Optimizing Deep Learning Model Efficiency Through Comparative Analysis of Compression Techniques: Knowledge Distillation, Quantization, and Pruning

*

Mikael Kieu[1] and Divyam Sood[2]
Msc. Applied Data Science
University of Gothenburg
[1]Email: guskiemi@student.gu.se
[2]Email: divyamsood.se@gmail.com

***Abstract*—As deep learning models have evolved to tackle increasingly complex tasks, their sizes have grown substantially. However, ensuring the accessibility of such large models beyond industry-grade hardware is essential. This study addresses this challenge by examining techniques to reduce model sizes while striking a balance between accuracy, speed, and efficiency. Specifically, we present a comprehensive evaluation of three prominent model compression techniques: Knowledge Distillation, Weight Pruning, and Quantization. The report begins with an overview of each technique, explaining their theoretical foundations and practical implications. The study then conducts an extensive empirical analysis, comparing the performance, computational efficiency, and model size reduction achieved by each method across a baseline CNN architecture (Resnet50) and a bird classification dataset containing 525 classes.**

As a result of our findings, we conclude that "Knowledge Distillation" is extremely effective in transferring knowledge from a complex teacher model to a simpler student model with minimal loss in accuracy, making it particularly well-suited to resource-constrained settings. Alternatively, "Weight Pruning", a technique that selectively removes less influential weights, shows significant model size reduction while maintaining high accuracy. Lastly, quantization, which reduces the precision of model parameters, effectively reduces model size without causing significant performance degradation. The results of our analysis emphasize the importance of taking into account the specific requirements and constraints of each scenario in order to better optimize model compression and facilitate the deployment of neural networks that are more efficient and scalable.

I. Introduction

As artificial intelligence and deep learning evolve, it has become increasingly important to develop models that are both efficient and lightweight. Model efficiency is driven by the desire to deploy practical solutions in resource-constrained environments, such as edge devices and cloud-based services.

Research and development efforts have been dedicated to achieving model efficiency, resulting in three noteworthy techniques: quantization, pruning, and knowledge distillation. It is important to note that each of these methods focuses on a different aspect of model optimization, but they all share the general goal of reducing computational demands and memory consumption while maintaining the model's predictive capability.

Quantization involves representing the model's weights and activations using fewer bits, which reduces the amount of storage and computation required during inference. Pruning simplifies the model architecture by removing parameters or neurons that do not have a significant impact on the inference. In knowledge distillation, a larger, complex model is leveraged to train a smaller model, transferring the rich information acquired by the larger model to the smaller model.

To understand the effects of these techniques, this study focuses on implementing these compression techniques on a ResNet-50 architecture. Specifically, the compresses models will be used to classify bird species obtained from [4], with the goal of achieving an equivalent level of performance while substantially reducing the size and parameter count of the model. The objective of this study is to demonstrate how these techniques can be used to deploy highly efficient models without compromising their predictive accuracy. Consequently, it contributes to the broader goal of making AI more accessible and practical for real-world applications, even under resource constraints.

Following is the outline: in Section II, we describe the relevant literature and theory behind the main techniques used in the project. In Section III, the methods used in this study are described, beginning with a discussion of preprocessing and then an explanation of the main methods used to compress the model obtained. A discussion of the results is provided in Section IV, in which the models performance is analyzed in relation to its memory size and parameter count. Additionally, the model speed will be assessed by examining both the training and inference speeds. Lastly, Section V concludes the project and discusses potential future work.

II. Background

A. Quantization

Quantization, the process of representing network weights and activations with a reduced number of bits, strikes a delicate balance between model accuracy and resource utilization [1]. By quantizing the model's parameters, one can drastically decrease memory consumption and accelerate inference, albeit at the cost of introducing quantization-induced errors. While floating-point numbers can represent a wide range of values with high precision, quantized integers can only represent a limited range of values with low precision. This error occurs as a result of approximating real numbers with a limited number of discrete values. A variety of techniques are available to minimize the impact of this error.

The quantization process can take many forms, including weight quantization (reducing the precision of model parameters), activation quantization (reducing the precision of intermediate activations), and hybrid quantization (combining different quantization methods for weights and activations) in order to optimize memory and computation.

Quantization typically has two forms of training:

1) Quantization-Aware Training (QAT) is a technique where the model is trained with quantization effects incorporated into the training process. This allows the model to learn to be robust to quantization and can mitigate the accuracy loss that typically accompanies quantization.
2) Post-Training Quantization (PTQ) applies quantization after the model has been trained with full precision. This method is simpler but may result in more substantial accuracy loss compared to QAT.

B. Knowledge distillation

The concept of knowledge distillation refers to a process where the knowledge acquired by a complex "teacher" model is transferred to a simpler model in order to address the inherent trade-off between model performance and model efficiency [2]. As its name implies, knowledge distillation is the process of distilling the expertise and generalization capabilities of a deep and complex model into a more compact and computationally efficient version.

The knowledge distillation process uses soft targets rather than focusing exclusively on the final predictions of the teacher model. Thus, the teacher's predictions are treated as probability distributions over classes rather than hard labels. Due to this softness, the student model can capture not only what the teacher predicts, but also the uncertainty and nuances associated with those predictions.

An important aspect to knowledge distillation is the loss function, which is typically determined by the Kullback-Leibler divergence or another similarity metric. It is used to determine the degree of alignment between the student's predictions and the teacher's soft targets. The student model strives to minimize this loss during training, thereby effectively learning from the teacher.

There is a temperature parameter (often abbreviated as "T") that determines how soft the teacher's predictions are. When T is higher, the teacher's predictions are softer and more informative, while when T is lower, they are more deterministic. A balance can be struck between fidelity to the teacher and generalization by tuning this parameter. In general, knowledge distillation results in improved performance for the student model as compared to training it from scratch. This is because the student learns not only from the labeled data but also from the rich insights and generalization abilities of the teacher.

C. Pruning in DNN

In neural networks, pruning is a crucial technique that is used to systematically eliminate network parameters which contribute minimally to the network's output. By reducing redundancy within the architecture, it is primarily intended to enhance the efficiency of the network. The application of pruning in a judicious manner results in a neural model that retains its predictive accuracy while drastically reducing computational demands during inference [3].

1) Lottery ticket hypothesis: The lottery ticket hypothesis [4] introduces an interesting concept for pruning a model. It states that within any randomly initialized feed-forward neural network, there exists a specific sub-network, often referred as the "winning ticket." This winning ticket possesses distinct properties that set it apart from the larger network. After training, the winning ticket will generalize better or in other words it will have higher test accuracy. Moreover, training of the "winning ticket" will require fewer iterations than training the original network. Lastly, the winning ticket typically has a significantly reduced number of parameters, usually less than 10% to 20% of the original network's parameters. Parameters are remarkable reduced in this hypothesis, which is one of its key features.

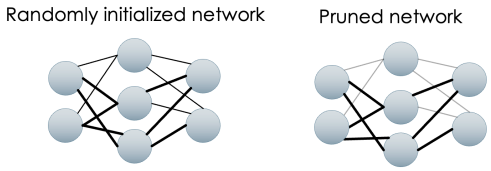

Fig. 1. Illustration of pruning a randomly initialized network based on the lowest absolute values. The width of the connecting lines represent absolute value size.

2) Finding winning ticket: Although the lottery ticket hypothesis asserts that winning tickets exist within neural networks, it does not provide an explicit recipe for how they are discovered. A systematic method based on pruning has been proposed by authors in [4] and it follows:

1) The process begins with the original neural network, initialized with random weights.
2) This network is trained for a fixed number of training iterations, allowing the weights to evolve and adapt to the given dataset.

3) Pruning is then employed to systematically eliminate weights, usually based on criteria like the lowest absolute values (L1 norm), and this is typically done on a layer-by-layer basis.
4) Throughout this process, a record of the pruning mask is maintained. The pruning mask indicates which weights were pruned, while the pruned weights themselves are discarded.
5) After pruning, researchers revert to the original network's weights and apply the pruning mask to retain only the relevant connections.
6) Steps 1 to 5 are repeated for multiple iterations, a process known as iterative pruning. These iterations ultimately lead to the discovery of a network with a significantly reduced number of non-pruned weights.

III. Methods

A. Dataset Selection and Characteristics

The model compression study presented here uses a dataset obtained from [4] that contains 525 bird species. It was intentionally chosen due to its inherent complexity, featuring a large number of classes, making it an ideal benchmark.

In this dataset, 84,635 training images, 2,625 test images (5 per species), and an equivalent number of validation images are included. Each image is uniformly formatted in JPG, maintaining a consistent dimension of 224 x 224 pixels, with RGB color channels. Moreover, each image focuses on a single bird species, and is thoughtfully framed to ensure that the bird occupies at least 50% of the image area.

B. Baseline Model - ResNet-50

In the initial phases of our experimentation, we explored various model architectures, including MobileNetV2, ResNet-50, SqueezeNet, and ResNet-152, with the goal of establishing a suitable baseline model. We systematically replaced the original classification head with a custom one capable of producing 525 outputs, corresponding to the number of unique bird species in the dataset. Three distinct training strategies were employed:

- Freezing all layers except the final one, utilizing pre-trained weights from ImageNet.
- Unfreezing all layers and training with pre-trained weights from ImageNet.
- Unfreezing all layers and training with randomly initialized weights.

Ultimately, our findings indicated that the second approach, involving the unfreezing of all layers and training with pre-trained ImageNet weights, yielded the highest accuracy, reaching an impressive 98%. The choice of the ResNet-50 architecture was based on its well-documented success in image classification tasks, thanks to its deep architecture and outstanding performance. Additionally, standard data augmentation techniques were applied during training, and the model was optimized using cross-entropy loss function.

C. Quantization

The baseline model was a ResNet-50 model previously trained on the bird classification dataset. To prepare the model for quantization, it was transitioned into evaluation mode and configured with a suitable quantization setting optimized for the 'x86' hardware platform.

The calibration phase followed, wherein a subset of training data was designated as a calibration dataset. Approximately 10 batches of this dataset were utilized to collect activation statistics for quantization parameter determination. Activation fusion was then applied to the model, specifically targeting convolutional layers with ReLU activation functions to consolidate operations and enhance computational efficiency. Subsequently, the prepared model underwent conversion to a quantized form using PyTorch's quantization.convert function. This transformation involved converting both model weights and activations into lower-precision formats, typically 8-bit integers, effectively reducing memory requirements. Following quantization, the performance of the quantized model was evaluated on the test dataset, assessing inference speed and classification accuracy, the results can be found in table III-C.

TABLE I
Comparison of baseline and quantized ResNet-50 architecture

	Inference time	**Accuracy**
Baseline	0.1672 s	98.63%
Quantized	0.069 s	98.43%

D. Knowledge distillation

This approach aimed to distill knowledge from a powerful ResNet-50 teacher model, originally trained on a comprehensive bird classification dataset, into a more compact SqueezeNet student model.

A pretrained ResNet-50 teacher model was loaded and acted as the source of "knowledge". In parallel, a SqueezeNet student model was built, specifically tailored for the classification of birds. Designed to be more lightweight with fewer parameters.

In the training phase, the student model was initially untrained and yielded an accuracy score of 70% on the validation set after 30 epoch, the results are shown in figure 2. During training, the student model's objective was twofold: mimic the teacher's predictions while maintaining competitive classification performance. The loss function employed was a combination of the Kullback-Leibler Divergence (KLDivLoss) between the softmax logits of the teacher and student models, scaled by temperature, and the standard cross-entropy loss. By combining these factors, the student model inherited the teacher's expertise and retained its own discriminatory abilities. The training process was carried out for 30 epochs, allowing the student model to progressively distill the knowledge from the teacher, as depicted in figure 3 .

As a result of knowledge distillation, a more lightweight and computationally efficient SqueezeNet student model was developed. Despite its smaller size, the model was still able

to absorb the valuable insights from the ResNet-50 teacher model.

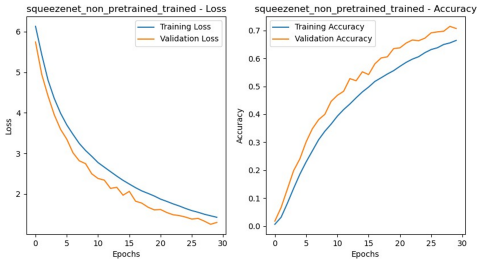

Fig. 2. Comparison of accuracy and loss for training and validation data on Squeezenet network

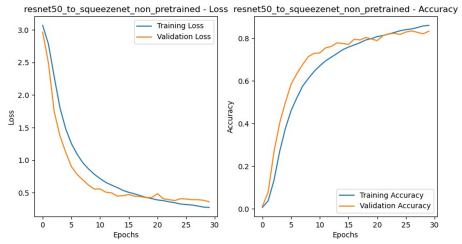

Fig. 3. Comparison of accuracy and loss for training and validation data on a distilled Squeezenet network

As shown in table II. the baseline model has an accuracy of 97% and SqueezeNet without distillation reached an accuracy score of 60% at 30 epochs. The accuracy of ResNet rose to 84% when ResNet50 was used as a teacher model, thus showing a significant improvement in accuracy along with a substantial reduction in size.

TABLE II
COMPARISON OF MODEL SIZE AND PARAMETER

	Parameters	Size
ResNet-50 (Teacher model)	24 583 757	93.98 Mb
SqueezeNet (Student model)	1 004 749	3.83 Mb
StudentNet (Student model)	281 613	1.07 Mb

E. Pruning

A GitHub repository on pruning [5] was influential in our work to prune the ResNet-50 model. The script was initially modified to adjust the first convolutional layer to make the model prunable. Additionally, the it was used for computing the initial model's accuracy while measuring global sparsity, which quantifies the percentage of zero-valued parameters. The implementation provides both layer-wise and global pruning options as part of the pruning process. Using layer-wise pruning, it systematically prunes a specified percentage of weights in each module (Conv2d or Linear) based on their L1 norms, effectively eliminating less important connections. Alternatively, global pruning prunes a specified percentage of weights across all layers at the same time. Following each pruning iteration, fine-tuning was performed in order to recover any accuracy loss due to pruning by adjusting the remaining weights while applying L1 and L2 regularization for sparsity and weight decay. Due to the excessive running time, this iterative process was limited to six iterations. As presented in table III, the finalized pruned model resulted in a global sparsity of 57.58% while achieving a higher accuracy score than the baseline model.

TABLE III
COMPARISON OF BASELINE AND PRUNED RESNET-50 ARCHITECTURE

Iteration	Global sparsity	Validation accuracy
0	0.00%	97.83%
1	19.56%	98.67%
2	35.19%	98.43%
3	47.72%	98.51%
4	57.78%	99.04%

IV. RESULTS

In conclusion, the series of techniques applied to the neural networks in this project have resulted in significant improvements in terms of efficiency and accuracy.

The implementation of knowledge distillation from a complex model (ResNet-50) to a smaller model (SqueezeNet) resulted in remarkable improvements in accuracy. The accuracy of SqueezeNet increased from 70% to 84% after distillation, demonstrating the effectiveness of knowledge transfer. Quantization was another powerful approach employed in this project. The original model size was substantially reduced while maintaining a high level of accuracy. In addition, pruning was essential to optimizing neural network architecture. In the pruned model, redundant layers were removed and the fine-tuned model was fine-tuned iteratively to produce greater accuracy than the baseline ResNet-50. Pruning not only results in reduced model complexity, but can also lead to improved performance, thereby enhancing the effectiveness and efficiency of the network. The code for this project has been published here: Github/ DmlProjectGroup36 along with a readme to replicate results and findings.

In summary, the combination of distillation, quantization, and pruning techniques has proven to be both resource-efficient and highly accurate, making them well-suited for various real-world applications where computational resources and model size are critical factors.

REFERENCES

[1] Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., & Bengio, Y. (2017). Quantized neural networks: Training neural networks with low precision weights and activations. Journal of Machine Learning Research, 18(1), 6869-6898.

[2] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[3] Han, S., Pool, J., Tran, J., Dally, W., & Abbeel, P. (2015). Learning both weights and connections for efficient neural network. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS) (pp. 1135-1143).

[4] Birds 525 species. https://www.kaggle.com/datasets/gpiosenka/100-bird-species. Accessed: 2023-10-05.

[5] A. Majumdar, Neural Network Pruning with Global Absolute Magnitude Pruning, GitHub, 2023. https://github.com/arjunmajumdar/Neural_Network_Pruning Accessed: 2023-10-05.