International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 03 | March - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Developing New AI Model Compression Techniques

Name: Balaji Soundararajan (Independent Researcher)
Email: esribalaji@gmail.com

Abstract
The rapid growth of artificial intelligence (AI) model complexity has created significant challenges for deployment on resource-constrained devices and customization by developers. Model compression techniques, such as pruning, quantization, and knowledge distillation, have emerged as critical solutions to reduce computational and memory demands while preserving accuracy. This work explores foundational and state-of-the-art approaches to AI model compression, emphasizing their role in enabling efficient edge computing, lowering energy consumption, and democratizing access to advanced AI capabilities. We discuss the trade-offs between model size, inference speed, and accuracy, and evaluate methods for deploying compressed models on mobile and IoT devices. Case studies on architectures like MobileNet and SqueezeNet demonstrate practical successes, while benchmark datasets and evaluation metrics highlight the need for standardized methodologies to assess compression efficacy. The paper concludes with guidelines for reliable experimentation and future directions in optimizing model efficiency without compromising performance.

Keywords
AI Model Compression, Pruning and Sparsity, Quantization, Knowledge Distillation, MobileNet, Edge Computing, Resource-Constrained Devices, Inference Optimization.

Introduction to AI Model Compression

The increasing complexity of artificial intelligence models makes it more difficult for regular developers to do any kind of customization on top of these models. This trend is worrying, as AI finds applications in several relevant domains and allows us to automate several tasks. The ability to compress these huge models in a way that neither distorts the overall model too much nor slows down the entire model opens a window for regular developers to create customizations. It also enables lower-end devices to use AI-powered models in a context where energy efficiency can be crucial. This text presents efforts in developing new techniques for model compression, from high school graduation projects to the most recent research work.

Background and Significance

Over the past decade, deep learning has produced state-of-the-art results in a wide variety of problems, including tasks like image classification, linguistic processing, and playing board games with superhuman proficiency. These models are, however, often very large, with implementation details consisting of millions of parameters. With some models reaching billions of parameters in size, it has become increasingly important to find ways to effectively reduce this memory footprint in order to make models more efficient and accessible.

A significant source of inaccessibility is that large models can exceed the capabilities of mobile devices, and we want to have low-latency AI through edge computing or run without an internet connection. A model that is too large for compute can have significant monetary costs, thus reducing effectiveness for new startups. Additionally, learning on massive models has led us to train AI models that can have negative societal effects, further exacerbating the model size concern. The problem of model size growth is even more pressing when we consider

that AI research competes for time and resources on the accelerating growth of data center resources for model training.

Fundamentals of AI Model Compression

This section provides various concepts and theories that are essential to understand AI model compression. These are all foundational concepts that underpin the development of advanced AI model compression techniques. Reading this full section before reading the rest of the essay will provide a strong holistic view of the landscape, and as such, will be invaluable for understanding the later discussions. Model compression, particularly based on sparsity, are fundamental concepts that are widely used in the development of various advanced model compression techniques. Artificial Neural Networks (ANNs) have gained a lot of attention in recent years as potential means of artificial intelligence. They produce superior results in various problems to humans and have the ability to tune some parameters in the network. However, this large number of parameters leads to redundancy. Eliminating this redundancy is important to improve various properties of the network, particularly its hardware implementation. As massive parameter redundancy exists in neural networks, various methods of compression, to enhance their efficiency, have been explored. Broadly, these can be classified as either compressing primarily the highest quantities from the network or by encoding a set of parameters in a way that receives a smaller amount of storage. Naturally, others' strategy will have a high pricing in both methods, though some methods are more resource minimal than others, like compressing only a portion of the parameters in the network, known as sparsity. However, compression of the network has several key advantages: 1. Computational efficiency is improved. In modern parallel computing devices, as well as in the human brain, using large quantities of computational resources, the heavy perceptual system is considered extremely necessary in biological beings for a robustly working clumping system. 2. Reduction in memory use. After a certain point, further compression of a network is much more useful in terms of hardware storage so that the information can be useful in other network learning or tasks. A whirlwind tour of the sparsity theory introduced by different pressures to regularize the training process is provided.

Pruning and Sparsity

Recently, there has been increasing interest in developing various model compression techniques that reduce the cost of using artificial intelligence (AI) models in deployed systems. Among them, pruning has been a pivotal approach to reduce the computational and memory requirements of models by eliminating non-essential weights, which are usually located in the later phase of training.

Pruning is a prevalent approach to compressing AI models. Pruning cuts out random weights which have a reasonable likelihood of being redundant. Then these weights are either cut down to zero, or 1 is added to their skill of inverse penalization. Sparsity is defined as the proportion of zero weights to the entire weights. It is achieved by pruning original neural networks. A neural network is trained from scratch with the entire training dataset and validation dataset to reduce the mean square error (MSE) loss using backpropagation.

Sparsity is achieved through pruning the original neural network to zero out the weights gradually. The effect of sparsity results in speedups of runtime performance and better memory efficiency. Implementing a pruned model on CPUs or GPUs noticeably increases inference speed compared with the dense model, and the pruned model shows better memory efficiency. For hardware acceleration, the original model needs to be sparsified. CPU or GPU acceleration is more likely to be powered in practice because the cost and convenience of those are significantly better than other hardware.

Techniques for Model Size Reduction

In this section, we briefly introduce the selected prior techniques for model size reduction, ordered by the type of pruning setting. Several of the proposed techniques for pruning, quantization, knowledge distillation, or weight clustering can be found in other tutorial work. We selected the techniques either because they set new state-of-the-art results, represent the majority of the proposed techniques for their specific group, or introduce a fundamentally new approach in the field. In an introductory manner, we skipped the technical details, but we encourage readers to find the complete procedures.

Pruning. In modern life, it is hard to find a person or a work without understanding that less is more. A similar philosophy holds in deep learning model compression. Pruning is the process of setting weight magnitudes to zero. To be more precise, a pruning rate is a percentage of the smallest weights that one selects to be reset to zero. The pruning percentage is a hyperparameter of the trained network that has to be defined before the training process begins and should be carefully set. The percentage greatly affects speed-accuracy trade-offs since the network must not only fit the training data with a good error, but more importantly, it must learn useful information that generalizes well to unseen examples.

Quantization and Weight Sharing

Quantization and weight sharing are the most widely used techniques for reducing the size of parameters in neural network models. Quantization uses a smaller number of bits to represent each weight and/or each activation, mainly by converting them into discrete integers. Weight sharing involves setting a large number of weights to the same value, either before or after quantization, to reduce the number of unique values shared among tens of millions of weights. Typically, with n-bit quantization, you need to find 2n values, because they often involve n-bit integer representations. Previous approaches used to reduce the size include the codebook initialization on a small subset of input, which is difficult to demonstrate the percentage of utilized values. Finding the optimal set of quantized values sometimes leads to an NP-hard optimization problem. Instead, for simplification, others introduced a stochastic dithering technique, which requires additional storage for dithering noise, or used the element-wise maximization problem algorithm, which achieves suboptimal quantization values for minimizing gradient awareness.

We propose that minimizing the mutual information flow from the weights to the model output is a better and simpler method, which can effectively encourage individual weights to have less redundancy, an essential factor in model compression. The mutual information estimator makes no assumption about the data distribution and can model complex weight distributions effectively. By minimizing mutual information, even binary values can represent the weight distribution while quantization does not suffer from the error accumulation problem due to converting weights to 32-bit floats. Furthermore, our algorithm can naturally provide a continuous relaxation that smoothly decays to a smoothness training objective, an intermediate generalization of CNN quantization techniques represented using scaling methods. Sufficient experiments show that our method consistently outperforms many quantization strengths for ImageNet with AlexNet backbone, a CelebA dataset, and large models for neural architecture search.

Improving Inference Time

Improving inference time is vital for the success of an application deploying AI models, especially in real-time processing applications. Inference time is the time taken to predict output based on unseen data for trained models, directly affecting user experience and the application's overall performance. It is comparatively more crucial to compress or optimize a model than to train the model in an application that involves inference many times on unseen data. Various model compression techniques and optimization strategies have therefore been devised to reduce the time taken to predict outputs for a given unseen sample [1]. Techniques include shrinking model size

using knowledge distillation or quantization, disabling parts of the model like deep compression, sparse neural networks, or pruning, or converting the original model to a faster one like labelling some layers as depthwise separable layers. In the context of deep learning, knowledge distillation is an innovative technique. It works by transferring knowledge from a larger model to the smaller one, allowing the smaller model to mimic the output of the larger model instead of mapping directly from the input. This results in the smaller model being able to approximate the output of the larger model while being smaller and therefore faster. There are many factors that can influence the inference time of a model spanning different levels (i.e., hardware, software, and algorithmic efficiency). They determine the speed by which a trained and compiled model generates predictions on unseen data, each with a unique way of measuring. Hardware capabilities and the particular hardware used are the foremost factors. The inference time of a model drops when a dedicated piece of hardware is used. Thus, performance directly depends on the hardware speciality, with clusters better suited for different tasks. Algorithmic efficiency and model architecture play a crucial role in determining the inference speed as well. Besides hardware and software-dependent factors, parameter count and sparsity can also affect the performance of a model.

Knowledge Distillation

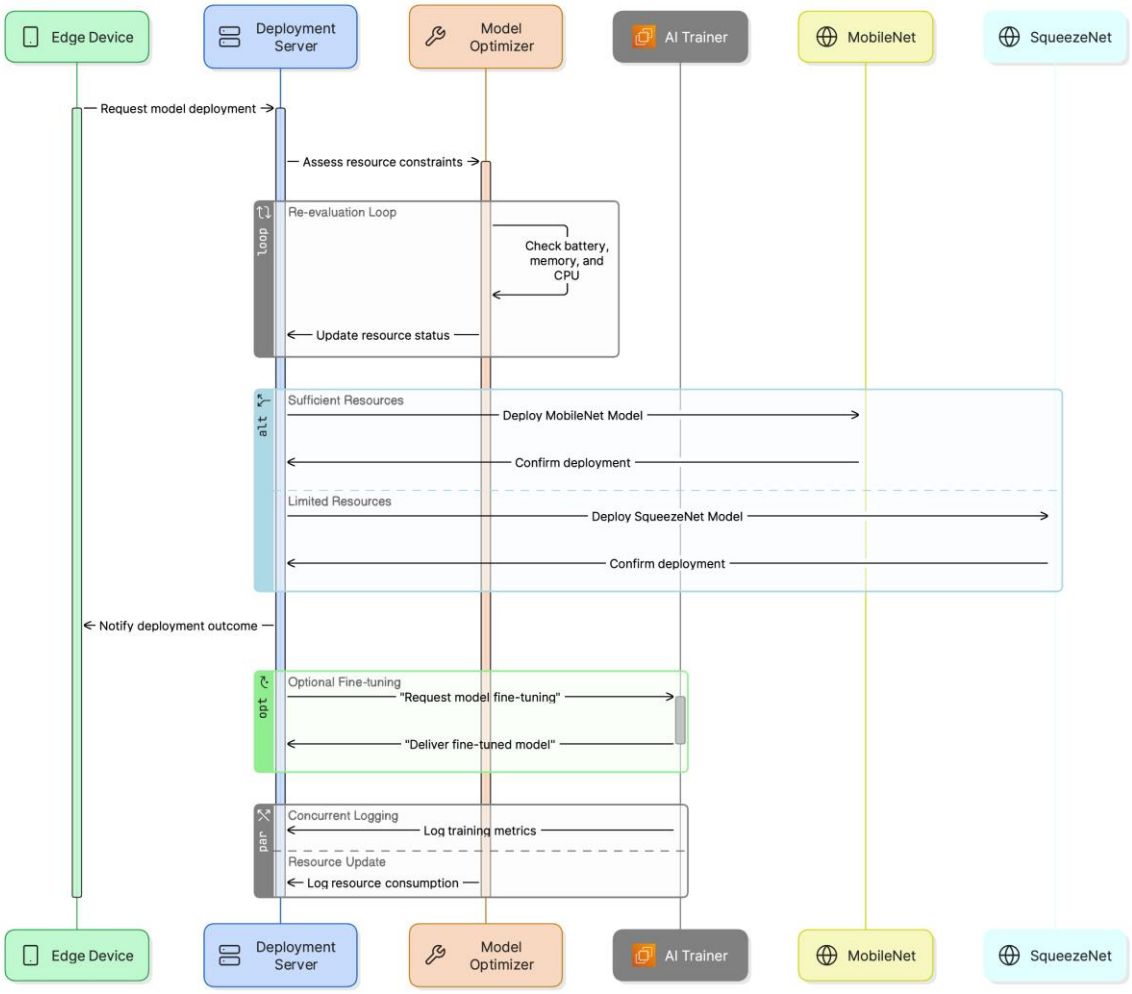
Although deep learning systems have shown impressive performance in various tasks, they are often characterized by large model sizes and high computational requirements. These characteristics make it difficult to deploy such models in real-time applications where computational resources are limited and inference should be performed with a short latency. Therefore, reducing the computational cost and memory footprint of deep learning models is highly desired. Knowledge distillation has proven successful in training smaller models with faster inference times, shortening the gap between practical deployability of models and cutting-edge performance. MobileNet demonstrated that a small model, trained to replicate the output of a larger model, could achieve much better computational efficiency, whilst still delivering good accuracy. Another recent work proposed Platform-Aware Neural Architecture Search (PA-NAS), using efficient performance evaluation techniques to find strong models with affordable inference cost. One of the key components of PA-NAS is using an efficient large-scale resource on-device evaluation to evaluate the model quality. Extreme network quantization and compression enabled ResNet-18 with real-time speed and device resources. ADMM-NN uses parameter quantization and pruning to accelerate model inference in an on-device environment.

The generalized distillation process involves training a smaller model (the student) to mimic a larger network (the teacher). The teacher model can be an Accurate large model or an ensemble of several models, and the student network can be designed to a fixed set of simpler topologies. The student model is trained to predict the true class labels (hard targets) in the initial stages of training but gradually learns to replicate the softened class probabilities (soft targets) outputted by the teacher. It proves effective to train the model in the presence of small perturbations around its predictions. One of the widely used techniques; the model predictions are labels themselves for the model training. Such training is known as soft labeling. Several variations of distillation have been proposed in literature, including methods based on features, adversarial loss, teacher assistant learning, attention transfer, and a framework that adapts the teacher model as the student learns. The proposed methodology will closely follow the traditional knowledge distillation process described by [2] to provide a broad overview of this technique and enable the subsequent discussion of model optimization and performance improvement strategies.

Enabling Deployment on Resource-Constrained Devices

Over past years there has been a trend to deploy increasingly powerful AI models in inherently limited resource settings, such as mobile devices and the Internet of Things. The widespread use of intelligent software on mobile devices and other edge devices brings multiple challenges, such as limited on-chip memory, weak processing power, and energy efficiency. On one hand, the models have to be extremely small since they need to share the limited on-chip memory with data processing. On the other hand, most devices are battery-driven, making energy

efficiency a crucial factor. These several problems demand effective model compression techniques for deep neural networks [3]. From weight and weight-CPU offload to kernel pointwise separability and DNN quantization there are a variety of ways to take into account these requirements. The related performance of these techniques has instead been studied in a separated way, which restricts the degree of granularity of the choices forced by the types of acceleration hardware.



In this section, the field of enabling the deployment of deep learning models in dedicated hardware for resource-constrained devices is systematically discussed. Concretely, the existing hardware platforms, models and frameworks for on-device AI are reviewed first. The related challenges (resource limitation and deployment bottleneck) are analyzed. Next, a number of strategies are discussed to alleviate the challenges, including model quantization, architecture optimization, scheduling of resource consumption, and using larger powerful AI server for training with device-friendly architecture. Finally, it is advocated that more research should be performed on the optimization of deployment strategies, architectures, and operating systems, including adapting application neural network structures and deployment configurations to new devices and applications.

MobileNet and SqueezeNet Architectures

High computational demand and the large model sizes of Deep Convolutional Neural Network (CNNs) hinder their deployment in resource constrained applications such as mobile and edge computing. Therefore there has been considerable interest in compressed models that are both computationally efficient and relatively small in size.

There is significant prior work on network compression of CNNs into smaller versions. MobileNet and SqueezeNet are selected as exemplary models as they showcase innovative architecture and weight sharing to obtain compact network, that are also deployable on mobile.

MobileNet leverages depthwise separable convolutions to compose models that achieve high accuracy while remaining computationally lightweight. A MobileNet model consists of a series of layers, all of which have a single input and output tensor. Each layer i in a MobileNet model has a Type, StrideS[i], a multiplier for the number of output filters called M[iM], an expand ratio M[eff][i], a set of kernel weights K[i], bias B[i], and a nonlinearity. There are eight types of layers that appear in MobileNet: 'Conv2D', 'Depthwise', 'SeparableConv', 'Add', 'Pad', 'Mean', 'Reshape' and 'Concat'. The 'Conv2D' layer corresponds to a normal 2D convolution. The 'Pad' layer "pads" a feature map with zeroes either on the right hand side or on the bottom. The 'Mean' layer is an average pooling operation along the spatial dimensions. The 'Concat' layer concatenates multiple inputs along their spatial dimensions. The stride of a 'Mean' layer should always be 2, and the stride and multiplier for a 'Mean', 'Pad', and 'Concat' layer should always be 1. There are 28 2D Conv layers and 11 1x1 Conv layers in this Tab. These layers reduce the spatial dimensions and increase the number of channels. Additional 3x3 or 5x5 depthwise separable convolution layers are added after expand layers to further shrink the spatial dimension. Depthwise separable convolutions consist of a 2D depthwise convolution followed by a pointwise convolution, and this operation is separable in that it is factored into a depthwise convolution with a kernel size of k, followed by a 1x1 pointwise convolution to the output of the depthwise convolution.

Experimental Evaluation and Case Studies

Despite the extensive studies on model compression, unfortunately, there is still a lack of a comprehensive and valid method of empirical evaluation to inspire trust into the effectiveness and trustworthiness of artificially compressed models within realistic systems. Various valid methodologies are proposed for evaluating model compression comprehensively in terms of model performance. First, quantitative metrics are presented. Experimental results on model performance metrics with variances carried out using the same or similar experimental settings. It offers model developers detailed observations of the influence of different hyperparameters, architectures, or designs on the model's performance when being compressed. Also, with the use of benchmark datasets, a fair comparison in terms of model performance between artificially compressed models and natively compact models is offered. Importantly, the evaluational framework points out future directions of continuously refining the evaluation methodologies while the ever-changing AI landscape, such as neural network architecture, hyperparameter, and application domain, evolves beyond that from the time of writing. Qualitative assessments of model performance are also presented. It shows the importance of beneficial effects of training from scratch using the assistance of artificially compressed models. As widely used in studies on model compression, the study uses benchmark datasets as well for facilitating researchers and engineers to compare and differentiate the performance of artificially compressed models and native compact models. Case studies aiming to fulfill a variety of real-world applications are presented. Overall, positive results are yielded in real-world applications. Three CNN models and two Transformer models with distinct topologies and scale are comprehensively used as baselines to implement experimental efforts, which provide valuable comparisons and insights into the effectiveness, robustness, and limitations of arbitrarily developed model compression techniques. Finally, trustworthiness and empirical leeriness about model compression are discussed, with an emphasis on the critical necessity of thorough empirical testing for the reliability of artificially compressed models in operationalized systems. Further still, the importance of continuously improving empirical evaluation methodologies for shedding light on the powerful AI model compression domain is addressed.

Benchmark Datasets and Metrics

Model compression techniques have gained broad attention as they play a pivotal role in the deployment of deep neural networks on device constrained applications. It is of paramount importance for users and developers to measure the effectiveness of AI model compression techniques. With this objective, the role of benchmark datasets and evaluation metrics are highlighted in this subsection. Benchmark datasets and metrics are depicted as two inseparable components for the evaluation of model compression techniques, indicating that they both influence and relate to architectures. Existing benchmark datasets and metrics are surveyed for evaluating AI model compression methods. Then, the challenges, principles, and future considerations on benchmark datasets and metrics are shared. Aiming at providing a guide for constructing benchmark datasets and metrics for new compression techniques, a benchmark dataset is built for adversarial training on noise robustness evaluation tasks [4].

In the deployment of AI technologies on device-constrained applications, developing efficient AI frameworks for real-time processing with low-latency, power consumption, and bandwidth are crucial. To this end, a wide range of efforts has been studied in recent years, such as hardware design for efficient inference, knowledge distillation, and neural architecture search. However, with the rapid development of deep neural networks, the modest gain in hardware devices in terms of floating-point operation speeds makes the deployment of complex and large AI models on device-constraint applications still challenging. It further promotes the adapt needs for designing efficient and small network architectures. With the quickly growing variety of model architectures, encouraging the development of sophisticated benchmark datasets and metrics for evaluating model architectures becomes necessary.

Conclusion:

It is essential to provide synthetic and compact representations for deep neural networks to deploy them in mobile systems or use them in real time. In the last years a number of model compression techniques proved to be very effective at that end. Unfortunately, a similar effort on evaluating and comparing the effectiveness of such techniques is still at its beginning [5]. Not always better results are obtained using the latest and more complex techniques in literature: this depends among other aspects on architecture, dataset or resources availability. It is crucial to understand how to run such experiments reliably; with this purpose, a series of guidelines are provided both at a low level, e.g., the critical aspect of controlling the training variability during multiple runs, and at a more general level, like what structured and organized comparisons should be done and reported to consistently evaluate the impact of a compression technique.

Deep learning-powered solutions have achieved unparalleled goals in multiple domains, such as image classification and recognition, speech and sound processing, language modeling and natural language processing. Nevertheless, hostile deployment of large and time-consuming deep neural networks remains a concern. As extensive literature and studies have demonstrated, performance, and consequently generalizability, are only poorly determined by the size and structure of the trained DNNs [6]. Subsequently, researchers have sought to uncover expressive and powerful aspects of DNNs which would allow designs of more efficient networks with smaller computational and memory demands that still provide satisfactory accuracy' here obtained by analyzing universally applicable aspects like training dynamics or expressibility properties of DNNs, which can drive future research and shaping of more efficient design paradigms from a broader perspective.

References:

[1] V. Sanz Marco, B. Taylor, Z. Wang, and Y. Elkhatib, "Optimizing Deep Learning Inference on Embedded Systems Through Adaptive Model Selection," 2020.

[2] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network.

[3] H. Cai, J. Lin, Y. Lin, Z. Liu et al., "Enable Deep Learning on Mobile Devices: Methods, Systems, and Applications," 2022.

[4] Y. Matsubara, R. Yang, M. Levorato, and S. Mandt, "SC2 Benchmark: Supervised Compression for Split Computing," 2022.

[5] Iandola, F. N., et al. (2016). SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters and <0.5MB Model Size.

[6] J. O' Neill, "An Overview of Neural Network Compression," 2020.