

Vanishing Contributions: A Unified Approach to Smoothly Transition Neural Models into Compressed Form

Lorenzo Nikiforos, *Graduate Student Member, IEEE*, Charalampos Antoniadis, *Member, IEEE*, Luciano Prono, *Member, IEEE*, Fabio Pareschi, *Senior Member, IEEE*, Riccardo Rovatti, *Fellow, IEEE*, and Gianluca Setti, *Fellow, IEEE*

Abstract—The increasing scale of deep neural networks has led to a growing need for compression techniques such as pruning, quantization, and low-rank decomposition. While these methods are very effective in reducing memory, computation and energy consumption, they often introduce severe accuracy degradation when applied directly. We introduce Vanishing Contributions (VCON), a general approach for smoothly transitioning neural models into compressed form. Rather than replacing the original network directly with its compressed version, VCON executes the two in parallel during fine-tuning. The contribution of the original (uncompressed) model is progressively reduced, while that of the compressed model is gradually increased. This smooth transition allows the network to adapt over time, improving stability and mitigating accuracy degradation. We evaluate VCON across computer vision and natural language processing benchmarks, in combination with multiple compression strategies. Across all scenarios, VCON leads to consistent improvements: typical gains exceed 3%, while some configuration exhibits accuracy boosts of 20%. VCON thus provides a generalizable method that can be applied to the existing compression techniques, with evidence of consistent gains across multiple benchmarks.

Index Terms—Pruning, Quantization, Low-rank decomposition, Fine-tuning, Computer vision, Natural language processing, Accuracy preservation, Transfer learning

I. INTRODUCTION

DEEP NEURAL NETWORKS (DNNs) have shown significant capabilities in solving a wide array of complex tasks across multiple domains. These tasks include, but are not limited to, computer vision, natural language processing, speech recognition, medical image analysis, autonomous driving, and even drug discovery [1], [2]. Their remarkable performance has made DNNs the state-of-the-art for many of these tasks.

However, DNNs come with a major drawback: their models require substantial computational resources, memory, and energy. This high resource consumption makes their deployment

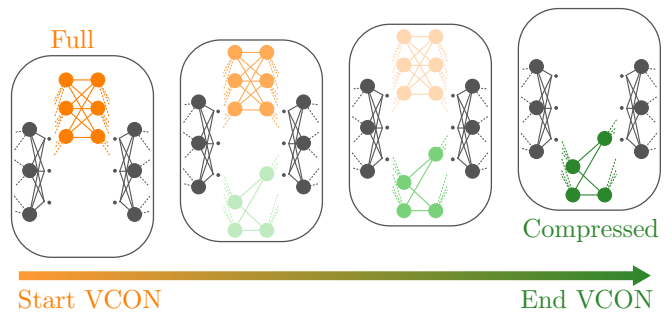


Fig. 1. Illustration of VCON: from left to right, the original layer (orange) slowly disappears while the compressed layer (green) contribution is gradually incorporated.

in resource-constrained environments challenging, such as the deployment on mobile devices, edge computing, or in real-time systems [2], [3].

Different studies have shown that DNN models are overparameterized [4], meaning that they contain more parameters and complexity than necessary to achieve optimal performance. It has been shown that a significant portion of the network's parameters or complexity can be removed without a substantial loss in performance [5]. This reduction is often referred to as DNN compression, and it plays a crucial role in making these models more efficient, both in terms of memory usage and computational power. The goal of DNN compression is to create lighter, more efficient models that retain as much as possible the predictive performance of the original model.

Several techniques have been developed to achieve DNN compression. Specifically, model pruning, parameters/data quantization and compressed representation techniques for the layers are among the most important and widely used approaches.

DNN pruning involves selectively removing certain parameters, neurons, filters, or entire layers from the network [3], [6]. The selection process is typically based on scoring mechanisms that quantify the importance of each network component, and helps identifying which parts of the network are redundant or have a minimal impact on performance. For example, neurons or connections with weights close to zero are often considered unimportant and can be pruned. By pruning unnecessary parts of the network, the overall size of the model is reduced, leading to a decrease in memory usage and computational load.

Code available at: <https://github.com/foros15/vanishing-contributions>

L. Nikiforos and L. Prono are with the Department of Electronic and Telecommunication, Politecnico di Torino, 10129 Torino, Italy. (e-mail: {lorenzo.nikiforos, luciano.prono}@polito.it).

C. Antoniadis and G. Setti are with King Abdullah University of Science and Technology (KAUST), Saudi Arabia. (e-mail: {charalampos.antoniadis, gianluca.setti}@kaust.edu.sa).

F. Pareschi is with the Department of Electronic and Telecommunication, Politecnico di Torino, 10129 Torino, Italy and also with the Advanced Research Center on Electronic Systems (ARCES), University of Bologna, 40125 Bologna, Italy. (e-mail: fabio.pareschi@polito.it)

R. Rovatti is with the Department of Electrical, Electronic, and Information Engineering, University of Bologna, 40136 Bologna, Italy and also with the Advanced Research Center on Electronic Systems (ARCES), University of Bologna, 40125 Bologna, Italy. (e-mail: riccardo.rovatti@unibo.it).

Another class of techniques for DNN compression relies on alternative data representations, including weight sharing, multiplexing, and low-rank decomposition [7]. The goal is to obtain a model that closely replicates the behavior of the original one, while representing the structure in a more efficient manner. These methods can significantly reduce the memory footprint of the network, enabling it to run more efficiently on devices with limited resources.

Lastly, quantization reduces the precision of the parameters used in the network. By using fewer bits to represent each parameter, the memory usage of the model is reduced, its computational cost is decreased [8], [9]. This process involves converting the floating-point weights into lower bit-width representations, such as using 8-bit integers instead of 32-bit floating-point numbers.

Even though various methods for DNN compression have been proposed in the literature, many of them lead to sub-optimal performance. In particular, compressed models often suffer from significant accuracy degradation or fail to generalize as well as their original counterparts, especially when aggressive compression techniques are applied.

In this paper, we propose a technique called Vanishing Contribution (VCON), a unified approach which enables a gradual transition from the original network to its compressed version. Instead of directly replacing the original model with the compressed one, we place both in parallel during fine-tuning. The contribution of the original (uncompressed) model is progressively reduced by scaling down the magnitude of its output, allowing a smooth and controlled shift toward the compressed model. Figure 1 shows a simple diagram illustrating the intuition behind the proposed method. Although this method was first proposed in [10], it was originally introduced only for the Multiply-and-Max-min (MAM) neuron structure. On the contrary, this work expands the original idea by demonstrating a wide range of alternative applications.

We demonstrate the versatility of this approach, which can be applied in various compression contexts and within different data domains. Specifically, we test the model with pruning, binary quantization and low-rank decomposition both for computer vision and natural language processing tasks. VCON is thus proven to be a generalizable method introducing consistent gains across multiple benchmarks while also being easy-to-implement. During fine-tuning, VCON temporarily runs the original and compressed blocks in parallel, but it does not modify the final network structure produced by conventional compression techniques.

The remainder of this paper is summarized as follows. First, we provide an overview of iterative methods in Section II, as they are based on concepts closely related to VCON. Then, Section III presents a detailed analysis of various compression techniques proposed in the literature, with particular emphasis on those adopted in this work. In Section IV, we thoroughly explain the formulation of VCON. Section V introduces a first simple experiment, which is described and analyzed in detail. Then, we demonstrate in Section VI the effectiveness of VCON across a wide range of applications. Section VII outlines the limitations of the proposed technique and directions for future research. Finally, we draw a conclusion.

II. RELATED WORKS

Several works in the literature explore DNN compression, proposing strategies to reduce model size while maintaining acceptable performance. Among them, a prominent class of techniques leverages iterative compression methods, which progressively compress the model during or across multiple training phases [11]–[18].

Iterative compression is based on the principle that a network can better adapt to structural changes if those changes are introduced gradually rather than all at once. Instead of applying a strong compression operation in a single step, iterative techniques apply small, incremental modifications, allowing the model to retrain and recover at each stage. This incremental process enables the network to redistribute important information across the remaining parameters, effectively maintaining its representational power despite progressive simplifications.

The literature on iterative compression provides examples with most of the compression techniques, such as pruning, quantization, and low-rank decomposition. Iterative pruning gradually removes the least important weights, inserting re-training phases between two subsequent pruning steps to recover the lost performance [11]–[13]. As an example, in [14] authors propose an approach that iteratively updates channel importance estimates through an affine combination during training, gradually identifying and pruning redundant channels in a stable manner. In another work [15], authors introduce Rank-based Pruning (RPG), an iterative weight pruning method guided by a rank-based objective that encourages higher matrix ranks for weights. It involves alternating phases in which interconnections are removed and in which are reintroduced, updating masks based on weight importance and gradients, and fine-tuning the sparse model.

Similarly, iterative quantization progressively reduces the precision of selected subsets of weights, starting from high-precision formats and moving toward lower-bit representations, while retraining the model after each quantization stage [16]. For example, in [17], the authors propose an iterative quantization approach that gradually quantizes neural network weights by classifying them based on gradient-based importance. Low-importance weights are quantized first, followed by fine-tuning, and the process is repeated to achieve significant compression while preserving accuracy.

In the case of low-rank decomposition, weight matrices are incrementally approximated with lower-rank factors, and retraining steps allow the network to adapt to the reduced parameterization [18]. In all these cases, the progressive nature of the compression, coupled with intermediate fine-tuning, helps preserve model accuracy despite substantial reductions in size and complexity.

Similar to the aforementioned approaches, the VCON method offers an alternative approach based on the same principle of gradually transitioning from the original model to the compressed one. However, they operate along fundamentally different dimensions. Iterative compression reduces the number of parameters in discrete stages. At each step, a subset of parameters is compressed, and the model is fine-tuned to recover performance before the next compression stage.

This process progressively shrinks the model by explicitly removing parameters. In contrast, VCON does not immediately discard parts of the original model. Instead, it gradually reduces their contribution by decreasing their magnitude over time, while simultaneously increasing the influence of the compressed counterpart. Rather than eliminating parameters outright, VCON smoothly shifts the computational load from the original model to the compressed one. This results in a unified approach that can be shared by different compression methods, offering the advantage of using the same simple compression framework.

III. COMPRESSION METHODS

In this section, we provide an overview of the existing compression methods that are compatible with the VCON technique.

A. Pruning

Pruning is generally performed following two main strategies: structured [19]–[22], and unstructured pruning [23]–[26]. The former involves the removal of neurons, filters, or entire blocks. This approach is hardware-friendly, meaning it does not require hardware adaptation to maintain computational performance. However, it typically removes a smaller portion of weights to preserve comparable accuracy to that of the original network.

On the contrary, unstructured pruning focuses on the removal of individual interconnections, regardless of their position in the network. This results in a higher percentage of weights being removed compared to the structured case, leading to a significant reduction in model size. However, this results in sparse, non-regular structures, which can cause a computational overhead due to inefficiencies in memory access, as the remaining non-zero weights are scattered across the network.

The selection of the parts to be removed is typically guided by a scoring system. Each removable entity (such as interconnections, neurons, filters, or blocks) is assigned a score that reflects its importance within the network. Entities are then removed in ascending order (from the least to the most important) until the desired pruning rate is achieved.

Pruning methods based on scoring can be applied at different levels of granularity, resulting into three pruning groups: global, layer-wise, and N:M pruning. In global pruning, scores are computed for all entities across the entire network, and the ones with the lowest scores are removed, regardless of the block/layer they belong to. In layer-wise pruning, scores are ranked and pruned separately within each layer, ensuring a more balanced distribution of weight sparsity. A third type, namely N:M pruning, introduces a structured constraint: within every local group of M interconnections, exactly N are kept, and the remaining are pruned.

Although numerous pruning methods have been proposed in the literature, here we focus on a subset of the techniques. In particular, we restrict our study on pruning strategies that rely on magnitude-based scoring. These methods are built on the assumption that interconnections associated to the smallest

absolute values of the weights – or the smallest norms of the weights associated to a neuron – contribute less to the overall output of the network and can therefore be removed.

B. Low-Rank Decomposition

Low-rank decomposition reduces both memory consumption and computational cost, and can be applied to both convolutional and fully connected layers [27]–[32]. In the case of fully connected layers, the weight matrix can be approximated as the product of two lower-rank matrices. Formally, let $\mathbf{W} \in \mathbb{R}^{n \times m}$ be a weight matrix. Its low-rank decomposition can be represented by two matrices $\mathbf{A} \in \mathbb{R}^{n \times r}$ and $\mathbf{B} \in \mathbb{R}^{r \times m}$ such that $r \ll \min(n, m)$. With this decomposition, we aim to replace $\mathbf{W}$ with the product $\mathbf{AB}$. The computational cost and the memory complexity of a fully connected layer is $O(mn)$, whereas its low-rank decomposition has complexity $O(r(m + n))$. This approach offers a benefit when $r(m + n) < mn$, implying $r < \frac{m+n}{mn}$.

A common approach to obtain a low-rank matrix is to use truncated Singular Value Decomposition (SVD). Specifically, matrix $\mathbf{W}$ can be factorized as $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{n \times n}$ and $\mathbf{V} \in \mathbb{R}^{m \times m}$ are unitary matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{n \times m}$ is a rectangular diagonal matrix whose diagonal elements are the singular values arranged in descending order. The truncated SVD keeps only the r largest singular values. The factorized matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}^T$ become $\hat{\mathbf{U}} \in \mathbb{R}^{n \times r}$, $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$ and $\hat{\mathbf{V}}^T \in \mathbb{R}^{r \times m}$. From this, we finally set $\mathbf{A} = \hat{\mathbf{U}}$ and $\mathbf{B} = \hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^T$. After factorization, fine-tuning is typically needed to achieve an accuracy comparable to the original model.

C. Quantization

Quantization is a widely employed technique for compressing DNNs, aiming to reduce the precision of weights and activations. In its general form, quantization maps each real-valued weight or activation $x \in \mathbb{R}$ to a value $\tilde{x}$ belonging to a finite discrete set $\mathcal{Q} \subset \mathbb{R}$. The size of the finite set $\mathcal{Q}$ is determined by the bit-width – i.e., the number of bits used to encode the information. For instance, 8-bit quantization allows 256 distinct levels, while lower bit-widths correspond to smaller sets.

Quantization techniques can be broadly classified into post-training quantization, where quantization is applied after the model has been trained, and quantization-aware training, where the model is trained while accounting for quantization effects, typically resulting in a higher final accuracy.

Finally, binary and ternary quantization [33]–[36] are extreme forms of quantization that constrain weights and/or activations to only two or three distinct values, respectively, drastically reducing model size and simplifying arithmetic operations. With binary quantization, weights are constrained to $\{-1, +1\}$. Given a real-valued weight $w \in \mathbb{R}$, the binary quantized value $\tilde{w}$ can be obtained through the simple sign function:

$$\tilde{w} = \text{sign}(w) = \begin{cases} +1, & \text{if } w \geq 0 \\ -1, & \text{otherwise.} \end{cases}$$

Although binary networks achieve significant reductions in memory and computational cost, they result in a non-negligible accuracy degradation. To mitigate this problem, each group of binary weights or activations is multiplied by a fixed scaling factor, used to restore their original overall scale and compensating for the limited precision [37]. Scaling factors can be learned during training or predefined, depending on the method used.

D. Training compressed models

In the literature, various training schedules are used in conjunction with model compression, including one-shot, post-shot, iterative, and Straight-Through Estimator (STE) approaches.

One-shot approaches [38], [39] imply that the model is compressed right after the initial training phase, without further adjustments. While these methods are fast and do not require a training dataset, they often lead to a significant drop in accuracy, as the compressed model is not given the chance to adapt to the task.

Post-shot fine-tuning [40], [41] is a more effective strategy, involving a fine-tuning phase that follows the compression of the model. This allows the model to recover some of the accuracy lost during compression. However, this approach requires an additional training phase to fine-tune the model after compression, requiring long computational times and large datasets to complete the compression process.

Similarly, iterative compression [14], [42], [43] progressively reduces the complexity of the model over multiple training iterations. For each step, the model is compressed by a small amount and then fine-tuned for some time. This gradual process allows the model to adjust more effectively to compression, reducing the performance drop compared to one-shot and post-shot methods.

Finally, STE [44] is commonly used to introduce the compression constraints during training, increasing the chance for the model to find the optimal compressed configuration. As an example, STE is applied to perform the so-called quantization-aware training: quantization is applied dynamically during the forward pass, while during the backward pass, parameters are restored to their full-precision version [45]. In the case of pruning, at the beginning of each training step, parts of the model are dynamically removed, but then restored right before the application of the parameters update [46].

IV. VANISHING CONTRIBUTION

In this section, we provide a detailed explanation of VCON, which enables a smooth transition from an original model to its compressed version by progressively reducing the influence of the former during training while increasing that of the latter.

A. Vanishing contribution definition

To define VCON, we first introduce a typical setting in model compression. We consider a model $\text{DNN}(\Theta^{(1)}, \dots, \Theta^{(L)}; f_{\Theta}^{(1)}, \dots, f_{\Theta}^{(L)})$, where L denotes the number of parameterized blocks in the network, and $\Theta^{(i)}$

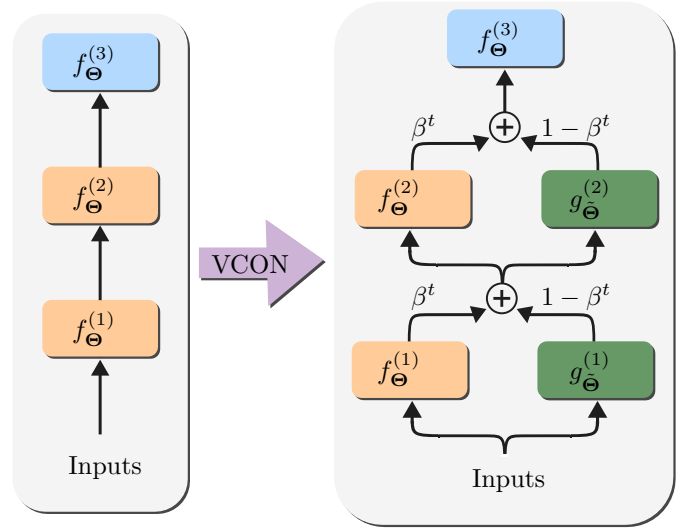


Fig. 2. Illustration of block-wise VCON: the first two blocks $f_{\Theta}^{(1)}$ and $f_{\Theta}^{(2)}$ are progressively replaced with their compressed counterparts $g_{\Theta}^{(i)}$, while the final block $f_{\Theta}^{(3)}$ remains uncompressed.

represents the set of parameters associated with the i -th block/function $f_{\Theta}^{(i)}(\cdot)$ (e.g., a linear or convolutional layer).

Given the set of L functions $f_{\Theta}^{(i)}(\cdot)$ with i from 1 to L , we compress a subset of them through a transformation operation $G(\cdot)$, yielding compressed versions $g_{\Theta}^{(i)}(\cdot)$. In our case, $G(\cdot)$ is interpreted as pruning, quantization, or low-rank decomposition. Specifically, we define

$$g_{\Theta}^{(i)}(\cdot) = G(f_{\Theta}^{(i)}; \Theta^{(i)}; \text{DNN}) \quad (1)$$

where $\tilde{\Theta}^{(i)}$ are the new parameters that parametrize $g_{\Theta}^{(i)}(\cdot)$. In some cases, such as pruning, there is a direct correspondence between $\Theta^{(i)}$ and $\tilde{\Theta}^{(i)}$. However, this is not always the case, as with low-rank decomposition.

Within the VCON framework, the original function $f_{\Theta}^{(i)}(\cdot)$ is replaced by a new function $\bar{g}_{\Theta, \tilde{\Theta}}^{(i), t}(\cdot)$, defined as the affine combination between the original function and its compressed counterpart, i.e.,

$$\bar{g}_{\Theta, \tilde{\Theta}}^{(i), t}(\cdot) = \beta^t f_{\Theta}^{(i)}(\cdot) + (1 - \beta^t) g_{\Theta}^{(i)}(\cdot), \quad (2)$$

where β^t is a scalar in the range $[0, 1]$ that progressively varies as a function of the training step t . In more detail, β^t controls the gradual transition from the original function $f_{\Theta}^{(i)}(\cdot)$ to its compressed version $g_{\Theta}^{(i)}(\cdot)$ as the training process advances. We remark that for $\beta = 1$ the resulting function is identical to the original function, while for $\beta = 0$ we get its compressed version only. Figure 2 illustrates a visual example.

The value of β^t is scheduled during the training process through a monotonically decreasing function, which starts at one, reaches zero after Q steps, and remains zero for $t \geq Q$. In this work, we use a linear scheduler, defined as

$$\beta^t = \max\left(1 - \frac{t}{Q}, 0\right). \quad (3)$$

This process facilitates a smooth and simultaneous transition by allowing independent compression of each block.

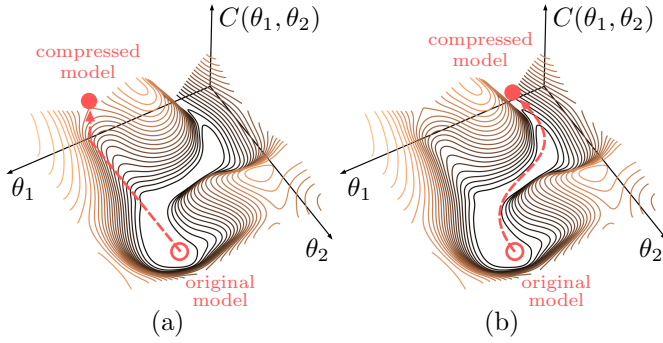


Fig. 3. Visual intuition of the VCON approach: when a model parameter is removed abruptly (a), the working point is projected directly onto the hyperplane defined by the remaining dimensions, which is suboptimal. In contrast, if the parameter is removed gradually (b), the working point shifts slowly toward the hyperplane of the remaining dimensions and the model is continuously updated, allowing a greater chance to reach a better local minimum.

B. An intuitive perspective on VCON effectiveness

To support the rationale behind VCON approach, we propose an intuitive view of the compression of a model. Figure 3 visually illustrates the proposed intuition. Let us have a model with two parameters θ_1 and θ_2 , of which one is to be removed by pruning. Then let us define the cost function as $C(\theta_1, \theta_2) : \mathbb{R}^2 \rightarrow \mathbb{R}$. Removing parameter θ_2 is equivalent to restricting the domain of C to $\{(\theta_1, 0) \in \mathbb{R}^2\}$, *abruptly* projecting the current working point to axis θ_1 . This operation cannot be fully controlled since *at-most* we can linearize the behavior of the model in the original working point and remove the parameter with the lowest local influence on the cost function. Conversely, by using VCON, the working point is *gradually* shifted towards the axis θ_1 , and the model is updated following the full profile of C . Although we lack a formal proof, our intuition is that this gradual reduction leaves room for the model to adapt its trajectory in the cost function landscape. This means that the model has a chance to find a better local minimum compared to what we may get with an abrupt change of the structure, even when fine-tuning is performed after compression. This vision can be easily extended to the generic multi-dimensional case, where removing a parameter results in the projection of the working point onto the hyperplane defined by the remaining axes.

V. PRELIMINARY EXPERIMENTS

In this section, we present a set of preliminary experiments to validate the effectiveness of the proposed VCON technique. The main goal of these experiments is to evaluate how the inclusion of VCON modifies the training dynamics and affects model performance, with a particular focus on the influence of the hyperparameter Q , which governs the duration of the transition from the original to the compressed model. To do so, we evaluate unstructured pruning on the ViT-T/16 model using the CIFAR-10 dataset. We apply a layer-wise pruning STE strategy with a compression ratio of 95% and we conduct a comparison with and without the use of the VCON technique. Details about dataset and training setup are in Appendix A and Appendix B, respectively.

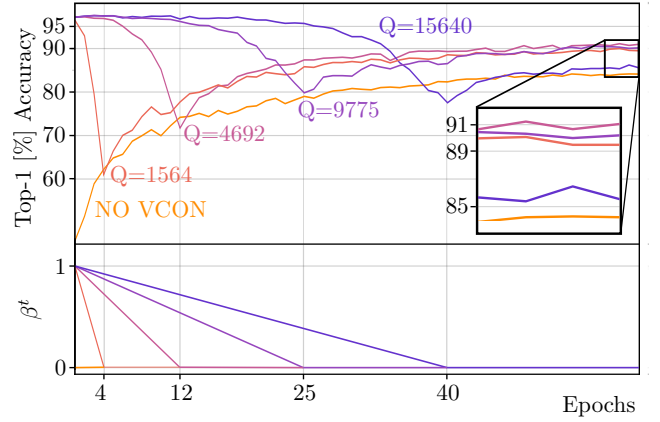


Fig. 4. Impact of Training Dynamics with VCON: evolution during training for different transition durations Q .

In the VCON setting, while the original blocks gradually fade out, the parallel counterpart progressively takes over and is trained throughout the process. Pruning via STE is naturally applied to the latter, which evolves dynamically during training. This reflects the core idea of VCON: enabling a smooth and flexible transition that integrates seamlessly with existing compression techniques. We performed the experiment using different values of Q and we complete the vanishing phase after 4, 12, 25, and 40 epochs ($Q = 1564, 4692, 9775, 15640$), respectively. Figure 4 shows the validation-set accuracy curves evolving during the training process.

All the configurations employing the VCON technique consistently outperform the baseline model without VCON, demonstrating the effectiveness of our approach in enhancing model performance. The improvement is evident across all values of the transition hyperparameter Q , confirming the general benefit of introducing a smooth transition.

A closer inspection of the training dynamics reveals an interesting and recurring pattern. For each value of Q , there is a noticeable drop in validation accuracy that aligns closely with the end of the transition period, where the original model blocks are fully faded out. This temporary drop is likely due to the sudden change in model representation as the compressed path fully takes over. However, what follows is equally notable: in all cases, the model resumes a steady ascent in accuracy, indicating that the network successfully adapts to the new structure once the transition is complete.

Among the tested configurations, a transition of 12 epochs appears to strike the best balance. A transition too short may not allow sufficient time for the compressed model to effectively assimilate the knowledge encoded in the original network, while an excessively extended one could delay the learning of the compressed blocks. Thus, our findings suggest that intermediate values of Q lead to smoother transitions and better overall performance, highlighting the importance of tuning this hyperparameter to achieve optimal results.

VI. EXPERIMENTAL RESULTS

In this section, we present the experimental evaluation of the VCON approach. To assess its effectiveness, we focus

TABLE I
VANISHING CONTRIBUTIONS RESULTS FOR STE-BASED PRUNING ON COMPUTER VISION MODELS

granularity	sparsity	ViT-S/16					ViT-B/16				
		CIFAR10 Top-1 [%]		CIFAR100 Top-1 [%]		#params	CIFAR100 Top-1 [%]		ImageNet-1k Top-1 [%]		#params
		Standard	VCON	Standard	VCON		Standard	VCON	Standard	VCON	
	0.0 (dense)	98.70	–	91.36	–	21.28M	91.90	–	81.10	–	85.06M
Unstr. layer	0.9	93.98	(+2.64) 96.62	75.70	(+8.22) 83.92	2.20M	83.16	(+3.50) 86.66	70.08	(+1.45) 71.53	8.61M
	0.95	89.46	(+5.12) 94.58	66.64	(+10.34) 76.98	1.14M	66.26	(+15.50) 81.76	39.65	(+21.45) 61.10	4.37M
	0.99	75.62	(+3.94) 79.56	49.30	(+1.26) 50.56	0.29M	53.92	(+6.74) 60.66	23.90	(+5.92) 29.82	0.97M
	0.875 (1:8)	94.18	(+2.02) 96.20	75.96	(+7.24) 83.18	2.73M	84.70	(+2.40) 87.10	68.73	(+1.15) 69.88	10.74M
Unstr. N:M	0.9375 (1:16)	88.60	(+5.56) 94.16	65.02	(+10.58) 75.60	1.40M	67.44	(+14.66) 82.00	39.68	(+19.53) 59.20	5.43M
	0.96875 (1:32)	80.48	(+9.34) 89.82	55.34	(+9.42) 64.76	0.74M	61.88	(+12.40) 74.28	32.48	(+13.41) 45.89	2.78M
Str. layer	0.4	98.01	(+0.47) 98.48	90.08	(-0.02) 90.06	12.77M	90.86	(+0.54) 91.40	77.49	(+0.16) 77.65	51.04M
	0.6	95.32	(+2.88) 98.20	89.42	(-0.42) 89.00	8.51M	90.84	(-0.26) 90.58	77.40	(+0.05) 77.45	34.02M
	0.8	94.42	(+3.12) 97.54	87.32	(+1.54) 88.86	4.27M	89.24	(+0.52) 89.76	76.60	(+0.54) 77.14	17.01M
	0.9	93.46	(+3.04) 96.50	77.08	(+6.18) 83.26	2.13M	82.70	(+3.88) 86.58	76.22	(+0.82) 76.94	8.51M

on three model compression techniques: pruning, quantization, and low-rank decomposition. These methods are applied across a combination of datasets and models in both computer vision and natural language processing domains. We apply compression either with a post-shot approach and with the VCON approach, and we compare the results. Datasets and implementation details are provided in Appendix A and Appendix B, respectively.

A. Pruning

We present a set of experiments to validate the effectiveness of our method when applied to pruning. Specifically, we employ magnitude-based scoring in conjunction with our VCON approach across three different granularity strategies, namely unstructured layer-wise, unstructured N:M, and structured pruning layer-wise. We perform the tests for several different pruning ratios, removing parameters equally in all the layers of the models (self-attention and multi-layer perceptron altogether). In all the cases, training was carried out using a STE-based approach. Within the VCON framework, each parallel block $g_{\Theta}^{(i)}$ is initialized in the same way as the corresponding block $f_{\Theta}^{(i)}$. Crucially, pruning via STE is applied exclusively to the parallel components, which are continuously updated and optimized throughout the transition.

The results for the experiment in the computer vision domain are reported in Table I, while Table II reports the results for the natural language models. Across the large majority of the experimental configurations, our VCON approach yields consistent improvements in classification accuracy when compared to the standard post-shot pruning baseline, demonstrating its broad applicability and stability. Even in scenarios where performance gains are not substantial, results remain at least on par with those obtained through direct STE-based pruning, further underscoring the robustness and reliability of the proposed strategy. Finally although not universally true, we observe a general tendency for the benefits of VCON to become more pronounced as pruning becomes more aggressive. This suggests that the smooth transition mechanism is particularly effective in preserving performance under high sparsity conditions.

B. Quantization

We evaluate the effectiveness of the proposed VCON method in the context of binary quantization. In this setting, network weights are constrained to two discrete values, -1 and +1. To handle the non-differentiability of the binarization function, we adopt an STE-based method during training. Rather than introducing learnable scaling parameters for each layer, which would require modifications to the architecture, we apply a fixed scaling based on the ℓ_2 norm of the weight matrices. As before, VCON is applied and each block of the original, full-precision model is accompanied by a parallel quantized branch that is gradually activated during training. The associated STE-based updates are applied only to this parallel branch, allowing it to adapt smoothly while inheriting the functional behavior of the original network.

Experimental results, summarized in Table III, cover both computer vision and natural language processing tasks. Each configuration is compared against a standard baseline in which STE-based quantization is applied directly to the original model from the beginning of training. Even in this case, results obtained with the VCON approach are generally superior to those achieved through standard binary quantization. In some cases, the performance remains effectively identical to the baseline, indicating that VCON does not introduce degradation even when VCON does not yield a clear benefit. Overall, the improvements appear more consistent in settings involving simpler datasets or smaller models, suggesting that the gradual transition facilitated by VCON may be particularly effective when model capacity or task complexity is limited. In contrast, larger models may be operating near their task-specific capacity, leaving little room for VCON to recover accuracy after compression.

C. Low-Rank Decomposition

We assess the performance of the proposed VCON framework when applied to low-rank decomposition. In this setting, each weight matrix of the original model is factorized using singular value decomposition (SVD) into a pair of lower-rank matrices, denoted as A and B , initialized as $A = \hat{U}$ and $B = \hat{\Sigma}\hat{V}^T$. To explore the trade-off between model

TABLE II
VANISHING CONTRIBUTIONS RESULTS FOR STE-BASED PRUNING ON NATURAL LANGUAGE MODELS

granularity	sparsity	BERT					distilBERT				
		QNLI Top-1 [%]		MNLI Top-1 [%]		#params	QNLI Top-1 [%]		MNLI Top-1 [%]		#params
		Standard	VCON	Standard	VCON		Standard	VCON	Standard	VCON	
	0.0 (dense)	91.08	–	83.84	–	85.02M	88.70	–	81.43	–	42.50M
Unstr. layer	0.9	81.09 (+4.43)	85.52	77.32 (+2.04)	79.36	8.50M	79.72 (+3.91)	83.63	76.05 (+1.45)	77.50	4.25M
	0.95	76.68 (+6.11)	82.79	73.85 (+4.06)	77.91	4.25M	73.62 (+7.52)	81.14	72.29 (+3.91)	76.20	2.12M
	0.99	61.50 (+17.91)	79.41	68.12 (+5.36)	73.48	0.85M	60.91 (+1.89)	61.10	61.28 (+9.61)	70.89	0.42M
	0.875 (1:8)	82.90 (+21.33)	85.76	77.83 (+7.24)	79.43	10.62M	80.56 (+2.82)	83.38	76.18 (+1.08)	77.26	5.31M
Unstr. N:M	0.9375 (1:16)	61.87 (+19.53)	83.20	73.61 (+3.81)	77.42	5.31M	61.58 (+19.86)	81.44	72.45 (+3.32)	75.77	2.65M
	0.96875 (1:32)	61.89 (+19.53)	81.42	70.94 (+4.66)	75.60	2.65M	61.32 (+14.94)	76.26	67.74 (+6.16)	73.90	1.32M
	0.4	83.58 (+3.22)	86.80	79.45 (+1.5)	80.95	51.01M	83.74 (+0.96)	84.70	76.87 (+1.27)	78.14	25.50M
Str. layer	0.6	82.64 (+1.16)	83.80	75.63 (+4.01)	79.64	34.00M	76.90 (+4.96)	81.86	72.73 (+4.27)	77.00	17.00M
	0.8	80.87 (+0.77)	81.64	72.63 (+4.19)	76.82	17.00M	73.66 (+7.36)	81.02	70.36 (+3.85)	74.21	8.50M
	0.9	61.79 (+18.69)	80.48	71.21 (+3.33)	74.54	8.50M	60.61 (+18.19)	78.80	66.90 (+4.64)	71.54	4.25M

TABLE III
VANISHING CONTRIBUTIONS RESULTS FOR BINARY QUANTIZATION

Model	Dataset	Full model	Top-1 [%]	
			Standard	VCON
ViT-T/16	CIFAR 10	97.84	93.18 (+2.28)	95.46
	CIFAR 100	87.92	74.54 (+4.56)	79.10
ViT-S/16	CIFAR 10	98.70	94.88 (+2.20)	97.08
	CIFAR 100	91.36	83.22 (+1.66)	84.88
ViT-B/16	CIFAR 100	91.90	89.02 (-0.12)	88.90
	ImageNet-1k	81.10	76.00 (-0.04)	75.96
BERT	QNLI	91.08	86.53 (+1.47)	88.00
	MNLI	83.84	80.00 (+0.89)	80.89
distilBERT	QNLI	88.70	85.00 (+0.61)	85.61
	MNLI	81.43	78.13 (+0.62)	78.75

compression and accuracy, we evaluate multiple rank configurations. The decomposed model is introduced at the beginning of training and placed in parallel with the original full-rank model. As training progresses, the contribution of the original weights is gradually reduced, while the low-rank path becomes increasingly dominant.

Results for the computer vision and natural language processing experiments are reported in Table IV and Table V, respectively. In each case, we compare the standard low-rank decomposition baseline, where factorized matrices are used from the start of training, with our VCON-based approach. Across all evaluated scenarios, VCON consistently matches or outperforms the standard low-rank baseline. These findings confirm the effectiveness of the gradual transition mechanism in the context of low-rank decomposition. While overall performance gains are evident, no clear trend is observed with respect to the selected rank values. Improvements appear across both low and high ranks, but the extent of the benefit varies depending on the specific model and task.

VII. LIMITATIONS AND FUTURE WORKS

While the proposed VCON framework demonstrates promising results across multiple tasks and compression techniques, several limitations remain and warrant further investigation.

We observed that certain combinations of models, datasets, and compression methods yield suboptimal or inconclusive results. This suggests that in some scenarios, the baseline compression techniques may already be operating close to their optimal point, leaving limited room for improvement. Alternatively, the introduced compression could undermine the training stability – i.e., even if there is an optimal configuration for the model, we are not able to reach it with the current training configuration – which may negate the benefits introduced by the VCON approach. A deeper analysis is required to understand these dynamics and to isolate the conditions under which our method is most effective. Exploring this direction is an important avenue for future work.

VIII. CONCLUSION

In this work, we explored and extensively tested Vanishing Contributions (VCON), a general training strategy that enables a smooth transition from an original DNN to its compressed version. VCON operates by progressively reducing the influence of the uncompressed model during training, while the compressed model is gradually activated in parallel.

The proposed VCON technique was evaluated through an extensive set of experiments across three major compression methods: pruning, quantization, and low-rank decomposition. Experiments were conducted on both computer vision (CIFAR-10, CIFAR-100, ImageNet-1k) and NLP tasks (QNLI, MNLI) using ViT, BERT, and DistilBERT architectures. For pruning, VCON was tested across different sparsity levels using an STE-based approach, with unstructured layer-wise, unstructured N:M, and structured layer-wise granularities. For quantization, the method was tested with an STE-based binary setting. Lastly, low-rank decomposition experiments were conducted by applying truncated SVD with varying rank constraints.

Overall, the experimental evaluation confirms that VCON provides a generally superior or at least comparable alternative to conventional compression strategies. The main strength of VCON lies in its broad compatibility and ease of integration with existing methods. It acts as a lightweight extension to common training procedures and can be seamlessly combined with state-of-the-art compression pipelines. As demonstrated,

TABLE IV
VANISHING CONTRIBUTIONS RESULTS FOR LOW-RANK COMPRESSION ON COMPUTER VISION MODELS

rank	ViT-S/16					ViT-B/16				
	CIFAR10 Top-1 [%]		CIFAR100 Top-1 [%]		#params	CIFAR100 Top-1 [%]		ImageNet-1k Top-1 [%]		#params
	Standard	VCON	Standard	VCON		Standard	VCON	Standard	VCON	
full	98.70	–	91.36	–	21.28M	91.90	–	81.10	–	85.06M
128	90.50 (+0.80)	91.30	68.22 (−0.16)	68.06	10.61M	87.28 (+0.10)	87.38	76.45 (−0.11)	76.34	21.23M
64	87.34 (+1.36)	88.70	64.60 (+1.02)	65.62	5.30M	79.98 (+2.50)	82.48	71.03 (+1.27)	72.30	10.61M
16	78.86 (+3.64)	82.50	52.78 (+4.06)	56.84	1.32M	60.94 (+8.32)	69.26	42.54 (+6.66)	49.20	2.65M
8	71.06 (+6.04)	77.10	43.90 (+6.26)	50.16	0.66M	57.50 (+9.16)	66.66	28.50 (+5.23)	33.73	1.32M
4	65.04 (+5.32)	70.36	35.48 (+2.76)	38.24	0.33M	28.80 (+8.42)	37.22	18.85 (+1.23)	20.08	0.66M

TABLE V
VANISHING CONTRIBUTIONS RESULTS FOR LOW-RANK COMPRESSION ON NATURAL LANGUAGE MODELS

rank	BERT					distilBERT				
	QNLI Top-1 [%]		MNLI Top-1 [%]		#params	QNLI Top-1 [%]		MNLI Top-1 [%]		#params
	Standard	VCON	Standard	VCON		Standard	VCON	Standard	VCON	
full	91.08	–	83.84	–	85.02M	88.70	–	81.43	–	42.50M
128	84.29 (+3.28)	87.57	80.08 (+0.10)	80.18	21.32M	84.53 (+0.92)	85.45	77.92 (+0.36)	78.28	10.66M
64	80.60 (+4.00)	84.60	75.03 (+3.71)	78.84	10.70M	79.53 (+3.41)	82.94	75.97 (+1.07)	77.04	5.35M
16	64.58 (+12.98)	77.56	69.53 (+3.52)	73.05	2.74M	62.84 (+6.0)	68.86	66.71 (+4.68)	71.39	1.37M
8	62.77 (+6.68)	69.45	66.25 (+3.34)	69.59	1.41M	62.23 (+2.44)	64.67	62.87 (+4.63)	67.50	0.70M
4	62.69 (+2.21)	64.90	63.77 (+3.26)	67.03	0.75M	61.96 (−0.18)	61.78	58.37 (+5.18)	63.55	0.37M

this approach improves the robustness and quality of compressed models without requiring complex modifications.

APPENDIX A DATASETS AND PREPROCESSING

To evaluate our models across both computer vision and natural language processing tasks, we employed a selection of well-established benchmark datasets. Below, we provide a description of each of them and the preprocessing technique used in our experiments.

In the case of the computer vision tasks, we employed the following datasets:

- CIFAR-10 [47] is a dataset consisting of 60 000 color images, each with a resolution of 32×32 pixels and three color channels (RGB). The images are uniformly distributed across 10 distinct classes. The dataset is split into 50 000 training images and 10 000 test images.
- CIFAR-100 [47] is a dataset also containing 60 000 32 × 32 RGB images and 100 classes. As for CIFAR-10, the dataset is split into 50 000 training images and 10 000 test images.
- ImageNet-1K [48] is a large-scale dataset composed of approximately 1.28 million training images and 50 000 validation images, categorized into 1000 object classes. Since the ground-truth labels for the official ImageNet test-set are not publicly available, we split the standard validation set as equally into our validation and test set.

We applied a set of preprocessing techniques for all the computer vision datasets considered in this work. We first resize the input images to 224 × 224 pixels, then we normalize them with standard dataset-specific statistics. We then apply data augmentation in the form of RandAugment, as provided by the PyTorch framework, and MixUp.

In the case of the language tasks, we employed the following datasets from the General Language Understanding Evaluation (GLUE) benchmarks:

- QNLI [49] is a binary classification task. Each example consists of a question and a context sentence extracted from a Wikipedia article. The task is to determine whether the sentence contains the correct answer to the question. The dataset includes 104 743 training examples, 5463 for validation, and 5463 for testing.
- MNLI [50] contains pairs of sentences composed of a premise and a hypothesis. The task is to determine the semantic relationship between them by assigning one of three labels: entailment, contradiction, or neutral. The training set comprises 392 702 examples, with validation and test sets each containing 9815 examples, further split into matched (in-domain) and mismatched (cross-domain) subsets.

APPENDIX B MODEL TRAINING HYPERPARAMETERS

Table VI shows the hyperparameters used for the different models and configurations examined in this work. Note that vision models are fine-tuned from models pretrained on ImageNet-21k, while language models are first fine-tuned for 8 epochs on the target datasets (QNLI, MNLI). Note that the value of Q for each setup can be inferred from “VCON epochs” and the number of training steps per epoch, which depends on the dataset.

REFERENCES

- [1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. doi:10.1038/nature14539

TABLE VI
TRAINING CONFIGURATIONS AND HYPERPARAMETERS

Configuration	ViT-T/16, ViT-S/16, ViT-B/16		BERT, distilBERT
	pruning, quantization	low-rank decomposition	
Batch size	128	128	16
Optimizer	Adam	Adam	AdamW
LR start	1e-4	1e-3	2e-5
LR scheduler	Cosine	ReduceOnPlateau(factor=0.3, patience=3)	EarlyStopping(patience=5)
Warmup ratio	0.016	0.24	—
LR start (warmup)	1e-6	1e-4	—
VCON epochs	12 (CIFAR-10, CIFAR-100) 1 (ImageNet-1k)	12	2
Total epochs	60 (CIFAR-10, CIFAR-100) 5 (ImageNet-1k)	50	8

- [2] Y. Wang, J. Wang, W. Zhang, Y. Zhan, S. Guo, Q. Zheng, and X. Wang, “A survey on deploying mobile deep learning applications: A systemic and technical perspective,” *Digital Communications and Networks*, vol. 8, no. 1, pp. 1–17, Feb. 2022. doi:10.1016/j.dcan.2021.06.001
- [3] H. Cheng, M. Zhang, and J. Q. Shi, “A Survey on Deep Neural Network Pruning-Taxonomy, Comparison, Analysis, and Recommendations,” Aug. 2024. doi:10.48550/arXiv.2308.06767
- [4] Z. Allen-Zhu, Y. Li, and Y. Liang, “Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers,” in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019.
- [5] S. Han, H. Mao, and W. J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding,” Feb. 2016. doi:10.48550/arXiv.1510.00149
- [6] S. Vadera and S. Ameen, “Methods for Pruning Deep Neural Networks,” *IEEE Access*, vol. 10, pp. 63 280–63 300, 2022. doi:10.1109/ACCESS.2022.3182659
- [7] R. Mishra, H. P. Gupta, and T. Dutta, “A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions,” Oct. 2020. doi:10.48550/arXiv.2010.03954
- [8] Y. Guo, “A Survey on Methods and Theories of Quantized Neural Networks,” Dec. 2018. doi:10.48550/arXiv.1808.04752
- [9] “[2112.06126] Neural Network Quantization for Efficient Inference: A Survey,” <https://arxiv.org/abs/2112.06126>.
- [10] L. Prono, P. Bich, C. Boretti, M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, “A Multiply-And-Max/Min Neuron Paradigm for Aggressively Prunable Deep Neural Networks,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 36, no. 8, pp. 14 414–14 427, Aug. 2025. doi:10.1109/TNNLS.2025.3527644
- [11] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, “Pruning neural networks without any data by iteratively conserving synaptic flow,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 6377–6389.
- [12] Y. Gordienko, Y. Kochura, V. Taran, N. Gordienko, A. Bugaiov, and S. Stirenko, “Adaptive Iterative Pruning for Accelerating Deep Neural Networks,” in *2019 XIth International Scientific and Practical Conference on Electronics and Information Technologies (ELIT)*, Sep. 2019, pp. 173–178. doi:10.1109/ELIT.2019.8892346
- [13] C. M. J. Tan and M. Motani, “DropNet: Reducing Neural Network Complexity via Iterative Pruning,” in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, Nov. 2020, pp. 9356–9366.
- [14] J. Pan, S. Yang, L. G. Foo, Q. Ke, H. Rahmani, Z. Fan, and J. Liu, “Progressive Channel-Shrinking Network,” *IEEE Transactions on Multimedia*, vol. 26, pp. 2016–2026, 2024. doi:10.1109/TMM.2023.3291197
- [15] Y. Tian, H. Chen, T. Guo, C. Xu, and Y. Wang, “Towards Higher Ranks via Adversarial Weight Pruning,” Nov. 2023. doi:10.48550/arXiv.2311.17493
- [16] S. Liao, J. Chen, Y. Wang, Q. Qiu, and B. Yuan, “Embedding Compression with Isotropic Iterative Quantization,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8336–8343, Apr. 2020. doi:10.1609/aaai.v34i05.6350
- [17] J. Meng, Z. Qu, W. Zhou, S. Hu, and B. Ye, “Gradient-Aware Incremental Neural Network Quantization,” in *Network and Parallel Computing*, X. Chen, G. Min, D. Guo, X. Xie, and L. Pu, Eds. Singapore: Springer Nature, 2025, pp. 430–441. doi:10.1007/978-981-96-2864-3_34
- [18] M. Kholiavchenko, “Iterative Low-Rank Approximation for CNN Compression,” Nov. 2019. doi:10.48550/arXiv.1803.08995
- [19] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, “Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration,” Jul. 2019. doi:10.48550/arXiv.1811.00250
- [20] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, “NISP: Pruning Networks using Neuron Importance Score Propagation,” Mar. 2018. doi:10.48550/arXiv.1711.05908
- [21] J.-H. Luo, J. Wu, and W. Lin, “ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression,” Jul. 2017. doi:10.48550/arXiv.1707.06342
- [22] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning Filters for Efficient ConvNets,” Mar. 2017. doi:10.48550/arXiv.1608.08710
- [23] M. Sun, Z. Liu, A. Bair, and J. Z. Kolter, “A Simple and Effective Pruning Approach for Large Language Models,” May 2024. doi:10.48550/arXiv.2306.11695
- [24] L. Liu, S. Zhang, Z. Kuang, A. Zhou, J.-H. Xue, X. Wang, Y. Chen, W. Yang, Q. Liao, and W. Zhang, “Group Fisher Pruning for Practical Network Compression,” Aug. 2021. doi:10.48550/arXiv.2108.00708
- [25] Y. Tang, Y. Wang, Y. Xu, Y. Deng, C. Xu, D. Tao, and C. Xu, “Manifold Regularized Dynamic Network Pruning,” Mar. 2021. doi:10.48550/arXiv.2103.05861
- [26] E. Frantar and D. Alistarh, “SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot,” Mar. 2023. doi:10.48550/arXiv.2301.00774
- [27] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, “Predicting Parameters in Deep Learning,” Oct. 2014. doi:10.48550/arXiv.1306.0543
- [28] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, “Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition,” Apr. 2015. doi:10.48550/arXiv.1412.6553
- [29] C. Li and C. J. R. Shi, “Constrained Optimization Based Low-Rank Approximation of Deep Neural Networks,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 732–747.
- [30] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, “Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets,” in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6655–6659. doi:10.1109/ICASSP.2013.6638949
- [31] X. Zhang, J. Zou, K. He, and J. Sun, “Accelerating Very Deep Convolutional Networks for Classification and Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1943–1955, Oct. 2016. doi:10.1109/TPAMI.2015.2502579
- [32] A. Novikov, D. Podoprikin, A. Osokin, and D. Vetrov, “Tensorizing Neural Networks,” Dec. 2015. doi:10.48550/arXiv.1509.06569
- [33] Z. Tu, X. Chen, P. Ren, and Y. Wang, “AdaBin: Improving Binary Neural Networks with Adaptive Binary Sets,” Oct. 2022. doi:10.48550/arXiv.2208.08084
- [34] E. Vargas, C. V. Correa, C. Hinojosa, and H. Arguello, “BiPer: Binary Neural Networks using a Periodic Function,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5684–5693.
- [35] T. Chen, Z. Li, W. Xu, Z. Zhu, D. Li, L. Tian, E. Barsoum, P. Wang, and J. Cheng, “TernaryLLM: Ternarized Large Language Model,” Jun. 2024. doi:10.48550/arXiv.2406.07177
- [36] S. Xu, Y. Li, T. Ma, B. Zeng, B. Zhang, P. Gao, and J. Lv, “TerViT: An Efficient Ternary Vision Transformer,” Jan. 2022. doi:10.48550/arXiv.2201.08050
- [37] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks,”

- in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 525–542. doi:10.1007/978-3-319-46493-0_32
- [38] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, “Low-bit Quantization of Neural Networks for Efficient Inference,” Mar. 2019. doi:10.48550/arXiv.1902.06822
- [39] T. Chen, B. Ji, T. Ding, B. Fang, G. Wang, Z. Zhu, L. Liang, Y. Shi, S. Yi, and X. Tu, “Only Train Once: A One-Shot Neural Network Training And Pruning Framework,” in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 19 637–19 651.
- [40] M. Mozaffari, A. Yazdanbakhsh, and M. M. Dehnavi, “SLiM: One-shot Quantization and Sparsity with Low-rank Approximation for LLM Weight Compression,” Aug. 2025. doi:10.48550/arXiv.2410.09615
- [41] P. Hu, X. Peng, H. Zhu, M. M. S. Aly, and J. Lin, “OPQ: Compressing Deep Neural Networks with One-shot Pruning-Quantization,” May 2022. doi:10.48550/arXiv.2205.11141
- [42] J. Chang, Y. Lu, P. Xue, Y. Xu, and Z. Wei, “Iterative clustering pruning for convolutional neural networks,” *Knowledge-Based Systems*, vol. 265, p. 110386, Apr. 2023. doi:10.1016/j.knosys.2023.110386
- [43] S. Ye, X. Feng, T. Zhang, X. Ma, S. Lin, Z. Li, K. Xu, W. Wen, S. Liu, J. Tang, M. Fardad, X. Lin, Y. Liu, and Y. Wang, “Progressive DNN Compression: A Key to Achieve Ultra-High Weight Pruning and Quantization Rates using ADMM,” Mar. 2019. doi:10.48550/arXiv.1903.09769
- [44] “[1308.3432] Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation,” <https://arxiv.org/abs/1308.3432>.
- [45] M. Huh, B. Cheung, P. Agrawal, and P. Isola, “Straightening Out the Straight-Through Estimator: Overcoming Optimization Challenges in Vector Quantized Networks,” May 2023. doi:10.48550/arXiv.2305.08842
- [46] Y. Hu, J. Zhu, and J. Chen, “S-STE: Continuous Pruning Function for Efficient 2:4 Sparse Pre-training,” Dec. 2024. doi:10.48550/arXiv.2409.09099
- [47] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” 2009.
- [48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. doi:10.1109/CVPR.2009.5206848
- [49] D. Demszky, K. Guu, and P. Liang, “Transforming Question Answering Datasets Into Natural Language Inference Datasets,” Sep. 2018. doi:10.48550/arXiv.1809.02922
- [50] A. Williams, N. Nangia, and S. Bowman, “A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 1112–1122. doi:10.18653/v1/N18-1101