a similar distribution). In this case we optimize the model having 3 quantized values. As we observe the continuous w values are distributed tightly around their quantized representations $\hat{w}$: as $w \to \hat{w}, L(w) \to L(\hat{w})$; in the end the accuracy of the quantized representation of the model approaches the accuracy of the continuous model. Additionally, as observed in Fig. 6a, also the entropy of the quantized model is minimized, achieving both a quantized trained model with high accuracy and high compressibility of its representation.

5.2. Comparison with the state-of-the-art

HEMP. We now compare our method with state-of-the-art methods for network compression. Our main goal is to minimize the size of the final compressed file while keeping the top-1 performance as close as possible to the baseline network's one. Therefore,

Table 3: Results on CIFAR-10 using different architectures.

Model	Method	Top-1 [%]	Size
ResNet-32	Baseline	93.10	1.9MB
	LOBSTER [28]	92.97	439.4kB
	HEMP	91.57	168.3kB
	HEMP+ LOBSTER [28]	92.55	86.2kB
MobileNet-v2	Baseline	93.67	9.4MB
	HEMP	92.80	872kB

Table 4: Results on ImageNet using different architectures.

Model	Method	Top-1 [%]	Size
ResNet-18	Baseline	69.76	46.8MB
	LOBSTER [28]	70.12	17.2MB
	Lin *et al.* [21]	68.30	5.6MB
	Shayer *et al.* [22]	63.50	2.9MB
	HEMP	68.80	3.6MB
	HEMP+LOBSTER [28]	69.70	2.5MB
ResNet-50	Baseline	76.13	102.5MB
	Wang *et al.* [29]	70.63	6.3MB
	Han *et al.*[24]	68.95	6.3MB
	Wiedemann *et al.* [12]	74.51	10.4MB
	Tung *et al.* [30]	73.7	6.7MB
	HEMP (high acc.)	74.52	9.1MB
	HEMP	71.33	5.5MB
MobileNet-v2	Baseline	72.1	13.5MB
	Tu *et al.*[31]	7.25	10.1MB
	He *et al.*[32]	9.8	4.95MB
	Tung *et al.*[30]	70.3	2.2MB
	HEMP	71.3	1.7MB
Custom, latency 6.11ms, energy 9.14mJ	APQ [33]	72.8	20.8MB
	APQ [33] + HEMP	72.5	3.04MB

our approach can be compared only with works that report the real final file size. To the best of our knowledge, only the methods reported in Tables 2, 3 and 4 can be included in this compression benchmark. Indeed, most of the pruning-based methods ([4, 5]) typically report pruning-rates only, which can not be directly mapped to file size: encoding sparse structures requires additional memory to store the coordinates for the un-pruned parameters. We implemented one state-of-the-art pruning method (LOBSTER [28]) to report pruning baseline storage memory achieved. Concerning quantization methods, existing approaches either focus on quantization to boost inference computation minimization ([7, 19, 21, 9, 8]) or do not report the final file size ([17, 10, 23, 11]). We also tried to directly compress the baseline file and we did not observe any compression gain. Therefore, to make reading easier, we did not report these numbers.

As a first experiment, we train LeNet-5 on MNIST (Table 2): despite the simplicity of the task, the reference LeNet-5 is notoriously over-parametrized for the learning task. Indeed, as expected, most of the state-of-the-art techniques are able to compress the model to approximately 40kB. In such a context, HEMP performs best, lowering the size of the compressed model to 27.5kB.

Then, we experiment with ResNet-32 and MobileNet-v2 on CIFAR-10 (as reported in Table 3), achieving also in this case significant compression: ResNet-32 size drops from 1.9MB to 168kB and MobileNet-v2 from 9.4MB to 822kB. Note that, other literature methods do not report experiments on CIFAR-10 on the proposed architectures. Nevertheless, HEMP approximately reduces the network size by a factor 11 for both architectures.

We also compress pretrained ResNet-18, ResNet-50 and MobileNet-v2 trained on ImageNet (Table 4). Also in this case, HEMP reaches competitive final file size, being able to compress ResNet-18 from 46.8MB to 3.6MB with minimal performance loss and ResNet-50 from 102.5MB to 5.5MB. For the ResNet-50 experiment, we also report partial result for high accuracy band, indicated as "high acc", to compare to Wiedemann *et al.* [12]: for the same accuracy, HEMP proves to drive the model to a higher compression. In the case of ResNet-18, [22] achieves a 0.5MB smaller compressed model, which is however set off by a 4.3% worse Top-1 error. Also in the case of very efficient architectures like Mobilenet-v2, HEMP is able to reduce significantly the storage memory occupation, moving from 13.5MB to 1.7MB only. Furthermore, the error drop is in this case very limited (0.8%) when compared to other techniques, like Tung *et al.* which, in the case of less optimized architectures like ResNet-50, do not have a large drop. While concurrent techniques rely on typical pruning+quantization strategies, aiming at *indirectly* eliminating the redundancy in the models, HEMP is *directly* optimizing over the existing redundancy.

Finally, we also tried to make HEMP cope with a different quantize and prune scheme. In particular, APQ [33] proves to be a perfect framework for our purpose, since it is a strategy performing both network architecture search, pruning and quantization. We have used HEMP in the most challenging scenario proposed by Wang *et al.*, with the lowest latency constraint (6.11ms) and the lowest energy consumption (9.14mJ) at inference time. In this case, we have fine-tuned the APQ's provided model for 5 epochs. Even in this case, HEMP is able to reduce the model's size, from the 20.8MB of the model to 3.04MB only, proving on-the-field its deployability as a companion besides

other quantization/pruning scheme, and non-exploiting any prior on the network's architecture.

Overall, these experiments show that HEMP strikes a competitive trade-off between compression ratio and performance on several architectures and datasets.

HEMP+LOBSTER. It has been observed that combining pruning and compression techniques enhances reduces the model final file size with little performance loss [12]. In our context, this translates into including two constraints to the learning:

- force the quantizer to have, for some ξ, the representation $r(\xi) = 0$ (or in simpler words, a quantization level corresponding to "0");
- include a pruning mechanism (permanent parameter set to "0").

Both of the constraints work independently from HEMP: indeed, HEMP is not a quantization technique, but it is thought to side any other learning strategy whose aim is to quantize the model's parameters (in such context, pruning "quantizes to zero" as many parameters as possible). Hence, we tried to side HEMP to LOBSTER [28], which is a state-of-the-art differentiable pruning strategy (hence, compatible within HEMP's framework).

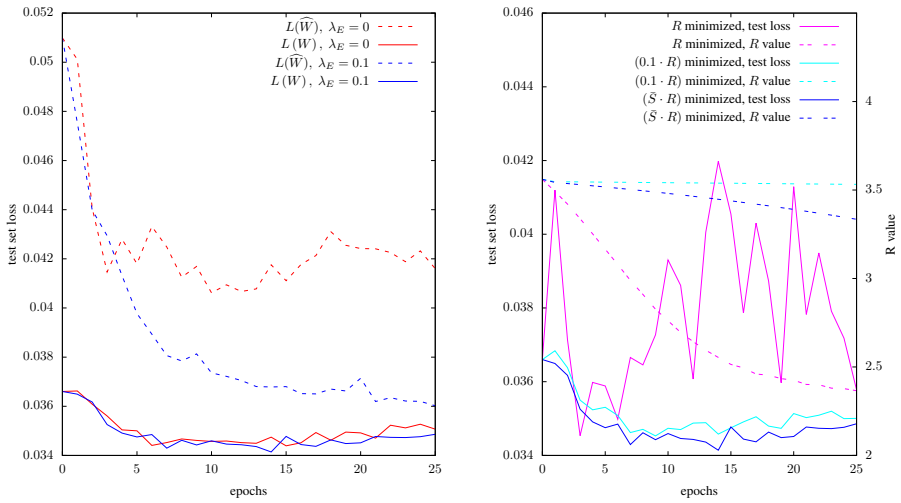
The results are as well reported in Tables 2, 3 and 4: it is evident that, including a prior on the optimal distribution of the parameters (removing all the un-necessary ones for the learning problem) helps HEMP to compress more. We have tested the setup HEMP + LOBSTER on one architecture per dataset: LeNet-5 (MNIST), ResNet32 (CIFAR-10) and ResNet-18 (ImageNet). While LOBSTER alone is able to achieve highly compressed models for toy datasets (like MNIST), it can not achieve high compression alone on more complex datasets. Still, siding a technique like LOBSTER to HEMP, boosts the compression of 10x for MNIST and ImageNet dataset and 4x for CIFAR-10.

HEMP minimizes the n-th entropy order (in these experiments, $n = 2$) - or in other words, maximizes the occurrence of certain sequences of quantization indices. The mapping of these quantization indices to quantization levels has to be determined outside HEMP: when we run experiments with "HEMP" alone, the loss minimization (in our case, the cross-entropy) automatically determines these levels - with the general-purpose Lloyd-max quantizer. However, pruning strategies include a prior on one of the quantization levels (the one corresponding to "0"), and this helps towards having a higher entropy minimization.

5.3. Ablation study

Here, we evaluate the impact of the reconstruction error term (25) and the overall insensitivity re-weighting (27) for the regularization function. Towards this end, we perform an ablation study on the ResNet-32 architecture trained on CIFAR-10.

Reconstruction error regularization. Fig. 8a (left), shows the ResNet-32 loss for the continuous and the quantized models ($L(W)$ and $L(\widehat{W})$ when the reconstruction error E is included or excluded ($\lambda_E = 0$) from the regularization function (24). We observe that both continuous models (solid lines) obtain similar performance on the test



(a) Reconstruction error regularization

(b) Insensitivity re-weighting

Figure 8: Test set losses for different trainings on ResNet-32 trained on CIFAR-10 (a), and effect of the insensitivity as re-weighting factor for R (b). Please notice that the blue line in both (a) and (b) refers to the same simulation, which refers to the standard HEMP training.

set. However, the quantized models (dashed lines) perform very differently. When the reconstruction error is not included in the training procedure (red lines), the quantized model reach a plateau with a high loss value showing that the network performs poorly on the test set. Conversely, when the reconstruction error is included (blue lines), the quantized model reaches a final loss closer to the continuous models. Indeed, regularizing also on (25) makes $w_{l,i} \rightarrow \hat{w}_{l,i} \; \forall \; l, i$, hence $L(W) \rightarrow L(\widehat{W})$. This experiment verifies the contribution of the error reconstruction regularization term towards the good performance of the quantized model.

Insensitivity-based re-weighting. Fig. 8b (right) shows the performance of the ResNet-32 model including or excluding the insensitivity re-weighting $\bar{S}_{l,i}$ for the regularization function (27). Here, we report the test set losses obtained by the continuous models (continuous lines) and the value for the overall R function (dashed lines). We observe a very unstable test loss without insensitivity re-scaling for R (magenta line). Hence, minimization with an overall 0.1 re-scaling for R is also shown (in cyan): in such case, the test loss on the continuous model remains low, but R is extremely slowly minimized. Using the insensitivity re-weighting (in blue) proves to be a good trade-off between keeping the test set loss low and both minimizing R. This behavior is what we expected: the insensitivity re-weighting, acting parameter-wise (ie. there is a different value per each parameter), dynamically tunes the re-weighting of the overall regularization function R, allowing faster minimization with minimal or no performance loss. This is why we could use the same λ_H and λ_R values for all the simulations, despite optimizing different architectures on different datasets. Such robustness of the hyper-

parameters over different dataset is a major practical strength of our approach.

6. Conclusion

We presented HEMP, an entropy coding-based framework for compressing neural networks parameters. Our formulation efficiently estimates entropy beyond the first order and can be employed as regularizer to minimize the quantized parameters' entropy in gradient based learning, directly on the continuous parameters. The experiments show that HEMP is not only an accurate proxy towards minimizing the entropy of the quantized parameters, but are also pivotal to model the quantized parameters statistics and improve the efficiency of entropy coding schemes. We also sided HEMP to LOBSTER, a state-of-the-art pruning strategy which introduces a prior on the weight's distribution which gives a further boost to the final model's compression.Future works include the integration of a quantization technique designed specifically for deep models to HEMP.

References

[1] S. Samarakoon, M. Bennis, W. Saad, M. Debbah, Distributed federated learning for ultra-reliable low-latency vehicular communications, IEEE Transactions on Communications.

[2] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, K. Keutzer, Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size, arXiv preprint arXiv:1602.07360.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 4510–4520.

[4] D. Molchanov, A. Ashukha, D. Vetrov, Variational dropout sparsifies deep neural networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 2498–2507.

[5] E. Tartaglione, S. Lepsøy, A. Fiandrotti, G. Francini, Learning sparse neural networks via sensitivity-driven regularization, in: Advances in neural information processing systems, 2018, pp. 3878–3888.

[6] C. Louizos, M. Welling, D. P. Kingma, Learning sparse neural networks through l_0 regularization, International Conference on Learning Representation (ICLR).

[7] M. Courbariaux, Y. Bengio, J.-P. David, Binaryconnect: Training deep neural networks with binary weights during propagations, in: Advances in neural information processing systems, 2015, pp. 3123–3131.

[8] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, Y. Zou, Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, arXiv preprint arXiv:1606.06160.

[9] A. Mishra, E. Nurvitadhi, J. J. Cook, D. Marr, Wrpn: wide reduced-precision networks, International Conference on Learning Representation (ICLR).

[10] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, D. Shin, Compression of deep convolutional neural networks for fast and low power mobile applications, International Conference on Learning Representation (ICLR).

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

[12] S. Wiedemann, H. Kirchhoffer, S. Matlage, P. Haase, A. Marban, T. Marinc, D. Neumann, T. Nguyen, H. Schwarz, T. Wiegand, D. Marpe, W. Samek, Deepcabac: A universal compression algorithm for deep neural networks, IEEE Journal of Selected Topics in Signal Processing.

[13] D. Oktay, J. Ballé, S. Singh, A. Shrivastava, Scalable model compression by entropy penalized reparameterization, arXiv preprint arXiv:1906.06624.

[14] Y. Cheng, F. X. Yu, R. S. Feris, S. Kumar, A. Choudhary, S.-F. Chang, An exploration of parameter redundancy in deep networks with circulant projections, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 2857–2865.

[15] C. Lee, Y.-B. Kim, H. Ji, Y. Lee, Y. Hur, H. Lim, On the redundancy in the rank of neural network parameters and its controllability, Applied Sciences 11 (2) (2021) 725.

[16] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2018, pp. 6848–6856.

[17] K. Ullrich, E. Meeds, M. Welling, Soft weight-sharing for neural network compression, International Conference on Learning Representation (ICLR).

[18] J. Frankle, M. Carbin, The lottery ticket hypothesis: Finding sparse, trainable neural networks, International Conference on Learning Representation (ICLR).

[19] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: Imagenet classification using binary convolutional neural networks, in: European conference on computer vision, Springer, 2016, pp. 525–542.

[20] C. Baldassi, F. Gerace, H. J. Kappen, C. Lucibello, L. Saglietti, E. Tartaglione, R. Zecchina, Role of synaptic stochasticity in training low-precision neural networks, Physical review letters 120 (26) (2018) 268103.

[21] X. Lin, C. Zhao, W. Pan, Towards accurate binary convolutional neural network, in: Advances in Neural Information Processing Systems, 2017, pp. 345–353.

[22] O. Shayer, D. Levi, E. Fetaya, Learning discrete weights using the local reparameterization trick, International Conference on Learning Representation (ICLR).

[23] V. Belagiannis, A. Farshad, F. Galasso, Adversarial network compression, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 0–0.

[24] S. Han, H. Mao, W. J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, International Conference on Learning Representation (ICLR).

[25] I. H. Witten, R. M. Neal, J. G. Cleary, Arithmetic coding for data compression, Communications of the ACM 30 (6) (1987) 520–540.

[26] G. Seroussi, A. Lempel, Lempel-ziv compression scheme with enhanced adapation, uS Patent 5,243,341 (Sep. 7 1993).

[27] I. Pavlov, Lzma sdk (software development kit) (2007).

[28] E. Tartaglione, A. Bragagnolo, A. Fiandrotti, M. Grangetto, Loss-based sensitivity regularization: towards deep sparse neural networks, arXiv preprint arXiv:2011.09905.

[29] K. Wang, Z. Liu, Y. Lin, J. Lin, S. Han, Haq: Hardware-aware automated quantization with mixed precision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2019, pp. 8612–8620.

[30] F. Tung, G. Mori, Deep neural network compression by in-parallel pruning-quantization, IEEE transactions on pattern analysis and machine intelligence 42 (3) (2020) 568–579.

[31] C.-H. Tu, J.-H. Lee, Y.-M. Chan, C.-S. Chen, Pruning depthwise separable convolutions for mobilenet compression, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.

[32] Y. He, Z. Pan, L. Li, Y. Shan, D. Cao, L. Chen, Real-time vehicle detection from short-range aerial image with compressed mobilenet, in: 2019 International Conference on Robotics and Automation (ICRA), IEEE, 2019, pp. 8339–8345.

[33] T. Wang, K. Wang, H. Cai, J. Lin, Z. Liu, S. Han, Apq: Joint search for nerwork architecture, pruning and quantization policy, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020.