International Journal of Information Technology (IJIT)
Volume 4, Issue 01, January-June 2023, pp. 65-71 Article ID: IJIT_4_01_008
Available online at https://iaeme.com/Home/issue/IJIT?Volume=4&Issue=1
Journal ID: 4573-3410, DOI: https://doi.org/10.17605/OSF.IO/TJ8MS

MAKING GENERATIVE AI MODELS SMALLER: MODEL COMPRESSION TECHNIQUES AND ITS CHALLENGES

Surabhi Sinha
Department of Computer Science, University of Southern California,
Los Angeles, USA

Mayukh Maitra
Department of Computer Science, Stony Brook University, New York, USA

ABSTRACT

From computer vision and NLP to speech recognition and autonomous driving, deep learning models have been remarkably efficient in many different areas. The size of such models, however, grows proportionally with the complexity and information they encompass. Several strong reasons have highlighted the importance of shrinking deep learning models. Deep learning model size reduction is critical for overcoming storage and computing constraints, enhancing efficiency and inference speed, lowering environmental impact, and resolving privacy and security concerns. As deep learning advances and finds applications in new fields, efforts to reduce model size will be critical to establishing scalable, accessible, and sustainable deep learning systems. In this work, we explore some of the model compression techniques such as quantization, model pruning, and low-rank factorization, and also highlight their limitations. In addition to building more efficient models, it is also paramount to make them sustainable and model compression plays a critical role in that.

Keywords: Model Size Reduction, Quantization, Pruning, Low-Rank Factorization, Hybrid-Precision.

Cite this Article: Surabhi Sinha and Mayukh Maitra, Making Generative AI Models Smaller: Model Compression Techniques and Its Challenges, *International Journal of Information Technology (IJIT)*, 4(1), 2023, pp. 65-71.
https://iaeme.com/Home/issue/IJIT?Volume=4&Issue=1

I. INTRODUCTION

Deep learning models have seen tremendous success in a variety of applications, including computer vision, natural language processing, speech recognition, and autonomous driving, to name just a few. The scale of these models, however, continues to grow in tandem with the complexity and depth to which they are developed. Breakthrough models such as AlexNet [1], GoogleNet [2], and VGG16 [3], etc. all increased in size from their previous competitors due to an increased number of parameters and complexity of the architectures.

The necessity to reduce the size of deep learning models has become increasingly evident for a variety of compelling reasons. Large model sizes present considerable storage, memory, and computing issues. Massive model training and deployment necessitate large computational resources, limiting their availability to academics and practitioners with limited access to high-end technology. Furthermore, their enormous model sizes make them difficult to deploy on resource-constrained devices like mobile phones, embedded systems, and edge devices. Reducing model size can ease these limits, allowing deep learning technology to be widely adopted and democratized.

Another important factor to consider is the environmental impact of huge deep learning models. Training these models uses a lot of computer resources, which increases energy consumption and carbon emissions. We may reduce the carbon footprint associated with training and deployment by reducing the model size and supporting more sustainable and environmentally friendly deep learning approaches. During the training of a single model, about 300 tonnes of CO2 equivalent emissions are produced as discussed in [4]. If we compare this to emissions produced by an average American over a year [5], training an average deep learning model releases over 17 times more CO2.

Furthermore, reducing the size of the model can assist address privacy issues while also mitigating any security problems. Large models may contain sensitive training data information, causing privacy problems, especially in cases involving personal or confidential data. Smaller models that eliminate extraneous characteristics or use compression techniques can reduce the chance of data leakage, improving data privacy and security.

In this work, we discuss several model size reduction techniques and observe their impact on deep learning model sizes. As deep learning continues to advance and find applications in diverse domains, efforts to reduce model size will be critical to establishing scalable, accessible, and sustainable deep learning systems.

II. MODEL COMPRESSION TECHNIQUES

Over the years with the increasing complexity of deep learning models, extensive research has been performed to develop model compression techniques in order to address the critical issues discussed in the previous section.

MODEL PRUNING

Pruning is the process of discovering and deleting redundant connections or parameters from a deep learning model. This can be accomplished by zeroing out minor weights or eliminating entire neurons or layers that contribute little to overall model performance. Pruning assists in reducing the number of parameters, resulting in a smaller model size. Within pruning there are multiple techniques as discussed below which enable modelers to perform model compression.

1. Weight Pruning: Weight pruning is the process of detecting and deleting minor or inconsequential weights from a model. The model gets sparser and more compact by setting these weights to zero or eliminating the related links. Weight trimming is frequently based on a threshold criterion, with weights less than a specific threshold pruned.

2. Magnitude-based pruning: Weight pruning commonly uses a magnitude-based technique. It entails sorting the model's weights by their magnitudes (absolute values) and deleting the weights with the smallest magnitudes. This procedure can be repeated iteratively, with each iteration trimming a given percentage of the smallest magnitude weights.
3. Structural Pruning: Instead of deleting individual weights, structural pruning removes entire neurons, filters, or layers from the model. It finds and prunes the model architecture's less important components. Various factors can be used to guide structural pruning, such as the relevance of neurons or filters based on their activations or contributions to the loss function.
4. Iterative Pruning: Iterative pruning is a technique that involves several rounds of pruning and retraining. The model is initially pruned to a desired sparsity level and then retrained to recover or improve its performance [6]. This alternating pruning and retraining procedure can be performed several times, gradually increasing sparsity while maintaining or enhancing performance.
5. Group-Wise Pruning: Pruning entire groups of weights or connections at once is referred to as group-wise pruning. This technique is frequently used in convolutional neural networks (CNNs) to prune weights within a filter or kernel simultaneously. Criteria such as the sum of weights within a group or their relevance based on activations can be used to drive group-wise pruning.

Despite its benefits, model pruning presents numerous issues that must be addressed. Deep learning model pruning removes weights, connections, neurons, or layers. Parameter removal may reduce model performance and information. Reduction in model performance requires fine-tuning or retraining to improve accuracy. Finding the right pruning and retraining timetable can be difficult and time-consuming. In addition to that all methods defined above have compression and performance trade-offs, so choosing the right one for a model or task can be difficult.

MODEL QUANTIZATION

Model quantization is a technique for reducing deep learning models' memory footprint and processing requirements by representing weights and activations with lower precision values. It substitutes lesser bit-width representations, such as 8-bit integers or even binary values, for higher precision data types, such as 32-bit floating-point numbers. Some of the model quantization techniques currently available are:

1. Fixed point quantization: The most prevalent type of model quantization is fixed-point quantization. It depicts weights and activations with fixed-point numbers that have a smaller bit width than floating-point numbers. Weights and activations, for example, can be quantized to 8-bit integers rather than 32-bit floating-point values. Fixed-point quantization necessitates the selection of appropriate scaling factors to return the quantized values to their original range.
2. Dynamic Quantization: In dynamic quantization, the model is quantized during the inference phase rather than the training phase. Because the quantization is done depending on the dynamic range of the inputs during inference, this approach allows for greater flexibility. Dynamic quantization can be used on a per-batch or per-sample basis to tailor the quantization levels to the specific inputs encountered during inference.
3. Post-training quantization: It is the practice of quantifying a learned model after it has been taught. It entails quantifying the model's weights and activations without retraining it. For our experiments, we performed post-training static and dynamic quantization in order to compare the impact of model quantization on model size. We utilized Pytorch’s dynamic quantization module [7] on various pre-trained models to compare size reductions from the original model.

This method only converts the weights to an int8 precision. For our experiment, we only quantized the Linear layer to int8 precision. In Figure 1 we can see that dynamic quantization helps reduce model sizes and the extent of model compression depends on the number of Linear layers present in the architecture. On average, for the 12 pre-trained models shown below, we observed a 56% reduction in model size.

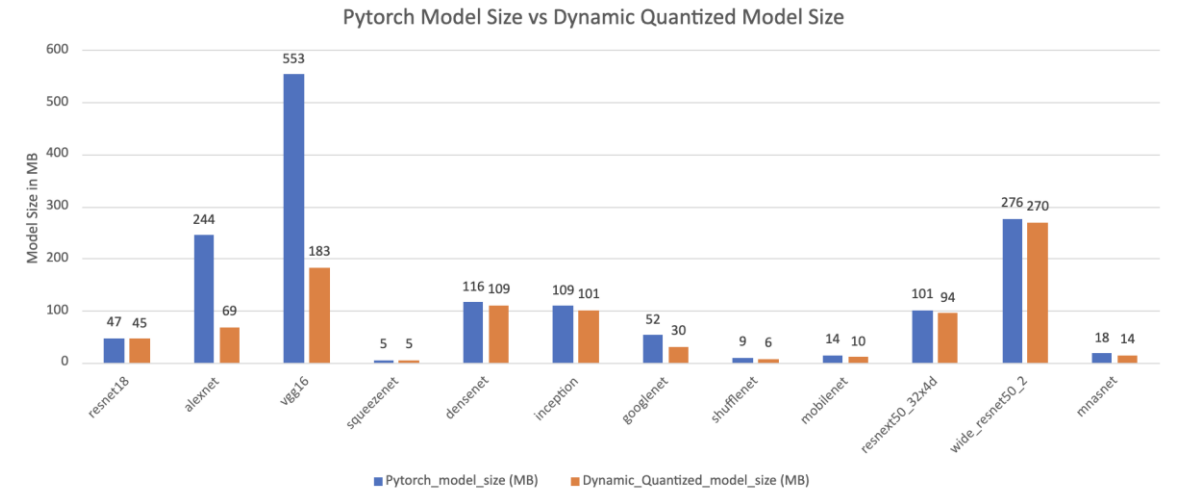

Figure 1. Model size comparison between Pytorch pre-trained models and their dynamically quantized versions

In addition to dynamic quantization, we also experimented with post-training static quantization [8] by Pytorch. Compared to its dynamic counterpart it converts both weights and activations to an int8 precision and has a higher impact on model compression as compared to the dynamic method. In Figure 2 we can see a greater reduction in sizes as compared to that in Figure 1. On average, we observed approximately a 74% reduction in model sizes compared to the original Pytorch model for the 12 pre-trained models selected for the experiment.

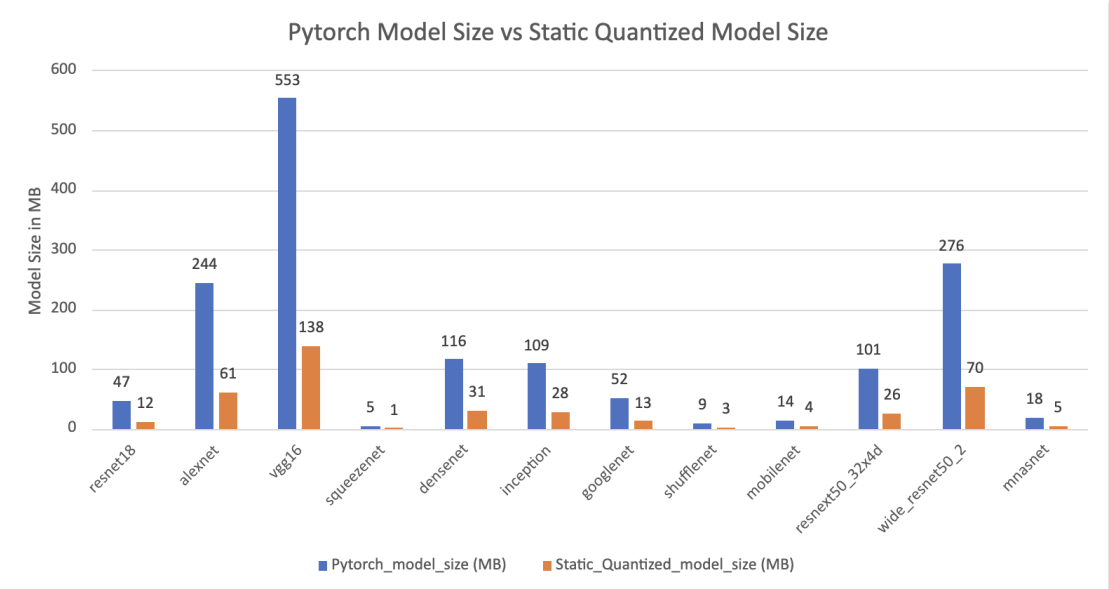

Figure 2. Model size comparison between Pytorch pre-trained models and their statically quantized versions

4. Hardware-Aware Quantization: Quantization that is hardware-aware considers the exact hardware or target platform on which the quantized model will be delivered. varying hardware platforms have varying requirements and quantization format support. These parameters are taken into account by hardware-aware quantization algorithms during the quantization process to optimize the model for the target hardware, ensuring efficient execution and maximizing performance.
5. Hybrid Precision Quantization: In hybrid precision quantization, distinct portions or layers of the model are quantized with varying precision levels. Some layers of the model may require higher precision to maintain performance, and not all sections of the model may be equally sensitive to quantization. Hybrid precision quantization enables fine-grained control over precision levels in different regions of the model, maximizing the accuracy-to-model-size trade-off.

Similar to model pruning, model quantization also has its limitations. The precision of weights and activations is reduced during quantization, which might result in a loss in model accuracy. Rounding errors and truncation are introduced during the quantization process, which may impair the model's capacity to capture fine-grained features. A significant difficulty in quantization is balancing the trade-off between model size reduction and retaining acceptable accuracy. In addition to that it is critical to select the best bit width for quantization. Using too few bits may result in considerable accuracy loss, whereas using too many bits may result in insufficient compression. Determining the best bit-width for different layers or portions of the model necessitates extensive testing and analysis which is often time and resource intensive.

LOW-RANK FACTORIZATION

Low-rank factorization is a model compression technique that uses low-rank matrices to approximate weight matrices in order to minimize the size and complexity of deep learning models. The basic concept is to divide a weight matrix into two or more smaller matrices that contain the most significant information. Performing low-rank factorization of even a single layer may lead up to a 30%-50% reduction in network parameters [9].

1. Tensor factorization: It takes low-rank factorization and applies it to higher-order weight tensors. Weight tensors in deep learning models, such as convolutional layers, can have three or more dimensions. It is feasible that a limited number of random projections effectively retains the spectral information in the tensor, allowing one to avoid the eigengap dependence that afflicted previous tensor-to-matrix reductions [10]. Tensor factorization approaches deconstruct a weight tensor into a set of lower-dimensional factor matrices or tensors, such as Tucker decomposition or CP decomposition. These factor matrices or tensors capture the weight tensor's important properties and linkages while lowering its size.
2. Matrix sketching: These approaches compute a low-rank sketch or projection of the original matrix to approximate a weight matrix. The sketch is a smaller matrix that keeps all of the original matrix's important properties. To minimize the size of weight matrices while keeping their critical qualities, various matrix sketching methods, such as random projection or structured sketching, are used. Matrix sketching methods are proven to be efficient in supervised classification of datasets having imbalanced datasets [11].

However, low-rank factorization has certain drawbacks. When employing low-rank factorization, aggressive compression might result in significant accuracy loss, especially if the rank is set too low. It is critical to balance compression with accuracy retention, and the optimal rank or compression ratio must be chosen depending on the unique model and job requirements.

III. CONCLUSION

In this work, we discussed a few of the model compression techniques and how they can help reduce model sizes along with their limitations. These strategies can be used individually or in combination to reduce the size of deep learning models, allowing for more efficient storage, faster inference, and better deployment on resource-constrained devices. The choice of technique used is often determined by the application's or scenario's specific requirements, constraints, and trade-offs. Deep learning's environmental effect can be considerably decreased by employing model compression techniques, making AI applications more sustainable and eco-friendly. Energy savings, more effective resource utilization, and longer device lifespan all contribute to a greener, more environmentally conscious AI ecosystem.

REFERENCES

[1] Krizhevsky A. et al. ImageNet Classification with Deep Convolutional Neural Networks, Advances in Neural Information Processing Systems 25 (NIPS 2012).

[2] Szegedy C. et al. Going Deeper with Convolutions, CVPR 2015.

[3] Simonyan K. et al. Very Deep Convolutional Networks for Large-Scale Image Recognition, 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings. 2015.

[4] Labbe M. et al. AI and climate change: The mixed impact of machine learning, TechTarget, Enterprise AI (link).

[5] Toews R., Deep Learning’s Carbon Emissions Problem, Forbes, Innovation AI (link).

[6] Pytorch Iterative Pruning, Pytorch (link).

[7] Post Training Dynamic Quantization, Pytorch (link)

[8] Post Training Static Quantization, Pytorch (link)

[9] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy and B. Ramabhadran, "Low-rank matrix factorization for Deep Neural Network training with high-dimensional output targets," 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 2013, pp. 6655-6659, Doi: 10.1109/ICASSP.2013.6638949.

[10] Kuleshov, V., Chaganty, A. T., & Liang, P. (2015). Tensor Factorization via Matrix Factorization. ArXiv. /abs/1501.07320

[11] Falcone, R., Anderlucci, L. & Montanari, A. Matrix sketching for supervised classification with imbalanced classes. Data Min Knowl Disc 36, 174–208 (2022). https://doi.org/10.1007/s10618-021-00791-3

Citation: Surabhi Sinha and Mayukh Maitra, Making Generative AI Models Smaller: Model Compression Techniques and Its Challenges, *International Journal of Information Technology (IJIT)*, 4(1), 2023, pp. 65-71

https://doi.org/10.17605/OSF.IO/TJ8MS

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJIT/VOLUME_4_ISSUE_1/IJIT_4_01_008.pdf

editor@iaeme.com