iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78048

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

Pruning in Model Compression: Techniques, Challenges, and Future Directions

Sajana T[1], Anagha P[2]
Assistant Professor, Department of Computer Applications, Chinmaya Institute of Technology, Kannur University, Kerala, India

Abstract: ***Deep learning models have achieved remarkable success across domains such as computer vision, natural language processing, and speech recognition. However, their increasing size and computational requirements pose significant challenges for deployment on resource-limited devices. Model compression techniques aim to reduce model size and computational cost without significantly compromising accuracy. Among these techniques, pruning has emerged as one of the most effective methods. This paper provides a comprehensive overview of pruning as a model compression strategy, exploring its principles, types, applications, and challenges. Comparative insights and future research directions are also discussed to highlight pruning's continuing relevance in the era of efficient AI.***

Keywords: Model Compression, Parameter Reduction, Pruning Methods, Quantization, Low-Rank Factorization, Parameter Sharing, Knowledge Transfer Methods, Architecture Optimization Methods,

I. INTRODUCTION

The rapid advancement of deep neural networks (DNNs) has resulted in remarkable improvements across various artificial intelligence tasks. Models such as ResNet, BERT, and GPT architectures often contain millions or even billions of parameters. Although these models achieve exceptional accuracy, their high computational and memory demands make them impractical for deployment on resource-constrained devices such as smartphones, IoT nodes, and embedded systems. To overcome these challenges, researchers have introduced model compression techniques that aim to reduce the complexity of DNNs while preserving their accuracy. Popular approaches include quantization, knowledge distillation, low-rank factorization, and pruning [1], [5]. Among these, Among these techniques, knowledge distillation transfers knowledge from a large teacher network to a smaller student network, improving efficiency without significant performance loss [6], [7], [8], [9]. Pruning is particularly favoured as it focuses on removing redundant parameters and connections within the network [2], [3], [4]. This process significantly reduces model size, accelerates inference, and lowers energy consumption, making real-time AI applications feasible on edge and low-power devices.

II. MODEL COMPRESSION OVERVIEW

Model compression refers to the process of simplifying a deep learning model without significantly degrading its performance. The main goal of model compression is to make large, resource-intensive models more efficient and suitable for deployment on devices with limited hardware capabilities, such as smartphones, IoT devices, and embedded systems [1].
Model compression methods generally fall into three categories:

A. Parameter Reduction Methods

Parameter Reduction Methods refer to techniques used in machine learning and deep learning to decrease the number of trainable parameters in a model without significantly affecting its performance. These methods directly shrink the size of the model while keeping its structure mostly the same. These methods are crucial for deploying models on devices with limited computational power (like mobile or edge devices) and for improving training efficiency [3].
Core Techniques are:

1) *Pruning:* Pruning removes parameters that contribute minimally to the model's output. It is based on the observation that many weights in deep networks have very small magnitudes and can be eliminated without degrading model performance. By removing redundant connections, pruning reduces the overall number of parameters and makes the network more efficient. In many cases, pruning is followed by a retraining phase to recover any lost accuracy and further optimize the remaining weights. As a result, pruning significantly decreases model size, computational cost, and inference time [1], [2], [3], [4].
2) *Quantization:* Quantization reduces the precision of model weights and activations. Typical conversions include:
   - 32-bit floating point → 16-bit floating point
   - 32-bit floating point → 8-bit integers (INT8)

Lower-precision arithmetic results in smaller models, reduced memory bandwidth, and faster inference, especially on CPUs and NPUs that support integer operations. Quantization-aware training (QAT) further improves robustness [1].

3) *Low-Rank Factorization :* Large weight matrices are decomposed into products of smaller, low-rank matrices. Since many weight matrices contain redundant information, low-rank approximations preserve performance while reducing the number of parameters and computation [1], [3].
4) *Parameter Sharing :* Parameter sharing reuses the same set of weights across different parts of the model. Examples include:

- Convolutional neural networks (CNNs), where filter weights are shared across spatial locations
- Recurrent neural networks (RNNs), where weights are shared across time steps
- Shared embeddings in transformer models

Through parameter sharing, models can achieve efficient learning with fewer parameters while preserving accuracy and generalization performance [5].

B. Knowledge Transfer Methods

Knowledge Transfer Methods refer to techniques that enable a machine learning or deep learning model to transfer knowledge learned from one task, dataset, or model to another. These methods are essential when labeled data is limited or when we want to reuse knowledge from previously trained models to improve efficiency and performance.
Knowledge distillation transfers knowledge from a large model (teacher) to a smaller one (student) [6].
Core Technique:

1) *Knowledge Distillation :* Knowledge distillation involves a large, high-capacity model known as the teacher, which guides the training of a smaller student model. Instead of training the student only with hard labels (0/1), the teacher provides soft labels, i.e., probability distributions over classes. Soft labels contain richer information and help the student model mimic the teacher's decision boundaries. This method achieves substantial compression while maintaining accuracy close to the original model [6], [8].
2) *Feature-Based Knowledge Transfer:* Feature-based knowledge transfer allows the student model to learn from the intermediate feature representations of the teacher model. Instead of only using the final output, the student is trained to mimic hidden layer activations or feature maps produced by the teacher. These internal representations contain important structural information about the data. By aligning its features with those of the teacher, the student model can learn more effective representations and improve performance even with fewer parameters [7], [9].
3) *Response-Based Knowledge Transfer:* Response-based knowledge transfer focuses on transferring the final output responses of the teacher model to the student model. The student learns to replicate the probability distribution generated by the teacher for each class. These responses provide additional information about class similarities and decision boundaries. As a result, the student model can achieve better generalization compared to training only with hard labels [6], [8].
4) *Parameter Transfer:* Parameter transfer involves copying or sharing certain parameters or layers from the teacher model to the student model. The student model may initialize its weights using the teacher's trained parameters and then fine-tune them for the target task. This approach helps the student model learn faster and often improves performance because it starts with already learned knowledge instead of random initialization [7].

C. Architecture Optimization Methods

Designing efficient architectures like MobileNet and EfficientNet inherently minimizes complexity. Architecture Optimization in deep learning refers to improving the design and structure of a neural network to achieve better performance such as higher accuracy, lower latency, or reduced computational cost without significantly increasing resources (e.g., memory, power, or inference time). It focuses on how the model is built, rather than just compressing or pruning it afterward [1][3][5].Examples of Efficient Architectures are

1) *MobileNet :* MobileNet uses depthwise separable convolutions, which separate spatial and channel-wise processing. This dramatically reduces computation compared to standard convolutions.
2) *EfficientNet :* EfficientNet introduces compound scaling, which uniformly scales depth, width, and resolution of the network. This balanced scaling produces highly efficient models with superior accuracy-to-size ratios.

3) *ShuffleNet and SqueezeNet :* These models incorporate strategies such as: Channel shuffling, Fire modules, Grouped convolutions. They achieve high accuracy while maintaining extremely small model sizes, making them ideal for mobile devices.
4) *Neural Architecture Search (NAS) :* NAS automatically searches for optimal network architectures using reinforcement learning or evolutionary algorithms. It can discover architectures that outperform manually designed models in efficiency and accuracy [1][3][4][5].

III. PRUNING TECHNIQUES

Pruning is a model compression technique that aims to make neural networks smaller and faster by removing unnecessary parameters (such as weights, neurons, or filters) that contribute little to the model's performance. The idea is based on the observation that many parameters in a trained model are redundant or have very small values, meaning they don't significantly affect the final output [1][3]. The major approaches include:

A. *Weight Pruning*

Weight Pruning technique used to reduce the size and complexity of neural networks by removing unimportant or redundant weights. It identifies connections with very small magnitudes that contribute little to the model's output and sets them to zero, creating a sparse network. This reduces memory usage, computational cost, and inference time without significantly affecting accuracy [1][5].
For instance, in the work of Han et al. (2015), "Deep Compression" demonstrated up to 9× model size reduction using iterative weight pruning followed by quantization and Huffman coding [1].

B. *Neuron Pruning*

Neuron pruning technique removes entire neurons (or nodes) from a neural network instead of just individual weights. It identifies neurons that contribute little to the final output, often determined by their low activation values or small outgoing weight magnitudes and eliminates them along with their connections. This reduces the number of computations and memory requirements, leading to a smaller, faster, and more efficient model [3][4]. This is useful in convolutional neural networks (CNNs), where unimportant filters can be removed to reduce computation while preserving feature diversity.
Neuron pruning is considered a form of structured pruning, as it removes whole components of the network, making it easier to accelerate on modern hardware while maintaining nearly the same prediction accuracy [4].

C. *Structured Pruning*

Structured pruning removes groups of parameters such as channels, filters, or layers while maintaining the network's overall structure. This method is hardware-friendly, enabling efficient parallelization on GPUs and accelerators [3]. Structured pruning removes entire structures such as neurons, filters, channels, or even layers from a neural network, rather than individual weights. This makes the resulting model smaller, faster, and easier to run efficiently on hardware like CPUs, GPUs, or mobile devices. The pruning process typically identifies low-importance structures based on criteria like low activation strength, small weight magnitude, or minimal contribution to the output [3][4]. Unlike unstructured pruning, which creates sparse weight matrices, structured pruning maintains regular network architecture, allowing better compatibility with existing deep learning frameworks and achieving significant speed and memory improvements with minimal accuracy loss.

D. *Dynamic Pruning*

Dynamic pruning adapts during inference by skipping unimportant computations depending on the input. This approach is beneficial for tasks with variable input complexity, such as object detection or speech recognition [3].

E. *Lottery Ticket Hypothesis*

Proposed by Frankle and Carbin (2019), this theory suggests that within large networks, there exist smaller "winning subnetworks" that can be trained to achieve performance comparable to the original model. Identifying and training these subnetworks effectively acts as a pruning mechanism [2].

IV. COMPARATIVE ANALYSIS AND RESULTS

Pruning has shown strong empirical results across benchmark models. For example, pruning ResNet-50 can reduce model parameters by up to 80–90% with only 1–2% loss in accuracy when combined with fine-tuning. Similarly, pruning techniques applied to BERT and GPT models have reduced inference time by up to 40% while maintaining high performance in NLP benchmarks.

A key factor influencing pruning effectiveness is retraining. After pruning, fine-tuning or retraining helps the network recover lost performance by adjusting the remaining weights. Tools like TensorFlow Model Optimization Toolkit and PyTorch's pruning API provide standardized implementations to apply pruning efficiently in practice.

V. CHALLENGES AND FUTURE DIRECTIONS

Despite its advantages, pruning still faces several challenges:

1) Trade-off Between Accuracy and Compression: Over-pruning can degrade model accuracy. Balancing efficiency and performance remains complex.

2) Lack of Standardization: Different pruning algorithms behave differently across architectures, leading to inconsistent results.

3) Hardware Constraints: Sparse representations created by unstructured pruning often fail to achieve actual speedups on certain hardware.

4) Automation: Automated or learning-based pruning methods are still in early stages and require further exploration.

Future research aims to integrate pruning with neural architecture search (NAS) and quantization-aware training to create fully optimized, deployment-ready models. The development of hardware-aware pruning strategies will further bridge the gap between theoretical and practical efficiency gains.

VI. CONCLUSION

Pruning remains one of the most practical and impactful techniques for model compression in deep learning. By systematically removing redundant parameters, it enables efficient deployment of neural networks on resource-constrained devices such as smartphones, IoT devices, and embedded systems without significant loss in prediction accuracy. This makes pruning highly valuable for real-world applications where memory, computational power, and energy consumption are limited.

In addition to reducing model size, pruning also improves inference speed and energy efficiency, allowing deep learning models to run faster and consume less power. Techniques such as weight pruning, neuron pruning, and structured pruning help simplify network architectures while maintaining their ability to learn complex patterns. Furthermore, modern approaches like dynamic pruning and the Lottery Ticket Hypothesis demonstrate that even smaller subnetworks within large models can achieve performance comparable to the original networks.

When combined with other compression methods such as quantization and knowledge distillation, pruning can further enhance model efficiency and scalability. As deep learning models continue to grow in size and complexity, the importance of effective model compression strategies will continue to increase.

Overall, continued research and innovation in structured, dynamic, and automated pruning techniques will play a crucial role in enabling scalable, sustainable, and accessible artificial intelligence systems. These advancements will help bridge the gap between powerful deep learning models and their deployment in real-world environments.

REFERENCES

[1] Han, S., Mao, H., & Dally, W. J. (2015). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. arXiv preprint arXiv:1510.00149.

[2] Frankle, J., & Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. International Conference on Learning Representations (ICLR).

[3] Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Guttag, J. (2020). What is the State of Neural Network Pruning? Proceedings of Machine Learning and Systems (MLSys).

[4] Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2017). Pruning Convolutional Neural Networks for Resource Efficient Inference. ICLR.

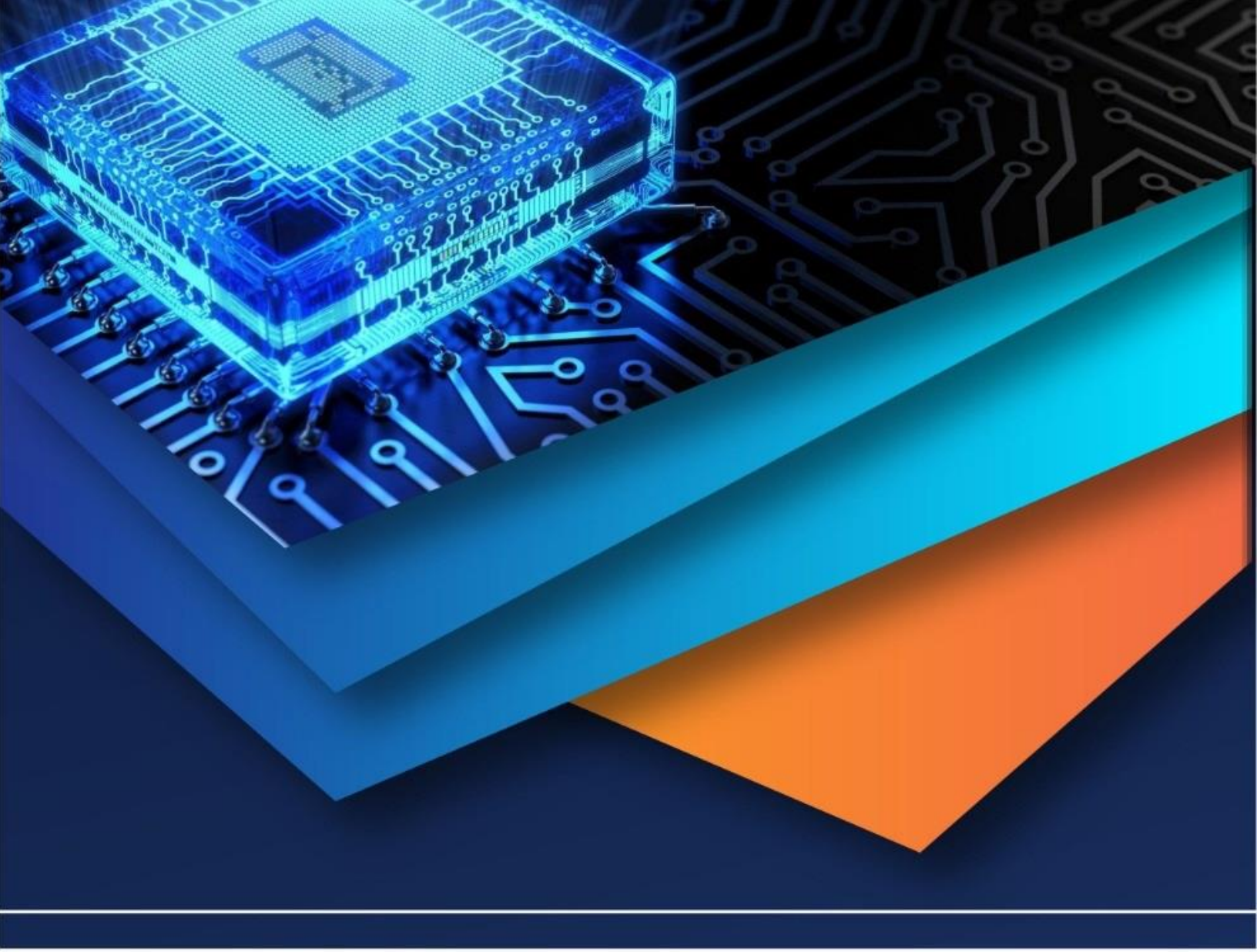
[5] Gale, T., Elsen, E., & Hooker, S. (2019). The State of Sparsity in Deep Neural Networks. arXiv preprint arXiv:1902.09574.

[6] G. Hinton, O. Vinyals, and J. Dean, "Distilling the Knowledge in a Neural Network," NIPS Deep Learning Workshop, 2015

[7] A. Romero et al., "FitNets: Hints for Thin Deep Nets," International Conference on Learning Representations (ICLR), 2015.

[8] Y. Kim and A. M. Rush, "Sequence-Level Knowledge Distillation," Proceedings of EMNLP, 2016.

[9] M. Ji, B. Heo, and S. Park, "Show, Attend and Distill: Knowledge Distillation via Attention-based Feature Matching," *arXiv preprint*, 2021.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY