# Quantization-Aware Regularizers for Deep Neural Networks Compression

1st Dario Malchiodi
*Dipartimento di Informatica*
*Università degli Studi di Milano*
Milan, Italy
dario.malchiodi@unimi.it

2nd Mattia Ferraretto
*Dipartimento di Informatica*
*Università degli Studi di Milano*
Milan, Italy
ferraretto.mattia@proton.me

3rd Marco Frasca
*Dipartimento di Informatica*
*Università degli Studi di Milano*
Milan, Italy
marco.frasca@unimi.it

***Abstract*—Deep Neural Networks reached state-of-the-art performance across numerous domains, but this progress has come at the cost of increasingly large and over-parameterized models, posing serious challenges for deployment on resource-constrained devices. As a result, model compression has become essential, and—among compression techniques—weight quantization is largely used and particularly effective, yet it typically introduces a non-negligible accuracy drop. However, it is usually applied to already trained models, without influencing how the parameter space is explored during the learning phase. In contrast, we introduce per-layer regularization terms that drive weights to naturally form clusters during training, integrating quantization awareness directly into the optimization process. This reduces the accuracy loss typically associated with quantization methods while preserving their compression potential. Furthermore, in our framework quantization representatives become network parameters, marking, to the best of our knowledge, the first approach to embed quantization parameters directly into the backpropagation procedure. Experiments on CIFAR-10 with AlexNet and VGG16 models confirm the effectiveness of the proposed strategy.**

***Index Terms*—DNN compression, weight quantization, weight sharing, quantization-aware regularization**

## I. Introduction

Over the past decade, Deep Neural Networks (DNNs) have achieved state-of-the-art performance across a wide range of complex tasks, spanning from computer vision [1] to natural language processing [2]. This rapid progress has come at the cost of exponentially increasing model complexity, with modern architectures often containing millions or even billions of parameters. Yet, despite this growth in scale, trained networks commonly exhibit substantial parameter redundancy [3]. Such over-parameterization leads to large memory footprints, high energy consumption, and increased computational latency, which makes deploying powerful models on resource-limited edge devices increasingly infeasible. As a result, DNN compression has become essential, aiming to lower the resource requirements of learned models [4], [5]. Prominent techniques include connection pruning [4], filter pruning [6], weight and activation quantization [7], low-rank factorization [8], and knowledge distillation [9]. In particular, *weight quantization* denotes reducing the number of bits used in order to represent network weights, for instance by lowering floating-point precision [10], down to the extreme of a single bit [11]. Weights can be quantized either by directly mapping them to the nearest value in a deterministic or stochastic manner [7], or by optimizing a given metric, such as entropy [12]. Resource savings can also be achieved *indirectly* through weight sharing, where only a small set of full-precision representative weights is stored and reused across multiple weight clusters [5], [13]–[15]. In this case, compression is obtained by employing suitable lossless encoding schemes that exploit the low-entropy distribution induced by weight quantization [13], [16]. Most existing methods, however, perform such quantization after standard gradient-based updates, even when quantization is used during training [10]. Only a limited number of works incorporate a quantization-aware objective directly into the loss, and these typically alternate between steps dedicated to accuracy and steps dedicated to compression [17].

In this work, we propose per-layer weight regularization loss terms specifically designed to drive weights to 'naturally' form clusters during DNN training, directly via gradient com-

M.F. and D.M. have been partially supported by the Italian MUR PRIN project "Multicriteria data structures and algorithms: from compressed to learned indexes, and beyond" (Prot. 2017WR7SHH). Additional support to D.M. has been granted by the National Plan for NRRP Complementary Investments (PNC) in the call for the funding of research initiatives for technologies and innovative trajectories in the health—project n. PNC0000003—AdvaNced Technologies for Human-centrEd Medicine (project acronym: ANTHEM).

putation and without any dedicated post-update steps. Notably, both previously discussed weight quantization families incur an accuracy loss compared to the uncompressed model. In contrast, our method—also compatible with those approaches—explores the parameter space while explicitly considering the subsequent quantization, thereby better limiting performance degradation while maintaining equivalent compression capability. Moreover, most existing DNN quantization techniques fix *a priori* the quantization levels or the representative values used for weight sharing [5], [12]. In contrast, we introduce a framework in which these representatives are learned jointly with the other DNN parameters during training. To the best of our knowledge, this is the first method that incorporates quantization parameters directly into the model as trainable variables. This is particularly relevant given recent evidence that most weights in convolutional and fully connected layers follow a bell-shaped distribution, which makes selecting appropriate quantization levels/representatives non-trivial [18]. Experimental results on the CIFAR-10 dataset using two popular architectures, AlexNet [19] and VGG16 [20], demonstrate the effectiveness of the proposed approach. The paper is organized as follows: Section II presents the proposed methodology, including the notation used and the preliminary definitions, while Section III describes the experimental results. Some concluding remarks end the paper.

## II. Methods

In this section, we present the notation, basic definitions, and four quantization-aware loss regularizers.

### A. Notation

Matrices and vectors are written in bold, such as $\boldsymbol{W}$ and $\boldsymbol{x}$, with their entries indicated by subscripts, e.g., $w_{ij}$ or $x_i$. The symbol $:=$ is used specifically to denote definitions, distinguishing them from standard equalities.

### B. Preliminary Definitions

A DNN represents a function $f_{\boldsymbol{W}} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_L}$ that maps an input vector $\boldsymbol{x} \in \mathbb{R}^{d_0}$ to an output vector $f_{\boldsymbol{W}}(\boldsymbol{x}) \in \mathbb{R}^{d_L}$, parameterized by $\boldsymbol{W}$. The parameter set is $\boldsymbol{W} = \{\boldsymbol{W}^{(1)}, \boldsymbol{W}^{(2)}, \ldots, \boldsymbol{W}^{(L)}\}$, where $\boldsymbol{W}^{(l)}$ is the weight matrix of the $l$-th layer with dimension $d_l$, for $l \in \{1, \ldots, L\}$, and $L$ is the total number of layers. In our framework, layers may be fully connected, convolutional, recurrent, pooling, dropout, or any other type. DNNs are inherently data-driven models designed to infer patterns from large sets of examples. Consider a dataset $\mathcal{D} := \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ of $N$ input–output pairs, where $\boldsymbol{x}_i \in \mathbb{R}^{d_0}$ and $\boldsymbol{y}_i \in \mathbb{R}^{d_L}$. The objective is to learn parameters $\boldsymbol{W}$ such that $f_{\boldsymbol{W}}$ closely captures the mapping from inputs $\boldsymbol{x}_i$ to their ground-truth targets $\boldsymbol{y}_i$. To this end, a loss function is used to evaluate performance by measuring how far the model's predictions deviate from the true labels in the dataset. Formally, training consists of minimizing the loss function $\mathcal{L}$:

$$\mathcal{L}(\boldsymbol{W}, \mathcal{D}) := \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} err(f_{\boldsymbol{W}}(\boldsymbol{x}), \boldsymbol{y}),$$

where $err : \mathbb{R}^{d_L} \times \mathbb{R}^{d_L} \to \mathbb{R}$ denotes a task-dependent loss (e.g., cross-entropy in classification). To compute

$$\hat{\boldsymbol{W}} := \arg\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \mathcal{D}), \tag{1}$$

modern deep learning approaches typically use gradient-based optimization together with backpropagation [21]. Recent work has revealed that most weights in convolutional and fully connected layers cluster around zero, forming an approximately bell-shaped distribution [18]. This distribution undermines post-training compression via weight quantization, since it leads to large approximation errors for many weights [12]. To address this issue, we propose two static and two dynamic weight regularization schemes explicitly designed to promote quantization-friendly weight distributions.

### C. Quantization-Aware Weight Regularization

Building on established regularization methods, we aimed to design new techniques tailored to network quantization. Our goal is a regularizer that mimics L1 penalization to support network pruning by driving some weights toward $0$ during training, enabling later magnitude-based pruning with a chosen threshold. Similarly, we hypothesized that an additional, tailored regularization term could drive weights to concentrate around preset values, rendering the network inherently "quantization-friendly" by the end of training. Consequently, the optimization objective extends (1) as follows:

$$\hat{\boldsymbol{W}} := \arg\min_{\boldsymbol{W}} \mathcal{L}(\boldsymbol{W}, \mathcal{D}) + \lambda\mathcal{R}(\boldsymbol{W}), \tag{2}$$

where $\mathcal{R}$ is the loss regularization term intended to create "basins of attraction" within a predefined weight range, whereas $\lambda > 0$ is a regularization parameter. $\mathcal{R}$ should tend to $0$ for weights located near these basins and increase as weights move away from them. This encourages weights to cluster naturally during training. Such a mechanism can help the network explore the parameter space more effectively

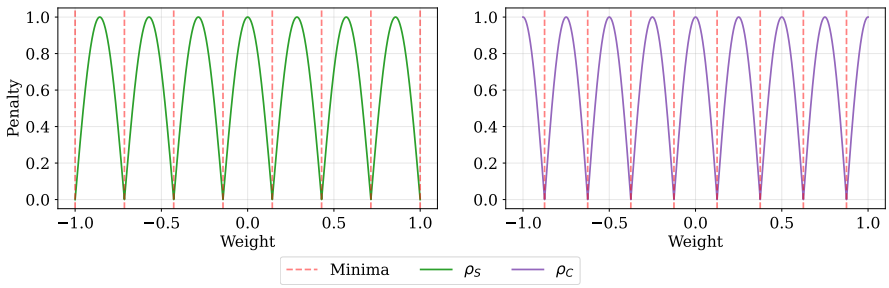


Figure 1: $\rho_S$ (left) and $\rho_C$ (right) penalty functions in the interval $[w = -1, W = 1]$ for $K = 8$.

than unregularized training, particularly when a quantization procedure will be applied afterward.

*1) Periodic Quantization-Aware Weight Regularization:* In the first approach, we introduce two periodic regularizers based on trigonometric functions. These regularizers produce a fixed number of basins, corresponding to the minima of the regularization term, which are evenly spaced across the weight interval. The first quantization-aware regularizer utilizes the absolute value of a sine function to create $K$ periodic minima. The sine regularizer $\mathcal{R}_S$ is defined as:

$$\mathcal{R}_S(\boldsymbol{W}, K, w, W) := \sum_{l=1}^{L} \frac{\sum_{i=1}^{d_{l-1}} \sum_{j=1}^{d_l} \rho_S(w_{ij}^{(l)}, K, w, W)}{d_{l-1} d_l},$$

with

$$\rho_S(\bar{w}, K, w, W) := \left| \sin \frac{\pi(K-1) \cdot (\bar{w} - w)}{W - w} \right|,$$

$K$ the number of minima desired, $w$ and $W$ the minimum and maximum weight allowed, respectively. As already mentioned, since the weights in DNN typically follow a bell distribution, a priori determining $w$ and $W$ does not represent a limitation (extremes need only to not be to close). The minima are evenly distributed in $[w, W]$, and the lower $K$, the stronger the quantization we are demanding to the network, the higher the subsequent potential DNN compression (Figure 1, left). It is worth noting that, to simplify the discussion, we are assuming the weight range to be the same across layers, but clearly we can easily tailor them to the specific layer type.

Similarly, the quantization-aware regularizer $\mathcal{R}_C$ leverages the periodicity of the cosine function and is defined as follows:

$$\mathcal{R}_C(\boldsymbol{W}, K, w, W) := \sum_{l=1}^{L} \frac{\sum_{i=1}^{d_{l-1}} \sum_{j=1}^{d_l} \rho_C(w_{ij}^{(l)}, K, w, W)}{d_{l-1} d_l},$$

where $\rho_C(\bar{w}, K, w, W) := \left| \cos \frac{\pi K(\bar{w} - w)}{W - w} \right|$. The main difference lies in the phase of the periodic function, which shifts the locations of the static basins of attraction, and the most external minima now do not lie exactly on $w$ and $W$ (Figure 1, right). The functions $\rho_S$ and $\rho_C$ have the advantage to be rapidly computed, since they do not require any additional parameter. However they have two main limitations: the static nature of their minima, which requires a predefined reference weight range, and the fact that these minima are evenly distributed in the interval $[w, W]$. Since the weights typically follow a bell-shaped distribution nearly centered at 0, it would be preferable to have basins that can be adapted to the local weight distribution. To address these limitations, we propose two solutions that replace static basins of attraction with learnable ones.

### *D. Dynamic and Learnable Quantization-Aware Weight Regularization*

The rationale behind this second category of quantization-friendly weight regularizers is to make the minima of the regularization function learnable during model training. To this end, we introduce $K$ additional learnable parameters $\boldsymbol{u} := (u_1, \ldots, u_K)$, which act as adaptive basins of attraction for the weights. In this framework, instead of pulling weights toward fixed, predefined values, we encourage them to cluster around $\boldsymbol{u}$. These parameters are updated at each iteration of the training algorithm, via the backpropagation procedure, just like the other model parameters $\boldsymbol{W}$, and the regularizer penalizes weights according to their difference with $u_1, \ldots, u_K$. From this standpoint, they slightly increase the complexity with respect to $R_S$ and $R_C$ regularizers (linearly with $K$). The first regularizer in this category, $\mathcal{R}_M$, aims at minimizing the minimum squared difference of weights with any of the minima $\boldsymbol{u}$. It is formally defined as it follows

$$\mathcal{R}_M(\boldsymbol{W}, \boldsymbol{u}) := \sum_{l=1}^{L} \frac{\sum_{i=1}^{d_{l-1}} \sum_{j=1}^{d_l} \rho_M(w_{ij}^{(l)}, \boldsymbol{u})}{d_{l-1} d_l},$$

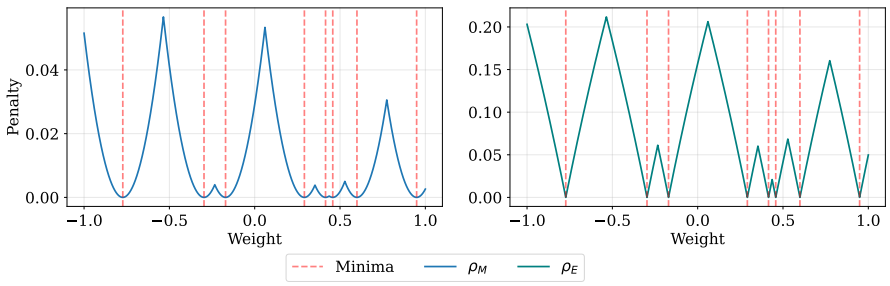


Figure 2: $\rho_M$ and $\rho_E$ functions in the interval [-1, 1] when minima ($\boldsymbol{u}$ assignment) are those denoted by the vertical dashed lines (K=8).

with $\rho_M(\bar{w}, \boldsymbol{u}) := \min_{r\in\{1,\dots,K\}}(\bar{w} - u_r)^2$. By minimizing the minimum distance to minima $\boldsymbol{u}$, we force each weight to fall in one of the clusters which will be formed during the training and relative to one representative in $\boldsymbol{u}$. Indeed, the penalty increases quadratically with the distance from such a minimum, strongly encouraging weights to cluster tightly. Figure 2 (left) illustrates the shape of $\rho_M$ for a given assignment of $\boldsymbol{u}$. The function is, in principle, unbounded, and its peaks can vary in magnitude depending on the distance from the nearest minimum. On the other side, when two minima lie very close to each other, weights falling between them may produce peaks that are too low to significantly influence the optimization process (as illustrated by the low peak in the figure). To address this issue, the second regularization function in this category, $\mathcal{R}_E$, leverages an exponential formulation to both bound the regularization term and amplify the penalty for weights that fall between two closely spaced minima. Formally, the $\mathcal{R}_E$ weight regularization is

$$\mathcal{R}_E(\boldsymbol{W}, \boldsymbol{u}) = \sum_{l=1}^{L} \frac{\sum_{i=1}^{d_{l-1}} \sum_{j=1}^{d_l} \rho_E(w_{ij}^{(l)}, \boldsymbol{u})}{d_{l-1} d_l} ,$$

where $\rho_E(\bar{w}, \boldsymbol{u}) := 1 - \exp\left(-\min_{r\in\{1,\dots,K\}} |\bar{w} - u_r|\right)$. The $\rho_E$ penalty is bounded between 0 and 1, and it approaches 1 asymptotically as the minimum distance increases. The use of the absolute difference also ensures that weight deviations smaller than 1 have a stronger effect than in the quadratic formulation. As shown in Figure 2 (right), for the same configuration of the $\boldsymbol{u}$ parameters (vertical dashed lines), the peaks between closely spaced minima now exhibit a much higher relative magnitude compared with the $\rho_M$ case. It is relevant to mention that, although the minimum function is not differentiable at every point, this does not create practical difficulties for gradient-based optimization. The only points where the derivative is not uniquely defined are those where two or more terms achieve the same minimum value. Modern optimization frameworks handle these cases automatically by using subgradients; since such non-differentiable points are isolated, this do not affect the overall optimization dynamics.

### *E. Quantization via Weight Sharing*

Once the weights are quantized, the storage requirements can be reduced in two main ways: either by decreasing the number of bits used to represent each weight, or by storing the representative weights in full precision and sharing them within each group through lossless structures that exploit the low entropy introduced by quantization [16]. We focus on the weight sharing, which in our setting has the advantage of preserving the full-precision values of the representatives, thereby better preserving the model's accuracy. This strategy considers a representative weight in each cluster (e.g., the centroid), and then substitutes each weight in that cluster with the corresponding representative. Since our goal is to assess the effectiveness of quantization-aware regularization, we first focus on model accuracy at a given quantization level rather than on the resulting compression rate, as lossless formats for low-entropy weight matrices depend primarily only on the number of distinct weights used.

## III. Experiments

In this section we describe the dataset and the DNNs used in the experiments, the corresponding settings and our results.

### *A. Datasets*

We focused on the CIFAR-10 [22] benchmark, containing 60K $32 \times 32$ images with three color channels (RGB), organized in 10 classes and split in 50K+10K images, respectively for training and testing. Pixel values were rescaled to the range $[0, 1]$ and centered by subtracting the per-channel mean.

### B. Deep Models

We tested our regularization framework on two well-known and publicly available architectures: AlexNet [19] (around 22M parameters) and VGG16 [20] (around 33M parameters), both adapted for the $32 \times 32 \times 3$ RGB input images. Except for some small adaptations to handle the different input dataset, models have been trained using the same configurations proposed in the original paper (details not shown for lack of space and available upon request).

### C. Performance Assessment

To measure the effectiveness of our quantization-tailored regularization in facilitating subsequent weight sharing, we quantized the same model twice: once by training it without regularization (*baseline*), and once by training it (with the same training configuration) using our quantization-aware regularizers. For the baseline, once trained the model, as typically done, we applied $k$-means to the resulting weights, and replaced weights in each cluster with the corresponding centroid [5]. For the regularized model, since weight representatives are already embedded in the regularization term, applying $k$-means is unnecessary. Clusters are instead formed by assigning each weight to the closest representative. To reduce possible distortions, the representative of each cluster is recomputed as the centroid of the assigned weights, which may differ slightly from the original representative specified in the regularization term (e.g., the $\boldsymbol{u}$ values used in the $\rho_M$ and $\rho_E$ terms). We evaluated two metrics on the test set: *pre-* and *post-tuning accuracy*, the latter measured after a few epochs of cumulative tuning of both baseline and regularized models, which updates only cluster centroids [13]. The final performance is measured in terms of the *accuracy ratio*, where the accuracy of the regularized models is used as the numerator. Values above 1 indicate a performance gain. In Figure 3, the baseline accuracy is shown as dashed lines.

### D. Results

To better investigate the behavior of our methodology, we conducted three experiments, regularizing (i) only convolutional layers, (ii) only the dense layers, and (iii) the entire model. We used two quantization levels ($K = 8$ and $K = 64$) to assess the impact of both more and less aggressive quantization. Figure 3 depicts the corresponding results in terms of accuracy ratio. To increase robustness, performance is averaged across three repetitions using different seeds. A first clear trend that emerges is that the pre-tuning accuracy is almost always higher when using our regularizers, up to $3\times$ on AlexNet and to $6\times$ on VGG16 better than the baseline, which, though expected, goes beyond our expectations. Conversely, the post-tuning gain in most experiments was not something we could take for granted. In particular, for $K = 8$ our regularizers always yield improvements on AlexNet, whereas on VGG16 this gain is evident only when compressing all layers (third row), likely because VGG16 is larger than AlexNet and therefore allows for stronger quantization. When $K = 64$, our approach outperforms the baseline mainly in the pre-tuning evaluation, while in the post-tuning setting, we observe only a slight improvement on AlexNet and a performance tie on VGG16. This may depend on the fact that, in general, VGG16 can be compressed more than AlexNet.

Concerning the regularizer comparison, the dynamic family ($\mathcal{R}_M$ and $\mathcal{R}_E$) yields slightly better results on AlexNet and, in most cases, dramatically better results on VGG16 than the static counterpart, highlighting the strong potential of having learnable (and therefore adaptable) quantization levels. Among individual families, $\mathcal{R}_E$—when not tied with $\mathcal{R}_M$—achieves higher accuracy in $4$ out of $5$ cases, confirming our intuition that it better handles close minima. In the static family, instead, $\mathcal{R}_S$ and $\mathcal{R}_C$ achieve very similar performances.

## Conclusion

In this work, we introduced one of the first approaches that explicitly regularize neural network weights with the subsequent quantization step in mind. To the best of our knowledge, it is also the first method that learns quantization levels as model parameters, optimized jointly with the weights through backpropagation, enabling a quantization process that adapts to the loss landscape and layer characteristics. Our experiments on AlexNet and VGG16 show substantial pre-tuning improvements—up to $3\times$ and $6\times$ gains over the baseline, respectively. These results indicate that the proposed regularizers successfully guide the model toward weight configurations that are inherently more quantization-friendly. Post-tuning gains also appear, especially for aggressive quantization ($K = 8$). Future work includes exploring weight and representative initialization strategies that exploit the typical bell-shaped distribution of neural network weights, potentially improving the synergy between initialization, regularization, and quantization. Additionally, evaluating our approach in combination with other compression techniques, and on larger datasets and more diverse architectures, may further reveal its generality and practical impact.

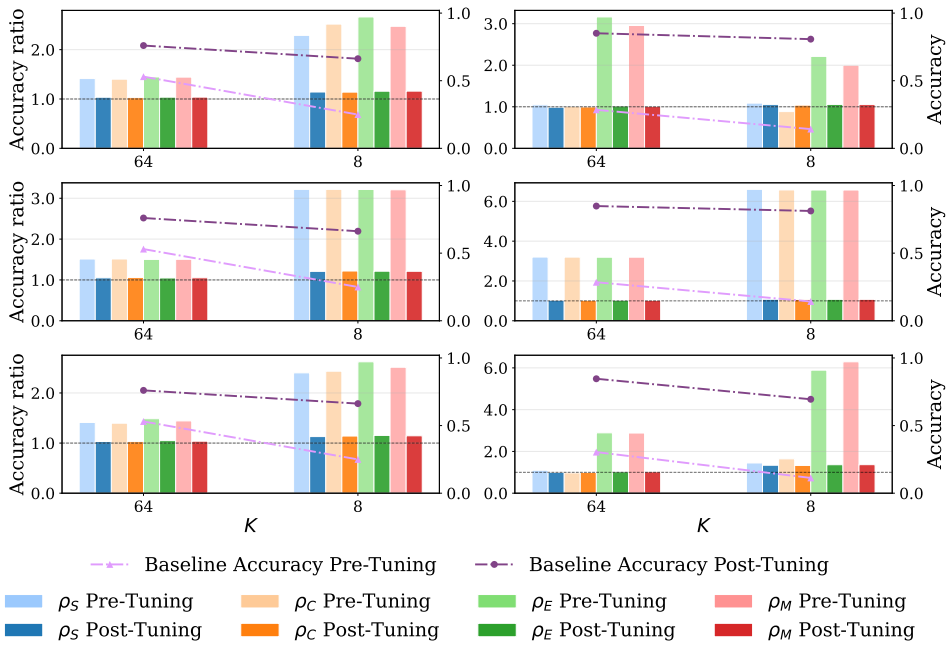


Figure 3: Accuracy ratio (regularized/baseline) for AlexNet (first column) and for VGG16 (second column) on CIFAR-10 data. Rows correspond, from top to bottom, to regularizing only convolutional layers, only dense layers, and all layers. Dotted lines denote a tie: all values above are gains for the regularized models. The second axis shows the baseline accuracy.

## References


[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[3] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," in *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[4] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient dnns," in *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[5] G. C. Marinò, A. Petrini, D. Malchiodi, and M. Frasca, "Deep neural networks compression: A comparative survey and choice recommendations," *Neurocomputing*, vol. 520, pp. 152–170, 2023.

[6] H. Li *et al.*, "Pruning filters for efficient convnets," in *Int. Conf. on Learning Representations*, 2017.

[7] I. Hubara *et al.*, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, no. 187, pp. 1–30, 2017.

[8] E. L. Denton *et al.*, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[9] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[10] F. Tung and G. Mori, "Deep neural network compression by in-parallel pruning-quantization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 3, pp. 568–579, 2020.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[12] E. Park, J. Ahn, and S. Yoo, "Weighted-entropy-based quantization for deep neural networks," in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7197–7205.

[13] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *ICLR*, 2015.

[14] G. C. Marinò, G. Ghidoli, M. Frasca, and D. Malchiodi, "Compression strategies and space-conscious representations for deep neural networks," in *25th Int. Conf. on Pattern Recognition (ICPR)*, 2021, pp. 9835–9842.

[15] G. C. Marinò, G. Ghidoli, M. Frasca, and D. Malchiodi, "Reproducing the sparse huffman address map compression for deep neural networks," in *Reproducible Research in Pattern Recognition*. Cham: Springer International Publishing, 2021, pp. 161–166.

[16] G. C. Marinò, F. Furia, D. Malchiodi, and M. Frasca, "Efficient and compact representations of deep neural networks via entropy coding," *IEEE Access*, vol. 11, pp. 106 103–106 125, 2023.

[17] H. Zhang, S. Ye, Y. Xu, and Y. Shi, "Lc: A flexible, extensible open-source toolkit for model compression," in *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, 2020.

[18] Y. Li, X. Dong, and W. Wang, "Additive powers-of-two quantization: An efficient non-uniform discretization for neural networks," in *International Conference on Learning Representations*, 2019.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[22] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.