Improved Bayesian Compression

Marco Federici
University of Amsterdam
marco.federici@student.uva.nl

Karen Ullrich
University of Amsterdam
karen.ullrich@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
welling.max@gmail.com

1 Variational Bayesian Networks for Compression

Compression of Neural Networks (NN) has become a highly studied topic in recent years. The main reason for this is the demand for industrial scale usage of NNs such as deploying them on mobile devices, storing them efficiently, transmitting them via band-limited channels and most importantly doing inference at scale. There have been two proposals that show strong results, both using empirical Bayesian priors: (i) Ullrich et al. [2017] show impressive compression results by use of an adaptive Mixture of Gaussian prior on independent delta distributed weights. This idea has initially been proposed as *Soft-Weight Sharing* by Nowlan and Hinton [1992] but was never demonstrated to compress before. (ii) Equivalently, Molchanov et al. [2017] use *Variational Dropout* [Kingma et al., 2015] to prune out independent Gaussian posterior weights with high uncertainties. To achieve high pruning rates the authors refined the originally proposed approximations to the KL-divergence and a different parametrization to increase the stability of the training procedure. In this work, we propose to join these two somewhat orthogonal compression schemes since (ii) seems to prune out more weights but does not provide a technique for quantization such as (i). We find our method to outperform both of the above.

2 Method

Given a dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^N$ and a model parametrized by a weight vector $\mathbf{w}$, the learning goal consists in the maximization of the posterior probability of the parameters given the data $p(\mathbf{w}|\mathcal{D})$. Since this quantity involves the computation of intractable integrals, the original objective is replaced with a lower bound obtained by introducing an approximate parametric posterior distribution $q_\phi(\mathbf{w})$. In the *Variational Inference* framework, the objective function is expressed as a *Variational Lower Bound*:

$$\mathcal{L}(\phi) = \underbrace{\sum_{i=1}^{N} \mathbb{E}_{\mathbf{w}\sim q_\phi(\mathbf{w})}\left[\log p(y_n|x_n, \mathbf{w})\right]}_{L_\mathcal{D}(\phi)} - D_{KL}\left(q_\phi(\mathbf{w})\,||p(\mathbf{w})\right) \tag{1}$$

The first term $L_\mathcal{D}(\phi)$ of the equation represents the log-likelihood of the model predictions, while the second part $D_{KL}(q_\phi(\mathbf{w})\,||p(\mathbf{w}))$ stands for the KL-Divergence between the weights approximate posterior $q_\phi(\mathbf{w})$ and their prior distribution $p(\mathbf{w})$. This term works as a regularizer whose effects on the training procedure and tractability depend on the chosen functional form for the two distributions. In this work we propose the use of a joint distribution over the D-dimensional weight vector $\mathbf{w}$ and their corresponding centers $\mathbf{m}$.

Table 1 shows the factorization and functional form of the prior and posterior distributions. Each conditional posterior $q_{\sigma_i}(w_i|m_i)$ is represented with a Gaussian distribution with a variance σ_i^2 around a center m_i defined as a delta peak determined by the parameter θ_i.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Table 1: Choice of probability distribution for the approximate posterior $q_{\boldsymbol{\phi}}\left(\mathbf{w},\mathbf{m}\right)$ and the prior distribution $p_{\boldsymbol{\psi}}\left(\mathbf{w},\mathbf{m}\right)$. $GM_{\boldsymbol{\psi}}\left(m_i\right)$ represent a Gaussian mixture model over m_i parametrized with $\boldsymbol{\psi}$.

Distribution factorization	Functional forms
$q_{\boldsymbol{\phi}}\left(\mathbf{w},\mathbf{m}\right)=\prod_{i=1}^{D}\left(q_{\sigma_i}\left(w_i\|m_i\right)q_{\theta_i}\left(m_i\right)\right)$	$q_{\sigma_i}\left(w_i\|m_i\right)=\mathcal{N}\left(w_i\|m_i,\sigma_i^2\right)$ $q_{\theta_i}\left(m_i\right)=\delta_{\theta_i}\left(m_i\right)$
$p_{\boldsymbol{\psi}}\left(\mathbf{w},\mathbf{m}\right)=\prod_{i=1}^{D}\left(p\left(w_i\right)\ p_{\boldsymbol{\psi}}\left(m_i\right)\right)$	$p\left(w_i\right)\propto 1/\|w_i\|$ $p_{\boldsymbol{\psi}}\left(m_i\right)=GM_{\boldsymbol{\psi}}\left(m_i\right)$

On the other hand, the joint prior is modeled as a product of independent distributions over $\mathbf{w}$ and $\mathbf{m}$. Each $p\left(w_i\right)$ represents a log-uniform distribution, while $p_{\boldsymbol{\psi}}\left(m_i\right)$ is a mixture of Gaussian distribution parametrized with $\boldsymbol{\psi}$ that represents the K mixing proportions $\boldsymbol{\pi}$, the mean of each Gaussian component $\boldsymbol{\mu}$ and their respective precision $\boldsymbol{\lambda}$. In this settings, the KL-Divergence between the prior and posterior distribution can be expressed as:

$$D_{KL}\left(q_{\boldsymbol{\phi}}\left(\mathbf{w},\mathbf{m}\right)||p_{\boldsymbol{\psi}}\left(\mathbf{w},\mathbf{m}\right)\right)=\sum_{i=1}^{D}\left(D_{KL}\left(\mathcal{N}\left(w_i|\theta_i,\sigma_i^2\right)||\frac{1}{|w_i|}\right)+\log GM_{\boldsymbol{\psi}}\left(m_i=\theta_i\right)\right)+C \tag{2}$$

Where $D_{KL}\left(\mathcal{N}\left(w_i|\theta_i,\sigma_i^2\right)||1/|w_i|\right)$ can be effectively approximated as described in Molchanov et al. [2017]. A full derivation can be found in appendix C.

The zero-centered heavy-tailed prior distribution on $\mathbf{w}$ induces sparsity in the parameters vector, at the same time the adaptive mixture model applied on the weight centers $\mathbf{m}$ forces a clustering behavior while adjusting the parameters ψ to better match their distribution. The final expression for the training objective is consequently represented by:

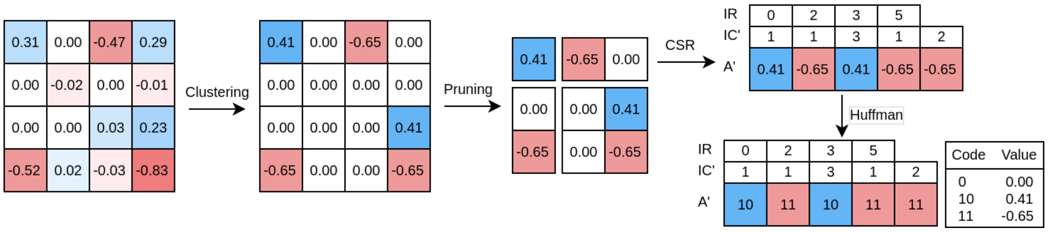
$$\begin{aligned}\mathcal{L}\left(\boldsymbol{\phi},\boldsymbol{\psi}\right)&=L_{\mathcal{D}}\left(\boldsymbol{\phi}\right)-D_{KL}\left(q_{\boldsymbol{\phi}}\left(\mathbf{w},\mathbf{m}\right)||p_{\boldsymbol{\psi}}\left(\mathbf{w},\mathbf{m}\right)\right)\\&=L_{\mathcal{D}}\left(\phi\right)-\sum_{i=1}^{D}\left(D_{KL}\left(\mathcal{N}\left(w_i|\theta_i,\sigma_i^2\right)||\frac{1}{|w_i|}\right)+\log GM_{\boldsymbol{\psi}}\left(m_i=\theta_i\right)\right)\end{aligned}$$

Additional details regarding the optimization procedure and the parameter initialization can be found in appendix A

3 Model Compression

In this section, we briefly describe the post-training model processing, with a focus on the compression scheme. The procedure (that is based on the pipeline described in Han et al. [2015]) is schematically represented in Figure 1 and consists in 4 steps: (i) *Weight Clustering*, (ii) *Model Pruning*, (iii) *Compressed Sparse Row Encoding* and (iv) *Huffman Encoding*.

Figure 1: Graphical representation of the model compression pipeline. First (i) the model weight are clustered into predetermined number of components, then (ii) empty rows and column are removed. (iii) The weight matrices are then converted into the Compressed Sparse Row format and (iv) the floating point representation of the remaining weights is replaced with an optimal Huffman code.



Weight Clustering At the end of the training procedure, the weight posterior means $\boldsymbol{\theta}$ are clustered according to the mixture model defined by the parameters $\boldsymbol{\psi}$. Each parameter θ_i is collapsed into the mean μ_k of the mixture component k that assigns it the highest probability. The resulting approximation doesn't compromise the model performances because the training objective maximizes the likelihood of $\boldsymbol{\theta}$ adapting $\boldsymbol{\psi}$ and vice-versa. Note that one of the mixture component is fixed to 0 to induce sparsity in the weight matrices. Additional details regarding this procedure can be found in Ullrich et al. [2017].

Model Pruning Empty rows and columns in the weight matrices can be removed from the model since they don't affect its prediction. If the j-th row of the weight matrix at layer i is filled with zeros and has a zero bias, it can be removed together with the j-th column of layer $i + 1$. Similarly, if the j-th column of layer i is empty, the j-th row of layer $i - 1$ and the corresponding entry in bias $i - 1$ can be pruned. The same procedure can be applied to convolutional layers by removing the empty channels.

Compressed Sparse Row Encoding The weight matrices can be encoded using more compact representation uses 3 lists to store the non zero entries (A), the cumulative sum of number of non-zero entries for each row (IR) and the column indexes of the non-zero entries (IC). Note that since the matrix are sparse, storing the offset between the columns with a fixed amount of bits (IC') is more efficient than storing the index. See appendix B for more details.

Huffman Encoding Since the non-zero entries can assume at most $K - 1$ different values, we can create an optimal encoding by using an Huffman coding scheme. This decreased the expected number of bits used to store the entries of A' from 32 (or 64 depending on the floating point number representation) to $k \leq \log K + 1$.

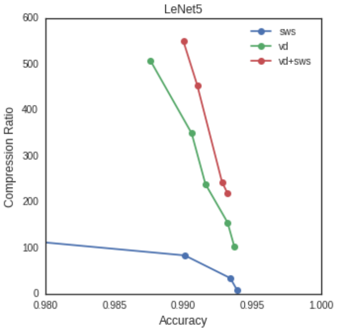
4 Experiments

In preliminary experiments, we compare the compression ratio obtained with the procedure described in the previous section and th accuracy achieved by different Bayesian compression techniques on the well studied MNIST dataset with dense (LeNet300-100) and convolutional (LeNet5) architectures. The details of the training procedure are described in appendix A.

The results presented in table 2 suggests that the joint procedure presented in this paper results in a dramatic increase of the compression ratio, achieving state-of-the-art results for both the dense and convolutional network architectures. Further work could address the interaction between the soft-weight sharing methodology and structured sparsity inducing techniques [Wen et al., 2016, Louizos et al., 2017] to reduce the overhead introduced by the compression format.

Table 2: Test accuracy (Acc), percentage of non-zero weights and compression ratio (CR) evaluated on the MNIST classification task. *Soft-weight sharing* (SWS) [Ullrich et al., 2017] and the *Variational Dropout* (VD) [Molchanov et al., 2017] approaches are compared with the Ridge regression (L2) and the combined approach proposed in this work (VD+SWS). All the accuracies are evaluated on the compressed models. The figure reports the best accuracies and compression ratios obtained by the three different techniques on the LeNet5 architecture by changing the values of the hyper-parameters.

Architecture	Training	Acc [%]	$\frac{\lvert W\neq 0\rvert}{\lvert W\rvert}$ [%]	CR
LeNet300-100	L2	**98.39**	100	1
	SWS	98.16	8.6	34
	VD	98.05	1.6	131
	VD+SWS	98.24	**1.5**	**161**
LeNet5	L2	99.13	100	1
	SWS	99.01	3.6	84
	VD	99.09	0.7	349
	VD+SWS	**99.14**	**0.5**	**482**



References

K. Ullrich, E. Meeds, and M. Welling. Soft Weight-Sharing for Neural Network Compression. *ICLR*, 2017.

Steven J. Nowlan and Geoffrey E. Hinton. Simplifying neural networks by soft weight-sharing. *Neural Comput.*, 4(4):473–493, July 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.4.473. URL `http://dx.doi.org/10.1162/neco.1992.4.4.473`.

D. Molchanov, A. Ashukha, and D. Vetrov. Variational Dropout Sparsifies Deep Neural Networks. *ICML*, January 2017.

D. P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. *ArXiv e-prints*, June 2015.

S. Han, H. Mao, and W. J. Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *ArXiv e-prints*, October 2015.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. *CoRR*, abs/1608.03665, 2016. URL `http://arxiv.org/abs/1608.03665`.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. *NIPS*, 2017.

Appendices

A Training

The full training objective equation is given by the variational lower bound in equation 1, where the two KL-Divergence terms have been scaled according to two coefficients τ_1 and τ_2 respectively:

$$\mathcal{L}\left(\boldsymbol{\phi}, \boldsymbol{\psi}\right) = L_{\mathcal{D}}\left(\phi\right) - \sum_{i=1}^{D} \left(\tau_1 D_{KL}\left(\mathcal{N}\left(w_i|\theta_i, \sigma_i^2\right) || \frac{1}{|w_i|}\right) + \tau_2 \log GM_{\boldsymbol{\psi}}\left(m_i = \theta_i\right)\right) + C$$

Starting from a pre-trained model, in a first warm-up phase we set $\tau_1 = 1$ and $\tau_2 = 0$. Note that this part of the training procedure matches the Sparse Variational Dropout methodology [Molchanov et al., 2017]. After reaching convergence (200 epochs in our experiments), we initialize the parameters $\boldsymbol{\psi}$ for the mixture model and the coefficient τ_2 is set to a value of $2\ 10^{-2}$ (τ_1 is kept to 1) to induce the clustering effect with the Soft Weight sharing procedure. This phase usually requires 50-100 epochs to reach convergence.

The mixture model used for our experiments uses 17 components, one of which has been fixed to zero with a fixed mixing proportion $\pi_0 = 0.999$. A gamma hyper-prior ($\alpha = 10^5$, $\beta = 10$) have been applied to the precision of the Gaussian components to ensure numerical stability.

The proposed parametrization stores the weight variance $\boldsymbol{\sigma}^2$, each mixture component precision $\boldsymbol{\lambda}$ and the mixing proportions $\boldsymbol{\pi}$ in the logarithmic space. All the models have been trained using ADAM optimizer, the learning rates and initialization values for the parameters are reported in Table 3

Table 3: Learning rates and initialization values corresponding to the model parameters. $\mathbf{w}$ represents the weight vector of a pre-trained model, while $\Delta\mu$ represent the distance between the means of the mixing components and it is obtained by dividing two times the standard deviation of the $\mathbf{w}$ distribution by the number of mixing components K. Note that the indexing i goes from 1 to D while k starts from $-\frac{K-1}{2}$ and reaches $\frac{K-1}{2}$.

	θ_i	$\log \sigma_i^2$	μ_k	$\log \lambda_k$	$\log \pi_k$
Initialization	w_i	-10	$k\ \Delta\mu$	$-2\log\left(0.9\ \Delta\mu\right)$	$\begin{cases}\log 0.999 & k=0\\ \log\frac{1-\pi_0}{K} & k\neq 0\end{cases}$
Learning Rate	$5\ 10^{-5}$	10^{-4}	10^{-4}	10^{-4}	$3\ 10^{-3}$

B Details on the Compression Scheme

B.1 Clustering

The first part of the compression pipeline slightly differs for the model trained without a mixture of Gaussian prior (VD in table 2). As suggested in Molchanov et al. [2017], once the model reaches convergence, all the parameters with a binary dropout rate $b_i = \sigma_i^2 / \left(\theta_i^2 + \sigma_i^2\right) \geq t$ are set to 0 ($t = 0.95$ in our experiments). Furthermore, since this methodology does not enforce clustering, in order to fairly compare the final compression ratio, the $\boldsymbol{\theta}$ values are clustered with a 64 mixture components (no training for $\boldsymbol{\theta}$ is involved) and quantized accordingly. The increased number of clusters has been selected to not compromise the performances of the discretized network. The other steps of the pipeline do not differ for the 3 evaluated methodologies.

B.2 Compressed Sparse Row Format

As mentioned in section 3, the Compressed Sparse Row format used in this work stores the column indexes by representing the offset with a fixed amount of bits (5 bits for the LeNet300-100 architecture, while 8 bits for the convolutional LeNet5 version). If the distance between two non-zero entries exceeds the pre-defined number of bits, a zero entry is inserted in A. Note that because of this procedure, the total number of codewords in the Huffman codebook is K. No optimization has been applied to the index array IC since it didn't result into significant improvement. For further details regarding the optimized CSR format, refer to Han et al. [2015] and Ullrich et al. [2017].

C Derivation of the KL-Divergence

In order to compute the KL-divergence between the joint prior and approximate posterior distribution, we use the factorization reported in table 1:

$$\begin{aligned}
-D_{KL}\left(q_{\boldsymbol{\phi}}\left(\mathbf{w},\mathbf{m}\right)||p_{\boldsymbol{\psi}}\left(\mathbf{w},\mathbf{m}\right)\right) &= \int\int q_{\boldsymbol{\phi}}\left(\mathbf{w},\mathbf{m}\right)\log\frac{p_{\boldsymbol{\psi}}\left(\mathbf{w},\mathbf{m}\right)}{q_{\boldsymbol{\phi}}\left(\mathbf{w},\mathbf{m}\right)}d\mathbf{w}\ d\mathbf{m}\\
&= \sum_{i=1}^{D}\int\int q_{\sigma_i}\left(w_i|m_i\right)\ q_{\theta_i}\left(m_i\right)\log\frac{p\left(w_i\right)p_{\boldsymbol{\psi}}\left(m_i\right)}{q_{\sigma_i}\left(w_i|m_i\right)\ q_{\theta_i}\left(m_i\right)}dm_i\ dw_i\\
&= \sum_{i=1}^{D}\int\left(\int q_{\sigma_i}\left(w_i|m_i\right)q_{\theta_i}\left(m_i\right)\log\frac{p\left(w_i\right)}{q_{\sigma_i}\left(w_i|m_i\right)}dm_i\right)dw_i\\
&\quad+\sum_{i=1}^{D}\int\left(\int q_{\sigma_i}\left(w_i|m_i\right)\ dw_i\right)q_{\theta_i}\left(m_i\right)\log\frac{p_{\boldsymbol{\psi}}\left(m_i\right)}{q_{\theta_i}\left(m_i\right)}dm_i
\end{aligned}$$

By plugging in $q_{\theta_i}(m_i) = \delta_{\theta_i}(m_i)$ and observing $\int q_{\sigma_i}(w_i|m_i)\ dw_i = 1$, we obtain:

$$\begin{aligned}
-D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{w},\mathbf{m})\,||p_{\boldsymbol{\psi}}(\mathbf{w},\mathbf{m})\right) &= \sum_{i=1}^{D}\int q_{\sigma_i}(w_i|m_i=\theta_i)\log\frac{p(w_i)}{q_{\sigma_i}(w_i|m_i=\theta_i)}dw_i \\
&+ \sum_{i=1}^{D}\int\ q_{\theta_i}(m_i)\log\frac{p_{\psi}(m_i)}{q_{\theta_i}(m_i)}dm_i \\
&= -\sum_{i=1}^{D}\left(D_{KL}\left(q_{\sigma_i}(w_i|m_i=\theta_i)\,||p(w_i)\right) + D_{KL}\left(q_{\theta_i}(m_i)\,||p_{\boldsymbol{\psi}}(m_i)\right)\right)
\end{aligned} \tag{3}$$

Where the first KL-divergence can be approximated according to Molchanov et al. [2017]:

$$\begin{aligned}
D_{KL}\left(q_{\sigma_i}(w_i|m_i=\theta_i)\,||p(w_i)\right) &= D_{KL}\left(\mathcal{N}\left(w_i|\theta_i,\sigma_i^2\right)||\frac{1}{|w_i|}\right) + C \\
&\approx k_1\sigma\left(k_2 + k_3\log\frac{\sigma_i^2}{\theta_i^2}\right) - 0.5\log\left(1+\frac{\theta_i^2}{\sigma^2}\right) + C
\end{aligned} \tag{4}$$

$$\text{With} \quad k_1 = 0.63576, \quad k_2 = 1.87320, \quad k_3 = 1.48695$$

While the second term can be computed by decomposing the KL-divergence into the entropy of the approximate posterior $\mathcal{H}\left(q_{\theta_i}(m_i)\right)$ and the cross-entropy between prior and posterior distributions $\mathcal{H}\left(q_{\theta_i}(m_i), p_{\boldsymbol{\psi}}(m_i)\right)$. Note that the entropy of a delta distribution does not depend on the parameter θ_i, therefore it can be considered to be constant.

$$\begin{aligned}
D_{KL}\left(q_{\theta_i}(m_i)\,||p_{\boldsymbol{\psi}}(m_i)\right) &= -\mathcal{H}\left(q_{\theta_i}(m_i)\right) + \mathcal{H}\left(q_{\theta_i}(m_i), p_{\boldsymbol{\psi}}(m_i)\right) \\
&= \underbrace{-\mathcal{H}\left(\delta_{\theta_i}(m_i)\right)}_{C} - \int\log p_{\boldsymbol{\psi}}(m_i)\,\delta_{\theta_i}(m_i)\,dm_i \\
&= -\log p_{\boldsymbol{\psi}}(m_i=\theta_i) + C \\
&= \log GM\left(m_i=\theta_i|\boldsymbol{\psi}\right) + C \\
&= \log\sum_{k=1}^{K}\pi_k\,\mathcal{N}\left(m_i=\theta_i|\mu_k,\lambda_k^{-1}\right) + C
\end{aligned} \tag{5}$$

Plugging-in the results from equations 4 and 5 into equation 3 we obtain the full expression for the KL-divergence reported in equation 2.