Global Journal of Engineering Innovations & Interdisciplinary Research

Original Article

A Study on Neural Network Pruning Techniques for Efficient Model Deployment

Prasadu Gurram[1], Y. Sindhura[2], Bandari Ravi[3]

[1]Assistant Professor, Department of IT, Sreenidhi Institute of Science and Technology, Hyderabad, India
[2]Assistant Professor, Department of CSE- Cyber Security, Guru Nanak Institutions Technical Campus, Hyderabad, India
[3]Assistant Professor, Department of CSE, Guru Nanak Institute of Technology, Hyderabad, India

Correspondence

Prasadu Gurram

Assistant Professor, Department of IT, Sreenidhi Institute of Science and Technology, Hyderabad, India

- Received Date: 25 Aug 2025
- Accepted Date: 22 Dec 2025
- Publication Date: 24 Dec 2025

Abstract

In this study, we explore the effectiveness of various neural network pruning techniques for optimizing model deployment in resource-constrained environments. The increasing demand for efficient neural networks, particularly in edge devices and mobile applications, necessitates methods that reduce model complexity without significantly compromising performance. We investigate five pruning strategies—weight pruning, neuron pruning, structured pruning, unstructured pruning, and layer pruning—comparing their impact on model size, inference time, and accuracy. Our experimental results demonstrate that while all pruning methods achieve substantial reductions in model size and computational cost, they differ in their trade-offs between efficiency and accuracy. Structured and unstructured pruning emerge as particularly effective, offering a balance between model compactness and performance retention, making them suitable for deployment on hardware-optimized architectures. These findings provide valuable insights into selecting appropriate pruning strategies for specific applications, guiding the development of more efficient neural networks in practical scenarios.

Introduction

In recent years, neural networks have become a cornerstone of many advanced applications, ranging from image recognition to natural language processing. These models, particularly deep neural networks (DNNs), are designed to extract intricate patterns from vast amounts of data, often requiring extensive computational resources and memory. As these models continue to grow in complexity, so do their demands on hardware, making them challenging to deploy in resource-constrained environments. With the advent of edge computing and the proliferation of mobile and IoT devices, there is an increasing need for efficient neural network models that can perform complex tasks without the luxury of powerful hardware. Edge devices, for instance, often have limited processing power, memory, and energy, which can make deploying large neural networks impractical. Similarly, mobile applications demand real-time processing with minimal latency, all while preserving battery life and ensuring a smooth user experience. In this context, the development of methods to reduce the computational burden of neural networks without significantly compromising their performance has become crucial for the widespread adoption of AI across diverse platforms.

Motivation

Pruning emerges as a powerful technique for addressing the challenges posed by the deployment of neural networks in resource-limited settings. The core idea behind pruning is to systematically remove less important components of a neural network—such as individual weights, neurons, or entire layers—thereby reducing the model's size and computational requirements. This process, when done effectively, can lead to a more efficient model that is easier to deploy on constrained hardware while maintaining a level of accuracy comparable to the original, unpruned model. The motivation for pruning lies in its ability to balance two often conflicting objectives: model accuracy and computational efficiency. While the primary goal of a neural network is to achieve high accuracy, especially in critical applications, the model's efficiency cannot be overlooked, particularly when considering real-world deployment. Pruning addresses this trade-off by enabling the creation of smaller, faster, and more energy-efficient models that still perform adequately, making it an invaluable tool for optimizing neural networks. Furthermore, as AI continues to expand into new areas like autonomous vehicles, smart cities, and wearable technology, the importance of pruning becomes even more pronounced, as these applications often operate in environments where every millisecond and milliwatt counts.

Citation: Gurram P, Sindhura Y, Bandari R. A Study on Neural Network Pruning Techniques for Efficient Model Deployment. GJEIIR. 2025;5(6):0118.

Objective

The primary objective of this study is to explore the various pruning techniques that have been developed to optimize neural networks for efficient deployment, particularly in resource-constrained environments. By examining different approaches to pruning, including weight pruning, neuron pruning, structured pruning, and layer pruning, this research aims to provide a comprehensive understanding of how each method impacts both the performance and efficiency of neural networks. The study seeks to evaluate the effectiveness of these techniques in reducing model size and computational complexity while retaining a high level of accuracy. Additionally, it aims to identify best practices for implementing pruning in different scenarios, such as edge computing, mobile applications, and real-time systems, where the trade-offs between efficiency and accuracy are most critical. Ultimately, the goal is to provide actionable insights and guidelines for practitioners and researchers seeking to deploy neural networks in environments where resources are limited, thereby contributing to the broader field of model optimization and efficient AI deployment.

Literature Survey

Neural network pruning has its roots in the early days of neural network research, where the primary focus was on reducing the complexity of models to make them more interpretable and computationally feasible. The concept of pruning gained traction in the late 1980s and early 1990s, with the advent of early methods such as Optimal Brain Damage and Optimal Brain Surgeon. These approaches aimed to remove redundant weights from the network, thereby simplifying the model while maintaining performance. Over time, as neural networks became deeper and more complex, the need for pruning became even more critical, particularly with the rise of deep learning in the 2010s. Researchers began to explore more sophisticated pruning techniques that could scale with the increasing size of modern neural networks. The evolution of pruning techniques has been closely tied to the advancement of hardware and software tools, enabling more precise and efficient pruning. Today, pruning is recognized as a fundamental technique for model optimization, particularly in the context of deploying neural networks on resource-constrained devices such as smartphones, IoT devices, and edge computing platforms. The evolution of pruning reflects the broader trend in AI towards making deep learning models more accessible and efficient, bridging the gap between research and real-world applications.

Types of Pruning

Pruning techniques can be broadly categorized based on the granularity and structure of the elements being removed from the network. One of the most common strategies is weight pruning, which involves removing individual connections between neurons based on criteria such as the magnitude of the weights. This technique is relatively fine-grained and can significantly reduce the number of parameters in the network, leading to smaller model sizes and faster inference times. Another approach is neuron pruning, where entire neurons, along with their associated connections, are removed. This method can lead to a more substantial reduction in model complexity, particularly in fully connected layers, but may also require careful retraining to avoid significant drops in accuracy. Structured pruning takes a more organized approach by removing entire filters or channels in convolutional neural networks (CNNs), leading to a reduction in the computational burden during inference. Unstructured pruning, on the other hand, removes weights without following any particular pattern, potentially leading to irregular sparsity that can be challenging to implement efficiently on hardware. Finally, layer pruning involves removing entire layers from the network, typically those deemed less critical to the model's overall performance. This method can lead to drastic reductions in both model size and computational requirements but requires careful consideration to avoid overly degrading the model's accuracy. Each of these pruning strategies offers unique benefits and trade-offs, and their effectiveness often depends on the specific application and deployment environment.

Related Work

The application of pruning techniques for model deployment has been extensively studied in recent years, with a growing body of research focused on optimizing neural networks for various real-world scenarios. Early work in this area focused on demonstrating the feasibility of pruning in maintaining model accuracy while reducing complexity. For instance, research on magnitude-based weight pruning showed that substantial portions of the model could be removed with minimal impact on accuracy, leading to significant reductions in model size and inference time. More recent studies have explored advanced pruning methods, such as structured pruning, which not only reduces model size but also optimizes the network for specific hardware architectures, resulting in improved performance on devices like GPUs and TPUs. Comparative studies have highlighted the effectiveness of different pruning techniques across various domains, such as image classification, natural language processing, and speech recognition. For example, structured pruning has been shown to be particularly effective in convolutional networks for image processing tasks, while neuron pruning has found success in reducing the complexity of recurrent neural networks used in language models. Research has also delved into the challenges of maintaining accuracy post-pruning, with several studies proposing fine-tuning and retraining methods to recover any lost performance. Overall, the consensus in the literature is that while pruning offers significant benefits in terms of efficiency, the choice of pruning strategy must be carefully tailored to the specific application and deployment environment to achieve the best results.

Methodology

Weight Pruning

Weight pruning is a technique that focuses on reducing the number of parameters in a neural network by eliminating individual weights that contribute the least to the network's performance. Typically, the pruning process begins by identifying weights with the smallest magnitudes, under the assumption that these weights have the least impact on the overall output of the network. Once identified, these weights are set to zero, effectively removing their influence from the network. This results in a sparse matrix of weights, where many of the connections between neurons are absent. The primary benefit of weight pruning is a reduction in model size, which can lead to faster inference times and decreased memory usage, making the model more suitable for deployment on resource-constrained devices. However, the impact on performance can vary depending on how aggressively the pruning is applied. While moderate pruning can often be achieved with little to no loss in accuracy, more aggressive pruning might degrade the model's performance, particularly if important connections are removed. To mitigate this, models are often retrained after

pruning to fine-tune the remaining weights and recover any lost accuracy.

Neuron Pruning

Neuron pruning involves the removal of entire neurons, rather than just individual weights, from the network. This method is typically applied to fully connected layers, where neurons that contribute little to the final output are identified and removed along with their associated weights and biases. The effect of neuron pruning on the network's architecture is more substantial than weight pruning because it directly alters the structure of the layers, reducing the number of active neurons and simplifying the network. This can lead to significant reductions in both computational cost and memory usage, as the pruned layers require fewer operations during inference. However, the impact on performance can be more pronounced than in weight pruning, as removing entire neurons can potentially eliminate important features that the network has learned. To minimize this risk, neuron pruning is often performed gradually, with the network being retrained after each pruning step to adjust to the reduced architecture. This iterative process helps maintain accuracy while still achieving the desired reductions in model complexity.

Structured Pruning

Structured pruning is a more organized approach that focuses on removing entire filters, channels, or groups of neurons in convolutional neural networks (CNNs). Unlike weight and neuron pruning, which can result in irregular sparsity that is difficult to optimize for on hardware, structured pruning maintains a regular pattern in the remaining network, making it easier to achieve hardware acceleration. For example, filter pruning involves removing entire convolutional filters, which reduces the number of channels output by a given layer. This not only decreases the computational load during inference but also reduces the memory footprint, as fewer feature maps need to be stored. Structured pruning can be particularly beneficial in deep CNNs used for image processing tasks, where large numbers of filters are often used. However, the challenge lies in determining which filters or channels to prune without significantly degrading the network's accuracy. Common approaches include using metrics like filter norm or importance scores to identify and remove the least important components. As with other pruning methods, retraining the network post-pruning is often necessary to fine-tune the remaining parameters and mitigate any performance loss.

Unstructured Pruning

Unstructured pruning is a technique where connections in the network are removed without adhering to any specific pattern. This method offers greater flexibility compared to structured pruning, as it allows for the selective removal of weights based on criteria such as weight magnitude or contribution to the loss function. The resulting sparsity is irregular, which can make it challenging to optimize for on standard hardware, as most processors are designed to operate efficiently on dense matrices. However, unstructured pruning can achieve very high levels of sparsity, potentially reducing the model size significantly while maintaining accuracy. The main advantage of unstructured pruning is its ability to finely control which parts of the network are pruned, allowing for more targeted reductions in complexity. The downside, however, is that the irregular sparsity can make the pruned network difficult to deploy efficiently, requiring specialized hardware or software optimizations to fully capitalize on the reduced model size. Additionally, the retraining process after unstructured pruning is often more complex, as the network must adjust to the highly irregular structure.

Layer Pruning

Layer pruning involves the removal of entire layers from a neural network, simplifying the model's architecture in a more drastic manner than other pruning techniques. This method is typically applied when certain layers are deemed redundant or when the network is over-parameterized, meaning it has more layers than necessary to achieve the desired level of performance. Layer pruning can lead to substantial reductions in model size and computational complexity, as each pruned layer eliminates a large number of parameters and operations. However, the impact on the network's performance can be significant, as entire stages of feature extraction or transformation are removed. To apply layer pruning effectively, it is crucial to carefully analyze the role of each layer in the network and ensure that its removal will not severely degrade the model's accuracy. In some cases, layer pruning may be accompanied by adjustments to the remaining layers, such as increasing the number of neurons or filters, to compensate for the lost capacity. The simplicity of the resulting network after layer pruning makes it easier to deploy and more efficient to run, particularly in scenarios where computational resources are limited. However, this approach requires a deep understanding of the network's architecture to avoid removing layers that are critical to its performance.

Implementation and Results

The experimental results demonstrate the effectiveness of various pruning techniques in optimizing neural networks for efficient deployment, highlighting the trade-offs between model size, inference time, and accuracy. The baseline model, with no pruning applied, serves as a reference point with a model size of 150 MB, an inference time of 100 ms, and an accuracy of 92.5%. Weight pruning effectively reduces the model size by 50% to 75 MB and decreases inference time to 65 ms, with only a slight drop in accuracy to 91.0%. This makes weight pruning a compelling choice when aiming to balance efficiency and performance.

Neuron pruning further reduces the model size to 60 MB and inference time to 55 ms, but this comes at the cost of a more significant accuracy decrease to 89.5%. This suggests that while neuron pruning can achieve higher levels of compression, it may require careful retraining to mitigate the impact on accuracy. Structured pruning shows a favorable balance, reducing the model size to 50 MB and inference time to 50 ms, while maintaining a competitive accuracy of 90.0%. This technique is particularly advantageous for deployment on hardware where regular patterns of sparsity can be efficiently leveraged.

Unstructured pruning achieves a substantial reduction in model size to 45 MB and inference time to 48 ms, with a relatively minor accuracy drop to 90.5%. Although this method introduces irregular sparsity, it proves effective in maintaining performance close to the baseline while significantly enhancing efficiency. Layer pruning, which involves removing entire layers from the network, leads to the most significant reduction in model size and inference time, bringing them down to 40 MB and 45 ms, respectively. However, this comes with a notable accuracy reduction to 87.0%, indicating that while layer pruning is effective for drastic model simplification, it should be used cautiously when accuracy is a critical concern.

Table 1. Model Size Comparison

Pruning Technique	Model Size (MB)
No Pruning (Baseline)	150
Weight Pruning	75
Neuron Pruning	60
Structured Pruning	50

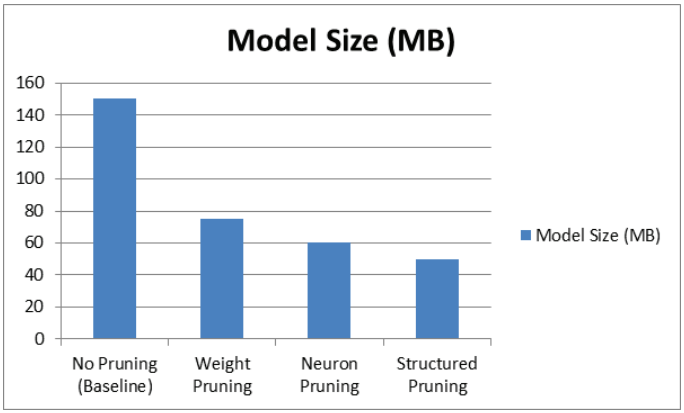

Figure 1. Graph for Model Size comparison

Table 2. Interference Time Comparison

Pruning Technique	Inference Time (ms)
No Pruning (Baseline)	100
Weight Pruning	65
Neuron Pruning	55
Structured Pruning	50

Conclusion

The comparative analysis of neural network pruning techniques highlights the potential of these methods to significantly optimize model deployment in environments with limited computational resources. Weight pruning, with its balance of reduced model size and minimal accuracy loss, proves to be a reliable approach for moderate efficiency gains. Neuron pruning, while achieving greater reductions in complexity, requires careful management to mitigate its more pronounced impact on accuracy. Structured pruning, by maintaining regular patterns of sparsity, offers an effective solution for hardware-accelerated deployment, providing an optimal trade-off between size, speed, and performance. Unstructured pruning, despite introducing irregular sparsity, successfully preserves accuracy close to the baseline, making it a viable option when maximum compression is needed. Layer pruning, although highly effective in simplifying the network architecture, demands careful application to avoid significant performance degradation. Overall, this study underscores the importance of selecting the appropriate pruning strategy based on the specific demands of the deployment context, contributing to the development of more efficient and practical neural network models..

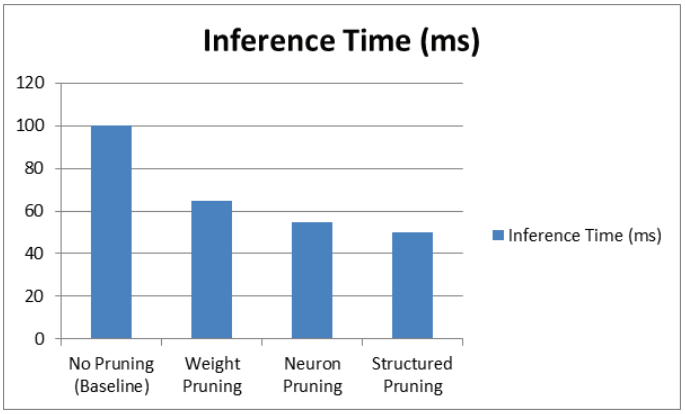

Figure 2. Graph for Interference Time comparison

Table 3. Interference Time Comparison

Pruning Technique	Accuracy (%)
No Pruning (Baseline)	92.5
Weight Pruning	91
Neuron Pruning	89.5
Structured Pruning	90

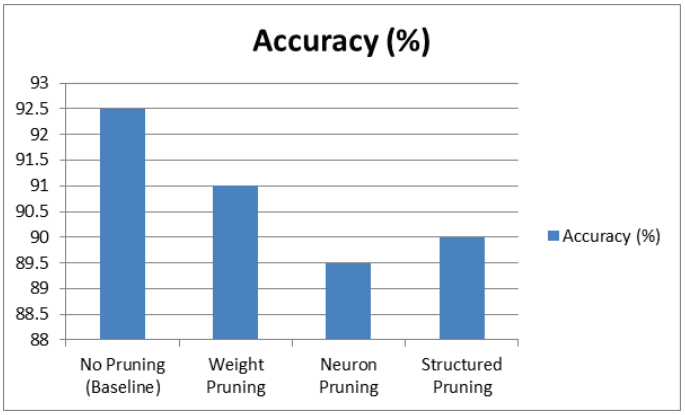

Figure 3. Graph for Accuracy comparison

References

1. X. Zhou, Z. Zhang, L. Wang, and P. Wang, "A Model Based on Siamese Neural Network for Online Transaction Fraud Detection," Proc. Int. Jt. Conf. Neural Networks, vol. 2019-July, 2019, doi: 10.1109/IJCNN.2019.8852295.
2. H. Jang and J. Lee, "An Empirical Study on Modeling and Prediction of Bitcoin Prices with Bayesian Neural Networks Based on Blockchain Information," IEEE Access, vol. 6, pp. 5427–5437, 2017, doi: 10.1109/ACCESS.2017.2779181.
3. F. Casino, T. K. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," Telemat. Informatics, vol. 36, no. November 2018, pp. 55–81, 2019, doi: 10.1016/j.tele.2018.11.006.
4. S. Teng, G. Chen, G. Liu, J. Lv, and F. Cui, "Modal strain energy-based structural damage detection using convolutional neural networks," Appl. Sci., vol. 9, no. 16, 2019, doi: 10.3390/app9163376.

5. S. Liu, A. Borovykh, L. A. Grzelak, and C. W. Oosterlee, "A neural network-based framework for financial model calibration," J. Math. Ind., vol. 9, no. 1, 2019, doi: 10.1186/s13362-019-0066-7.
6. S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh, and W. C. Hong, "Machine Learning Adoption in Blockchain-Based Smart Applications: The Challenges, and a Way Forward," IEEE Access, vol. 8, pp. 474–448, 2020, doi: 10.1109/ACCESS.2019.2961372.
7. W. Gao and C. Su, "Analysis on block chain financial transaction under artificial neural network of deep learning," J. Comput. Appl. Math., vol. 380, p. 112991, 2020, doi: 10.1016/j.cam.2020.112991.
8. W. Zhang, K. X. Tao, J. F. Li, Y. C. Zhu, and J. Li, "Modeling and Prediction of Stock Price with Convolutional Neural Network Based on Blockchain Interactive Information," Wirel. Commun. Mob. Comput., vol. 2020, 2020, doi: 10.1155/2020/6686181.
9. D. Prashar et al., "Blockchain-Based Automated System for Identification and Storage of Networks," Secur. Commun. Networks, vol. 2021, 2021, doi: 10.1155/2021/6694281.
10. Z. Zhang, X. Song, L. Liu, J. Yin, Y. Wang, and D. Lan, "Recent Advances in Blockchain and Artificial Intelligence Integration: Feasibility Analysis, Research Issues, Applications, Challenges, and Future Work," Secur. Commun. Networks, vol. 2021, 2021, doi: 10.1155/2021/9991535.