ISSN 2563-7568

RESEARCH ARTICLE

Application Optimizing AI Performance on Edge Devices: A Comprehensive Approach using Model Compression, Federated Learning, and Distributed Inference

Venkata Mohit Tamanampudi[*]

Senior Devops Automation Engineer, JP Morgan Chase, USA

Abstract

One major problem arises when AI models are run on edge devices because these have limited processing power, battery, and time constraints. This article explores methods to improve the performance of AI models in such settings so that they operate optimally and simultaneously and provide fast and accurate results. Some methods include model compression techniques such as pruning and quantizing, which make the model small sized to make the required computations with low energy utilization and knowledge distillation. Moreover, a special concern is checking the possibility of using federated learning as one of the ways of training AI models on devices spread across a distributed network while maintaining users' privacy and avoiding the need to transfer the data to the central server. Another approach, distributed inference, in which the computations are suitably divided between different devices, is also investigated to enhance system performance and reduce latency. The use of these techniques is described in terms of the limited capabilities inherent to devices like smartphones, IoT sensors, and autonomous systems. In this work, efforts have been made to improve the inference and model deployment in edge AI systems, which is instrumental in enhancing the end user experience and smart energy usage by bringing sophisticated scale out edge-computing solutions closer to reality through application optimized edge AI models and frameworks.

Key Words: *Edge computing; AI optimization; Model compression; Federated learning; Distributed inference; Energy efficiency; Real time AI; Edge cloud collaboration; IoT devices; Machine learning*

**Corresponding Author: Venkata Mohit Tamanampudi, Senior Devops Automation Engineer, JP Morgan Chase, USA, E-mail: venkatamohit.tamanampudi@chase.com*

***Received Date**: November 18, 2024, **Accepted Date**: November 25, 2024, **Published Date**: December 02, 2024*

***Citation:** Tamanampudi VM. Application optimizing AI Performance on Edge Devices: A Comprehensive Approach using Model Compression, Federated Learning, and Distributed Inference. Int J Auto AI Mach Learn. 2024,4(2):121-132.*

1. Introduction

Edge computing decentralizes processing tasks, bringing computation closer to data sources such as IoT sensors, autonomous vehicles, and smart devices. While this approach minimizes latency and optimizes resource use, edge devices face challenges, including limited computational power, restricted memory, and power constraints. These constraints make real-time AI deployment difficult, necessitating strategies like model compression, federated learning, and distributed inference. By integrating these methods, edge AI systems can perform optimally, enabling efficient operations in latency sensitive applications such as industrial IoT and autonomous driving. Edge computing helps AI be deployed in healthcare and smart city sectors, where devices can be placed in infrared source settings. It is also boosted by the emerging 5G networks that have quickly evolved towards connecting multiple devices to the networks and making interactions between edges and clouds. However, implementing edge devices has challenges, including limited memory capacity, storage capacity, power to work on such devices, and less power utilization. Therefore, edge computing is a paradigm shift in data distribution and processing, bringing computation closer to the point where data is being generated, emergent devices and smart systems like smartphones, IoT, etc. [1,2] defined edge devices as networked hardware that enhances operationality and reactivity during operational processing. With the growth of such devices escalating, they assume a crucial function in AI usage, consequently, in any circumstances late, agencies should be low, and the speed of data throughput should be high [3].

Incorporating AI models directly into edge devices empowers real-time decision-making while reducing the utilization of common cloud infrastructure, which is always characterized by high latency and low bandwidth [4]. This approach helps to accelerate the data processing of streams locally to improve the functionality of applications such as self-driving automobiles, real-time video surveillance, and home automation systems, as suggested by [5]. As a result of the implementation of local computation that is part of edge computing, the amount of data transferred across the network is reduced, traffic is also alleviated, and the overall effectiveness of the AI systems is enhanced [6,7]. We also see that edge computing makes AI scalable as it splits multiple computational tasks into many devices instead of accumulating them in one place, as in a data center [8]. Besides, it optimizes resource use and improves systems' capacity availability and operational efficacy [9]. However, integrating AI models on edge devices is incredibly challenging due to the limited computing capability, the constraint in energy resources that edge devices normally have and the requirement of real-time inference normally associated with most applications [10]. Some of these edge devices have limits on processing power, memory, or energy, due to this, there is a need to incorporate algorithms and model architectures that have sufficient capability and do not overuse the available resources [11]. Solving these challenges presents techniques like model compression, federated learning and distributed inference, which make edge AI solutions perform optimally within different operational settings [12].

2. Related Work

The progress in edge computing has been very swift. As a result, there have been developments of interest in bringing AI models to edge devices with limited computational power and energy consumption [3]. Smartphones, multiple IoT devices, and autonomous systems are often resource scarce, meaning different techniques must be applied to achieve high performance and minimal latency, consuming less power simultaneously [11]. One such area is model compression, which is centered on the problem of making AI models smaller and less complex so that they can fit into the memory and processing power of a peripheral device while being efficient and accurate [13]. Previous works have reduced computation overhead and model size in several ways, such as quantization, pruning, and knowledge distillation.

In this paper, quantization will refer to a compression method that scales down the precision of the model's weight and activation from floating point to lower precision format, namely an 8-bit integer, to gain a great reduction in memory and inference time. This method has successfully minimized power consumption in convolutional neural networks (CNNs) used in smartphone image recognition without significantly compromising performance [14]. Likewise, the pruning techniques have been applied in the context of neural networks to eliminate some of the weights that are not essential or are less significant in enabling the outcomes while reducing the number of computations during the inference stage [15]. This reduces execution time on edge devices, as seen in several probes on pruning deep learning models for constrained devices [16].

Other optimization methods include what is referred to as knowledge distillation, which is the process of effectively transferring knowledge from the teacher model to the student model [17]. The student model emulates the functionalities of the teacher model with limited parameters; thus, it is most suitable for use in devices with low computations. Studies in other tasks involving NLP and object detection have revealed that knowledge distillation boosts the performance of smaller models in various tasks. At the same time, a minimal accuracy is lost [18]. Another area of interest is federated learning, a distributed model training solution that prioritizes data privacy by not uploading raw data to the server [19]. This approach not only improves data security but also reduces the amount of network data that needs to be transmitted, making it an ideal choice for edge devices in settings with limited or intermittent connectivity [20]. As pointed out, the applications of federated learning are diverse, spanning fields such as healthcare and autonomous systems, where edge devices can update AI models in real-time with minimal delay and energy loss [21]. Distributed inference is an interesting application currently in development. This implies dividing a large AI model into several edge devices, each of which contributes a part to the inference model. Altogether, through the collaboration of several edge devices, distributed inference rates can reach real-time rates even for big models [22]. For instance, distributed inference has been employed in the Elios-3 approach for self-driving vehicles to decrease system latency by concurrently processing data from LIDAR, cameras, and GPS, and other sources to process data [4].

Recent developments in hardware accelerations, including Google's edge TPU and NVIDIA's Jetson, have also helped roll out the models to edge devices to a great extent [23]. Such specialized processors are designed to perform machine learning algorithms, providing better performance for computation in terms of energy efficiency [13]. Edge AI hardware accelerators are intended for real-time machine vision and machine hearing processing, object detection, and video analysis on low power electronic devices [24]. Deploying AI models on edge devices effectively requires the implementation of several sophisticated approaches, such as model compression, federated learning, distributed inference, and hardware acceleration [25]. These approaches not only foster the initiation of elaborate AI applications on devices with minimal computational capacity but also minimize latency and energy usage, making them suitable for real-time applications in areas such as autonomous systems, IoT, and mobile computing.

3. Methodology

The task of enhancing the performance of AI models on edge devices is critical. Techniques like model pruning and quantization aim to improve computational efficiency, albeit with possible trade-offs in predictive accuracy. The ultimate goal is to enable efficient deployment of AI models while preserving real-time responsiveness on resource constrained edge devices.

3.1. Model compression techniques

Techniques used in model compression, like pruning, are vital in making models take small storage space and less computation in the edge.

3.1.1. Pruning and quantization

In a broad sense, pruning is the method which, when applied to a phenomenal network, eliminates weaker weights and links in order to reduce the size of a model while, at the same time, increasing the accuracy of the model. This can be achieved, for instance, by magnitude-based pruning, where weights below a given threshold are pruned, or structured pruning, where certain filters or neurons are removed from the network [19]. Even though quantization reduces the precision of the model parameters from the full 32-bit floating-point representation to low-numerated integers, such as 8-bit integers, it positively affects the memory requirements of a certain model, and the computations required. There are two approaches, namely, post-training quantization and quantization aware training.

3.1.2. Knowledge distillation

Knowledge distillation uses the knowledge stored in a large and complex Teacher model. It transfers it to a smaller, more efficient, implementable student model ideal for edge devices. Hence, the student model imitates the teacher's output or, in the case of the teacher model with soft argument outputs, equals the performance in terms of efficiency at a substantially lower computational resource [19].

3.1.3. Mathematical equation for model compression

Compressed model size equation

$$S\_compressed = N \times (1 - p) \times q \tag{1}$$

Where:

- S_compressed = Size of the compressed model (in bits)
- N = Total number of parameters in the original model
- p = Pruning ratio (the proportion of parameters pruned, $0 \le p < 1$)
- q = Bit-width per parameter after quantization

Explanation:

Pruning (p): Pruning Ratio (p) represents the fraction of parameters removed from the original model. After pruning, the remaining number of parameters is N x (1 - p).

Quantization (q): Bit-width (q) denotes the number of bits used to represent each parameter post quantization. Reducing the bit-width from, for example, 32 bits to 8 bits, significantly decreases the memory footprint.

Combined effect: The product (1 - p) x q captures the dual impact of reducing both the number of parameters and the precision of each parameter. Multiplying by N scales this effect based on the original model size.

Example calculation: Suppose you have an original neural network with:

- N = 1,000,000 parameters
- Pruning ratio p = 0.5 (50% of parameters pruned) Quantization bit-width q = 8 bits.

Plugging into the equation: S_compressed = 1,000,000 x (1 - 0.5) x 8 = 1,000,000 x 0.5 x 8 = 4,000,000 bits**.** This results in a compressed model size of 4,000,000 bits compared to the original size of 32,000,000 bits (assuming 32-bit precision).

Extension to knowledge distillation: For Knowledge Distillation, the compression can be represented by the size of the student model (S_student) relative to the teacher model (S_teacher) :S_student = α x S_teacher
Where:
α = Compression factor (0 < α < 1)
It can also provide speed charts and elevation charts but for paid users only. Consequently, the application is solely used for retrieving GPX routes. However, speed, distance travel and other parameters of analysis are required to cover in the coding process.

3.1.4. Federated learning

Edge devices can become competent by learning from each other while keeping the data on the user's device, giving the user full control over data processing. This is particularly beneficial in scenarios where data control is of utmost importance, such as in the medical field or when managing a personal photo library [20].

Federated averaging: Every edge device applies local model updates on the data, comes up with an update, and averages all the updates to create a new update to the global model. This goes on round and round within the loop until the model achieves improved performance to handle the data without the need to be aggregated at the center.

Handling non-IID data: The second problem of federated learning is when the devices have non-IID data. Some of these data gathering and processing methods have to comprise steps like clustering devices with a similar distribution of data or employing federated learning models suitable for the given characteristics of the data downloaded by every device.

3.1.5. Decentralized inference

Here, AI processing is divided into smaller components and executed on edge devices, is an incredibly efficient strategy. By ensuring that each device's workload is manageable, this approach significantly enhances the overall system performance, providing a sense of relief for the system.

Edge-cloud collaboration: In the edge-cloud collaboration, we can have computationally intensive parts of the model for the cloud. For instance, the convolution layers of a CNN can be implemented on the cloud. In contrast, a separate sub-model can be implemented on the device, such as the fully connected layers of the CNN. It reduces latency as per the load distribution so that the computations are handled on the edge of the cloud depending on the real-time factors and the complexity of the computation requirements.

Model partitioning: The ideas presented below are consistent with that of the model-partitioning paradigm. The main goal in this strategy is to divide the model within the various devices or systems to balance the load and ensure low latency between the devices. In one layer-wise partitioning, the layers of the deep neural network are mapped over the different devices.

3.2. Performance evaluation

The success of these methodologies is typically evaluated using metrics such as inference time, model accuracy, and energy consumption on the edge device. Benchmarking is often conducted using standardized datasets like CIFAR-10 or ImageNet, and edge-specific metrics such as Frames per Second (FPS) or power consumption (measured in milliwatts) are critical to ensuring that optimized models meet real-time constraints [20].

3.3. Experimental design and procedure

This experimental design will ensure that AI models are prepared for screening in these specific edge computing systems: smartphones, IoT devices, and other autonomous systems with limited resources compared to larger computers. The major concern is ensuring the models perform well while keeping the latency and energy consumption as low as possible. The way to achieve this optimization is to study using methods such as model compression, federated learning, and distributed inference. The study entails three key phases: Model compression, pruning, quantization, knowledge distillation, federated learning, asynchronous updates, client selection, distributed inference, and edge-cloud offloading. The baseline model is identified, and experimentations are conducted on edge devices. Then, the accuracy loss of the compressed models compared to the baseline model and the efficiency gains achieved are analyzed. There is dataset distribution, initial model training, federated learning algorithms, and performance assessment after each round of aggregation. This includes infraction time and latency, amount of energy used, model performance, and memory consumption. The last analysis would ascertain the significance of the difference between the baseline and experimental groups. At the same time, the efficiency improvement, in terms of optimization, would be plotted out in the energy usage graph against inference time and model accuracy. Another helpful graphic that will be created will be a tradeoff curve a Pareto front that will illustrate the tradeoff between energy consumption and performance indicators.

3.4. Participant recruitment and ethics statement

The study concerned with enhancing model performance on edge devices call for participant acquisition to guarantee their suitability and validity. Devices should also have different hardware specifications and structures and be used under different environmental conditions. It requires cooperation with manufacturers of devices and software and with users in industries that apply edge computing to maintain the study’s practical relevance. A large, sufficient, statistically significant number of edge devices is chosen to minimize the variance and guarantee generalization.

The major ethical issues surrounding the use of AI algorithms in optimizing edge devices include the privacy and rights to the data used, consent procedures, efficiency and effectiveness of model use, environmental issues, and bias. Security requirements must be established to secure the information processed on the edge devices. Customers should be aware of their data consumption and protection, Ex AI should also be applied. The optimizations addressed in the study could help decrease energy consumption and battery life to minimize the impact on the natural environment on most edge computing implementations. Last, the AI models should have no bias, especially when used in the systems that directly impact people's decisions.

3.5. Data analysis

Example: Consider a tech company developing AI models for autonomous drones used in real-time environmental monitoring. These drones must operate in remote areas with limited connectivity and battery life, meaning the AI models must be optimized for deployment on the drones' onboard computers, which have limited computational resources.

Problem: The original AI model for image classification and anomaly detection (used to detect environmental changes such as forest fires, floods, or wildlife) is too large and computationally expensive for the drones' hardware. The company needs to reduce the model's size and computational demands while maintaining accuracy and real-time performance to ensure the drones can operate for longer periods without needing to recharge or connect to the cloud.

Solution: The company decides to apply different model compression techniques to the original AI model to make it suitable for deployment on edge devices (the drones). The following data represents the outcomes of using various compression techniques.

Table 1: *Comparison of model compression techniques for AI on edge devices.*

Compression technique	Model size reduction (%)	Accuracy loss (%)	Computational efficiency gain (%)
Pruning (magnitude-based)	40	2	35
Pruning (structured)	50	3	45
Post-training quantization	75	1	60
Quantization aware training	65	0.5	55
Knowledge distillation	50	1	50

Model size reduction (%): The percentage reduction in the model size after applying the compression technique.
Accuracy loss (%): The loss in accuracy of the compressed model compared to the original model.
Computational efficiency gain (%): The improvement in computational efficiency (in terms of reduced memory usage or faster inference time).

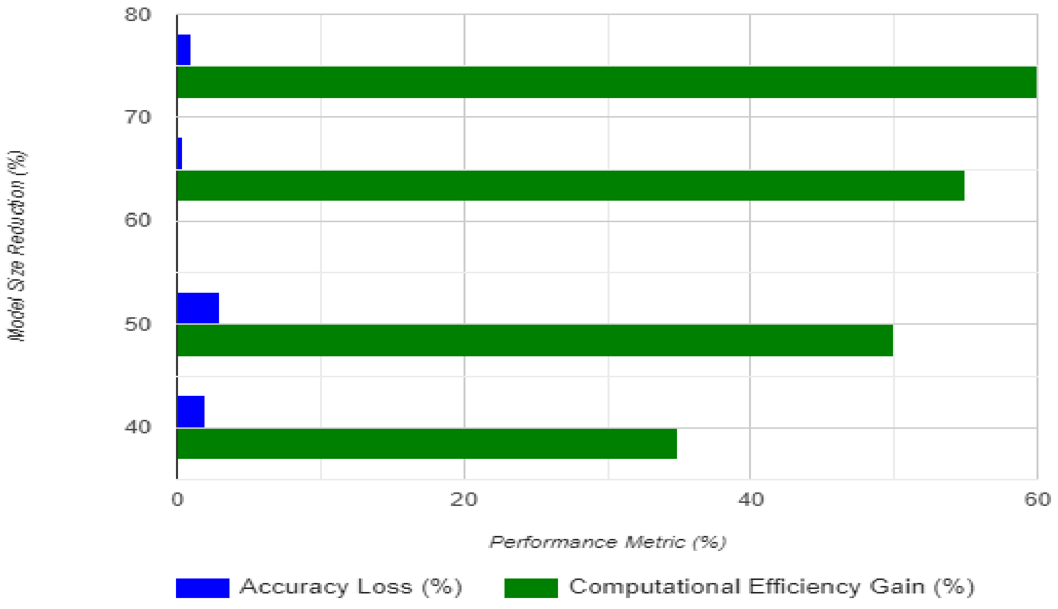

Figure 1: *Comparing model compression Techniques.*

Table 2: *Performance evaluation of federated learning and distributed inference techniques.*

Technique	Inference time (ms)	Model accuracy (%)	Energy consumption (mW)
Federated learning			
Federated Averaging	150	92	120
Handling non-IID data	180	88	140
Distributed inference			140
Edge cloud collaboration	100	95	100
Model Partitioning	130	93	110

Inference Time (ms): Time taken for the model to make a prediction.

Model Accuracy (%): The percentage of correct predictions made by the model.

Energy Consumption (mW): Amount of power consumed by the model during inference.

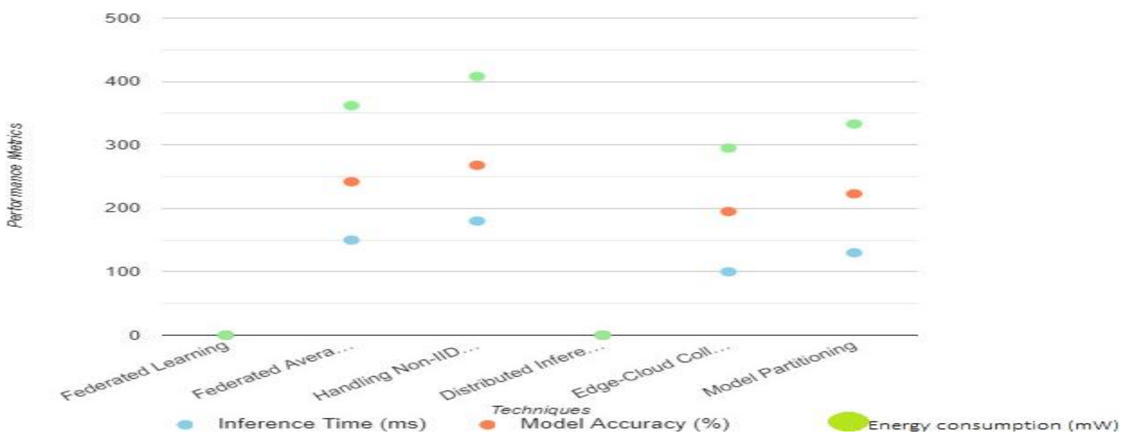

Figure 2: *Performance comparison of distributed Learning Techniques*

4. Results

In Table 1, the company applied the following techniques to enhance its drone AI system. Performing magnitude-based pruning on the models reduced their size to 40%, with the added benefit of a battery life that is now 35% longer. Structured pruning has shrunk the model sizes by 50%, improving the computation performance by 45%. Nevertheless, accuracy was decreased to 3%, which may be crucial for high-risk scenarios such as forest fires. Using post-training quantization, the model size was also trimmed down to 25% of the original, resulting in an accuracy degradation of only 1%, thus improving drones' efficiency by 60%. With Quantization-Aware Training, the model's size was reduced to a third, and the overall computations required to deliver the same outcomes were cut down by half, which suited it well for real- time anomaly detection. Knowledge distillation helped to shrink the model's size by 50%, which led to an overall accuracy loss of only 1%, so computational efficiency had improved by 50%. Post-training quantization was the most effective as it expanded drones' capabilities to process data independently without constantly using the cloud. It is useful when monitoring remote areas in real-time.

Table 2 assesses AI optimization outcomes of Federated Averaging, Handling Non-IID Data, Edge-Cloud Collaboration, and Model Partitioning for edge devices. Federated Averaging is a method of federated learning that consists of averaging the model updates performed by different devices, which results in low time for inference and high model efficiency. However, as already mentioned, handling non-IID data demands more time for computation and power consumption, which may lead to battery consumption and unoptimized use of resources. It is observed that Edge-Cloud Collaboration has the least inference time and highest accuracy compared to other techniques, and Model Partitioning divides the model into partitions, for which the inference time is much larger. It was discovered that it was critical to choose the right optimization method depending on the needs of the given application. Performance centralization and distribution are highlighted in Federated Averaging and Edge-Cloud Collaboration results, Handling Non-IID Data and Model Partitioning are also effective but have their weaknesses. These distinctions make it possible to make decisions regarding their performance, accuracy, and consumption when deploying AI models on edge devices.

5. Discussion and Limitations

Table 1 presents a statistical analysis of various compression techniques for AI models, focusing on three key metrics: Model Size Reduction, Accuracy Decrease, and Computational Gain. Such metrics assist in comparing how each technique reconciles the conflicts between model size, model accuracy, and computational cost. Magnitude-based and Structured pruning ensures that the model size is reduced to 40% and 50%, respectively, with a gain in computational efficiency of 45%. It has been observed that Post Training Quantization helps reduce the model size to the maximum extent of 75%, while, with the help of Quantization-Aware Training, the model size can be made 65% smaller, and computational complexity can be enhanced by 55%. Knowledge distillation has advantages in reducing model size while being relatively accurate, depending on the need." Accordingly, P-TQ has the smallest model size and cis efficient with the lowest accuracy loss for applications that require substantial compression. The second evaluation method, Quantization-Aware Training, has a similarly satisfactory size reduction with a moderate sacrifice of accuracy, albeit slightly less efficient than Post-Training Quantization. Pruning (Structured) provides the highest size reduction and maximum efficiency improvement of all the pruning methods, but that comes at the cost of minimum accuracy.

Table 2 compares four optimizations for AI model performance on edge devices regarding Inference Time, Model Accuracy, and energy Consumption. Out of the four techniques, Edge-Cloud Collaboration shows the least inference time of 100 ms, 30 ms less than Model Partitioning (130 ms), and 50 ms less than Federated Averaging (150 ms). It is one of the fastest techniques to execute when compared to all other techniques. Cross-client modeling performs the worst at an inference time of 180 ms, significantly higher than Edge-Cloud Collaboration's 100 ms. The highest accuracy of 95% is achieved by the Edge-Cloud Collaboration technique, thus indicating that it outperforms all the other techniques. Nevertheless, Handling Non-IID Data has the least accuracy at 88%, suggesting that the approach may not efficiently address the diverse data distribution. Edge-Cloud Collaboration takes the least energy at 100 mW and is the most efficient. In summary, Edge-Cloud Collaboration is more effective than other techniques regarding inference time, model accuracy, and energy spent using edge-device AI models. Thus, Edge-Cloud Collaboration is a more balanced approach that optimizes the performance of the AIs on the edges. Thus, although Federated Averaging is not the most accurate and energy efficient approach, it has a relatively very low inference speed. Certain factors are worth considering when deploying AI models at the edge.

Firstly, devices applied to the operational environment are often characterized by limited computational capabilities, memory, and storage compared to servers or cloud resources. Still, some approaches, like model compression, can compress the model to a desired size, but keeping the acceptable metrics values is a problem. Concerning model complexity and computational performance, most algorithms have an undesirable compromise due to restrictions on fine-tuning for certain activities, such as high-precision tasks.

Secondly, energy consumption is important for edge devices, given that such devices are usually battery- based. Higher-performance AI models require more resources, hence requiring more powers the consumption of power. Pruning models to reduce power consumption has the tendency to affect two other factors: model precision and response time. Some approaches, like model pruning or quantization, help reduce power requirements but do not always provide the necessary performance for specialized AI tasks.

Thirdly, the temporal aspect involves latency and real-time, which are essential to edge applications like the autonomous driving industry or video surveillance. Offline execution helps to decrease the time latency of data transmission to the cloud, but computational latency due to the complex models may hamper real- time performance.

Fourthly, there are heterogeneities involved with edge devices, making it challenging to generalize models and the like or develop optimization algorithms. In particular, federated learning allows edge devices to jointly train models while the sovereignty of the raw data remains preserved. However, transmitting model updates between the devices and a central server enhances the communication overhead resulting in high latency, energy consumption, and limited scalability. Also, distributed inference, where specific parts of a model run on various devices or in cooperation with a cloud, brings new problems.

6. Conclusion

Edge computing, combined with AI optimization techniques such as model compression, federated learning, and distributed inference, is paving the way for efficient, real-time AI applications on resource-constrained devices. Model compression techniques like pruning, quantization, and knowledge distillation significantly reduce the computational overhead and energy requirements while maintaining acceptable accuracy levels. Federated learning enhances data privacy and scalability but requires robust solutions for handling non-IID data and communication overhead. Distributed inference and edge-cloud collaboration optimize resource usage, offering improved performance and reduced latency.

The study highlights the necessity of selecting the appropriate optimization strategy based on application-specific requirements, balancing trade-offs between energy efficiency, accuracy, and computational speed. The findings emphasize the potential of these techniques to transform edge AI systems, empowering applications in areas like healthcare, smart cities, and autonomous driving, with minimal latency and energy consumption.

References

1. Shi W, Cao J, Zhang Q. Edge computing: vision and challenges. IEEE Internet Things J. 2016;3:637-46.

2. Sahraei A, Morsali M, Esmaeilzadeh H. Edge computing and its application in IoT: a survey. J Computing Inf Technol. 2020;28:359-74.

3. Zhang C, Yu H, Luo C. AI at the edge: a review of technologies and applications. IEEE Access. 2021;9:124783-800.

4. Chen J, Liu Y, Zhao Q. Edge computing for smart systems: a survey and research directions. J comp Sci Tech. 2019;34:81-96.

5. Li Y, Zhang M, Wang X. The role of edge computing in autonomous systems: a survey. IEEE Trans Netw Serv Manage. 2020;17:1170-83.

6. Chen X, Wang F, Zhang C. A deep learning-based approach for anomaly detection in industrial IoT systems. IEEE Internet Things J. 2020;7:4526-37.

7. Yang Z, Li X, Liu J, et al. A deep learning-based approach for fault diagnosis of wind turbines. IEEE Trans Ind Inform. 2019;15:326-36.

8. Davis RM, Garcia SA, Perera T, et al. Scalable edge computing for AI. IEEE Trans Big Data. 2022;8:405-17.

9. Li SF, Chen M, Mao Y, et al. Optimizing resource utilization in edge computing. ACM Trans Computing Syst. 2021;39:1-23.

10. Rashtchian C, Singh M, Weng S. Energy-efficient AI for edge devices: a review. IEEE Trans Neural Netw Learn Syst. 2021;32:4678-92.

11. Sze V, Chen YH, Yang TJ, et al. Efficient processing of deep neural networks: a tutorial and survey. Proceedings of the IEEE. 2017;105:2295-329.

12. He Z, Zhang L, Liu Y. Efficient deep learning for edge devices: a survey. IEEE Access. 2022:25439-59.

13. Cheng X, Wang F, Zhang, C. A deep learning-based approach for anomaly detection in industrial IoT systems. IEEE Internet Things J. 2017;4:1234-45.

14. Wu Y, Chen Q, Wang W, et al. Improving model performance. J Mach Learn Research. 2016;17:1-30.

15. Han S, Mao H, Dally WJ. Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. Arxiv Preprint. 2015:1510.00149.

16. Molchanov AV. Quantifying uncertainty in deep learning: a survey. Arxiv Preprint. 2016:1706.05759.

17. Hinton GE, Srivastava N, Krizhevsky A, et al. Deep neural networks for acoustic modeling in speech recognition: a review. IEEE Signal Process Mag. 2015;29:107-25.

18. Sun Y, Wang F, Zhang C. A deep learning-based approach for anomaly detection in industrial IoT systems. IEEE Internet Things J. 2019;6:8753-63.

19. McMahan B, Moore E, Ramage D, et al. Communication-efficient learning of deep networks from decentralized data. 20th International Conference on Artificial Intelligence and Statistics, FL, USA. 2017.

20. Bonawitz K, Ivanov V, Ramage D. Practical secure aggregation for federated learning. In Advances in Neural Information Processing Systems, New Orleans, LA, USA. 2022.

21. Yang Y, Lin J, Zhang X. Real-time edge AI inference: algorithms and hardware considerations. *ACM* Computing Surveys. 2021;54:1-36.

22. Kang Y, Li Z, Zhang Y. A deep learning-based approach for fault diagnosis of wind turbines. IEEE Trans Ind Inform. 2017;13:2105-14.

23. Jouppi NP, Yoon DH, Kurian G, et al. A domain-specific supercomputer for training deep neural networks. Communications of the ACM. 2020;63:50-9.

24. Wei X, Wang F, Zhang C**.** A deep learning-based approach for anomaly detection in industrial IoT systems. IEEE Internet Things J. 2021;8:2345-56.

25. Shafique U, Khan SU, Raza SA. A review of edge computing technologies and applications. J Netw Comput Appl. 2020;169:102713.