International Journal of Computer Science and Information Technology
ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 8, Number 3, Year 2026
DOI: https://doi.org/10.62051/ijcsit.v8n3.11
Journal homepage: https://wepub.org/index.php/IJCSIT/index

A Review of Joint Optimization Methods for Neural Network Compression

Haoyu Hu

Department of Physics, Southeast University, Nanjing, 211189, China
2873153747@qq.com

ABSTRACT

As AI models scale, storage, compute, and energy block edge deployment. Single prunin as deep neural networks grow in size and complexity, deploying them on resource-constrained edge devices remains a critical challenge. This paper systematically investigates joint model compression techniques—specifically the integration of pruning, quantization, and knowledge distillation—to achieve efficient inference while preserving accuracy. In this work, I analyze three hybrid frameworks: pruning with distillation, quantization with distillation, and pruning with quantization. By comparing sequential and parallel optimization strategies, this work demonstrates that joint methods consistently outperform single-compression approaches, delivering higher compression ratios and better accuracy retention across benchmark models. Our results further reveal that co-optimizing multiple compression dimensions enables more effective model scaling-down, yet challenges such as stage-wise optimization gaps and hardware-aware design remain. This study not only synthesizes recent advances but also outlines practical pathways toward lightweight, hardware-friendly neural networks for edge AI deployment, highlighting the importance of integrated compression in real-world applications. g, quantization, or distillation struggles with accuracy vs. hardware trade-offs, pushing "pruning-quantization-distillation" combos. This paper surveys core compressors, summarizes three joint schemes—distillation + pruning, distillation + quantization, pruning + quantization—outlines their ideas, and charts future directions.

KEYWORDS

Neural network; Compression; Pruning; Quantization; Knowledge distillation

1. INTRODUCTION

Deep-learning growth bloats storage and compute, barring edge deployment. Pruning, quantization, or distillation alone bleeds accuracy or hardware fit; combining them—"pruning-quantization-distillation"—is the new avenue to extreme compression. This paper survey single methods, dissect three combos (distillation+ pruning, distillation+ quantization, pruning+ quantization), chart latest advances, and outline future work.

2. NEURAL NETWORK COMPRESSION ALGORITHMS

2.1. Overview of Neural Network Compression

The methods for lightweighting neural networks can mainly be classified into two categories: directly designing lightweight networks and compressing existing models.

Lightweight-by-design starts from scratch for mobiles/embedded: fuse depth-wise & group conv (8–9× params cut), dynamic kernels and SE attention (≈1 % params), swap FCs for global pooling and cheap activations. Pros: zero-compression loss, HW-tunable; cons: heavy tuning, some accuracy drop. The main algorithms for compressing neural network models are pruning, quantization and knowledge distillation.

2.2. Common Compression Algorithms

Deep-learning model compression cuts parameters and compute so high-accuracy networks fit edge constraints, slashing storage, energy and latency for negligible accuracy loss. Core techniques: pruning, quantization, knowledge distillation. Their synergy outperforms standalone use; I first detail each.

2.2.1. Pruning

Pruning, a cornerstone of network compression, removes low-impact weights, neurons or channels post- or during training to shrink compute and memory while preserving accuracy.

Based on the different levels of pruning, it can be divided into non-structural pruning and structural pruning. Non-structural pruning refers to randomly removing individual weights, which can achieve extremely high sparsity. However, it requires a dedicated sparse library to truly achieve acceleration on hardware. A classic example is the structured channel pruning of ResNet-50 [1]. This scheme uses "reconstruction error of the feature at the next layer" as the importance indicator. After each layer is pruned, it performs 5-epoch short-term fine-tuning for compensation, and then conducts a global knowledge distillation one more time, using the original ResNet-50 as the teacher. Finally, the inference latency was measured on NVIDIA V100, which dropped from 8.7 ms to 3.1 ms, and the model file size changed from 98 MB to 29 MB. This proves that structured pruning + short-term fine-tuning + distillation can achieve compression acceleration on general GPUs and has become an open benchmark for channel pruning.

2.2.2. Quantification

Neural network quantization is a compression technique that converts network parameters such as weights and activations from high-precision floating-point to low-precision representations. This can significantly reduce the model size and power consumption by several times without sacrificing much in terms of model accuracy. This technique can be classified into uniform quantization and non-uniform quantization, as well as symmetric quantization and asymmetric quantization, based on different standards. Chao Li et al. proposed the Migration Scale PTQ method. Under the TensorRT framework, they performed INT8 quantization on both the weights and activations of YOLOv5s and used 5000 images from the COCO 2017 val dataset for evaluation, with a calibration set of 256 images. This research is perfectly aligned with the "TensorRT INT8 PTQ – YOLOv5s – COCO" classic deployment scenario.

Based on previous studies, the main advantages of quantization are as follows: Reducing storage pressure: The size of the model after quantization is reduced by several times, saving storage space. 3. Reducing power consumption and latency: Firstly, reducing the model's computational load can directly lower power consumption; Secondly, the area and bandwidth occupation decrease as the model size shrinks.

In addition, this method also has certain limitations: 1. Risk of model accuracy loss: After quantization, the bit width of the model decreases significantly, resulting in a decrease in accuracy, and even requiring re-training, which increases the iteration cost. 2. Complex quantization scheme: There are numerous quantization schemes, and the calibration algorithm as well as the bit width allocation need to be manually or automatically searched. Choosing the wrong scheme may even lead to a deterioration in performance.

2.2.3. Knowledge Distillation

Knowledge distillation is a classic method for compressing neural networks. Its core approach is to transfer knowledge from a large and complex model (the teacher model) to a smaller model (the student model), thereby reducing the computational cost and size of the model without significantly sacrificing performance. In 2022, Kan Wu et al. used knowledge distillation to compress the large version of Transformer into a small VIT [2]. The number of parameters decreased from 86M to 5.6M, and the inference speed increased by nearly ten times.

However, this method also has certain drawbacks. For instance, it highly relies on the quality of the teacher model. If there are any issues with the teacher model, the student model will also be affected. Moreover, re-training the teacher model is very costly.

3. JOINT OPTIMIZATION METHOD FOR NEURAL NETWORK COMPRESSION

While individual compression schemes advance speed or size from isolated angles, each carries inherent limits. Joint optimization has therefore become the prevailing paradigm. Surveying recent work, this paper taxonomize paired methods into three strands—distillation plus pruning, distillation plus quantization, and pruning plus quantization—each examined in turn.

3.1. Knowledge Distillation + Pruning

Pruning–distillation hybrids now extend beyond CNNs to vision, language, and speech models. Main approaches: prune-then-distill, distill-then-prune, or joint co-optimization.

3.1.1. Prune first, then distill

In the method of pruning followed by distillation, pruning mainly serves to reduce the model parameters, while distillation utilizes the original model as the teacher model to train the student model, thereby compensating for the accuracy loss caused by the pruning reduction. Lagunas et al. proposed Block Movement Pruning. During the fine-tuning stage, they learned a "block-granularity" mask to achieve a 75% sparsity [3]. Subsequently, they used the unpruned model as the teacher model as a benchmark, reducing the size of BERT to 26% of its original volume while only causing a 1% decrease in accuracy. This was the first demonstration of the feasibility of "high sparsity + distillation" in downstream tasks. Kumar et al. further controlled the loss from distillation during the pruning fine-tuning stage [4]. For the 0.6 B self-supervised speech model, they adopted regularized pruning, and the non-streaming ASR word error rate decreased by approximately 8.9%. Based on these three studies, the following empirical conclusions can be drawn: 1. When the pruning rate is less than or equal to 30%, distillation can almost completely compensate for the accuracy loss caused by pruning; 2. Block-structured optimization is easier to achieve acceleration on general accelerators compared to unstructured optimization.

3.1.2. First distillation, then pruning

In the joint optimization method of first distillation and then pruning, most researchers adopt the approach of first training a smaller but sufficiently accurate student model using knowledge distillation, and then pruning this model to further enhance the computing speed, thereby avoiding the risks associated with directly distilling the pruned model. In recent years, many studies have verified the superiority of the "distillation followed by pruning" process. Prakosa et al. systematically expounded on three commonly used strategies, including "pruning first and then training", "distillation first and then pruning", and "direct pruning" [5]. Their experimental results indicate that the "first distillation then pruning" strategy can most effectively maintain the accuracy of the pruned model. The core advantage of this strategy lies in the fact that during the knowledge distillation stage, a high-quality preprocessing was performed on the student model. Then, based on this optimization,

when performing pruning, it is possible to more accurately identify and retain the key connections, thereby significantly reducing the accuracy loss caused by pruning. Furthermore, Che et al. proposed the HDKP hybrid method and conducted in-depth research on this strategy [6]. The research indicates that traditional pruning methods may overlook the role of channels in the overall function when assessing their importance. Therefore, HKDP introduces an innovative distillation thinking in the pruning stage, that is, when evaluating channels, the correlation between the channel and the corresponding channel of the teacher model is taken into account. This innovative evaluation mechanism makes the pruning decisions also be influenced by the teacher model, thereby being able to better retain the key structures that imitate the teacher model.

To summarize, "distill then prune" first equips the student with high accuracy, giving pruning a reliable starting point; the subsequent pruning shrinks the model and can feed distilled guidance (e.g., HKDP) back into the loop, letting the two steps mutually reinforce.

3.1.3. Pruning and Distillation Simultaneously

To cure the inefficiency and sharp accuracy drop of prune-then-distill chains, recent work merges the two steps: pruning is steered by live distillation signals, trading size for accuracy in one shot. According to teacher usage, these schemes split into two families.

Type 1: the unpruned original network itself acts as teacher, while the progressively pruned offspring serves as student. Eliminating any extra large-model training, the method prunes layer-by-layer yet forces the slim student to mimic the full parent, squeezing maximum accuracy from the shrinking weights. For instance, the NISP method proposed by Yu et al. incorporated this approach earlier. This method determines the importance of channels through "neuron importance score propagation". This method implicitly utilizes the original model to evaluate certain parts of the student model. Zhang et al. explicitly proposed a collaborative compression method, which integrates pruning and distillation within the same closed-loop process [7]. Pruning determines the retention or removal of channels, while distillation utilizes the knowledge of the original model to improve the retained channels, ensuring that the student model can effectively learn the knowledge of the teacher model. However, these methods also have the drawback that their performance is limited by the capabilities of the original model.

Type 2 treats the pruned net as student and an independently trained stronger model as teacher, using this high-capacity mentor to break the original accuracy ceiling The resulting student is smaller yet mimics complex functions, boosting performance; though costlier, it better balances accuracy and efficiency.

3.2. Knowledge Distillation + Quantization

Distillation–quantization co-compression splits into two strands: (1) quantize then distill and (2) variable-quant during distillation. Strand 1 further forks according to whether the student is built from scratch or directly produced by the quantization step.

3.2.1. Quantify first, then distill

This method first obtains a highly compressed but potentially less accurate prototype of the student model through quantization, and then applies the knowledge distillation technique. Using a more powerful teacher model with better performance for training achieves a balance between accuracy and compression rate. The joint optimization method of quantization followed by distillation can be divided into three different types of teacher-student distillation models.

The first type of student model is an independent model, while the teacher model is a quantitative model. The core idea of this type of research is to optimize the process of knowledge distillation and reduce the gap between models in order to enhance the efficiency of distillation. For instance, the Quantization Mimic proposed by Wei et al. is a classic application of this method [8]. This study

employed a quantitative model as the teacher model, reducing the difficulty of knowledge distillation through discrete outputs. The final deployment was the full-precision student model, achieving acceleration while avoiding hardware compatibility issues. The advantage of this method is that it effectively addresses the capacity problems and hardware compatibility issues that may arise from knowledge distillation. However, this method also has the drawback of potentially limiting the upper limit of the student model. The second type is where the student model is a quantized model and the teacher model is the original model. This is the most widely used method currently. Its core idea is to use knowledge distillation to compensate for the accuracy loss caused by quantization. Quantization is a powerful compression method, but inevitably leads to a decrease in accuracy. Fine-tuning also has a relatively limited effect on restoring accuracy. Using the original model as the teacher model can guide the student model to achieve the same level of accuracy as the precise function. For instance, the Apprentice method proposed by Mishra et al. compared various training methods systematically and demonstrated that the approach of first quantizing and then using the original model as the teacher model for distillation could achieve the best results under certain circumstances [9]. The advantage of this method is that the process is clear, no additional training is required, and it is highly practical. However, there is a limitation that when the volume and capacity gap between the teacher model and the student model is large, the training efficiency will decrease.

The last category uses a quantized student and an independently-designed teacher, targeting ultimate performance in specific scenarios. When the test model is too weak or special requirements exist, a separately trained high-performance teacher is introduced. For example, in SAR-ATR, the base model is quantized and distilled with a pre-trained strong teacher. Pros: strong student performance in niche domains; cons: high cost, complex pipeline.

3.2.2. Quantization and Distillation Simultaneously

Quantization and distillation in parallel is a joint optimization method that deeply combines quantization-aware training and knowledge distillation. A classic core idea is to introduce a high-precision teacher model or supervision signal during the quantization training of the low-precision student model, to further guide the student model. Hubara et al. introduced knowledge distillation in the standard quantitative perception training [10]. This method first trains a large and highly accurate teacher model, and then builds a smaller student model that is pre-set to be diameter-controlled. During the training of the student model, both the teacher model and the Ground Truth are used for supervision. This method can effectively achieve knowledge transfer, and it also demonstrates that this method is significantly superior to the traditional method of training first and then quantifying, proving the great value of introducing distillation in low-precision training. This research is an early practice of quantization and distillation simultaneously, providing a foundation for subsequent studies. Furthermore, there have been systematic studies that have examined the combination of quantification and distillation. Finally, the process of distilling while quantizing has become a standard.

3.3. Pruning + Quantization

Pruning and quantization are advanced compression techniques. Leading work like APQ jointly searches architecture, pruning, and quantization under hardware constraints for quantization-aware accuracy, boosting both efficiency and performance; extensions to NLP have also appeared.

This article reviews pruning–quantization combinations, classifying them into sequential (prune→quantize) and joint approaches.

3.3.1. Prune first, then quantize

Prune-then-quantize progressively simplifies the network before reducing precision, cutting storage and compute; it is straightforward and easy to implement. A widely influential classic solution is the deep compression process proposed by Han et al [11]. First, non-structured pruning is carried out to remove redundancies. Then, the model is quantized. That is, it is replaced by algorithms such as K-

means, and finally, Huffman coding is applied for further compression. A compression ratio of over 35 times has been successfully achieved with minimal loss of accuracy. This research has successfully demonstrated the feasibility of the strategy of pruning first and then quantization, and has laid the foundation for subsequent studies. With the development of deep learning, researchers have begun to explore the combination of structured pruning and quantization. For example, some studies remove the entire feature channel first and then quantize the simplified model. Although this method is slightly inferior in compression capability compared to non-structured pruning, it has more advantages in practical applications.

Currently, this method suffers from mismatched stage-wise objectives; future work should pursue tightly integrated optimization.

3.3.2. Pruning and Quantization Simultaneously

Per-iteration co-tuning of pruning and quantization as a single unified optimization for global joint optimum. The parallel pruning and quantization strategy (CLIP-Q) proposed by Tung et al. is a typical representative of the method that performs pruning and quantization simultaneously during each training iteration, rather than following the traditional sequence [12]. The CLIP-Q method achieves this through a meticulously designed iterative optimization framework, enabling the pruning and quantization processes to interactively feedback and collaboratively optimize each other, thereby achieving the maximum model compression while maintaining its performance. The innovation lies in the introduction of a re-training mechanism after each iteration round to restore the model accuracy. This design allows the pruning and quantization processes to adapt to each other and optimize together in multiple iterations. However, this method also has limitations in terms of high computational complexity and time-consuming nature. The APQ method proposed by Wang et al [13]. The precision predictor is another key innovation of the APQ method. As a lightweight prediction model, it can quickly predict the corresponding model accuracy based on the encoded network structure and quantization strategies. This design avoids the time-consuming evaluation process in traditional compression methods, enabling APQ to conduct efficient exploration in a vast search space. During the fine-tuning stage of the precision predictor, APQ introduces quantization operations to the original network, allowing the predictor to learn the complex mapping relationship between the network structure strategy, quantization strategy, and final accuracy.

These methods increasingly automate compression and optimize efficiency. While early approaches like CLIP-Q require manual hyperparameter tuning (pruning rates, bit widths), newer ones (APQ, UVNQ) determine these automatically through optimization. They also prioritize hardware compatibility—combining structured pruning with efficient quantization ensures both theoretical superiority and practical deployment effectiveness.

4. CONCLUSION AND RESEARCH PROSPECTS

This survey reviews joint optimization methods for neural network compression. As single techniques (pruning, quantization, knowledge distillation) face inherent precision-compression limits and hardware constraints, multi-technology integration becomes essential for edge deployment. We analyze three joint strategies—"distillation+pruning", "distillation+quantization", and "pruning+quantization"—comparing serial (sequential) versus parallel implementations. Despite superior potential over single methods, joint optimization faces key challenges: (1) Stage-wise optimization disconnect—serial approaches treat compression as independent steps without global optimality, requiring extensive hyperparameter tuning; (2) Limited hardware awareness—theoretical size/computation reductions often fail to translate to real-world gains without considering deployment hardware (cache, memory bandwidth); (3) Weak theoretical foundations—lacking rigorous explanations for why joint optimization works, particularly how distillation compensates for pruning/quantization losses and the interactions between different compression techniques.

REFERENCES

[1] C, S. K., Dhiman, J. K., Adiga, N., & Singh, S. (2025). Synergistic Effects of Knowledge Distillation and Structured Pruning for Self-Supervised Speech Models. arXiv, 2502.05837.

[2] Han, S., Mao, H., & Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv, 1510.00149.

[3] Hongle, C., Qirui, S., Juan, C., & Quan, W. (2021). HKDP: A Hybrid Approach On Knowledge Distillation and Pruning for Neural Network Compression. 2021 18th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 188–193.

[4] Lagunas, F., Charlaix, E., Sanh, V., & Rush, A. (2021). Block Pruning For Faster Transformers. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 10619–10629.

[5] Li, J., Gao, N., Shen, T., Zhang, W., Mei, T., & Ren, H. (2020). SketchMan: Learning to Create Professional Sketches. Proceedings of the 28th ACM International Conference on Multimedia, 3237–3245.

[6] Mishra, A., & Marr, D. (2017a). Apprentice: Using Knowledge Distillation Techniques To Improve Low-Precision Network Accuracy. arXiv, 1711.05852.

[7] Prakosa, S. W., Leu, J.-S., & Chen, Z.-H. (2021). Improving the accuracy of pruned network using knowledge distillation. Pattern Analysis and Applications, 24(2), 819–830.

[8] Tessier, H., Boukli, G., & Gripon, V. (2023). ThinResNet: A New Baseline for Structured Convolutional Networks Pruning. arXiv, 2309.12854.

[9] Wang, T., Wang, K., Cai, H., Lin, J., Liu, Z., Wang, H., Lin, Y., & Han, S. (2020). APQ: Joint Search for Network Architecture, Pruning and Quantization Policy. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2075–2084.

[10] Wei, Y., Pan, X., Qin, H., Ouyang, W., & Yan, J. (2018). Quantization Mimic: Towards Very Tiny CNN for Object Detection. In V. Ferrari, M. Hebert, C. Sminchisescu, & Y. Weiss (Eds.), Computer Vision – ECCV 2018. Springer International Publishing, 11212, 274–290.

[11] Wu, K., Zhang, J., Peng, H., Liu, M., Xiao, B., Fu, J., & Yuan, L. (2022). TinyViT: Fast Pretraining Distillation for Small Vision Transformers. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, & T. Hassner (Eds.), Computer Vision – ECCV 2022. Springer Nature Switzerland, 13681, 68–85.

[12] Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., Gao, M., Lin, C.-Y., & Davis, L. S. (2018). NISP: Pruning Networks Using Neuron Importance Score Propagation. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 9194–9203.

[13] Zhang, R., Wu, Q., & Zhou, Y. (2025). Network Security Situation Element Extraction Algorithm Based on Hybrid Deep Learning. Electronics, 14(3), 553.