LEARNING SPARSE AUTO-ENCODERS FOR GREEN AI IMAGE CODING

Cyprien Gille *, Frédéric Guyard †, Marc Antonini *, member IEEE, and Michel Barlaud *, Fellow IEEE

* Université Côte d’Azur, I3S, CNRS, Sophia Antipolis, France
† Orange Labs, Sophia Antipolis, France

ABSTRACT

Recently, convolutional auto-encoders (CAE) were introduced for image coding. They achieved performance improvements over the state-of-the-art JPEG2000 method. However, these performances were obtained using massive CAEs featuring a large number of parameters and whose training required heavy computational power.

In this paper, we address the problem of lossy image compression using a CAE with a small memory footprint and low computational power usage. In this work, we propose a constrained approach and a new structured sparse learning method. We design an algorithm and test it on three constraints: the classical $\ell_1$ constraint, the $\ell_{1,\infty}$ and the new $\ell_{1,1}$ constraint. Experimental results show that the $\ell_{1,1}$ constraint provides the best structured sparsity, resulting in a high reduction of memory (82%) and computational cost reduction (25%), with similar rate-distortion performance as with dense networks.

I. INTRODUCTION

Since Balle’s [1] and Theis’ works [2] in 2017, most new lossy image coding methods use convolutional neural networks, such as convolutional autoencoders (CAE) [3], [4], [5]. CAEs are discriminating models that map feature points from a high dimensional space to points in a low dimensional latent space. They were introduced in the field of neural networks several years ago, their most efficient application at the time being dimensionality reduction and denoising [6]. One of the main advantages of an autoencoder is the projection of the data in the low dimensional latent space: when a model properly learns to construct a latent space, it naturally identifies general, high-level relevant features. In a lossy image coding scheme, the latent variable is losslessly compressed using entropy coding solutions, such as the well-known arithmetic coding algorithm. End-to-end training of a CAE coding scheme reaches image coding performances competitive with JPEG 2000 (wavelet transform and bit plane coding) [7]. These are compelling results, as JPEG 2000 represents the state-of-the-art for standardized image compression algorithms\(^1\). Autoencoder-based methods specifically are becoming more and more effective: In a span of a few years, their performances have gone from JPEG to JPEG 2000. Considering the performances of these new CAEs for image coding, the JPEG standardization group has introduced the study of a new machine learning-based image coding standard, JPEG AI\(^2\). Note that the performances of these CAEs are achieved at the cost of a high complexity and large memory usage. In fact, energy consumption is the main bottleneck for running CAEs while respecting an energy footprint or carbon impact constraint [8], [9]. Fortunately, it is known that CAEs are largely over-parameterized, and that in practice relatively few network weights are necessary to accurately learn image features.

Since 2016, numerous methods have been proposed in order to remove network weights (weight sparsification) during the training phase [10], [11]. These methods generally do produce sparse weight matrices, unfortunately with random sparse connectivity. To address this issue, many methods based on LASSO, group LASSO and exclusive LASSO were proposed [12], [13], [14] in order to simultaneously sparsify neurons and enforce parameter sharing. However, all proximal regularization methods quoted above require the computation of the LASSO path, which is time consuming [15]. In order to deal with this issue, we proposed instead a constrained approach in [16], where the constraint is directly related to the number of zero-weights.

In this work, we extend the aforementioned approach to a CAE in the context of image coding in order to reduce its memory requirement and computational footprint while keeping the best rate-distortion trade-off as possible. We designed an algorithm for the sparsification of the CAE in section II with three constraints in mind: the classical $\ell_1$ constraint, the $\ell_{1,\infty}$ constraint [17] and the structured $\ell_{1,1}$ constraint [18],[19].

In section III, we present experimental results with sparsification of the encoder. Finally, section IV concludes the paper and provides a discussion and some perspectives.

II. LEARNING A SPARSE AUTOENCODER USING A STRUCTURED CONSTRAINT

Figure 1 shows the architecture of a CAE network where $X$ is the input data, $Z$ the latent variable and $\hat{X}$ the reconstructed

\(^1\)https://jpeg.org/jpeg2000/index.html

\(^2\)https://jpeg.org/jpegai/index.html
The main difference with the criterion proposed in [2] is the following convex criterion 2 (see [18] for more details):

\[ \ell_1(\|v\|_1) = \begin{cases} \|v\| & \text{if } \|v\| \leq \eta \\ \eta & \text{otherwise} \end{cases} \]

Thus we propose the new \( \ell_{1,1} \) projection as follows. We first compute the radius \( t_i \) and then project the rows using the \( \ell_{1,1} \) adaptive constraint \( t_i \). Note that the proposed new \( \ell_{1,1} \) projection algorithm 1 corresponds to minimizing the following convex criterion 2 (see [18] for more details):

\[
\min_{t_i \leq \eta} \sum_{i=1}^{d} \left( \sum_{j=1}^{k} |v_{i,j} - t_i| \right)^2 + \varepsilon \sum_{i=1}^{d} \sum_{j=1}^{k} (v_{i,j} - w_{i,j})^2.
\]

Note that in the case of a CAE, contrarily to fully connected networks, \( W \) originally corresponds to a tensor instead of a matrix. We thus need to flatten the inside dimensions of the weight tensors to turn them into two-dimensional arrays. We then run the double descent Algorithm 2 [21], [22] where instead of the weight thresholding done by state-of-the-art algorithms, we use our \( \ell_{1,1} \) projection from Algorithm 1.

**Algorithm 1** Projection of the \( l \times d \) matrix \( V \) onto the \( \ell_{1,1} \)-ball of radius \( \eta \).

**Input**: \( V, \eta \)

**for** \( i = 1, \ldots, d \) **do**

\[ t_i := \text{proj}_{\ell_1}(\|v\|_1) \]

\[ w_i := \text{proj}_{\ell_1}(v_i, t_i) \]

**end for**

**Output**: \( W \)

**Algorithm 2** Double descent algorithm. \( \phi \) is the total loss as defined in (1), \( \nabla \phi(W, M_0) \) is the gradient masked by the binary mask \( M_0 \). \( A \) is the Adam optimizer, \( N \) is the total number of epochs and \( \gamma \) is the learning rate.

**# First descent**

**Input**: \( W_{init}, \gamma, \eta \)

**for** \( n = 1, \ldots, N \) **do**

\[ W \leftarrow A(W, \gamma, \nabla \phi(W)) \]

**end for**

**# Projection**

**for** \( i = 1, \ldots, d \) **do**

\[ t_i := \text{proj}_{\ell_1}(\|v\|_1) \]

\[ w_i := \text{proj}_{\ell_1}(v_i, t_i) \]

**end for**

\( (M_0)_{ij} := \mathbb{1}_{x \neq 0}(w_{ij}) \)

**Output**: \( M_0 \)

**# Second descent**

**Input**: \( W_{init}, M_0, \gamma \)

**for** \( n = 1, \ldots, N \) **do**

\[ W \leftarrow A(W, \gamma, \nabla \phi(W, M_0)) \]

**end for**

**Output**: \( W \)

Our implementation of algorithm 2 as well as all training code is available at 3.

**III. EXPERIMENTAL RESULTS**

The proposed method was implemented in PyTorch using the python code implementation of a convolutional autoencoder proposed in [2]4. Note that the classical computational cost measure evaluates FLOPS (floating point operations per second), in which additions and multiplications are counted separately. However, a lot of modern hardware can compute the multiply-add operation in a single instruction. Therefore, we instead use MACCs (multiply-accumulate)

3https://github.com/CyprienGille/Sparse-Convolutional-AutoEncoder

4https://github.com/alexandru-dinu/cae
operations) as our computational cost measure (multiplication and addition are counted as a single instruction\(^5\)).

We trained the compressive autoencoder on 473 \(2048 \times 2048\) images obtained from Flickr\(^8\), divided into \(128 \times 128\) patches. We use the 24-image Kodak PhotoCD dataset for testing \(^7\). All models were trained using 8 cores of an AMD EPYC 7313 CPU, 128GB of RAM and an NVIDIA A100 GPU (40GiB of HMB2e memory, 1.5TB/s of bandwidth, 432 Tensor Cores). Performing 100 Epochs of training takes about 5 hours.

We choose as our baseline a CAE network trained using the classical Adam optimizer in PyTorch, and compared its performance (relative MACCs and loss as a function of sparsity, PSNR and Mean SSIM as a function of the bitrate) to our masked gradient optimizer with \(\ell_1\), \(\ell_{1,1}\) and \(\ell_{1,\infty}\) constraints. For the \(\ell_{1,\infty}\) projection, we implemented the "Active Set" method from \([17]\). For the PSNR function, we used its implementation in CompressAI \([23]\)^6. Note that MSSIM refers to the Mean Structural SIMilarity, as introduced in \([24]\). Considering that the high image quality and low distortions are difficult to assess within an article, we provide here a link to our website (https://www.i3s.unice.fr/~barlaud/Demo-CAE.html) so that the reader can download the images and evaluate their quality on a high definition screen. From now on, we use \(S\) to denote the encoder sparsity proportion, i.e. the ratio of zero-weights over the total number of weights in the encoder.

In this experiment, we only sparsify the encoder layers of the CAE. This is a practical case, where the power and memory limitations apply mostly to the sender as is the case for satellites \([25]\), \([26]\), drones, or cameras used to report from an isolated country.

Figure 2 displays the relative number of MACCs with respect to the aforementioned baseline of a non-sparsified network, as a function of the density \(1 - S\). The \(\ell_1\) constraint does not provide any computational cost improvement while the \(\ell_{1,1}\) constraint significantly reduces MACCs. This is due to the fact that the \(\ell_{1,1}\) constraint (contrarily to \(\ell_1\)) creates a structured sparsity \([20]\), setting to zero groups of neighbouring weights, inhibiting filters and thus pruning operations off the CAE. The \(\ell_{1,\infty}\) constraint reduces MACCs even more, but comes with a reduction of the performance of the network.

Let us now define the relative PSNR loss with respect to the reference without projection as:

\[
10 \times (\log_{10}(MSE_{ref}) - \log_{10}(MSE_L)) \text{ in } dB
\]

for a constraint \(L = \ell_1\), \(L = \ell_{1,1}\), or \(L = \ell_{1,\infty}\).

Table I shows the relative loss of the different models for a value of the sparsity \(S \) around 83\%. The table shows that the \(\ell_{1,1}\) constraint leads to a slightly higher loss than \(\ell_1\). However, this slight decrease in performance comes with a significant decrease in computational cost (30\% less MACCs for a sparsity of 86\%), as shown in Figure 2.

The figures 3 and 4 show a slight PSNR loss of less than 1dB, and similarly close MSSIM scores. This loss translates perceptually into a slight reinforcement of the image grain, which is more noticeable for projection \(\ell_{1,\infty}\).

\[\text{Fig. 2: Projection on encoder layers : Relative MACCs as a function of the density.}\]

| Constraint | \(\ell_1\) | \(\ell_{1,1}\) | \(\ell_{1,\infty}\) |
|------------|-----------|-------------|-------------|
| \(S\) (%) | 82.12 | 83.05 | 84.77 |
| MACCs reduction | 0 | 27 | 40 |
| Memory reduction | 81 | 82 | 64 |
| Relative Loss (dB) | -1.15 | -1.2 | -1.7 |

We then display the bitrate-distortion curves for the CAEs from Table I.

\[\text{Fig. 3: Projection on encoder layers : PSNR as a function of the bitrate for decoded Kodak test images}\]

The aim of this study was not to present a new compressive network with better performance than state-of-the-art networks, but rather to prove that, for any given network,

---

\(^5\)https://machinethink.net/blog/how-fast-is-my-model/

\(^6\)https://github.com/CyprienGille/flickr-compression-dataset

\(^7\)http://www.r0k.us/graphics/kodak/

\(^8\)https://github.com/InterDigitalInc/CompressAI
the $\ell_{1,1}$ constraint and the double descent algorithm can be used as a way to efficiently and effectively reduce both storage and power consumption costs, with minimal impact on the network’s performance. We focus in this paper on high-quality image compression. In satellite imagery, the available energy is a crucial resource, making the image sender a prime benefactor of energy-sparing compression solutions. Our sparcification method using the the $\ell_{1,1}$ constraint on the encoder layers, i.e. the on-board segment of the network led to almost the same loss while reducing MACCs and energy consumption by 25% and memory requirements by 82.03% respectively.

Figure 6 shows that the CAE with low energy consumption outperforms JPEG and JPEG2K by at least 4dB at high bit-rates (4bpp). Note that our CAE was optimized for high bit-rates: we can see the translation of the 47 dB of PSNR to unperceiveable differences on test images in figure 7. In this case (bitrate around 4bpp), we can say that we have a near lossless compression even while reducing the energy and memory costs of the network.

Other examples include drone imagery, media transmission from a remote or hazardous country. This result is also very encouraging in a time where the ecological impact of neural networks is considerable [8], [27].

Both projection $\ell_1$ and $\ell_{1,1}$ decrease the memory footprint of the network, but only the $\ell_{1,1}$ constraint decreases its computational cost without degrading the performance. We have shown the interest of our method of sparsification to reduce the energy and memory costs of a CAE network. Further works will include application of our sparcification technique to state-of-the-art compression models [4]. Sparsification of all layers of the network is reported in [28].
V. REFERENCES

[1] J. Ballé, V. Laparra, and E. Simoncelli, “End-to-end optimized image compression,” ICLR Conference Toulon France, 2017.
[2] L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy image compression with compressive autoencoders,” ICLR Conference Toulon, 2017.
[3] J. Ballé, D. Minnen, S. Singh, J. Hwang, and N. Johnston, “End-to-end optimized image compression,” ICLR Conference Vancouver Canada, 2018.
[4] D. Minnen, J. Ballé, and G. Toderici, “Joint autoregressive and hierarchical priors for learned image compression,” NEURIPS, Montreal Canada, 2018.
[5] F. Mentzer, G. Toderici, M. Tschannen, and E. Agustsson, “High-fidelity generative image compression,” NEURIPS, 2020.
[6] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion,” J. Mach. Learn. Res., vol. 11, pp. 3371–3408, 2010.
[7] C. Zhengxue, S. Heming, T. Masaru, and K. Jiro, “Deep convolutional autoencoder-based lossy image compression,” arXiv:1804.09535v1 [cs.CV] April, 2018.
[8] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, “Green ai,” 2019.
[9] D. Patterson, J. Gonzalez, U. Hölzle, Q. H. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, “The carbon footprint of machine learning training will plateau, then shrink,” 2022.
[10] E. Tartaglione, S. Lepsøy, A. Fiandrotti, and G. Francini, “Learning sparse neural networks via sensitivity-driven regularization,” in Advances in Neural Information Processing Systems, 2018, pp. 3878–3888.
[11] H. Zhou, J. M. Alvarez, and F. Porikli, “Less is more: Towards compact cnns,” in European Conference on Computer Vision. Springer, 2016, pp. 662–677.
[12] J. M. Alvarez and M. Salzmann, “Learning the number of neurons in deep networks,” in Advances in Neural Information Processing Systems, 2016, pp. 2270–2278.
[13] Z. Huang and N. Wang, “Data-driven sparse structure selection for deep neural networks,” in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 304–320.
[14] U. Oswal, C. Cox, M. Lambon-Ralph, T. Rogers, and R. Nowak, “Representational similarity learning with application to brain networks,” in International Conference on Machine Learning, 2016, pp. 1041–1049.
[15] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, “The entire regularization path for the support vector machine,” Journal of Machine Learning Research, vol. 5, pp. 1391–1415, 2004.
[16] M. Barlaud, W. Belhajali, P. Combettes, and L. Fillatre, “Classification and regression using an outer approxima-