International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 17 (2025) pp. 189–206

Article

Lightweight Deep Learning Models For Edge Devices—A Survey

Aminu Musa [1,*], Habeebah Adamu Kakudi [2], Mohammed Hassan [2], Mohamed Hamada [3], Usman Umar [4] and Maryam Lawan Salisu [1]

[1] Department of Computer Science, Faculty of Computing, Federal University Dutse, Dutse 720001, Nigeria
[2] Department of Computer Science, Faculty of Computing, Bayero University Kano, Kano 700281, Nigeria
[3] Software Engineering Lab, University of Aizu, Fukushima 965-8580, Japan
[4] Department of Computer Science, Faculty of Science, Federal University Kashere, Gombe 771103, Nigeria
* Correspondence author: musa.aminu@fud.edu.ng

Received date: 3 March 2024; Accepted date: 21 March 2024; Published online: 6 January 2025

Abstract: As edge computing gains attention across various domains, the demand for lightweight deep learning models capable of running efficiently on resource-constrained edge devices has surged. This survey investigates the landscape of lightweight deep learning models tailored for edge computing environments. The survey explores various model compression techniques used to design and optimize deep learning models for edge deployment, including model quantization, pruning, and knowledge distillation. Emphasis is placed on strategies to reduce model size, computational complexity, and memory footprint while maintaining satisfactory performance levels. Additionally, the study examines the performances of these techniques on three real-life datasets evaluating lightweight deep learning models, highlighting the importance of balanced datasets representative of edge device deployment scenarios. Furthermore, this survey provides a comprehensive overview of the current state of lightweight deep learning models for edge devices, offering insights into design considerations, optimization techniques, and performance evaluation methodologies. The findings show that most of the compression techniques suffer from performance degradation, proving the existence of a trade-off between compression and performance. Therefore, we proposed a hybrid lossless-compressed model by combining pruning quantization, and knowledge distillation, to reduce parameters and weights, resulting in a lightweight model. The proposed model is three times smaller than the vanilla CNN model and achieved a state-of-the-art accuracy of 97% after compression, which shows the effectiveness of our approach. These results will serve as a valuable resource for researchers and practitioners aiming to develop efficient and scalable deep learning solutions for edge computing applications.

Keywords: model compression; deep learning; pruning; quantization; knowledge distillation; edge devices

1. Introduction

Deep learning (DL) has revolutionized various fields, including computer vision, natural language processing, and signal processing. However, the computational complexity and resource demands of traditional DL models often limit their deployment on edge devices [1]. The proliferation of edge computing has revolutionized the way data is processed, analyzed, and utilized at the network's periphery, closer to the data source. This paradigm shift has sparked a growing interest in deploying deep learning models directly onto edge devices, such as smartphones, IoT devices, and edge servers, enabling real-time inference and decision-making capabilities [2].

Edge devices are resource-constrained systems like smartphones, wearable sensors, and Internet-of-Things (IoT) gadgets, typically characterized by limited processing power, memory, and battery life [3]. While cloud computing can handle the heavy lifting for powerful models, it introduces latency, privacy concerns, and reliance on constant network connectivity [4].

However, the deployment of traditional deep learning models on edge devices presents significant challenges. These challenges stem from the resource limitations inherent in edge computing environments. Consequently, there

is a pressing need for lightweight deep-learning models that can operate efficiently on edge devices while delivering satisfactory performance [5,6].

In response to this need, researchers have developed a diverse array of techniques to design and optimize deep learning models specifically for edge deployment. These techniques encompass model quantization, pruning, knowledge distillation, and architecture design, among others, with the overarching goal of reducing the model size and computational complexity while preserving accuracy and performance [7,8]. Therefore, lightweight deep learning models have emerged as a crucial area of research, aiming to bridge the gap between powerful, complex models and the limited capabilities of edge devices. These models achieve comparable accuracy with traditional models with a slight drop in performance, while significantly reducing computational cost, memory footprint, and power consumption, making them suitable for a wide range of real-time applications on edge devices [9].

Lightweight models are based on the model compression techniques which have proven to be effective strategies to reduce the size and computational complexity of deep learning models while maintaining an acceptable level of performance. Model compression involves the application of various techniques to reduce the number of parameters and the overall memory footprint of a deep-learning model [10].

This survey provides a comprehensive overview of lightweight deep-learning models for edge devices. We delve into various design strategies and optimization techniques employed to compress and accelerate deep learning models, analyze their strengths and limitations, and showcase their diverse applications on the edge. Furthermore, we discuss open challenges and promising future directions in this rapidly evolving field.

Moreover, this research investigates the issue of performance degradation observed in the majority of deep-learning model compression techniques by evaluating the efficacy of these techniques across three distinct real-world datasets for benchmarking. Ultimately, it introduces a hybrid lossless model, which integrates pruning, quantization, and knowledge distillation techniques to address this challenge effectively.

2. Techniques for Developing Lightweight Models

Many well-known methods in the field of model compression have been made possible by the growing interest in creating compressed deep learning models that can retain the accuracy of traditional models while lowering complexities. Each of these techniques presents distinct advantages and trade-offs, contributing to the diverse landscape of approaches for compressing deep learning models. We offer a carefully thought-out and creative taxonomy. Our suggested taxonomy offers an organized framework for comprehending the variety of lightweight model approaches for edge devices, as shown in Figure 1. This investigation includes a detailed analysis of several well-known methods, including quantization, pruning, knowledge distillation, and Architecture Design.

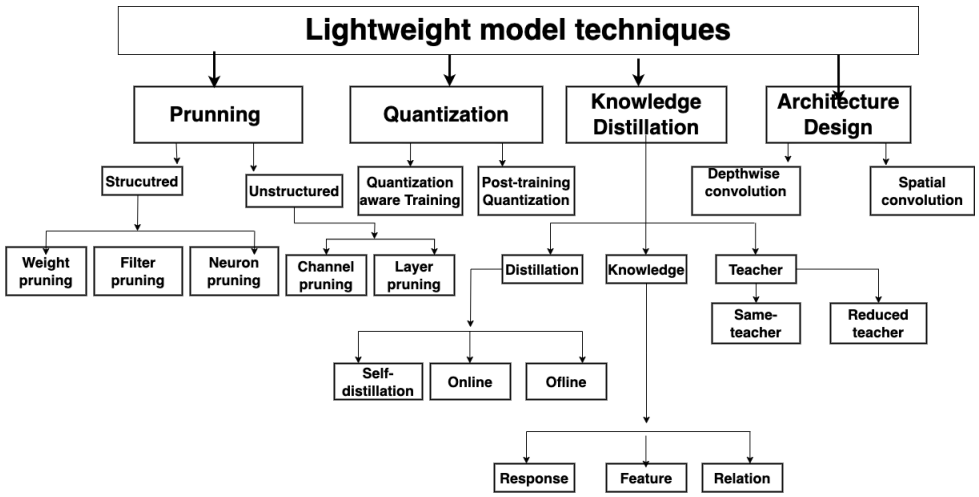

Figure 1. Taxonomy of Lightweight Model Techniques.

2.1. Prunning

Network pruning is a major technique that has been successfully used to reduce redundant parameters in deep neural network models [11]. The technique works by removing unnecessary connections, thus reducing the number of parameters and computation costs [12]. The basic idea behind network pruning is that those redundant parameters provide less contribution to the model performance. Therefore, by removing the redundant parameters, the model

size, memory footprints, and computational requirements can be reduced without affecting the model performance. Several architectures and architecture-specific pruning methods have been proposed in recent years, as found in the work of [13,14]. But structured and unstructured pruning are the two major classifications of network pruning [15].

The concept of network pruning is depicted in Figure 2. which described how redundant weights can be removed from the entire neural network.

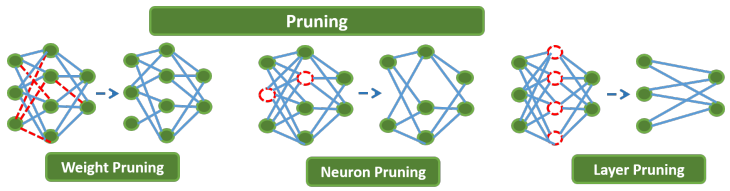

Figure 2. Concept of network pruning.

2.1.1. Structured Prunning

Structured pruning is aimed at reducing the size of deep learning models by identifying and removing entire structured components, such as channels, filters, or neurons, rather than individual parameters [16]. Unlike unstructured pruning, which removes individual parameters such as layers regardless of their location within the model, structured pruning preserves the model's original structure, enabling more efficient implementation and deployment [17].

One of the key advantages of structured pruning is its compatibility with hardware accelerators and deployment on resource-constrained edge devices [18]. By preserving the structured nature of the model, structured pruning facilitates streamlined inference, as hardware accelerators can exploit the sparsity patterns induced by pruning to optimize memory access and computational efficiency [19].

A variety of structured pruning was used to successfully compress the AlexNet model by a factor of 9 without sacrificing accuracy. The resulting pruned model has a significantly reduced memory size and small computational complexity while maintaining a comparable level of performance to the original model [20].

Structured pruning techniques can be applied independently or in combination with other compression methods, such as quantization or knowledge distillation, to further enhance model compression and efficiency [21]. Moreover, recent advancements in structured pruning algorithms have led to more sophisticated pruning strategies, such as iterative pruning with retraining and network slimming, enabling deeper model compression with minimal performance degradation [22]. Overall, structured pruning offers a promising approach for developing lightweight deep learning models tailored for deployment on edge devices, facilitating efficient inference and real-time processing in resource-constrained environments. By exploiting the structured nature of deep learning models, structured pruning enables substantial model compression without compromising predictive performance, paving the way for scalable and energy-efficient edge computing solutions [23].

2.1.2. Unstructured Pruning

Unstructured pruning simplifies a network by removing individual weights or neurons based on their importance, typically using a threshold to set insignificant ones to zero [24]. While this approach directly reduces the model size, it disregards the inherent structure of the network, creating an irregular and sparse model. This irregularity complicates both storage and computation of the pruned model, often requiring specialized techniques for efficient handling [25]. Additionally, unstructured pruning frequently necessitates substantial retraining to recover the lost accuracy, which can be particularly resource-intensive on edge devices with limited processing power [26].

One of the main advantages of unstructured pruning is its flexibility and simplicity in implementation. Since it operates at the level of individual parameters, unstructured pruning does not require modifying the model's architecture or structure, making it compatible with a wide range of deep learning models and frameworks [27].

Despite its effectiveness in reducing model size, unstructured pruning presents challenges in deployment, particularly on hardware accelerators or edge devices with limited memory and computational resources. The irregular sparsity patterns induced by unstructured pruning can lead to suboptimal memory access patterns and inefficient utilization of hardware resources.

Generally, Pruning has been widely used in developing lightweight models for edge devices. In Table 1, we present the recent literature on pruning-based lightweight models.

Table 1. Pruning-based Lightweight Models for Edge devices.

Methods	Pruning Type	Performance
Pruning via geometric mean [28]	Structured	No Performance drop
Pruned layer reconstruction [29]	Structured	5.13× speed-up with only 0.65% top-5 accuracy drop
Lightweight Bi-LSTM [30]	Structured	36% pruning ratio and improve accuracy by 3%
Block removal strategy [31]	Unstructured	Achieved 5.95x compression rate with 90.5% accuracy
Weight pruning strategy [32]	Structured	83.5% Top-1 accuracy using a pruned Mnas-Net with 12MB size
Low-variance features pruning [33]	Unstructured	Improves the performance and efficacy of malware detection models
Lightweight diffusion model [34]	Unstructured	Enables approximately a 50% reduction in FLOPs
Selective structure removal [35]	Structured	The compressed models still exhibit satisfactory capabilities
Automatic channel pruning [36]	Unstructured	Achieved size=16.3MB, FLOPS=2.31G, and mAP=71.2 using SSD model
Feature welding [37]	Unstructured	About 49 FPS on the CPU
Parameter searching [38]	Unstructured	The pruned P3DNet is 39.54% faster than MobileNet v3 and 50.65% faster than ShuffleNet V2
Channel importance [39]	Unstructured	Satisfactory balance between computational efficiency and detection accuracy

The key observation from Table 1 is that structured pruning techniques generally exhibit superior performance compared to their unstructured counterparts, as evidenced by several works [24,25,28,31]. For instance, the structured pruning approach using geometric mean achieves high accuracy retention on VGG-16 with a 64% compression rate [24]. Similarly, the lightweight Bi-LSTM architecture demonstrates negligible accuracy degradation while reducing model size significantly [27]. This suggests that structured pruning might be more effective in preserving model accuracy while reducing model size, making it a suitable choice for edge devices.

In contrast, unstructured pruning techniques like low-variance features pruning and automatic channel pruning tend to prioritize computational cost reduction over maintaining accuracy, as reflected by their focus on metrics like FLOPs reduction instead of mentioning preserved accuracy [33–35].

It's important to acknowledge that the table presents a limited selection of techniques, and their performance can vary depending on the specific model and dataset used [24,25,27,28,31,33,35]. Additionally, it doesn't explicitly address the trade-offs between different approaches, such as their impact on inference speed or memory footprint.

In conclusion, the table provides evidence that structured pruning techniques might be more suitable for achieving both accuracy preservation and size reduction when developing lightweight deep learning models for edge devices [24,27]. However, it is crucial to carefully evaluate different techniques based on the specific application's requirements and consider relevant performance metrics beyond those presented in the table.

2.2. Knowledge Distillation

The intuition behind Knowledge distillation (KD) involves training a smaller model(student) to mimic the behavior and predictions of a larger model (teacher) [40]. The student model learns from the softened probabilities or feature representations generated by the teacher model during training. This technique enables knowledge transfer from a larger model to a smaller one, resulting in a more compact yet effective model [41]. Knowledge distillation (KD) is a new approach for developing a lightweight model by transferring knowledge from a complex teacher model to a simpler student model. It has gained popularity for its simplicity, and promising performance [42]. The concept of knowledge distillation is visualized in Figure 3.

knowledge distillation offers a flexible framework for transferring various types of knowledge from a teacher model to a student model, enabling the creation of compact and efficient models with comparable performance to their larger counterparts. In knowledge distillation, the knowledge can either be Response-based [43], Feature-based [44], or Relation [45].

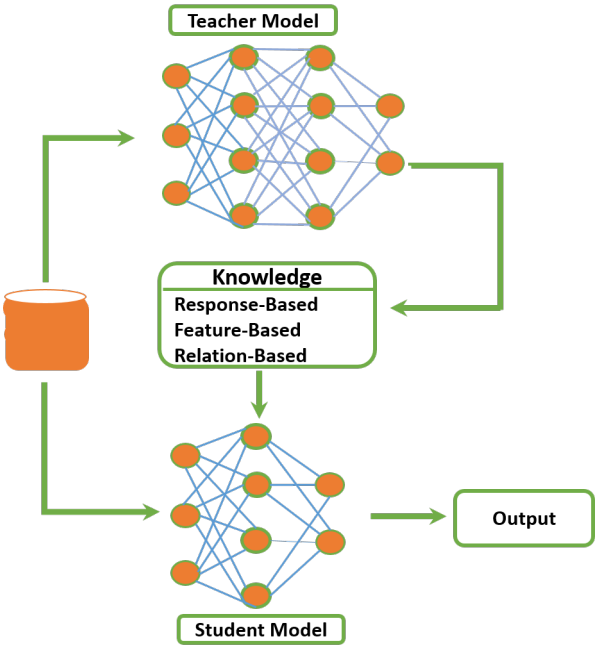

Figure 3. Concept of Knowledge Distillation.

2.2.1. Response-Based

In response-based, knowledge often describes the neuron's reaction to the teacher model's final output layer. The primary concept is to precisely imitate the teacher model's final predictions. Response-based knowledge distillation is a common technique for model compression that is easy to use and efficient for a variety of activities and applications. while Learning several levels of feature representation with increasing abstraction is a strong suit for deep neural networks. According to Bengio et al., [46], this is referred to as representation learning. As a result, feature maps, the output of intermediate layers, and the output of the final layer can be utilized as information to oversee the training of the student model. Particularly for the training of thinner and deeper networks, feature-based knowledge from the intermediate layers is a useful extension of response-based knowledge.

2.2.2. Relation-Based

knowledge distillation refers to the transfer of relational information or dependencies between classes from the teacher model to the student model. This type of knowledge distillation aims to capture the complex relationships and correlations that exist between different classes in the data, enabling the student model to better understand the underlying structure of the task at hand [47].

2.2.3. Feature-Based

In both response and feature-based knowledge distillation, the focus is primarily on transferring soft labels or feature embeddings from the teacher model to the student model. However, in relation-based knowledge distillation, the emphasis shifts towards capturing the relationships between classes, which may not be explicitly encoded in the soft labels or feature representations [44].

Knowledge distillation offers added advantages over the rest of the techniques in the sense that it does not have restrictions over the underlying hardware, and the resulting lightweight model can be deployed with hardware accelerators, to improve inference speed. Consequently, one of the major challenges of KD includes hyperparameter selection and understanding the accuracy-compression trade-off [41].

Recent studies in KD Future include developing advanced distillation techniques and exploring transfer options across domains [48]. Overall, knowledge distillation enables efficient model deployment and holds promise for various application domains. Table 2 presents a summary of works found in the literature that utilized knowledge distillation techniques for lightweight models.

Table 2. Summary of Lightweight Models Using Knowledge Distillation.

Authors	Response	Feature	Relation	Performance
Musa et al. [30]	[✓]			Improved Accuracy
Cheng. D et al. [32]			[✓]	Strike a balance between model size and performance
Chen et al. [48]		[✓]		Achieved enhanced results on Inception and MobileNet of 72.25%, and 64.14%
Kang et al. [49]	[✓]	[✓]	[✓]	Outperforms other knowledge distillation models
Guo et al. [50]	[✓]			Feasible for many real-time deployments
Liu et al. [47]			[✓]	Effective and efficient for resource-constrained environments
Mishra et al. [51]		[✓]		Improves inference time on edge devices
Wang et al. [52]	[✓]	[✓]		Addresses the trade-off between accuracy and efficiency
Park et al. [45]			[✓]	Improves educated student models with a significant margin
Xu et al. [44]		[✓]		Performance better than other variants of KD
Yang et al. [43]	[✓]			Suitable for IoT and Embedded devices
Musa et al. [41]	[✓]			Achieved better performance in terms of accuracy and model size
Musa et al. [1]	[✓]			Reduces energy and computational overhead of teacher model
Zhang et al. [53]			[✓]	Improve performance with reduced training cost
Yim et al. [54]			[✓]	Student DNN outperforms the original DNN

From Table 2 Several studies ([30,32,43,45,46,48,49]) reported improved accuracy and performance using KD. For instance, Musa et al. [30] achieved enhanced accuracy, while Chen et al. [45] observed significant improvements in student models' performance. These advancements highlight KD's effectiveness in enhancing model performance.

Moreover, A recurring theme across the studies is the trade-off between accuracy and efficiency (model size and inference time). While some studies prioritized accuracy ([30,43,45,46,49]), others aimed to strike a balance between both aspects ([32,47,48]). Notably, Cheng et al. [32] achieved balanced results on Inception and MobileNet deployments, demonstrating the potential of KD for practical applications.

The majority of the studies explored KD in the context of resource-constrained environments, like edge devices and internet-of-things (IoT) applications ([32,47–49]). Chen et al. [45] addressed the accuracy-efficiency trade-off and achieved efficient inference time on edge devices. Similarly, Yang et al. [39] found their method suitable for IoT and embedded devices, while Guo et al. [47] reported reduced energy and computational overhead. These findings suggest that KD can be effective in resource-constrained settings.

Another potential benefit of KD is reduced training costs. Musa et al. [1] observed improved performance with reduced training cost, suggesting that KD can be a cost-effective approach to model development.

However, While the studies presented promising findings, some limitations are worth considering. Firstly, the studies employed different benchmark datasets and evaluation metrics, making direct comparisons challenging. Secondly, the table primarily focuses on accuracy and efficiency, and other factors like memory footprint and complexities were not mentioned in many of the works, making it difficult to extensively explore the literature from that perspective.

Therefore, the studies summarized in the table demonstrate the potential of knowledge distillation for various improvements, including enhanced accuracy, performance, and efficiency. While acknowledging limitations, this discussion section highlights the promising capabilities of KD and paves the way for future research directions.

2.3. Quantization

Quantization, on the other hand, reduced the precision of weights and activation to utilize lower-bit representations, thereby reducing memory requirements [55]. Furthermore, model quantization is a model size reduction technique that converts model weights from high-precision floating-point representation to low-precision floating-point or integer representation, such as 16-bit or 8-bit. By converting the weights of a model from high-precision floating-point representation to lower precision, the model size and inference speed (latency) can improve by a significant factor without sacrificing too much accuracy [56].

Quantization is a widely used technique for compressing deep learning models, enabling efficient deployment on resource-constrained edge devices and hardware accelerators. By reducing the precision of parameters and activations, quantization achieves significant reductions in memory storage, bandwidth requirements, and computational complexity, while minimizing the impact on model accuracy [57].

All model parameters can be converted into low precision during quantization. In other instances selected parameters such as weights, and activations as described in Figure 4 which depict neural network quantization processes [58].

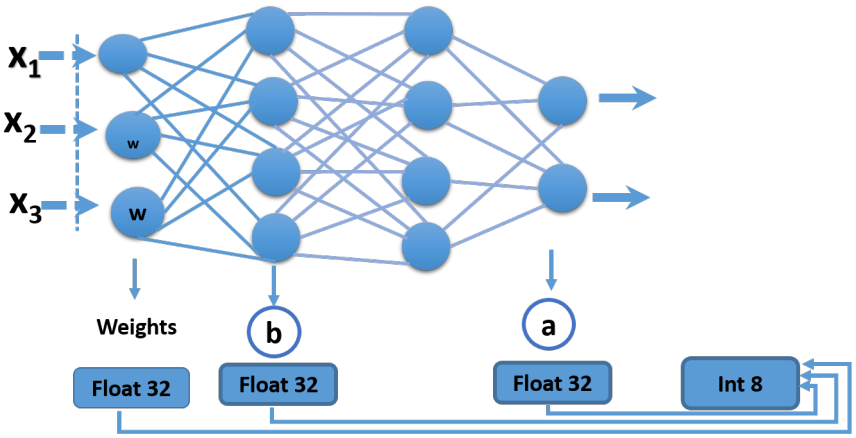

Figure 4. Concept of quantization.

To obtain a lightweight model, researchers employed quantization techniques [59]. Also, it was reported that quantization does not only reduce memory bandwidth requirement, it also improves the performance of a model and

increases cache utilization [60]. Quantization can be achieved through quantization-aware training or post-training quantization.

2.3.1. Quantization-aware Training(QAT)

This involves training the deep learning model while considering the effects of quantization from the outset [61]. During training, the model is trained with simulated quantization effects, allowing it to learn to adapt to the reduced precision of parameters and activations [62]. This approach typically involves modifying the training process to incorporate quantization-aware optimization techniques, such as quantization-aware backpropagation and activation quantization. By training the model with quantization-aware techniques, it can achieve better accuracy and performance when quantized for inference[63].

2.3.2. Post-Training Quantization(PTQ):

Post-training quantization, on the other hand, involves quantizing the pre-trained deep learning model after it has been trained with full precision (typically 32-bit floating-point numbers). Once the model has been trained, its parameters and activations are quantized to lower precision (e.g., 8-bit integers or fixed-point numbers) without retraining [64]. This quantization process is applied after training and is often followed by fine-tuning to recover any accuracy loss incurred by quantization. Post-training quantization is simpler to implement than quantization-aware training and is suitable for models already trained with standard techniques [65]. Despite the promises of the two variants of quantization, they both have some notable drawbacks. For instance, Quantization-aware training allows the model to learn to adapt to quantization effects during training, potentially resulting in better accuracy and performance compared to post-training quantization [66]. However, Post-training quantization is simpler to implement and does not require modifications to the training process. It can be applied to pre-trained models without the need for retraining, making it more practical for existing models and frameworks [67].

Overall, the choice between quantization-aware training and post-training quantization depends on factors such as model complexity, training resources, and the desired balance between accuracy and efficiency. Both approaches offer effective means of compressing deep learning models for deployment on resource-constrained edge devices and hardware accelerators.

In Table 3, we present a summary of the literature on quantization-based lightweight models. The table provides a concise summary of relevant literature on quantization techniques for model compression in deep learning. Studies are categorized based on their approach (quantization-aware training or post-training quantization), and key findings are outlined, including reductions in model size, improvements in inference speed, and impact on accuracy.

Table 3. Summary of Literature on Lightweight Models Using Quantization.

Authors	**Approach**	**Findings**
[68]	QAT	only 0.1% accuracy degradation.
[58]	QAT	Achieved 82.5% accuracy on image classification
[69]	PTQ	reduced model size up to 5x while maintaining full model performance
[70]	PTQ	30x fewer parameters, with higher performance on Div2K
[71]	QAT	minimal training overhead using mobileNet architecture.
[72]	QAT	Achieved a high speed, lightweight, and relatively high accuracy
[73]	PQT	2-fold decrease in memory usage from 15.51 MB down to 7.68 MB
[74]	Hybrid	up to 16× weight size reduction
[75]	QAT	Good compression ratios with negligible accuracy degradation
[76]	QAT	reduces the model size while maintaining optimal performance
[77]	PQT	Investigated various PQT methods. Concludes that activations are better.

We highlight recent advances in various quantization techniques employed to develop lightweight deep-learning models for edge devices, as summarized in Table 3. Post-training quantization (PTQ) and Quantization-aware training (QAT) emerge as popular choices, achieving good compression ratios and maintaining model accuracy ([64,65,67,71, 72]). PTQ is particularly favorable due to its minimal training overhead [66]. Combining QAT with pruning strategies can reduce model size [70]. While PTQ shows promise, activating quantization might offer even greater benefits, as suggested by the findings in [73]. It's crucial to remember that this table represents just a subset of the ongoing research in quantization. Nonetheless, quantization techniques demonstrate considerable potential for creating lightweight deep learning models well-suited for edge devices by effectively reducing model size and memory footprint while preserving accuracy, thus paving the way for real-time applications on resource-constrained platforms.

2.4. Reserch Trends

In this section, we comprehensively analyze emerging research trends related to lightweight models for edge computing. Leveraging multiple databases, including but not limited to Springer, IEEE Xplore, Science Direct, ACM Digital Library, and arXiv, we have compiled a diverse collection of papers spanning recent years. By synthesizing findings across these databases, we aim to provide a holistic view of the evolving landscape of research in this domain.

The works from the literature were curated from various databases to ensure a broad coverage of the research landscape. Each database offers unique insights into different aspects of lightweight models, including model architectures, optimization techniques, deployment strategies, and application domains. By combining data from multiple sources, we gain a richer understanding of the diverse perspectives and approaches adopted by researchers worldwide. Figure 5 presents a barplot of the frequency of the literature found in each database, while a line plot is provided in Figure 6. to give insight into the trends of the publication per year.

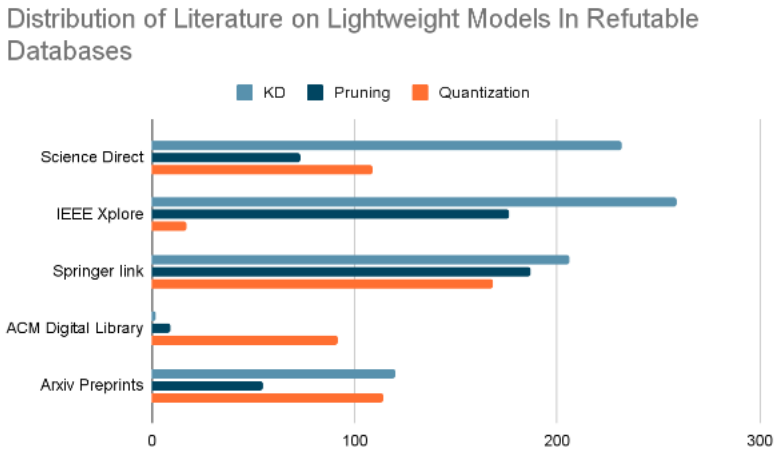

Figure 5. Frequency of Research found in the databases.

Our analysis reveals the prevalence of research on lightweight models in various academic databases. Science Direct has been the most prominent source, housing over 200 publications on lightweight models, significantly surpassing other databases like IEEE Xplore and Springer Link which hold a moderate quantity between 100 and 200 publications. ACM Digital Library and ArXiv Preprints appear as the least frequented sources, each containing less than 100 publications on the topic. It is crucial to acknowledge that this graph only represents a limited set of reputable databases, and other sources might hold valuable research not captured here. Nevertheless, this analysis suggests that Science Direct serves as a primary source for research on lightweight models, highlighting the importance of consulting diverse databases for a comprehensive understanding of this evolving field.

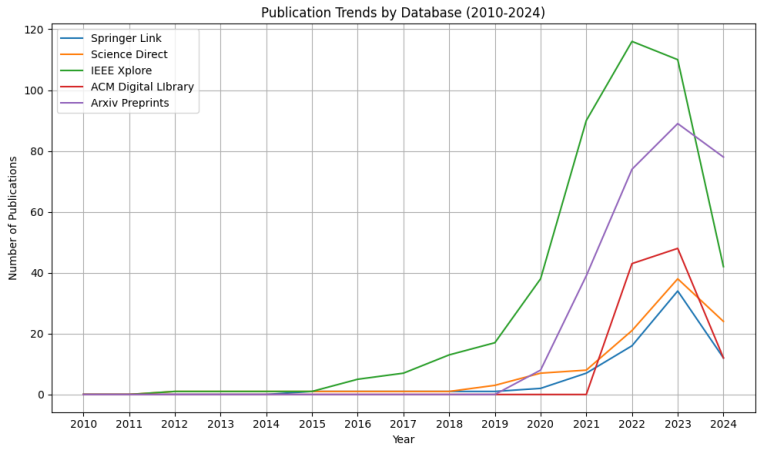

Figure 6. Publication Trends.

Additionally, an analysis of publication trends across various reputable databases, as depicted in figure 6 reveals a growing surge in research about lightweight deep learning models. From 2010 to 2024, all five databases showcased on the chart experienced a significant rise in the number of publications. This surge signifies an interest in this field and its potential applications. Notably, Springer Link and Science Direct emerged as the frontrunners, consistently housing the highest volume of publications, with Science Direct exhibiting a particularly steep rise in recent years. While IEEE Xplore and ACM Digital Library displayed a comparable pattern of growth, their overall publication counts remained lower. Interestingly, although Arxiv Preprints held the fewest publications historically, it exhibited the most rapid growth in recent years, suggesting its growing prominence as a platform for sharing research in this domain. It is crucial to acknowledge that the graph solely reflects the quantity, not the specific content or quality of the publications within each database. Nevertheless, this analysis underscores the flourishing field of lightweight deep learning models and the increasing availability of research across diverse databases, highlighting the significance of consulting a variety of sources for a comprehensive understanding of this evolving domain.

In conclusion, several works on model compression were found in the literature. These works were analyzed concerning the technique used. Consequently, the studies usually evaluated their methodology on baseline databases such as Cifar10 or Mnist. Additionally, a significant number of the techniques have a common notable drawback, which is a performance drop after compression. proving the existence of a trade-off between compression and model performance. To produce a lossless lightweight model, recent techniques are proposed that utilize separable convolutions towards the lightweight model. This methodology was seen in the work of [78–81]. However, not all kernels can be separated into small kernels, limiting the use of such techniques to a few architectures. Therefore, we proposed a hybrid model combining Knowledge Distillation, pruning, and post-training quantization. We also evaluated the model on three different real-life datasets to mimic real scenarios.

3. Proposed Hybrid Model Methodology

This section introduces the datasets used in the study and describes how the large deep-learning model is compressed using various compression techniques. The section also outlined how a hybrid lightweight model is proposed by combining pruning and quantization. Finally, the section discusses the metrics used to evaluate and compare the different models' performances.

3.1. Dataset

Two real-life datasets were collected for model evaluation, pothole detection, and paddy rice maturity detection. In contrast, the soybean weed detection dataset is an open-source dataset obtained online at [82]. pothole and rice maturity datasets were carefully collected and curated to ensure the diversity and representativeness of the target classes. The first dataset is the pothole detection dataset [83]. Pothole images were collected and organized into two directories, potholes, and good road surfaces, which have 1,245 and 2,943 images, respectively.

The second dataset is the rice yield detection dataset, comprising images of rice at different maturity stages. the dataset is organized into yield and unyield directories with 1,450 and 1,285 images, respectively.

The third dataset is the soybean weed detection dataset. The dataset has three directories, namely corn, soybean, and weed, with a total of 3,200 images.

3.2. Data Pre-Processing

All the datasets used in this study come in different sizes and have different shapes. A pre-processing step is necessary to transform the images into a format that is acceptable to the proposed model. The datasets are imbalanced because the images available in one class are significantly larger than those in the other. This condition may lead to overfitting. Therefore, data augmentation techniques, such as rotation, scaling, and flipping were applied to create augmented images to complement the class with lower instances and to increase the size and diversity of the datasets.

3.3. Model Definition and Training

Initially, a large CNN model was built from scratch. consisting of numerous convolutional layers and millions of parameters. The model was trained on the three selected datasets, employing appropriate parameter tuning techniques and loss functions to achieve better performance. This model is regarded as a cumbersome model. While the performance of the model is satisfactory, the size is problematic, especially if the model is to be deployed on edge devices.

To produce a lightweight model that can be deployed on edge devices, the cumbersome model is compressed using the three widely used model compression techniques discussed in Section 2. The proposed lightweight models are evaluated using standard metrics. Lastly, a hybrid lightweight model is proposed by combining pruning, quantization and knowledge distillation.

The proposed lightweight hybrid model received knowledge from the teacher using knowledge distillation. The weights of the hybrid model are pruned using a 0.5 pruning ratio, which means literary setting the none non-significant weights to zero. This results in a model with 50% of the original model weight. The model was trained and fine-tuned until adequate accuracy on the training and validation set was obtained. Finally, the pruned model was quantized from float 32 precision to int 8 and retrained for 2 epochs using pruned weights. The description of the entire experimental workflow and processes is presented in the system flowchart in Figure 7

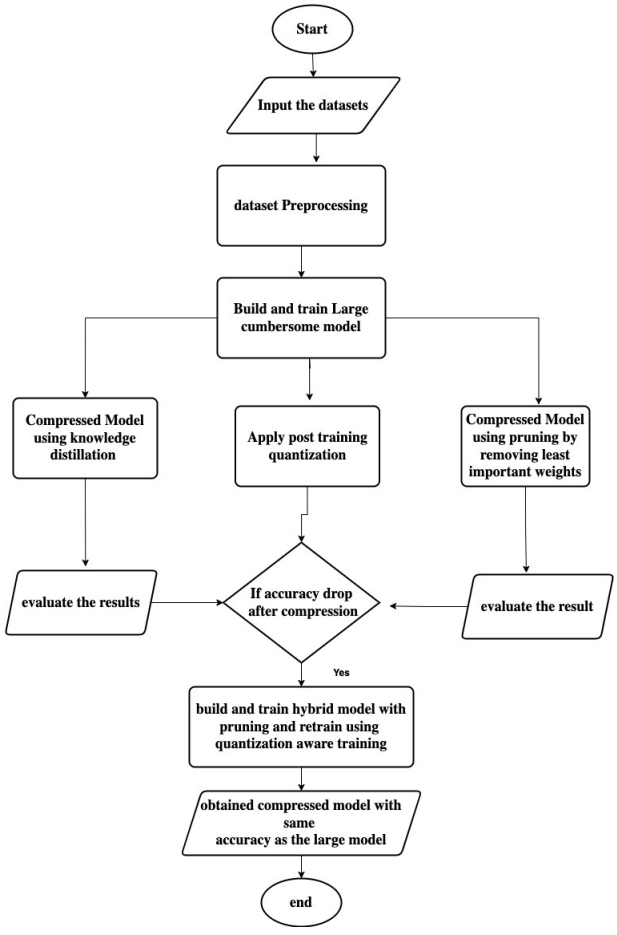

Figure 7. Proposed model Flowchart.

To further describe the proposed hybrid model, Algorithm 1 gives a general walk-through of the model formulation process.

Algorithm 1 : Proposed Hybrid Model

Data: Large model
Result: Lightweight model
initialized: pruning ratio $\leftarrow \beta$
obtained weights and parameters of large model γ, α ;
compute threshold $\epsilon \leftarrow (\beta * \text{sum } (\gamma))$;
compute pruned weight (β, γ, ϵ);
while *training* **do**
    For all γ;
    if $\beta \leq \epsilon$ **then**
        $\gamma \leftarrow 0$;
    else
        if $\beta > \epsilon$ **then**
            $\gamma == \gamma$;
            Update parameter $\alpha \leftarrow \gamma$;
return pruned model
while *weights are pruned* $\doteq$*true* **do**
    convert model precision from fp32 to int8;
    re-train pruned model with quantized parameters;
return pruned and quantized model

3.4. Performance Evaluation

The compressed models obtained from each experiment using a single compression technique were evaluated on separate test sets, for three different classification tasks using accuracy, no. of parameters, model size, and training time. The performance of the proposed hybrid model was recorded alongside the rest of the models. All the models' training and validation accuracy were also compared to evaluate any potential performance degradation.

4. Result and Discussion

The performance of the compressed models obtained from different compression techniques was compared with each other and with the original uncompressed model. The trade-offs between model size reduction and classification accuracy were analyzed for each compression technique. The hybrid lossless compressed model was evaluated for its effectiveness in achieving a lightweight model with state-of-the-art accuracy.

4.1. Experimental Setting

In this paper, we evaluated our approach on real-life datasets, not on the baseline datasets, to show the robustness and effectiveness of our approach when used in real-life scenarios.

All the models were implemented using Python programming language and the TensorFlow framework. The experiments were conducted on the Google Colab platform with GPU V100 backend. Therefore some metrics such as training time might vary when trying to replicate the experiments on different hardware.

In all the experiments, data augmentation was used utilizing common augmentation strategies such as rotation, shifting, and mirroring. the models were trained using stochastic gradient descent (SGD), with a starting learning rate of 0.01. Both models are trained from scratch with an Adam optimizer and a batch size of 32. All models are trained for 10 epochs each.

4.2. Experimental Result

The experimental results, including accuracy, size reduction, and computational requirements, were recorded and analyzed. The performances of the proposed hybrid lossless compressed model were compared with the traditional compression methods. The results are presented in Tables 4, 5, and 6 respectively. The accuracy and loss curves of the proposed model in all three experiments conducted are shown in Figure 8. All three traditional compression techniques performed remarkably in reducing the model size and cutting down training time. However, some of the techniques achieved high compression rates at the cost of accuracy, which shows that traditional compression reduced the model performance.

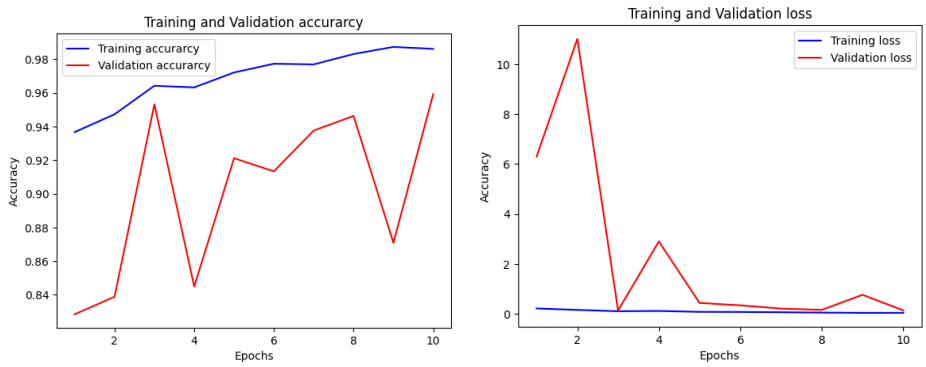

Figure 8. Accuracy and loss curves of the proposed model.

Table 4. Performance of the models on pothole detection task.

Models	Accuracy	No. of Params	Model Size	Training Time
Large CNN	0.9743	52,768,432	603 MB	17 m
KD model	0.9828	126,569	15.3 MB	3 m
Pruned Model	0.9023	52,711,489	195 MB	7 m
Quantized	0.9241	2,936,589	35 MB	20 m
Proposed hybrid	0.9774	142,345	3.3 MB	3 m

Table 5. Performance of the models on paddy rice detection.

Models	Accuracy	No. of Params	Model Size	Training Time
Large CNN	0.8109	52,714,439	603 MB	17 m
KD model	0.7994	126,569	15.3 MB	3 m
Pruned Model	0.7612	52,711,489	195 MB	7 m
Quantized	0.7900	2,936,589	35 MB	19 m
Proposed hybrid	0.8203	142,345	3.3 MB	6 m

Table 6. Performance of the models on soybean weed detection.

Models	Accuracy	No. of Params	Model Size	Training Time
Large CNN	0.9593	52,714,439	603 MB	17 m
KD model	0.9219	126,569	15.3 MB	3 m
Pruned Model	0.9367	52,711,489	195 MB	6 m
Quantized	0.9726	2,936,589	35 MB	19 m
Proposed hybrid	0.9703	142,345	3.3 MB	6 m

4.3. Discussion

The findings from the experiments were discussed in terms of the effectiveness of each compression technique and its impact on model size and performance. The potential limitations and future directions for improvement were addressed.

From the pothole detection experiment, it can be observed in Table I that the performances of the models compressed using pruning and quantization techniques were reduced significantly. While knowledge distillation (KD) compressed model performance was improved, the proposed hybrid model manages to keep performance the same.

Consequently, in the rice maturity detection task, the performance of the models changed completely. The large CNN model was able to achieve 81%, while the KD compressed model performance reduced hugely,

Interestingly, in the third experiment on soybean weed detection, the performances of the proposed hybrid model, KD, and quantized models improved upon the performance of the large model. while pruning performance degrades.

One notable observation from all the experiments is that the performance of the traditional model compression techniques varies from task to task, depending on the task type and the dataset's nature. There is a dilemma when selecting which model compression to use, as the performance changes drastically based on the task at hand.

Another observation from the experimental results shows that among the traditional compression methods, the KD compression technique gives the lightest model with just 11 MB on disk, 100k+ parameters, and a few minutes of training time. However, the proposed model was able to achieve better performance with just 3.3 MB size on disk, the same training time as the KD model, and slightly higher parameters than KD. While the rest of the model's size based on pruning and quantization was still relatively small compared to the large model, the training time and the number of parameters are somewhat similar.

However, despite all the variability of the model performance when compressed, the proposed hybrid approach was able to maintain a similar performance with the large model, or even improve it, in some scenarios. Therefore, based on the results from the three different experiments conducted in this study, it can be concluded that the proposed hybrid approach is a lossless model compression technique. The approach has proven effective in achieving a lightweight model with a lower memory footprint, less training time, and state-of-the-art performance on image classification. The proposed model size and performance indicate that the model can be deployed on edge devices efficiently.

5. Conclusion and Future Direction

In this work, we propose a novel hybrid pruning-quantized model effective for deployment on edge devices. To demonstrate the robustness of the proposed model, the model was trained and tested on various image classification tasks using diverse datasets. The proposed approach can be used to train and compress deep neural networks effectively. The results have demonstrated the usefulness of our strategy and verified that the proposed model can preserve the redundant CNN's performance while reducing its complexity.

It is recommended that, in the future, the proposed model be used in different domains such as NLP to measure its cross-domain performance. It is imperative to also investigate the performance of such a hybrid model on different domain-specific tasks without limiting the experiment to image classification only. Model safety and fairness concern can also be investigated.

Author Contributions
All authors contributed to the study's conceptualization and methodology design. Material preparation, data collection, and analysis were performed by Christian Usman Umar and Maryam Lawan. Model Development was done by Aminu Musa. The first draft of the manuscript was written by Aminu Musa and Habeebah Adamu Kakudi, and all authors commented on previous versions. All authors read and approved the final manuscript.

Funding
This research was partially funded by the Google exploreCSR 2023 award.

Conflicts of Interest
The authors declare no conflicts of interest.

Data Availability Statement
The data and code are available in the GitHub repo: https://github.com/El-amin/Hybrid-Lightweight-Model.

References

1. A. Musa, M. Hassan, M. Hamada, and F. Aliyu, "Low-power deep learning model for plant disease detection for smart-hydroponics using knowledge distillation techniques," *J. Low Power Electron*, vol. 12, p. 24, 2022. [Online]. Available: https://doi.org/10.3390/jlpea12020024.
2. A. Goel, C. Tung, Y.-H. Lu, and G. K. A. Thiruvathukal. Survey of Methods for Low-Power Deep Learning and Computer Vision. arXiv, 2019.
3. R. Rout, P. Parida, and S. Dash, "Automatic skin lesion segmentation using a hybrid deep learning network," *International Journal of Computer Information Systems & Industrial Management Applications*, vol. 15, pp. 238–249, 2023.
4. M. Merenda, C. Porcaro, and D. Iero, "Edge machine learning for ai-enabled iot devices: A review," *Sensors*, vol. 20, p. 2533, 04 2020.
5. K. Sharma and S. Bhusnur, "Prediction of oxygen content using deep learning cnn architecture for the classification of blast furnace gas fired boiler images." *International Journal of Computer Information Systems & Industrial Management Applications*, vol. 15, 2023.
6. D. T. Luong, D. D. Anh, T. T. Hanh, H. T. L. Huong, T. X. Thang *et al.*, "Detection, classification, and counting blood cells using yolov8." *International Journal of Computer Information Systems & Industrial Management Applications*, vol. 15, 2023.
7. M. A. Rahman, M. Hamada, and J. Shin, "The impact of state-of-the-art techniques for lossless still image compression," *Electronics*, vol. 10, no. 3, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/3/360.
8. D. K. NC, K. Suresh *et al.*, "Ensemble learning for static hand gesture recognition using hog and lbp features on rgb-d data." *International Journal of Computer Information Systems & Industrial Management Applications*, vol. 15, 2023.
9. J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization, transfer learning," vol. 2017, 2021.
10. C. Xi, X. Zhiqiang, and C. Yuyang, "Introduction to model compression knowledge distillation." 6th International Conference on Intelligent Computing and Signal Processing (ICSP), 2021.
11. S. Han, H. Mao, and D. W. J, "Deep compression: Compressing deep neural networks with pruning trained quantization and huffman coding," 2015.
12. Y. Nie, R. Cocci, C. Zhao, Y. Diao, and P. Shenoy, "Spire: efficient data inference and compression over rfid streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, 2012.
13. A. Silva, D. Fernandes, R. Névoa, J. Monteiro, P. Novais, P. Girão, and T. Afonso, "Resource-constrained onboard inference of object detection and localisation in point clouds targeting self-driving applications," *Sensors*, vol. 3, p. 7933, 2021.
14. A. Painsky and S. Rosset, "Lossless compression of random forests," *Journal of Computer Science and Technology*, vol. 34, pp. 494–506, 2019.
15. S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks," Advances in Neural Information Processing Systems, pp, pp. 1135–1143, 2016.
16. D. Wang, L. Zhou, X. Zhang, X. Bai, and J. Zhou, "Exploring linear relationship in feature map subspace for convnets compression," 2018.
17. H. Cheng, M. Zhang, and J. Q. Shi, "A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations," 2023.
18. A. Musa, M. Hamada, and M. Hassan, "A theoretical framework towards building a lightweight model for pothole detection using knowledge distillation approach," *SHS Web Conf.*, vol. 139, p. 03002, 2022. [Online]. Available: https://doi.org/10.1051/shsconf/202213903002.
19. T. Dettmers and L. Zettlemoyer, "Sparse networks from scratch: Faster training without losing performance," 2019.
20. M. Hamada, N. B. Odu, and M. Hassan, "A fuzzy-based approach for modelling preferences of users in multi-criteria recommender systems." 2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), 09 2018.
21. M. Aminu, H. Mohamed, H. Mohammed, U. Usman, A. Alex, and A. Mahendran, "A hybrid lightweight deep learning model for edge devices: Combining knowledge distillation, pruning, and quantization," in *Proceedings of the 15th Nature and Biologically Inspired Computing and Pattern Recognition (NaBIC 2023)*. Cham:

Springer Nature Switzerland, 2023, pp. 33–43.
22. C. M. J. Tan and M. Motani, "DropNet: Reducing neural network complexity via iterative pruning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 9356–9366. [Online]. Available: https://proceedings.mlr.press/v119/tan20a.html.
23. J. Liu, S. Tripathi, U. Kurup, and M. Shah, "Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey," 2020.
24. T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, pp. 191–207.
25. N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: Single-shot network pruning based on connection sensitivity," 2019.
26. C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," 2020.
27. Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "Model compression and acceleration for deep neural networks: The principles, progress, and challenges," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 126–136, 2018.
28. *Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration*, 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR. IEEE, 06 2019.
29. W. Chen, Y. Zhang, D. Xie, and S. Pu, "A layer decomposition-recomposition framework for neuron pruning towards accurate lightweight networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 3355–3362.
30. J. Sun, X. Zhang, and J. Wang, "Lightweight bidirectional long short-term memory based on automated model pruning with application to bearing remaining useful life prediction," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105662, 2023.
31. E. Hong, K. Lim, T.-W. Oh, and H. Jang, "Lightweight image steganalysis with block-wise pruning," *Scientific Reports*, vol. 13, no. 1, p. 16148, 2023.
32. K. Xiang, L. Peng, H. Yang, M. Li, Z. Cao, S. Jiang, and G. Qu, "A novel weight pruning strategy for light weight neural networks with application to the diagnosis of skin disease," *Applied Soft Computing*, vol. 111, p. 107707, 2021.
33. J. Carter, S. Mancoridis, and E. Galinkin, "Fast, lightweight iot anomaly detection using feature pruning and pca," in *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*, 2022, pp. 133–138.
34. G. Fang, X. Ma, and X. Wang, "Structural pruning for diffusion models," *Advances in neural information processing systems*, vol. 36, 2024.
35. X. Ma, G. Fang, and X. Wang, "Llm-pruner: On the structural pruning of large language models," *Advances in neural information processing systems*, vol. 36, 2024.
36. Y. Yao, W. Yang, and H. Zhu, "Creating lightweight object detectors with model compression for deployment on edge devices," *arXiv preprint arXiv:1905.01787*, 2019.
37. Y. Zou and C. Liu, "A light-weight object detection method based on knowledge distillation and model pruning for seam tracking system," *Measurement*, vol. 220, p. 113438, 2023.
38. H. Shi, W. Ma, Z. Xu, and P. Lin, "A novel integrated strategy of easy pruning, parameter searching, and re-parameterization for lightweight intelligent lithology identification," *Expert Systems with Applications*, p. 120657, 2023.
39. J. Zhang, P. Wang, Z. Zhao, and F. Su, "Pruned-yolo: Learning efficient object detector using model pruning," in *International Conference on Artificial Neural Networks*. Springer, 2021, pp. 34–45.
40. G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. arXiv, 2015.
41. A. Musa, M. Hassan, M. Hamada, H. A. Kakudi, M. F. I. Amin, and Y. Watanobe, "A lightweight cnn-based pothole detection model for embedded systems using knowledge distillation." in *SoMeT*, 2022, pp. 519–530.
42. A. Musa, F. M. Adam, U. Ibrahim, and A. Y. Zandam, "Learning from small datasets: An efficient deep learning model for covid-19 detection from chest x-ray using dataset distillation technique," 2022, pp. 1–6.
43. C. Yang, X. Yu, Z. An, and Y. Xu, "Categories of response-based, feature-based, and relation-based knowledge distillation," in *Advancements in Knowledge Distillation: Towards New Horizons of Intelligent Systems*. Springer, 2023, pp. 1–32.
44. K. Xu, L. Rui, Y. Li, and L. Gu, "Feature normalized knowledge distillation for image classification," in *European conference on computer vision*. Springer, 2020, pp. 664–680.

45. W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 3967–3976.
46. Y. LeCun, Y. Bengio, and G. Hinton, ""deep learning,"," *Nature*, vol. 521, pp. 436–444, 2015.
47. Y. Liu, W. Zhang, and J. Wang, "Learning from a lightweight teacher for efficient knowledge distillation," *arXiv preprint arXiv:2005.09163*, 2020.
48. W. Chen, L. Gao, X. Li, and W. Shen, "Lightweight convolutional neural network with knowledge distillation for cervical cells classification," *Biomedical Signal Processing and Control*, vol. 71, p. 103177, 2022.
49. J. Kang and J. Gwak, "Ensemble learning of lightweight deep learning models using knowledge distillation for image classification," *Mathematics*, vol. 8, no. 10, p. 1652, 2020.
50. J.-M. Guo, J.-S. Yang, S. Seshathiri, and H.-W. Wu, "A light-weight cnn for object detection with sparse model and knowledge distillation," *Electronics*, vol. 11, no. 4, p. 575, 2022.
51. R. Mishra and H. P. Gupta, "Designing and training of lightweight neural networks on edge devices using early halting in knowledge distillation," *IEEE Transactions on Mobile Computing*, 2023.
52. W. Wang, C. Su, G. Han, and H. Zhang, "A lightweight crack segmentation network based on knowledge distillation," *Journal of Building Engineering*, vol. 76, p. 107200, 2023.
53. Z. Zhang, X. Shu, B. Yu, T. Liu, J. Zhao, Q. Li, and L. Guo, "Distilling knowledge from well-informed soft labels for neural relation extraction," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 9620–9627.
54. J. Yim, D. Joo, J. Bae, and J. Kim, "A gift from knowledge distillation: Fast optimization, network minimization and transfer learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4133–4141.
55. R. Movva*123, J. Lei, S. Longpre, A. Gupta, and C. DuBois2, "Combining compressions for multiplicative size scaling on natural language tasks," Massachusetts Institute of Technology, slongpre, 2022.
56. M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," 2016, pp. 525–542.
57. A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=S1XolQbRW.
58. A. Fan, P. Stock, B. Graham, E. Grave, R. Gribonval, H. Jegou, and A. Joulin, "Training with quantization noise for extreme model compression," 2021.
59. S. Anwar, K. Hwang, and W. Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1131–1135, 2015.
60. V. Vanhoucke, A. Senior, and M. Mao, "Improving the speed of neural networks on cpus," *In Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*, vol. 1, p. 4, 2011.
61. H. Jin, D. Wu, S. Zhang, X. Zou, S. Jin, D. Tao, Q. Liao, and W. Xia, "Design of a Quantization-Based DNN Delta Compression Framework for Model Snapshots and Federated Learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 3, pp. 923–937, Mar. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10018182/.
62. T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, "A comprehensive survey on model compression and acceleration," *Artificial Intelligence Review*, vol. 53, no. 7, pp. 5113–5155, Oct. 2020. [Online]. Available: http://link.springer.com/10.1007/s10462-020-09816-7.
63. M. . Carreira-Perpiñán and Y. Idelbayev, "Model compression as constrained optimization, with application to neural nets. part ii: quantization," 2017.
64. Z. Liu, Y. Wang, K. Han, W. Zhang, S. Ma, and W. Gao, "Post-training quantization for vision transformer," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 092–28 103, 2021.
65. Y. Nahshan, B. Chmiel, C. Baskin, E. Zheltonozhskii, R. Banner, A. M. Bronstein, and A. Mendelson, "Loss aware post-training quantization," *Machine Learning*, vol. 110, no. 11-12, pp. 3245–3262, 2021.
66. I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Accurate post training quantization with small calibration sets," in *International Conference on Machine Learning*. PMLR, 2021, pp. 4466–4475.
67. M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? adaptive rounding for post-training quantization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7197–7206.

68. R. Ding, Z. Liu, T.-W. Chin, D. Marculescu, and R. D. Blanton, "Flightnns: Lightweight quantized deep neural networks for fast and accurate inference," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–6.
69. F. Boutros, N. Damer, and A. Kuijper, "Quantface: Towards lightweight face recognition by synthetic data low-bit quantization," in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022, pp. 855–862.
70. M. Ayazoglu, "Extremely lightweight quantization robust real-time single-image super resolution for mobile devices," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2472–2479.
71. H. Dbouk, H. Sanghvi, M. Mehendale, and N. Shanbhag, "Dbq: A differentiable branch quantizer for lightweight deep neural networks," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*. Springer, 2020, pp. 90–106.
72. J. Wang, "Lightweight and real-time object detection model on edge devices with model quantization." in *Journal of Physics: Conference Series*, vol. 1748, no. 3. IOP Publishing, 2021, p. 032055.
73. Q. Huang and Z. Tang, "High-performance and lightweight ai model for robot vacuum cleaners with low bitwidth strong non-uniform quantization," *AI*, vol. 4, no. 3, pp. 531–550, 2023.
74. C. Gong, Y. Chen, Y. Lu, T. Li, C. Hao, and D. Chen, "Vecq: Minimal loss dnn model compression with vectorized weight quantization," *IEEE Transactions on Computers*, vol. 70, no. 5, pp. 696–710, 2020.
75. A. Aaron, M. Hassan, M. Hamada, and H. Kakudi, "A lightweight deep learning model for identifying weeds in corn and soybean using quantization," *Engineering Proceedings*, vol. 56, no. 1, p. 318, 2023.
76. A. F. Rakib, R. Rahman, A. A. Razi, and A. T. Hasan, "A lightweight quantized cnn model for plant disease recognition," *Arabian Journal for Science and Engineering*, pp. 1–12, 2023.
77. X. Zhao, Y. Wang, X. Cai, C. Liu, and L. Zhang, "Linear symmetric quantization of neural networks for low-precision integer hardware," 2020.
78. T. Sheng, C. Feng, S. Zhuo, X. Zhang, L. Shen, and M. Aleksic, "A quantization-friendly separable convolution for mobilenets," in *2018 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*. IEEE, 2018, pp. 14–18.
79. F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.
80. S. Agrawal, S. Debnath, S. Sagnika, S. Bilgaiyan, and S. Gupta, "Hyperspectral image compression using modified convolutional autoencoder." *International Journal of Computer Information Systems & Industrial Management Applications*, vol. 15, 2023.
81. A. K. Pallikonda and P. S. Varma, "Multi-class classification of Alzheimer's disease stages using squeezenet-based approach for automated diagnosis." *International Journal of Computer Information Systems & Industrial Management Applications*, vol. 15, 2023.
82. F. Peccia, "Weed detection in soybean crops," Kaggle.com, 2018. [Online]. Available: https://www.kaggle.com/datasets/fpeccia/weed-detection-in-soybean-crops.
83. A. Musa, "Pothole detection dataset," 2023. [Online]. Available: https://www.kaggle.com/dsv/6013125.