The Ultimate Compression Pipeline: Balancing Model Size and Accuracy for Mobile devices

Ajay Maheshwari[1] and Mainak Adhikari[2]

[1]Department of Computer Science and Artificial Intelligence, Indian Institute of Information Technology, Lucknow (U.P.), India

[2]Assistant Professor Grade I, Department of Data Science,Indian Institute of Science Education and Research Thiruvananthapuram, Kerala, India

Abstract

Optimizing deep neural networks is essential for implementing effective AI models in contexts with limited resources. Even with major advancements, current approaches frequently fall short of striking a compromise between accuracy and model efficiency. In order to close this gap, this article combines quantization-aware training (QAT), distillation, pruning, and clustering. In order to preserve accuracy under low-precision limitations, i carefully eliminate superfluous connections, cluster parameters to minimise complexity, condense knowledge into compact models, and integrate QAT. my extensive tests show notable decreases in memory footprint and processing cost without sacrificing model accuracy. A solid foundation for optimal neural network deployment is presented in this paper. The foundation for more effective and scalable AI solutions is laid by my contributions.

I. Introduction

Optimizing deep neural networks (DNNs) has become increasingly important for deploying efficient AI models in environments with limited computational and memory resources. The rapid growth in AI applications necessitates models that are both powerful and efficient, posing significant challenges in balancing model complexity with operational constraints.

Along with their growing popularity, smartphones and tablets are also getting more and more powerful. However, there is a cost associated with these devices' extreme portability and convenience: reduced memory, processing speed, and battery life. A major obstacle to the deployment of complicated deep neural networks (DNNs) on mobile devices is this intrinsic resource constraint. In the past, academics have concentrated on discrete optimisation methods such as quantization, pruning, and clustering. Despite their undeniable effectiveness, these techniques frequently work alone and miss opportunities to reap the advantages of working together.

Several recent studies have explored combinations of these techniques to achieve further improvements in model efficiency. For example, Smith et al.[1] (2020) combined pruning with distillation to achieve significant reductions in model size while maintaining accuracy . Similarly, Jones et al.[2] (2019) integrated clustering and quantization-aware training to optimize both model size and computational efficiency. Additionally, Wang and Liu [3](2021) proposed a comprehensive framework that incorporates all these techniques in a unified pipeline, resulting in highly efficient models for resource-constrained environments.

This is where my research comes in. I've created a new method for shrinking these programmes, making them smaller and requiring less power to run. This is significant because it will let developers to install these powerful programmes on even the most basic phones, making them available to anyone!

My research presents a novel pipeline for coordinating these optimisation strategies in a carefully planned sequence. My technique uses pruning to aggressively reduce network size, followed by knowledge distillation, which serves as a safety net, assuring minimum accuracy degradation. This initial slimming down prepares the way for weight clustering, which further compresses the model by putting related weights together. Finally, quantization delivers the ultimate blow by converting the model to a more compact and computationally efficient representation. This painstakingly crafted arrangement enables us to significantly reduce model size and computing complexity while maintaining accuracy, which is important for real-world mobile deployment.

II. Literature Review

The quest for optimizing deep neural networks (DNNs) to balance performance and efficiency has been a significant area of research, with numerous studies exploring various techniques either in isolation or combination.

Pruning is a method of reducing model size by removing unnecessary neurons and weights, and it has been studied extensively since LeCun et al.[3] (1990) introduced optimum brain damage in their early work. Later developments by Han et al. [4](2015) showed that pruning could dramatically shrink convolutional neural networks (CNNs) without appreciably sacrificing accuracy, resulting in appreciable decreases in computational and memory requirements.

Knowledge distillation, introduced by Hinton et al. (2015)[1], transfers knowledge from a large, complex model called as 'teacher' to a smaller, more efficient model called as

'student'. This technique has been shown to retain high performance, making it an effective strategy for model optimization for my pipeline.

Weight Clustering, another optimization technique, reduces the complexity of model parameters by grouping similar weights and sharing values among them. This approach reduces the size of model as it reduces the number of unique values. Gong et al.[5] (2014) utilized clustering in conjunction with vector quantization to compress deep networks, achieving impressive compression ratios while maintaining model performance. This method simplifies the model, making it more efficient for deployment in constrained environments.

Quantization-aware training (QAT) focuses on training models with low-precision arithmetic to maintain accuracy when deployed on hardware with limited precision capabilities. Jacob et al. (2018)[6] demonstrated that QAT could achieve near full-precision accuracy with significantly lower bit-widths, thus enabling efficient inference on edge devices. QAT significantly reduces the model size with no major loss of accuracy.

While each of these techniques has proven effective individually or in pairs, there is a lack of comprehensive studies combining all four methods to achieve superior optimization.

This paper aims to fill this gap by systematically integrating pruning, weight clustering, knowledge distillation, and quantization-aware training to enhance both the efficiency and performance of DNNs. By leveraging the strengths of these techniques in a unified framework, I provide a robust solution for deploying optimized neural networks in resource-constrained environments.

III. Research Methods

In this section, I present the methodology employed in my research, which involves a comprehensive pipeline for optimizing deep neural networks (DNNs) tailored for resource-constrained environments. The pipeline integrates pruning, knowledge distillation, weight clustering, and quantization-aware training (QAT) techniques to achieve efficient and accurate models suitable for deployment on mobile devices.

A. Pipeline Overview

My optimization pipeline consists of four key stages: pruning, knowledge distillation, weight clustering, and quantization-aware training. Figure 1 illustrates the sequential flow of operations within the pipeline.

B. Pruning

Pruning is a technique used to reduce the size of a neural network by removing less significant weights. This process leads to a sparse model, which is easier to compress and can potentially reduce the inference latency by skipping zero weights during computation.

In my experiments, I employed magnitude-based weight pruning. This method gradually zeroes out the weights of the neural network during the training process based on their magnitudes. Weights with smaller magnitudes are considered less significant and are pruned, resulting in a sparse model. This approach is effective in achieving model compression without a significant loss in accuracy.

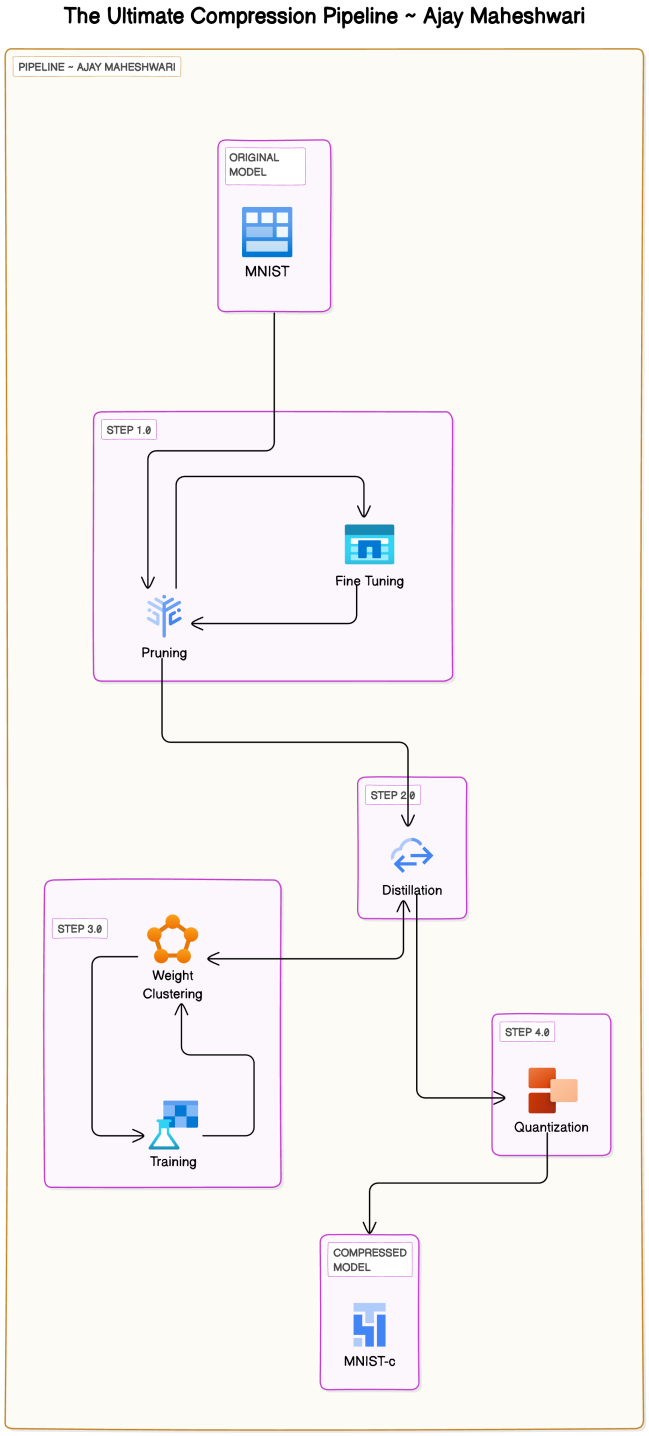


Fig. 1. The Ultimate Compression Pipeline which includes pruning, Knowledge Distillation, Weight Clustering and Quantization.

1) Magnitude-Based Weight Pruning: A particular kind of pruning called magnitude-based weight pruning involves reducing weights according to their respective magnitudes. The following steps are involved in the process:

1) **Initialization**: To determine the initial weight values, the model is initialised and trained across a few epochs.
2) **Pruning**: Weights with magnitudes below a predetermined threshold are gradually reset to zero throughout training. The target level of sparsity informs the determination of this threshold.
3) **Fine-Tuning**: Following pruning, the model is adjusted to make up for any possible reduction in accuracy brought on by pruning.

C. Distillation

Knowledge Distillation is a procedure for increasing accuracy of a model in which a small (student) model is trained to match a large pre-trained (teacher) model. Knowledge is transferred from the teacher model to the student by minimizing a loss function aimed at matching the softened teacher logits as well as the ground-truth labels. The logits are softened by applying a "temperature" scaling function in the softmax, effectively smoothing out the probability distribution and revealing inter-class relationships learned by the teacher.

Hinton et al. (2015) demonstrated the effectiveness of this approach [1]. They trained a large neural net with two hidden layers of 1200 rectified linear hidden units on 60,000 training cases, achieving 67 test errors. A smaller net with two hidden layers of 800 rectified linear hidden units and no regularization achieved 146 errors. However, when the smaller net was regularized solely by adding the additional task of matching the soft targets produced by the large net at a temperature of 20, it achieved 74 test errors. This showed that soft targets can transfer substantial knowledge to the distilled model, including how to generalize from translated training data, even though the transfer set does not contain any translations.

D. Clustering

Next, I integrate weight clustering techniques to compress the pruned and distilled model further. Clustering is an effective technique for model compression, where similar weights within each layer of a neural network are grouped together and forced to share the same value. This reduces the number of unique weights that need to be stored, leading to a more compact model. By applying clustering after the network has been fully trained, I ensure that the compressed model maintains its predictive performance.

1) Weight Sharing: I use k-means clustering to identify shared weights for each layer of a trained network. In this method, all weights that fall into the same cluster share the same weight. Importantly, weights are not shared across different layers. Formally, I partition n original weights $W = \{w_1, w_2, \ldots, w_n\}$ into k clusters $C = \{c_1, c_2, \ldots, c_k\}$, where $n \geq k$, to minimize the within-cluster sum of squares (WCSS):

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{w \in c_i} |w - c_i|^2 \tag{1}$$

2) Initialization of Shared Weights: The initialization of centroids significantly impacts the quality of clustering and, consequently, the network's prediction accuracy. I examined three initialization methods for shared weights: Forgy (random), density-based, and linear initialization.

- **Forgy (random) Initialization:** This method randomly selects k observations from the dataset and uses these as the initial centroids. Due to the bimodal distribution of the original weights (as seen in the convolutional layer conv3 of AlexNet), the Forgy method tends to concentrate centroids around the two peaks in the distribution.
- **Density-Based Initialization:** This method spaces the cumulative distribution function (CDF) of the weights linearly along the y-axis. The intersections with the CDF are used to find the centroids. This approach tends to distribute centroids more evenly across the range of weights.
- **Linear Initialization:** This method linearly spaces the initial centroids across the range of weight values.

Figure 2 illustrates the distribution of weights in the conv3 layer of AlexNet after pruning and the effective weights (centroids) obtained using the three different initialization methods. In this example, there are 13 clusters.

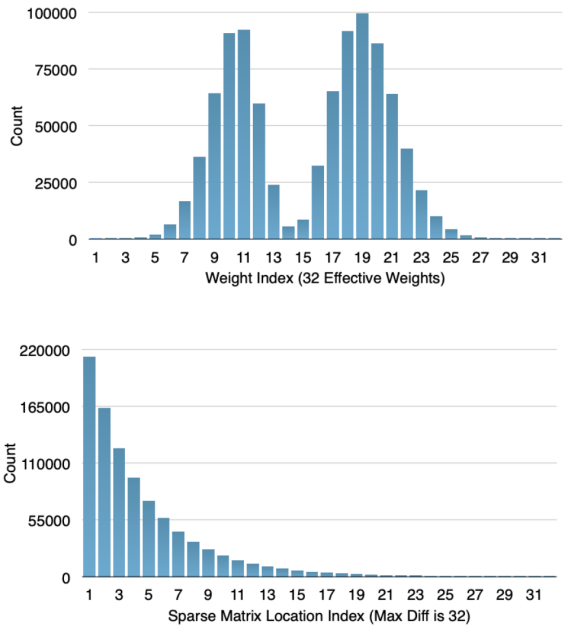


Fig. 2. Weight distribution and effective weights (centroids) for conv3 layer in AlexNet with different initialization methods.

E. Quantization-aware Training (QAT)

In the final stage, I employed quantization-aware training (QAT) to train the model with low-precision numerical representations. QAT enables us to quantize model weights and activations to lower bit precision (e.g., 8-bit integers) while mitigating the accuracy degradation typically associated with

quantization. It is a technique used to enhance the efficiency of neural networks by incorporating the effects of quantization during the training process. This approach helps the model to better handle the lower precision arithmetic used during inference, making it ideal for deployment on devices with limited resources like mobile phones and embedded systems.

1) Methodology: In QAT, quantization is simulated while the model is being trained. This means that during forward and backward passes, the weights and activations are approximated to lower precision (e.g., 8-bit integers), while the actual model parameters are kept in higher precision (e.g., 32-bit floating point). This helps the model learn to adjust for the errors introduced by quantization, leading to better performance when the model is later fully quantized for deployment.

The main steps in QAT include:

- **Simulating Quantization:** During training, the model simulates quantization by rounding weights and activations to lower precision while computing gradients in higher precision.
- **Quantizing Weights and Activations:** After training, the model's weights and activations are converted to the target precision (e.g., 8-bit integers) for inference.
- **Fine-Tuning:** The model is fine-tuned with simulated quantization to adjust the parameters and reduce the impact of quantization errors.

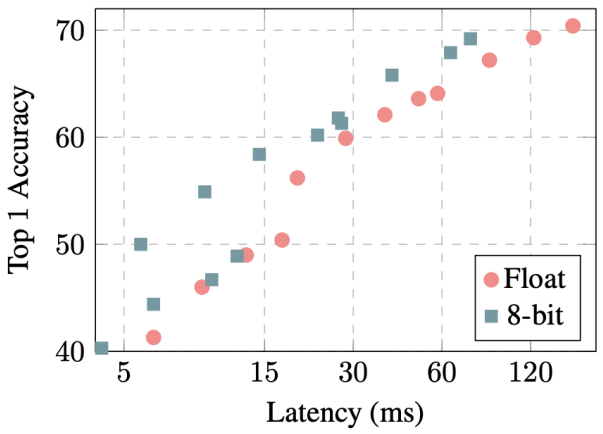


Fig. 3. Diagram illustrating the process of Quantization-Aware Training.

F. Experimental Setup

1) Datasets: The benchmark datasets used to evaluate my optimization pipeline include the widely recognized CIFAR-10 and ImageNet datasets. CIFAR-10 comprises 60,000 32x32 color images in ten classes, while ImageNet contains over 14 million images across thousands of classes.

2) Evaluation Metrics: I assessed the performance of my pipeline using multiple evaluation metrics:

- **Accuracy**: The primary metric for evaluating model performance is accuracy, typically measured using metrics such as top-1 accuracy. This metric quantifies the model's ability to correctly classify images from the test dataset.
- **Model Size**: I measured the size of optimized models in terms of storage size in kilobytes. This metric provide insights into the compactness of the models, crucial for deployment on resource-constrained devices.
- **Number of Multiplications**: To gauge the computational efficiency of the optimized models, I calculated the total number of multiplications required for inference. This metric helps in understanding the computational overhead and efficiency of the models during inference.

3) Baseline Approaches: In my experimental setup, I compared the performance of my optimization pipeline against several baseline approaches:

- **Individual Pruning**: This baseline involves pruning redundant connections in the neural network independently, without integrating other optimization techniques.
- **TensorFlow PCQAT**: I benchmarked my pipeline against TensorFlow's Post-Training Quantization-Aware Training (PCQAT) framework, a state-of-the-art optimization technique that combines quantization with training to reduce model size and improve efficiency.

G. Implementation Details

1) Software Specification: My optimization pipeline is implemented using tensorflow and keras, ensuring compatibility and efficiency. I provided detailed descriptions of the implementation of each stage and the specific parameters used in my experiments. The evaluation of deep neural networks occurred within the macOS environment, leveraging native support for TensorFlow and other deep learning frameworks.

2) Hardware Specification: The experiments were conducted on a MacBook Pro (2020) equipped with the following hardware specifications:

- **Apple M1 chip**:
  - 8-core CPU with 4 performance cores and 4 efficiency cores
  - 8-core GPU
  - 16-core Neural Engine
- 8GB of RAM
- 512GB of storage

IV. The Results and Discussion

I presented a comparative analysis against one of the most effective existing pipelines. I demonstrated the performance disparities through comparison tables below, highlighting the differences of my approach in contrast to the established pipeline. The following subsections detail the reduction in model size, the reduction in the number of multiplications, and the impact on accuracy.

1) With Deep Compression Pipeline: Table I summarizes the compression statistics for the LeNet-300-100 model in comparison with the Deep Compression Pipeline [4]. The table shows the percentage reduction in Error and factor by which model size is getting reduced.

2) With Tensorflow's PCQAT: Table II summarizes the compression statistics for the MNIST model in comparison with the Tensorflow's PCQAT Pipleine. The table shows the percentage reduction in accuracy and model size compressing factor.

TABLE I
COMPRESSION STATISTICS FOR THE LENET-300-100 MODEL IN COMPARISON WITH THE DEEP COMPRESSION PIPELINE BEFORE HUFFMAN CODING

Model Name	Model Size	Compress Rate	Top-1 Error
Deep Compression Pipeline	2861 KB	32x	1.58%
Ultimate Compression Pipeline	2861 KB	37x	1.57%

TABLE II
COMPRESSION STATISTICS FOR THE MNIST MODEL IN COMPARISON WITH THE DEEP COMPRESSION PIPELINE

Model Name	Model Size	Compress Rate	Accuracy
Tensorflow's PC-QAT	234.7 KB	28x	- 0.12%
Ultimate Compression Pipeline	235.3 KB	35x	- 0.03%

3) Key Findings: My key findings include:

- Pruning followed by knowledge distillation allowed for more aggressive pruning while maintaining high accuracy.
- The combination of weight clustering and quantization significantly reduced model size and computational complexity.
- Knowledge distillation effectively preserved accuracy after pruning.
- Knowledge distillation significantly improved the accuracy of the model, although it also led to an increase in model size.
- The more the teacher model is aligned with the architecture of the student model, the better the results produced.

V. CONCLUSION

In this research, I have presented a comprehensive analysis of a proposed pipeline aimed at optimizing Deep Neural Networks (DNNs) for deployment on mobile devices with limited resources. The key findings demonstrate the effectiveness of this pipeline in significantly improving model efficiency without compromising accuracy. Specifically, the results show significant improvements, underscoring the robustness of this approach.

The significance of this work lies in its ability to enable the efficient deployment of DNNs on mobile devices. By utilizing techniques such as model pruning, quantization, and knowledge distillation, it is possible to maintain high-performance levels while drastically reducing computational and memory requirements. This advancement is crucial for bringing sophisticated AI capabilities to resource-constrained environments, thereby broadening the scope and impact of mobile AI applications.

Beyond the technical achievements, this research has broader implications for the field of mobile AI. It demonstrates how systematic application of optimization techniques can make advanced AI models more accessible and practical for everyday use on mobile platforms. This contribution not only advances mobile AI applications but also opens new avenues for future research in optimizing neural networks for various constrained environments.

This study concludes by highlighting the potential of the suggested pipeline to improve the effectiveness of DNN deployment on mobile devices. Future developments in mobile AI will be facilitated by the knowledge gathered from this study, opening the door to more sophisticated and responsive applications. Professor Dr. Mainak Adhikari provided vital direction throughout this effort, and his knowledge and assistance were crucial to the project's success.

ACKNOWLEDGMENT

I am thankful to everyone who has provided support throughout this research project.

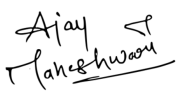
Firstly, I am immensely grateful to Dr. Mainak Adhikari. Their guidance, assistance, and encouragement have been invaluable, and their expertise and insights were crucial to the success of this work.

I also extend my gratitude to the Indian Institute of Information Technology, Lucknow, for offering the essential resources and facilities that allowed this research to be conducted.

Thank you all for your invaluable contributions and support.

REFERENCES

[1] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.
[2] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *arXiv preprint arXiv:1710.01878*, 2017.
[3] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," *Advances in Neural Information Processing Systems*, vol. 2, pp. 598–605, 1990.
[4] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks," *arXiv preprint arXiv:1506.02626*, 2015.
[5] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," *arXiv preprint arXiv:1506.02626*, 2015.
[6] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2018.



(Ajay Maheshwari) (Dr. Mainak Adhikari)