Entropy-Constrained Training of Deep Neural Networks

Simon Wiedemann
Dept. of Video Coding and Analytics
Fraunhofer HHI
Berlin, Germany
simon.wiedemann@hhi.fraunhofer.de

Arturo Marban
Dept. of Video Coding and Analytics
Fraunhofer HHI
Berlin, Germany
arturo.marban@hhi-extern.fraunhofer.de

Klaus-Robert Müller
Machine Learning Group
TU Berlin
Berlin, Germany
klaus-robert.mueller@tu-berlin.de

Wojciech Samek
Dept. of Video Coding and Analytics
Fraunhofer HHI
Berlin, Germany
wojciech.samek@hhi.fraunhofer.de

***Abstract*—Motivated by the Minimum Description Length (MDL) principle, we first derive an expression for the entropy of a neural network which measures its complexity explicitly in terms of its bit-size. Then, we formalize the problem of neural network compression as an entropy-constrained optimization objective. This objective generalizes many of the currently proposed compression techniques in the literature, in that pruning or reducing the cardinality of the weight elements can be seen as special cases of entropy reduction methods. Furthermore, we derive a continuous relaxation of the objective, which allows us to minimize it using gradient-based optimization techniques. Finally, we show that we can reach compression results, which are competitive with those obtained using state-of-the-art techniques, on different network architectures and data sets, e.g. achieving $\times$71 compression gains on a VGG-like architecture.**

***Index Terms*—Neural network compression, entropy minimization, pruning, cardinality reduction.**

I. Introduction

It is well established that deep neural networks excel on a wide range of machine learning tasks [1]. However, common training practices require to equip neural networks with millions of parameters in order to attain good generalisation performances. This often results in great storage requirements for representing the networks (in the order of hundreds of MB), which greatly difficults (or even prohibits) their deployment into resource constrained devices and their transmission in restricted communication channels. Moreover, several recent works [2] have shown that most deep networks are overparametrized for the given task and that they have the capacity to memorize entire data sets [3]. This motivates the use of regularizers that penalize the networks complexity, lowering the risk of overfitting and thus, favour models that generalize well.

Compression, that is, lowering the amount of information required to uniquely reconstruct the networks parameters, is a very natural way to limit the networks complexity. Indeed, the Minimum Description Length (MDL) principle states that if two or more models achieve same prediction performance, then, one should always choose the model that requires the least amount of bits to describe it [4], [5]. Moreover, compression also offers direct practical advantages. Networks with small memory footprint do not only require lower communication costs (which is of utmost importance in, e.g., federated learning [6]–[8]), but can also perform inference requiring significantly less time and energy resources [2].

Furthermore, we know from the source coding literature [9]–[11] that the *entropy* of a probabilistic source measures the minimum average bit-length needed in order to represent the data generated by that source in a lossless manner. In addition, recent work have focused on designing efficient data structures that exploit the fact that the network has low entropy[1], achieving lower memory, time and energy footprint for performing inference [12]–[15]. Hence, due to theoretical as well as practical reasons, it appears only natural to measure the bit-size of neural networks in terms of their entropy and to constrain their training explicitly by it.

Hence, our contributions can be summarized as follows:

- We formulate the task of neural network compression under an entropy-constrained optimization objective. The derived entropy term explicitly measures the minimum average bit-size required for representing the weight elements, generalizing many of the compression techniques proposed so far in the literature.
- We derive a continuous relaxation of the entropy-constrained objective, which allows to minimize it using gradient-based optimization techniques. We conjecture that the continuous objective upper bounds the desired discrete one, and provide experimental evidence for it.
- Our experimental results show that we can highly compress different types of neural network architectures while

[1]The entropy is measured relative to the empirical probability mass distribution of the weight elements.

minimally impacting their accuracy.

II. The MDL principle as a general framework for deep neural network compression

We start by formalising the general learning problem under the MDL principle, since the objective of model compression arises naturally under this paradigm.

Assume a supervised learning setting where given a particular data set, $\mathcal{D}_N = \{(x_i, y_i) | x_i \in \mathcal{X}, y_i \in \mathcal{Y}, i \in \{1, ..., N = |\mathcal{X}| = |\mathcal{Y}|\}\}$, we want to learn to predict the elements y_i of the output (or label) set $\mathcal{Y}$ from the elements x_i of the input set $\mathcal{X}$. The MDL principle equates the task of "learning" with the task of compression. That is, the more regularities we find in the data, the more we can use these to compress it and ultimately, generalize better to unseen data that entail the same regularities. Therefore, the above supervised learning goal can be restated as to find the compression algorithm that compresses the most the set of labels $\mathcal{Y}$ given the input set $\mathcal{X}$.

Due to the fundamental correspondence between probability distributions and bit-lengths [5], we can precisely formalize this idea. Namely, let $\mathcal{W} = \{p(\mathcal{Y}|\mathcal{X}, W) | W \in \mathbb{R}^n, n \in \mathbb{N}\}$ be a set of parametric conditional probability distributions over the label set $\mathcal{Y}$, where we denote as $W \equiv p(\mathcal{Y}|\mathcal{X}, W)$ a particular model or point hypothesis in $\mathcal{W}$ (e.g., a trained neural network). Then, one possible way to encode the labels $\mathcal{Y}$ is by using a so called *two-part code*. That is, we first describe a particular point hypothesis W in $\mathcal{W}$ using $L_{\mathcal{W}}(W)$ bits and then optimally encode its prediction errors using $L(\mathcal{Y}|\mathcal{X}, W) = -\log_2 p(\mathcal{Y}|\mathcal{X}, W)$ bits. Hence, the MDL principle states that we should find,

$$W^* = \min_W \underbrace{-\log_2 p(\mathcal{Y}|\mathcal{X}, W)}_{encoding\ prediction\ error} + \alpha \underbrace{L_{\mathcal{W}}(W)}_{encoding\ model} \quad (1)$$

where $0 < \alpha \in \mathbb{R}$.

Notice, that (1) aims to minimize the prediction errors of the model *and* the explicit bit-length of the model. Therefore, the MDL principle is a natural framework for the task of neural network compression. Now we need to define a description or code that assigns an unique bit-string to each model W.

Firstly, we consider the entire set of discrete neural networks $\mathbb{W}$ with a particular and fixed topology. That is, all considered neural networks have same topology which does not change over time, and their weight element values are restricted to be elements from a finite set of real numbers. Thus, w_i denotes the i-th weight element of the neural network, which can take one of K possible values, i.e., $\omega_i \in \Omega = \{\omega_0, ..., \omega_{K-1}\}$. We furthermore assume that the network has n parameters, i.e., $i \in \{1, ..., n\}$. Whilst this definition may seem a considerable constraint at first sight, notice that $\mathbb{W}$ entails all neural networks that are representable by any digital device[2]. Moreover, focusing on networks with fixed topology is reasonable because: 1) storing the particular topology should be negligible compared to storing the weight values and 2) in many practical settings one aims to compress a network parameters with an already given topology (e.g. communication scenario).

Secondly, we assume that the decoder[3] has no prior knowledge regarding the correlations between the input and output set. Consequently, it can neither make any reasonable a priori assumptions regarding the correlations between the weight element values of the neural network. This scenario is very common in most real world cases. For instance, most digital processor units implicitly make such assumption by not prioritising any particular network configuration. Moreover, this is also a common assumption in machine learning problems.

Thus, the only reasonable prior over the weight elements is an n-long discrete independent random process. That is, let $P_\Omega = \{p_0, ..., p_{K-1}\}$ be a particular probability mass distribution (PMD) over the finite set Ω. Then, we model the joint probability distribution as $P(W) = P(w_1 = \omega_{k_1}, ..., w_n = \omega_{k_n}) = \prod_i^n p_{k_i}$, where $p_{k_i} \in P_\Omega$ is the probability of the i-th weight element outputting the value $\omega_{k_i} \in \Omega$. Hence, the respective bit-length can be defined as $L_{\mathbb{W}}(W) = -\log_2 P(W) = \sum_i^n -\log_2 p_{k_i}$. However, the resulting code depends on the choice of the PMD, which we do not know a priori.

A reasonable choice of PMD in this case would be to assume an uniform prior over Ω, i.e., $p_k = 1/K$. However, it assigns a bit-string of constant size to each weight element and consequently to each model W. Concretely, $L_{\mathbb{W}}(W) = n \log_2 K$, and if $K = 2^{32}$ we obtain the amount of bits needed to store the network if we store each weight element in, e.g., its respective single precision floating point representation. In this case, the minimization (1) becomes the standard maximum likelihood estimation objective and consequently no model compression is performed in the process.

However, we can design more suitable codes for the case of deep neural networks. Namely, for each particular neural network, W, we could first estimate the probability distributions of its weight values and subsequently use these estimates in the compression process. This may work well since the number of parameters in most state-of-the-art deep neural network models is typically $n = \mathcal{O}(10^7)$ and therefore, we can expect the estimation to be a good approximation of the real distribution.

A. Universal coding of discrete deep neural networks

Hence, we propose to first calculate the maximum likelihood estimate of the PMD of W, and then optimally encode the element values of W relative to it. Such coding techniques belong to the subject of *universal coding* and their properties are well studied in the literature [4], [5], [16], [17]. Concretely, from [5] we know that if we apply an universal two-part code for the task probability density estimation by using histograms, the respective regret (or maximum redundancy) per data point decreases sublinearly as the number of data points increases. That is, the maximum redundancy of the code decreases at a

[2]Assuming a sufficiently large K.

[3]The entity which is able to recover back the label set $\mathcal{Y}$ from its compressed bitstream representation.

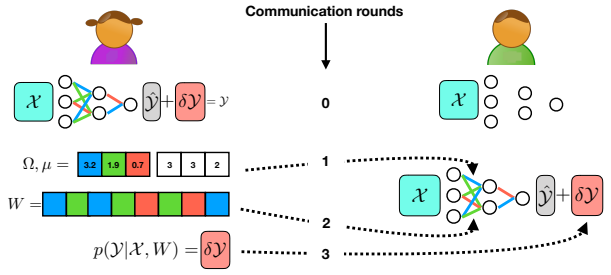


Fig. 1: The task of compression is analogous to the task of communication, which is depicted in this figure. Alice wants to send to Bob a particular output set $\mathcal{Y}$. Assume that Alice and Bob have previously agreed on the input set $\mathcal{X}$ and a particular neural network topology. Then, Alice will send $\mathcal{Y}$ performing three steps: 1) She trains a particular network model $p(\mathcal{Y}|\mathcal{X}, W) \equiv W$, encodes the unique elements $\Omega = \{3.2, 1.9, 0.7\}$ (blue, green, red) that appear in the network using an uniform code, along with their maximum likelihood estimate of their probability mass distribution $\mu = (3, 3, 2)$, and sends these to Bob. 2) She encodes the weight values using an entropy-coder with regards to μ and sends them to Bob. Since Bob also knows μ and Ω, he can uniquely recover the weight values. 3) Alice encodes the prediction errors of W using $-\log_2 p(\mathcal{Y}|\mathcal{X}, W)$ bits and sends these to Bob. With that, Bob can uniquely reconstruct the output set $\mathcal{Y}$.

rate of order $\mathcal{O}(\frac{\log_2 n}{n})$, n being the number of data points. This means that we can expect the regret to be almost 0 if we apply such code to the weights of deep neural networks, and consequently, the empirical PMD is almost identical to the actual PMD that generated the weight values (assuming that such a distribution actually generated the weight values).

Hence, we propose the following coding scheme: Let $\mu = \{\hat{p}_0, ..., \hat{p}_{K-1}\}$ denote the empirical probability mass distribution (EPMD) of the elements Ω in W, thus, $\hat{p}_k = \#(\omega_k)/n$ where $\#(\cdot)$ denotes the counting operator. Then, we

1) calculate μ and encode its elements along with Ω using an uniform code, and
2) compress each w_i using $-\log_2 \sum_k I(i = k)\mu_k$ bits respectively.

Here, $I(i = k)$ denotes the indicator function, being 1 if $w_i = \omega_k$, and 0 otherwise. Fig. 1 sketches this coding scheme.

Thus, the resulting bit-length of a particular neural network model W can be written as,

$$L_{\mathbb{W}}(W) = L(\mathbb{I}_W) + L(\mu) + L(\Omega) \tag{2}$$

where $L(\mathbb{I}_W) = \sum_{i=1}^{n} -\log_2 \sum_k I(i = k)\mu_k$, $L(\mu) = K \log_2 n$, and $L(\Omega) = Kb$ with b being the bit-precision used to represent the elements in Ω.

B. Entropy-constrained minimization objective

Notice, that $L(\mathbb{I}_W) = n \sum_k -\mu_k \log_2 \mu_k = nH(\mu)$, with $H(\mu)$ being the entropy of the EPMD μ. Thus, (2) states that the bit-length of each weight element w_i is equal to its minimum average bit-length relative to μ, plus additional redundant terms that result from having to send the estimates. Hence, since the bit-length of the estimation $L(\mu)$ and $L(\Omega)$ is constant, the MDL principle states that we aim to find the neural network that minimizes the objective

$$W^* = \min_{W \in \mathbb{W}} -\log_2 p(\mathcal{Y}|\mathcal{X}, W) + \alpha n H(\mu), \quad 0 < \alpha \in \mathbb{R} \tag{3}$$

As a side note, we want to mention that most common lossless entropy coders encode the outputs of an independent random process in the same manner as described above. Thus, the above description of the neural network explicitly expresses the amount of bits required to store its weight elements if we would apply, e.g. the Huffman code or an arithmetic coder for encoding its elements [11].

III. Training of deep neural networks under an entropy-constrained objective

We would now like to apply gradient-based optimization techniques to minimize (3), since they are scalable to large and deep networks and have shown capable in finding good local minima that generalize well [1], [18]. However, the minimization objective stated in (3) is non-differentiable. Therefore, in the following sections, we firstly derive a continuous relaxation of (3), and subsequently reformulate it as a variational minimization objective. Then, we will conjecture that, under certain conditions, the variational objective upper bounds the discrete one.

A. Continuous relaxation

Notice that the only reason (3) is non-differentiable is due to the indicator operator implicitly entailed in it. Hence, we propose to smooth it by a continuous parametric probability distribution. Namely,

$$I(i = k) = \begin{cases} 1, & \text{if } w_i = \omega_k \\ 0, & \text{else} \end{cases}$$
$$\xrightarrow{\text{cont. relax.}} P_i = \{P(i = k|\theta_{ik}) | \sum_k P(i = k|\theta_{ik}) = 1\} \tag{4}$$

with $P(i = k|\theta_{ik}) = P_{ik}$ being the probability that the weight element w_i takes the discrete value ω_k, parametrized by θ_{ik}. Thus, the continuous relaxation now models an entire set of discrete neural networks, whose probability of being sampled is specified by the joint probability distribution $P_\theta = \prod_i^n P_i$.

Subsequently, we naturally replace each estimate μ_k of the PMD, and thus the entropy, by

$$\mu_k = \frac{1}{n} \sum_i I(i = k) \xrightarrow{\text{cont. relax.}} P_k = \frac{1}{n} \sum_i P_{ik}$$
$$H(\mu) = \sum_k -\mu_k \log_2 \mu_k \rightarrow H(P) = \sum_k -P_k \log_2 P_k \tag{5}$$

and reformulate (3) as the following variational minimization objective

$$\theta^* = \min_\theta \mathbb{E}_{P_\theta}[-\log_2 p(\mathcal{Y}|\mathcal{X}, W \sim P_\theta)] + \alpha n H(P) \tag{6}$$

Algorithm 1 Forward pass sampling from a discrete probability distribution of the layers' weight tensor.

procedure FORWARDPASS(a) ▷ per layer
  Calculate the mean weight tensor, ν_W, and its respective variance, σ_W^2.
  Forward pass layers' mean, ν_z, and standard deviation, σ_z, as specified in (7) and (8).
  Sample a tensor $\epsilon \sim \mathcal{N}(0,1)$ with the same dimension as the preactivation layers.
  Calculate the preactivation values as $Z = \nu_z + \sigma_z \odot \epsilon$ ▷ $\odot$ denotes the Hadamard product.

Thus, we now aim to minimize the *averaged* prediction error of the network, as taken relative to our model P_θ, constrained by the respective relaxation of the entropy (5).

However, we can still not apply gradient-based optimization techniques in order to minimize (6) as calculating the mean of the log-likelihood is infeasible for deep neural networks.

B. Training neural networks under the entropy-constrained variational objective

Hence, in order to train the network under the objective (6) we apply similar scalable approximation techniques as proposed by the bayesian neural networks literature [19]–[22]. Concretely, we approximate the mean of the log-likelihood by an unbiased Monte-Carlo estimator and sample from the preactivation values of the network, whose sufficient statistics depend on the mean and variances of the weights and input activations of the respective layer. That is, each preactivation value is now modeled as $Z = \nu_z + \sigma_z\ \mathcal{N}(0,1)$, where

$$\nu_z = \nu_W \cdot a \tag{7}$$

$$\sigma_z = \sqrt{\sigma_W^2 \cdot a^2} \tag{8}$$

and $(\nu_W)_i = \sum_k \omega_k P_{ik}$, $(\sigma_W)_i^2 = \sum_k \omega_k^2 P_{ik} - (\nu_W)_i$ are the mean and variances of the weights of the respective layer, and a are the activation values of the previous layer.

From a source coding point of view this procedure simulates a stochastic quantization scheme of the weights relative to the joint probability model P_θ, which can be expressed as Gaussian noise in the preactivation layers due to the central limit theorem. The respective pseudocode can be seen in algorithm 1. Notice, that now (6) is differentiable with respect to the parameters θ and discrete values Ω.

After training, we then select the most likely point hypothesis as our discrete model. Henceforth, we will refer to this operation as the maximum-a-posteriori (MAP) quantization step, and denote it as $W^* = q(\theta^*)$. If all P_{ik} are defined in terms of a radial function relative to a continuous parameter θ_i and ω_k, then the MAP quantization corresponds to a nearest-neighbor quantization scheme of the parameter θ_i relative to the discrete set Ω (as depicted in Fig. 2).

C. Relation between the variational and discrete objectives

The motivation behind minimizing (6) is not restricted to the scalability property. We conjecture that it upper bounds the loss value (3) of the MAP point hypothesis W^* for sufficiently small loss values.

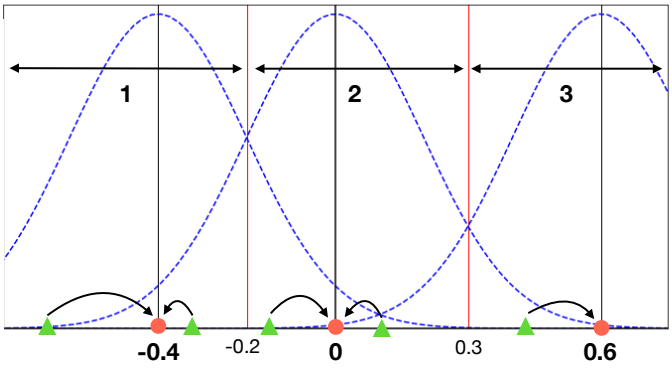


Fig. 2: Quantization of the weight values corresponds to the maximum-a-posteriori (MAP) operator applied to their parametric discrete probability distribution. In our experiments we parametrized the probability distributions using Gaussian kernels, whose MAP operator is equivalent to a nearest-neighbor quantization scheme, as depicted in the above diagram. For instance, all continuous weight element values (green triangles) that lie in region 1 will be quantized to the maximum probable value of that region, namely the value $\omega_1 = -0.4$ (red dot). Analogously for the regions 2 and 3. For constant variance, the decision thresholds (red lines) lie exactly in the middle between two elements of $\Omega = \{-0.4, 0, 0.6\}$.

Conjecture 1: Let θ^* be the parameters of the joint probability model $P_{\theta^*} = \prod_i^n P_i$ which minimizes the variational objective (6), where each P_i is defined as in (4). Then, if its variational loss value is sufficiently small, it upper bounds the loss value of the respective MAP point hypothesis $W^* = q(\theta^*)$. That is, there exist a $0 < \epsilon \in \mathbb{R}$, where

$$\begin{aligned}\mathcal{L}(P_{\theta^*}) &= \mathbb{E}_{P_{\theta^*}}[-\log_2 p(\mathcal{Y}|\mathcal{X}, W \sim P_\theta^*)] + \alpha n H(P(\theta^*)) \\ &> \mathcal{L}(W^*) = -\log_2 p(\mathcal{Y}|\mathcal{X}, W^*) + \alpha n H(\mu^*)\end{aligned} \tag{9}$$

with high probability, if $\mathcal{L}(P_{\theta^*}) < \epsilon$. μ^* refers to the EPMD of the point hypothesis W^*.

Although we do not provide a rigorous proof of conjecture 1, in the following we give a proof idea in order to support the claim.

Proof idea: Firstly, notice that $P_\theta \xrightarrow{\text{var}(P_\theta)\to 0} \mu$. That is, the PMD of the continuous relaxation P_θ converges to the EPMD μ of the MAP point hypothesis in the limit of zero variance. Consequently, we argue that the entropy of the EPMD $H(\mu)$ is always smaller or equal to the entropy of its continuous relaxation $H(P)$ (as defined in (5)),

$$H(P) \geq H(\mu).$$

The intuition behind the above inequality is that we can reduce the entropy $H(P)$ by reducing the variance of each element of the random process P_θ (thus, the variance of each weight element) and, in the limit where each element has 0 variance, we obtain the MAP distribution μ.

Secondly, we argue that the mean value of the log-likelihood is with high probability larger than the prediction error of the MAP estimate at the minima. This becomes more clear if we rewrite the mean as

$$\mathbb{E}_{P_\theta}[-\log_2 p(\mathcal{Y}|\mathcal{X}, W \sim P_\theta)] = L^* P_1 + \bar{L}(1 - P_1) \quad (10)$$

where $L^* = -\log_2 p(\mathcal{Y}|\mathcal{X}, W^*)$ is the prediction error of the MAP point estimate, P_1 its respective probability of being sampled and $\bar{L} = \mathbb{E}_{P_\theta \setminus P_1}[-\log_2 p(\mathcal{Y}|\mathcal{X}, W \sim P_\theta \setminus P_1)]$ the mean of the prediction errors as taken relative to all the other point estimates with their respective conjugate probabilities $P_\theta \setminus P_1$. Hence, the likelihood of the MAP point estimate is smaller iff $L^* < \bar{L}$. This is likely to be true iff the variance of P_θ is sufficiently small[4], since $P_1 \xrightarrow{\text{var}(P_\theta)\to 0} 1$.

Lastly, minimizing (9) implies minimizing the entropy of P_θ which implicitly minimizes its variance, thus, pushing the probability P_1 to 1 and minimizing the term L^* with high probability.

□

Hence, conjecture 1 states that if we find an estimator θ^* that reaches a sufficiently low loss value, then the respective MAP point hypothesis W^* has a lower loss value with high probability. In the experimental section, we provide evidence that conjecture 1 holds true.

IV. Related work

There is a plethora of literature proposing different techniques and approaches for compressing the weight parameters of deep neural network [2]. Among them, some have focused on designing algorithms that sparsify the network structure [21], [23]–[27]. Others, on reducing the cardinality of the networks parameters [28]–[33]. And some have proposed general compression frameworks with the objective to do both, sparsification and cardinality reduction [12], [22], [34]. Our entropy-constrained minimization objective (3) (and its continuous relaxation (6)) aims to generalize those works. Either maximizing sparsity or minimizing the cardinality of the weight values, both imply minimization of our derived entropy-term. Moreover, our framework allows learning neural networks that are neither sparse nor their set of weight values have low cardinality, but still, require low bit-lengths to represent them. Thus, our framework is able to learn a larger set of networks without increasing their complexity.

Other related work have also focused on designing regularizers that lower the implicit information entailed in the networks parameters, based on minimizing a variational lower bound objective [19], [20], [35]. Although minimizing the variational lower bound is also well motivated from a MDL point of view [36], the resulting coding scheme is often impractical for real world scenarios. Therefore, [22], [34], [37] focused on designing suitable priors and posteriors that allow to apply practical coding schemes on the weight parameters after the variational lower bound has been minimized. This includes a final step where a lossless entropy coder is applied to the network's parameter as proposed by [12]. Therefore, their proposed framework does only *implicitly* minimize the bit-lengths of the resulting model. In contrast, our entropy-constrained objective (3) *explicitly* states the bit-size of the resulting network. Moreover, our continuous entropy term (5) does not correspond to the minimization of a Kullback-Leibler term, but to the following cross-entropy term instead

$$\begin{aligned} nH(P) &= KL(q(\theta)||p(\theta)) + H(q(\theta)) \\ &= \mathbb{E}_{q(\theta)}[-\log_2 p(\theta)] \end{aligned}$$

with $q(\theta) = P_\theta = \prod_i^n P_i$ and $p(\theta) = \prod_k^K P_k$, where P_i and P_k are defined as in (4) and (5) respectively. This is again well motivated from a practical coding point of view. Notice how the prior $p(\theta)$ entails all weight element values in each P_k, and therefore implicitly takes correlations between them into account.

Finally, whilst we use similar training techniques for learning discrete neural networks as proposed by [32], [33], they only focused on learning ternary and binary networks. Hence, our work can also be seen as a generalization, which allows for arbitrary cardinality of the discrete set and adds an entropy term to the cost, which prioritizes non-uniform weight distributions as opposed to the training objectives stated in [32], [33].

V. Experimental results

A. Experimental setup

In our experiments we chose to parametrize each P_{ik} with the Gaussian kernel. That is,

$$\begin{aligned} P_{ik} &= \frac{G_{\sigma_i}(w_i, \omega_k)}{Z}, \\ \text{with } G_{\sigma_i}(w_i, \omega_k) &= \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{w_i - \omega_k}{\sigma_i}\right)^2} \\ \text{and } Z &= \sum_k G_{\sigma_i}(w_i, \omega_k) \end{aligned} \quad (11)$$

with parameters $\theta_i = (w_i, \omega_i, \sigma_i)$, $w_i \in \mathbb{R}$, $\omega_i \in \Omega$ and $0 < \sigma_i \in \mathbb{R}$. We chose a different set of discrete values Ω^l for each layer l, where we denote their respective cardinality as $|\Omega^l| = K_l \in \mathbb{N}$. Henceforth, we introduce the notation $\bar{K} = [K_0, K_1, ..., K_l, ..., K_{L-1}]$ in order to specify the cardinality of each layer. Since each layer has a different set of discrete values, we calculated the overall entropy of the network as the sum of the entropies of each individual weight element.

The experiments were performed on the MNIST dataset, using the LeNet-300-100 and LeNet-5 networks, and on the CIFAR-10 dataset, using a VGG16 network like architecture (with 15 layers), referred to as VGG-Cifar-10[5]. These networks are commonly used to benchmark different compression methods [12] [21] [22] [37]. We investigated two compression approaches:

[4]This argument can be relaxed into P_1 expressing the probability of the set of points W_j near W^*, such that their respective loss $L_j \approx L^*$.

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

1) Direct entropy-constrained training: Starting from a pre-trained model, we minimized the Shannon entropy of the network parameters with the variational entropy-constrained objective (6) (by applying algorithm 1 to each layer). We used ADAM [38] as the optimizer with a linearly decaying learning rate. We also linearly increased (from 0 to up to a number less than 1) the regularization coefficient α in (6). Afterwards, in the evaluation stage, the network weights were quantized using the values of the vector Ω^l, available in the l-th layer. We set their respective initial cardinalities, $\bar{K}$, as follows: (i) LeNet-300-100, $\bar{K} = [3, 3, 33]$, (ii) LeNet-5, $\bar{K} = [5, 5, 33]$, and (iii) VGG-Cifar-10, $\bar{K} = [33, 17, 15, 11, 7, 7, 7, 5, 5, 5, 3, 3, 3, 17, 33]$. The quantized network performance was monitored with the classification error, compression ratio, and sparsity (calculated on the weights).

2) Pre-sparsifying the models: Firstly, we sparsified the networks by applying the sparse variational dropout technique [21]. To this end, the network parameters were initialized from scratch and ADAM was applied as the optimizer with a linearly decaying learning rate. Also, we linearly increased the regularizer of the Kullback-Leibler divergence approximation during the sparsification stage (from 0 up to a number less than 1). After sparsification, we applied our entropy-constrained training approach to the neural networks. In this context, $\bar{K}$ was defined for each network as: (i) LeNet-300-100, $\bar{K} = [21, 21, 31]$, (ii) LeNet-5, $\bar{K} = [17, 17, 31]$, and (iii) VGG-Cifar-10, $\bar{K} = [27, .., 27, 33]$.

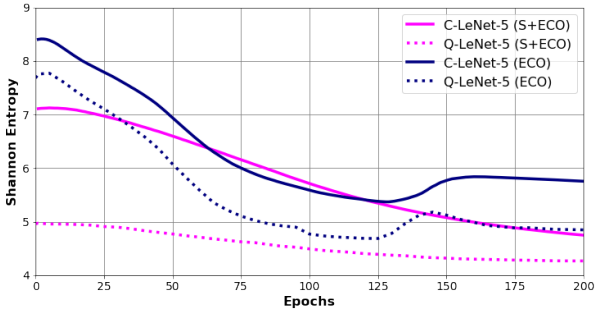
Notation remark: Henceforth, we refer to *C-model-name* or *Q-model-name* to either the continuous or quantized model. Furthermore, *ECO* refers to models that have been trained directly under the entropy-constrained objective, whereas with *S+ECO* we refer to models that have been a priori sparsified. For instance, *Q-LeNet-5 (S+ECO)* refers to the quantized LeNet-5 model, which has firstly been sparsified and then trained under the entropy-constrained objective.

B. Verifying conjecture 1

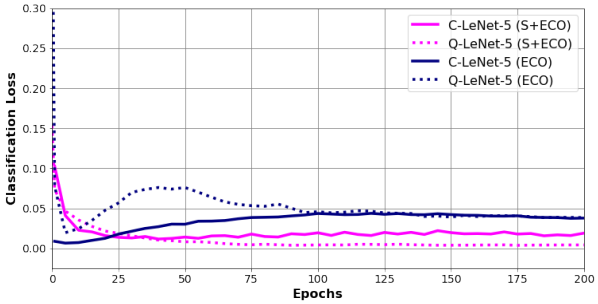
Figs. 3a, 3b and 4 verify some of the arguments stated in the proof idea of conjecture 1. Firstly, Fig. 3a shows how the entropy of the quantized model is always lower than the entropy of the continuous parametrization. Thus, the entropy of the continuous relaxation is always upper bounding the entropy of the quantized model. We also confirmed this on the other architectures.

Secondly, from Fig. 4 we see how the loss automatically tends to minimize the variance of the continuous probability distribution during training. This increases the probability of selecting the quantized model (or close models) during training and, thus, minimizing its cost value as well.

Lastly, from Fig. 3b we see how indeed, this can result in the variational classification loss eventually bounding the loss of the quantized network. We verified a similar trend on the LeNet-300-100 architecture. In particular, as we predicted, by comparing figures 4 and 3b we see how the difference between the variational and the quantized loss converges to the same value, as the variance of the continuous probability distribution becomes smaller.



(a)



(b)

Fig. 3: Trend of the average log-likelihood (on the training set) and entropy of the continuous model and the quantized model during training of the LeNet-5 architecture. (a) The entropy of the quantized model is always bounded by the entropy of the continuous model during the entire training procedure. (b) The log-likelihood of the quantized model will eventually be bounded by the average likelihood of the continuous one.

C. Generalizing sparsity and cardinality reduction

Fig. 5 shows PMDs of the last fully connected layer of the *Q-LeNet-5 (ECO)* and *Q-LeNet-5 (S+ECO)* models after training. Interestingly, directly minimizing the entropy-constrained objective admits solutions where the resulting weight distribution is not necessarily sparse, neither its cardinality is particularly lower than at the starting point. In contrast, by pre-sparsifiying the network, we constrain the set of solutions to only those which are close to the spike-and-slab distribution.

Nevertheless, from Table I we see that our entropy term is also strongly encouraging sparsity, since overall we attain similar pruning results as current state-of-the-art pruning techniques.

D. Network compression

Table I summarizes the achieved compression gains. As we can see, the results are comparable to the current state-of-the-art compression techniques. Interestingly, pre-sparsifying the VGG network seems to significantly improve its final accuracy. We believe that this is due to the variance of the continuous model, which may be too high during the training procedure. For large deep networks with millions of parameters, such

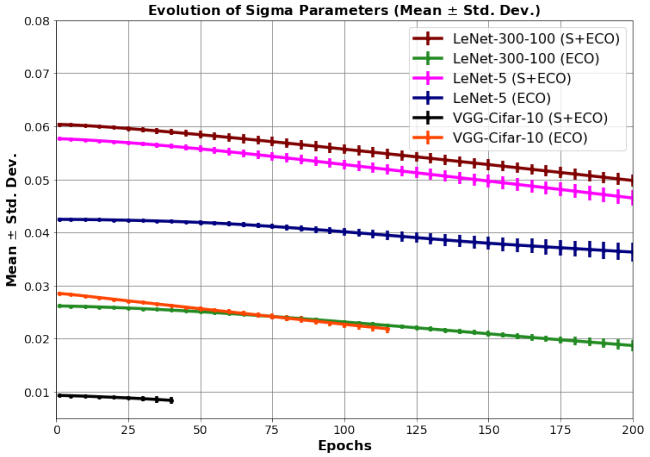


Fig. 4: Trend of the variance of the continuous probability distribution during training. Each point shows the mean and standard deviation of the variance, as taken over all weight element values of the network. The plots shows how the network is trying to reduce its variance during training.

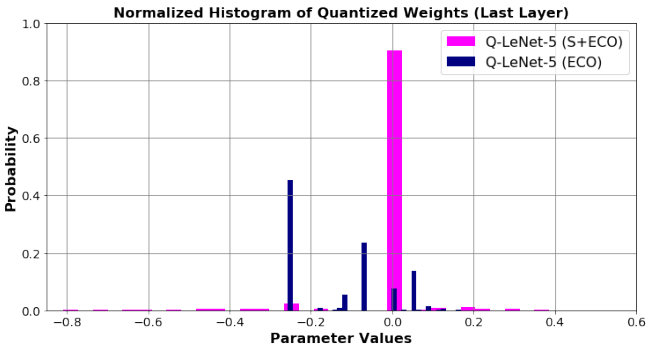


Fig. 5: Probability mass distribution (PMD) of the weight element values of the last layer of the LeNet-5 model after training. The PMD of the *Q-LeNet-5 (S+ECO)* is very sparse. In contrast, the PMD of the trained *Q-LeNet-5 (ECO)* architecture is neither sparse nor the cardinality of its values is particularly low.

as the VGG model, the probability that the quantized model P_1 (or close models to it) is (are) sampled during training reduces significantly. Consequently, the variational loss value was not upper bounding the cost of the quantized network during training, as can be seen in Fig. 6. However, notice that the variance of the pre-sparsified VGG network is significantly lower during the entire training procedure (Fig. 4), hence, closing the gap between the variational and quantized losses, and consequently, attaining higher accuracies on the quantized model.

As a side note, we want to stress that the dependency between P_1 and the number of parameters in the network is not reflected in Fig. 4. Therefore, the comparison between the variances across architectures is not informative.

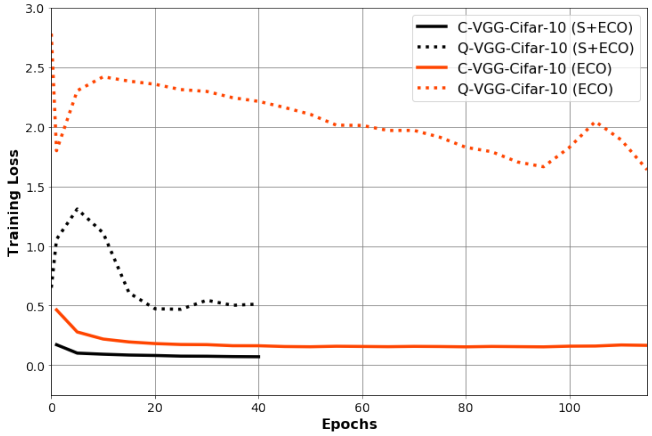


Fig. 6: Trend of the classification loss of the VGG architecture when trained on the CIFAR10 data set. The plot shows how the continuous loss does not upper bound the quantized loss during training. This may explain the loss in accuracy performance in Table I as compared to previous literature. However, we see a correlation between the variance and the difference on the loss values, in that lower variance significantly reduces the difference (see Fig. 4).

TABLE I: Compression gains attained from training the networks under the entropy-constrained minimization objective. We also compare the results to other compression techniques.

Model	Error [%]	$\frac{\lvert W \neq 0 \rvert}{\lvert W \rvert}$ [%]	CR
S + ECO			
LeNet-300-100	1.80	3.28	92
LeNet-5	0.90	1.11	209
VGG-Cifar-10	10.91	6.77	**71**
ECO			
LeNet-300-100	2.24	4.03	**102**
LeNet-5	0.97	1.94	**235**
VGG-Cifar-10	14.27	6.90	95
Deep Compression [12]			
LeNet-300-100	1.58	8	40
LeNet-5	0.74	8	39
Soft Weight Sharing [37]			
LeNet-300-100	1.94	4.3	64
LeNet-5	0.97	0.5	162
Sparse Variational Dropout [21]			
LeNet-300-100	1.92	1.47	68
LeNet-5	0.75	0.35	280
VGG-Cifar-10	7.30	1.53	65
Bayesian Compression [22]			
LeNet-300-100	1.80	2.20	113
LeNet-5	1.00	0.60	771
VGG-Cifar-10	9.20	5.50	116

VI. Conclusion

We formalized the task of neural network compression under the Minimum Description Length (MDL) principle, which resulted in an entropy-constrained minimization objective for training deep neural networks. We argued that this objective generalizes many of the compression techniques proposed in the literature, in that it automatically encourages pruning and cardinality reduction of the weight elements. Lastly, we showed that our compression framework achieves $\times 102$, $\times 235$

and $\times 71$ compression gains on the LeNet-300-100, LeNet-5 and VGG-Cifar-10 networks, respectively, a result that is comparable to that obtained with current state-of-the-art compression techniques.

In future work, we will consider improvements to our method. For instance, we believe that we can attain better accuracy-compression results by adding regularizers to the loss that encourage learning network distributions with small variance (since the proper minimization of the discrete objective loss is guaranteed). Furthermore, we will investigate the use of entropy-based compression techniques in federated learning scenarios and for improving the interpretability [39] [40] of deep models.

ACKNOWLEDGEMENT

This work was supported by the Fraunhofer Society through the MPI-FhG collaboration project "Theory & Practice for Reduced Learning Machines". This research was also supported by the German Ministry for Education and Research as Berlin Big Data Centre (01IS14013A) and Berlin Center for Machine Learning (01IS18037I). Partial funding by DFG is acknowledged (EXC 2046/1, project-ID: 390685689). This work was also supported by the Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (No. 2017-0-00451).

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

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

[3] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv:1611.03530*, 2016.

[4] J. Rissanen, "Paper: Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[5] P. Grünwald and J. Rissanen, *The Minimum Description Length Principle*. Adaptive computation and machine learning, MIT Press, 2007.

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

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

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

[9] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[10] T. M. Cover and J. A. Thomas, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. New York, NY, USA: Wiley-Interscience, 2006.

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

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

[13] S. Basu and L. R. Varshney, "Universal source coding of deep neural networks," in *Data Compression Conference (DCC)*, pp. 310–319, 2017.

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

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

[16] A. Barron, J. Rissanen, and B. Yu, "The minimum description length principle in coding and modeling," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2743–2760, 2006.

[17] Q. Xie and A. R. Barron, "Asymptotic minimax regret for data compression, gambling, and prediction," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 431–445, 2000.

[18] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade - Second Edition, Springer LNCS 7700*, pp. 9–48, Springer, 2012.

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

[20] A. Achille and S. Soatto, "On the emergence of invariance and disentangling in deep representations," *arXiv:1706.01350*, 2017.

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

[22] C. Louizos, K. Ullrich, and M. Welling, "Bayesian Compression for Deep Learning," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3290–3300, 2017.

[23] Y. L. Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 598–605, 1990.

[24] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *IEEE International Conference on Neural Networks*, pp. 293–299, 1993.

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

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

[27] C. Louizos, M. Welling, and D. P. Kingma, "Learning Sparse Neural Networks through L_0 Regularization," *arXiv:1712.01312*, 2017.

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

[29] M. Courbariaux, Y. Bengio, and J. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *arXiv:1511.00363*, 2015.

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

[31] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv:1605.04711*, 2016.

[32] O. Shayer, D. Levi, and E. Fetaya, "Learning discrete weights using the local reparameterization trick," *arXiv:1710.07739*, 2017.

[33] J. Achterhold, J. M. Köhler, A. Schmeink, and T. Genewein, "Variational network quantization," in *International Conference on Representation Learning (ICLR)*, 2018.

[34] M. Federici, K. Ullrich, and M. Welling, "Improved Bayesian Compression," *arXiv:1711.06494*, 2017.

[35] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Conference on Computational Learning Theory (COLT)*, pp. 5–13, 1993.

[36] H. V. Antti Honkela, "Variational learning and bits-back coding: An information-theoretic view to bayesian learning," 2004.

[37] K. Ullrich, E. Meeds, and M. Welling, "Soft Weight-Sharing for Neural Network Compression," *arXiv:1702.04008*, 2017.

[38] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[39] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," *ITU Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services*, vol. 1, no. 1, pp. 39–48, 2018.

[40] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Communications*, vol. 10, p. 1096, 2019.