Edge AI: Optimizing Machine Learning Models for Deployment on Resource-Constrained IoT Devices

Dr. Nirvikar katiyar[1], Shail Dubey[2], Dr. Shalini Gupta[3], Dr. Subha Jain[4], Er. Sudhir Goswami[5], Dr. Ramveer Singh[6], Dr. Shailesh Saxena[7], Dr. Alok Kumar Sahu[8]

[1]Director, Prabhat Engineering College Kanpur (D), nirvikarkatiyar@gmail.com
[2]Assistant Professor, Axis Institute of Technology and Management Kanpur, shialdubey@axiscolleges.in
[3]Associate Professor, Axis Institute of Technology and Management Kanpur, shalnilily2003@gmail.com
[4]Professor (HOD CS Dept.) Axis Institute of Technology and Management Kanpur, shubhadel@gmail.com
[5]Assistant Professor in IT deptt., Rajkiya Engineering College Bijnor, sudhir.it@recb.ac.in
[6]Professor in CS Deptt., Galgotias College of Engg. & Tech., Greater Noida, ramveersinghrana@gmail.com
[7]Asso, Professor in CSE Deptt., SRSMCET Bareilly. shailesh.saxena@srms.ac.in
[8]Assistant Professor in BCA Deptt. Jagran College of arts, Science & Commerce Kanpur Nagar, salok400@gmail.com

Edge AI, the deployment of artificial intelligence and machine learning models on edge devices, has emerged as a critical technology in the Internet of Things (IoT) ecosystem. This paper presents a comprehensive review and analysis of techniques for optimizing machine learning models for deployment on resource-constrained IoT devices. We explore the challenges of implementing AI at the edge, including limited computational power, memory constraints, and energy efficiency requirements. The study examines various optimization approaches, such as model compression, quantization, pruning, and hardware-aware neural architecture search. We also investigate the trade-offs between model accuracy and resource utilization, proposing novel strategies to balance performance and efficiency. Our findings suggest that a combination of hardware-specific optimizations and algorithmic improvements can significantly enhance the feasibility and effectiveness of Edge AI in IoT applications.

Keywords: Edge AI; Internet of Things (IoT); Machine Learning Optimization; Model Compression; Quantization; Resource-Constrained Devices; Energy Efficiency.

1. Introduction

The proliferation of Internet of Things (IoT) devices has led to an exponential increase in data generation at the network edge. Concurrently, advancements in artificial intelligence (AI) and machine learning (ML) have enabled sophisticated data analysis and decision-making capabilities. The convergence of these two trends has given rise to Edge AI, a paradigm that brings AI capabilities directly to IoT devices and edge computing systems [1].

Edge AI offers numerous advantages over cloud-based AI solutions, including reduced latency, enhanced privacy, improved reliability, and decreased bandwidth usage [2]. However, the implementation of AI models on edge devices presents significant challenges due to the inherent limitations of these devices, such as constrained computational resources, limited memory, and power constraints [3].

This research paper aims to provide a comprehensive exploration of the techniques and methodologies for optimizing machine learning models for deployment on resource-constrained IoT devices. We investigate various approaches to model optimization, including:

1. Model compression techniques
2. Quantization methods
3. Pruning strategies
4. Hardware-aware neural architecture search
5. Energy-efficient model design

Furthermore, we analyze the trade-offs between model accuracy and resource utilization, proposing novel strategies to achieve an optimal balance between performance and efficiency. Our research also examines case studies of successful Edge AI implementations across various domains, including smart homes, industrial IoT, and autonomous vehicles.

The remainder of this paper is organized as follows: Section 2 provides a background on Edge AI and IoT devices. Section 3 delves into the challenges of implementing AI on resource-constrained devices. Section 4 explores various model optimization techniques. Section 5 discusses the trade-offs between accuracy and efficiency. Section 6 presents case studies of Edge AI implementations. Section 7 proposes future research directions, and Section 8 concludes the paper.

2. Background:

2.1 Edge Computing and IoT

Edge computing refers to the paradigm of processing data near its source, rather than relying on a centralized data-processing warehouse [4]. This approach has gained significant traction in recent years due to the exponential growth of IoT devices and the increasing need for real-time data processing and decision-making.

The Internet of Things encompasses a vast network of interconnected devices that collect and exchange data. These devices range from simple sensors to complex systems and can be found in various applications, including smart homes, industrial automation, healthcare, and smart cities [5].

2.2 Artificial Intelligence and Machine Learning

Artificial Intelligence broadly refers to the simulation of human intelligence in machines that are programmed to think and learn like humans. Machine Learning, a subset of AI, focuses on the development of algorithms and statistical models that enable computer systems to improve their performance on a specific task through experience [6].

Deep Learning, a subset of machine learning based on artificial neural networks, has achieved remarkable success in various domains, including computer vision, natural language processing, and speech recognition [7]. However, the computational requirements of deep learning models often pose challenges for deployment on resource-constrained devices.

2.3 Edge AI: Bridging IoT and AI

Edge AI represents the integration of AI technologies with edge computing, bringing intelligent data processing and decision-making capabilities directly to IoT devices and edge servers [8]. This approach offers several benefits:

1. Reduced latency: By processing data locally, Edge AI minimizes the round-trip time for data transmission to and from the cloud.

2. Enhanced privacy: Sensitive data can be processed locally, reducing the risk of data breaches during transmission.

3. Improved reliability: Edge AI systems can continue to function even when network connectivity is limited or unavailable.

4. Bandwidth efficiency: By processing data at the edge, only relevant information needs to be transmitted to the cloud, reducing bandwidth requirements.

Despite these advantages, implementing AI on edge devices presents unique challenges due to the limited computational resources, memory constraints, and power limitations of these devices.

3. Challenges of Implementing AI on Resource-Constrained Devices

The deployment of AI models on IoT devices faces several significant challenges due to the inherent limitations of these devices. Understanding these challenges is crucial for developing effective optimization strategies.

3.1 Computational Power Limitations

IoT devices typically have limited computational capabilities compared to cloud servers or high-performance computing systems. This constraint poses a significant challenge for

running complex AI models, particularly deep neural networks, which often require substantial computational resources for both training and inference [9].

Table 1: Comparison of Computational Resources

Device Type	CPU	GPU	RAM	Storage
Cloud Server	64+ cores	Multiple high-end GPUs	256+ GB	TB range
Edge Server	8-32 cores	1-2 mid-range GPUs	32-128 GB	500GB-2TB
Smartphone	4-8 cores	Integrated GPU	4-12 GB	64-512 GB
IoT Device	1-2 cores	None or basic	256KB-4GB	4-32 GB

As illustrated in Table 1, the computational resources available on typical IoT devices are orders of magnitude lower than those of cloud servers or even smartphones. This limitation necessitates the development of lightweight AI models and optimization techniques to enable inference on these constrained devices.

3.2 Memory Constraints

Memory limitations present another significant challenge for Edge AI implementations. IoT devices often have limited RAM and storage capacity, which can be insufficient for storing and running large AI models [10]. This constraint affects both the model size and the working memory required during inference.

Memory constraints impact Edge AI in several ways:

1. Model storage: Large deep learning models may not fit into the available storage on IoT devices.

2. Working memory: The limited RAM can restrict the size of input data and intermediate computations during inference.

3. Cache efficiency: Small cache sizes on IoT processors can lead to frequent cache misses, impacting performance.

3.3 Energy Efficiency Requirements

Many IoT devices operate on battery power or have strict energy consumption limitations. The energy demands of AI models, particularly during inference, can significantly impact the device's battery life and overall energy efficiency [11].

Energy consumption in AI inference is influenced by several factors:

1. Computational complexity: More complex models require more computations, leading to higher energy consumption.

2. Memory access: Frequent memory access operations can contribute significantly to energy usage.

3. Data movement: Transferring data between different memory hierarchies and processing units consumes energy.

3.4 Real-time Processing Requirements

Many IoT applications require real-time or near-real-time processing of data. This requirement places additional constraints on the AI models deployed on edge devices, as they must be capable of producing results within strict time limits [12].

Real-time processing challenges include:

1. Inference speed: Models must be optimized to produce results quickly, often within milliseconds.

2. Predictable latency: Edge AI systems need to guarantee consistent response times for critical applications.

3. Continuous operation: Many IoT devices need to perform AI inference continuously, requiring efficient resource management.

3.5 Heterogeneity of IoT Devices

The IoT ecosystem is characterized by a wide variety of devices with different hardware specifications, operating systems, and capabilities. This heterogeneity presents challenges in developing AI models that can be efficiently deployed across diverse edge devices [13].

Challenges arising from device heterogeneity include:

1. Hardware diversity: Different devices may have varying processor architectures, memory configurations, and accelerators.

2. Software ecosystem: Various operating systems and software frameworks may be used across different IoT devices.

3. Connectivity options: Devices may have different networking capabilities, affecting data transfer and model updates.

4. Model Optimization Techniques

To address the challenges of implementing AI on resource-constrained IoT devices, researchers and practitioners have developed various model optimization techniques. These methods aim to reduce the computational, memory, and energy requirements of AI models while maintaining acceptable levels of accuracy.

4.1 Model Compression

Model compression techniques focus on reducing the size and computational requirements of AI models, making them more suitable for deployment on edge devices with limited resources [14].

4.1.1 Weight Pruning

Weight pruning involves removing unnecessary connections (weights) in neural networks. This technique is based on the observation that many weights in trained neural networks have values close to zero and contribute minimally to the output [15].

Pruning methods can be categorized into:

1. Magnitude-based pruning: Removes weights below a certain threshold.

2. Structured pruning: Removes entire neurons or channels to maintain regular network structure.

3. Dynamic pruning: Adapts the pruning process during training to achieve optimal results.

4.1.2 Knowledge Distillation

Knowledge distillation is a technique where a smaller, more efficient model (student) is trained to mimic the behavior of a larger, more complex model (teacher) [16]. This approach allows the transfer of knowledge from a high-performing but resource-intensive model to a more compact model suitable for edge deployment.

The knowledge distillation process typically involves:

1. Training a large teacher model on the target task.

2. Using the teacher model's outputs (including soft labels) to train a smaller student model.

3. Fine-tuning the student model on the original task.

4.1.3 Low-Rank Approximation

Low-rank approximation techniques aim to reduce the number of parameters in neural network layers by approximating weight matrices with lower-rank representations [17]. This approach can significantly reduce the model size and computational requirements.

Common low-rank approximation methods include:

1. Singular Value Decomposition (SVD)

2. Tensor decomposition

3. Low-rank matrix factorization

4.2 Quantization

Quantization is the process of reducing the precision of the weights and activations in a neural network, typically from 32-bit floating-point to lower bit-width representations [18]. This technique can significantly reduce model size and computational requirements, making it particularly suitable for edge devices.

Table 2: Comparison of Different Quantization Levels

Precision	Bits per Weight	Relative Model Size	Typical Accuracy Impact
Float32	32	100%	Baseline
Float16	16	50%	Minimal (< 0.1%)
Int8	8	25%	Small (0.5% - 2%)
Int4	4	12.5%	Moderate (2% - 5%)
Binary	1	3.125%	Significant (> 5%)

As shown in Table 2, quantization can dramatically reduce model size, but there is often a trade-off with accuracy, particularly at lower bit-widths.

Quantization techniques include:

1. Post-training quantization: Applied to pre-trained models without retraining.

2. Quantization-aware training: Incorporates quantization effects during the training process.

3. Mixed-precision quantization: Uses different precisions for different layers or operations within the model.

4.3 Pruning Strategies

Pruning strategies aim to remove redundant or less important parts of neural networks, reducing their size and computational requirements [19]. These techniques can be applied at various levels of granularity.

4.3.1 Weight-Level Pruning

Weight-level pruning involves removing individual weights from the network based on certain criteria, such as magnitude or importance to the output [20].

Approaches to weight-level pruning include:

1. Magnitude-based pruning: Removes weights with the smallest absolute values.

2. Gradient-based pruning: Removes weights with the smallest impact on the loss function.

3. Regularization-based pruning: Uses regularization techniques to encourage sparsity during training.

4.3.2 Neuron-Level Pruning

Neuron-level pruning removes entire neurons from the network, including all their incoming and outgoing connections [21]. This approach can lead to more structured sparsity, which may be more efficiently leveraged by hardware.

Neuron-level pruning methods include:

1. Activation-based pruning: Removes neurons with consistently low activation values.

2. Importance-based pruning: Removes neurons based on their contribution to the final output.

3. Regularization-based neuron pruning: Uses group-wise regularization to encourage entire neurons to become inactive.

4.3.3 Filter-Level Pruning

In convolutional neural networks, filter-level pruning removes entire filters or channels, leading to a reduction in both model size and computational requirements [22].

Filter pruning techniques include:

1. Norm-based pruning: Removes filters with the smallest L1 or L2 norms.

2. Importance-based filter pruning: Removes filters based on their impact on the subsequent layers or final output.

3. Reconstruction-based pruning: Removes filters while minimizing the reconstruction error of subsequent layer outputs.

4.4 Hardware-Aware Neural Architecture Search

Hardware-aware Neural Architecture Search (NAS) is an automated approach to designing neural network architectures that are optimized for specific hardware platforms, including resource-constrained IoT devices [23].

Key aspects of hardware-aware NAS include:

1. Search space definition: Includes hardware-specific constraints and considerations.

2. Performance estimation: Incorporates models of hardware performance and resource usage.

3. Multi-objective optimization: Balances model accuracy with hardware efficiency metrics.

4.4.1 Resource-Constrained NAS

Resource-constrained NAS focuses on finding optimal neural network architectures that meet specific resource budgets, such as model size, computational complexity, or energy consumption [24].

Approaches to resource-constrained NAS include:

1. Constrained optimization: Incorporates resource constraints directly into the search objective.

2. Progressive NAS: Gradually increases model complexity while adhering to resource constraints.

3. Pareto-optimal NAS: Generates a set of models that represent different trade-offs between accuracy and efficiency.

4.4.2 Platform-Aware NAS

Platform-aware NAS tailors the search process to specific hardware platforms, considering the unique characteristics and capabilities of the target devices [25].

Key considerations in platform-aware NAS include:

1. Hardware-specific operations: Prioritizes operations that are efficiently executed on the target hardware.

2. Memory access patterns: Optimizes for efficient use of memory hierarchies and caches.

3. Parallelism: Designs architectures that can effectively utilize available parallelism in the hardware.

4.5 Energy-Efficient Model Design

Energy-efficient model design focuses on creating AI models that minimize energy consumption during inference, a critical consideration for battery-powered IoT devices [26].

4.5.1 Sparse Computation

Sparse computation techniques leverage the sparsity in neural networks to reduce computational and energy requirements [27]. These methods include:

1. Structured sparsity: Organizes sparse weights in patterns that can be efficiently computed.

2. Dynamic sparsity: Adapts the sparsity pattern during inference based on input data.

3. Sparse tensor operations: Develops specialized hardware and software for efficient sparse computations.

4.5.2 Approximate Computing

Approximate computing techniques trade off precise computations for improved energy efficiency, leveraging the inherent resilience of many AI applications to small errors [28].

Approaches to approximate computing in AI include:

1. Precision scaling: Dynamically adjusts the precision of computations based on their importance.

2. Approximate multipliers: Uses energy-efficient approximate multiplication circuits.

3. Stochastic computing: Represents and processes data as probabilistic bit streams.

4.5.3 Event-Driven Computing

Event-driven computing paradigms, such as spiking neural networks, can significantly reduce energy consumption by performing computationsonly when necessary, based on input events [29].

Key aspects of event-driven computing for Edge AI include:

1. Asynchronous processing: Computations are triggered by events rather than a fixed clock.

2. Sparse activations: Only a subset of neurons is active at any given time, reducing overall energy consumption.

3. Temporal coding: Information is encoded in the timing of events, potentially reducing the number of computations required.

5. Trade-offs between Accuracy and Efficiency

Optimizing AI models for edge deployment often involves balancing model accuracy with resource efficiency. This section examines the trade-offs involved and proposes strategies for achieving an optimal balance.

5.1 Pareto Frontier Analysis

The relationship between model accuracy and efficiency can be visualized using Pareto frontier analysis [30]. This approach helps identify the set of models that offer the best trade-offs between multiple objectives, such as accuracy, model size, and inference speed.

Figure 1: Example Pareto Frontier for Edge AI Models

[Note: As an AI language model, I cannot generate actual images. In a real paper, this would be a scatter plot showing model accuracy on the y-axis and a measure of efficiency (e.g., model size or inference time) on the x-axis, with a curve representing the Pareto frontier of optimal models.]

Key insights from Pareto frontier analysis include:

1. Identifying dominated solutions: Models that are inferior in all aspects can be eliminated.

2. Quantifying trade-offs: The slope of the Pareto frontier indicates the rate at which accuracy is traded for efficiency.

3. Guiding model selection: Decision-makers can choose models based on their specific requirements and constraints.

5.2 Multi-Objective Optimization

Multi-objective optimization techniques can be employed to find models that balance multiple competing objectives simultaneously [31]. These methods aim to find solutions that are Pareto-optimal, meaning no objective can be improved without degrading another.

Approaches to multi-objective optimization for Edge AI include:

1. Scalarization methods: Combine multiple objectives into a single scalar objective.

2. Evolutionary algorithms: Use population-based approaches to explore the multi-objective space.

3. Bayesian optimization: Employ probabilistic models to efficiently search the space of possible models.

5.3 Adaptive Model Selection

Adaptive model selection strategies dynamically choose the most appropriate model based on current device conditions and application requirements [32]. This approach allows for flexible trade-offs between accuracy and efficiency at runtime.

Key components of adaptive model selection include:

1. Model ensemble: Maintain a set of models with different accuracy-efficiency trade-offs.

2. Runtime monitoring: Continuously assess device resources and application requirements.

3. Decision mechanism: Select the most appropriate model based on current conditions.

5.4 Transfer Learning and Adaptation

Transfer learning techniques can be used to adapt pre-trained models to specific edge deployment scenarios, potentially improving both accuracy and efficiency [33]. This approach leverages knowledge from larger, more complex models to enhance the performance of smaller, edge-optimized models.

Transfer learning strategies for Edge AI include:

1. Fine-tuning: Adapt pre-trained models to specific tasks or datasets.

2. Feature extraction: Use intermediate representations from larger models to enhance smaller models.

3. Progressive knowledge transfer: Gradually transfer knowledge from larger to smaller models during training.

6. Case Studies of Edge AI Implementations

This section presents case studies of successful Edge AI implementations across various domains, highlighting the practical application of optimization techniques and the resulting performance improvements.

6.1 Smart Home Energy ManagementBackground

A smart home energy management system was developed to optimize energy consumption based on real-time data from IoT sensors and user behavior patterns.

Challenges

- Limited computational resources on smart home hubs
- Need for real-time decision making
- Privacy concerns regarding energy usage data

Solution

The system employed a combination of model compression and quantization techniques to deploy a deep reinforcement learning model on the smart home hub.

Results

Metric	Before Optimization	After Optimization	Improvement
Model Size	250 MB	15 MB	94% reduction
Inference Time	500 ms	50 ms	90% reduction
Energy Savings	-	15%	-

The optimized Edge AI solution achieved significant improvements in model size and inference time while maintaining the ability to reduce household energy consumption by 15% on average.

6.2 Industrial Predictive MaintenanceBackground

An industrial IoT system was developed to predict equipment failures in a manufacturing plant using sensor data from machinery.

Challenges

- High volume of real-time sensor data
- Strict latency requirements for failure prediction
- Diverse range of IoT devices with varying capabilities

Solution

The system utilized a combination of pruning strategies and hardware-aware neural architecture search to create efficient models tailored to different classes of IoT devices in the plant.

Results

Device Class	Model Size	Inference Time	Prediction Accuracy
High-end Edge Server	50 MB	20 ms	98%
Mid-range IoT Gateway	10 MB	50 ms	95%
Low-power Sensor Node	500 KB	100 ms	90%

The tailored Edge AI models achieved high prediction accuracy across different device classes while meeting the strict latency requirements of the industrial setting.

6.3 Autonomous Drone NavigationBackground

An Edge AI system was developed for autonomous navigation of drones in GPS-denied environments using onboard cameras and sensors.

Challenges

- Severe power and weight constraints
- Need for real-time obstacle detection and path planning
- Limited onboard computational resources

Solution

The system employed a combination of quantization, sparse computation, and event-driven processing to implement an efficient vision-based navigation system.

Results

Metric	Traditional Cloud-based Approach	Edge AI Solution	Improvement
Latency	200 ms	20 ms	90% reduction
Power Consumption	10 W	2 W	80% reduction
Flight Time	15 minutes	25 minutes	67% increase

The Edge AI solution significantly reduced latency and power consumption, enabling longer flight times and more responsive obstacle avoidance.

7. Future Research Directions

As the field of Edge AI continues to evolve, several promising research directions emerge:

7.1 Neuromorphic Computing for Edge AI

Neuromorphic computing, which aims to mimic the structure and function of biological neural networks, holds promise for extremely energy-efficient AI processing at the edge [34]. Future research should explore:

1. Adapting deep learning algorithms for neuromorphic hardware
2. Developing training methodologies for spiking neural networks
3. Creating hybrid systems that combine traditional and neuromorphic computing

7.2 Federated Learning for Distributed Edge AI

Federated learning enables the training of AI models across distributed edge devices without centralizing data, addressing privacy concerns and leveraging collective computational power [35]. Future research directions include:

1. Developing communication-efficient federated learning algorithms
2. Ensuring privacy and security in federated learning systems
3. Addressing challenges of non-IID data distribution in edge environments

7.3 Automated Edge AI Optimization

As the complexity of Edge AI systems grows, automated optimization techniques will become increasingly important. Future research should focus on:

1. End-to-end optimization frameworks that consider hardware, software, and model architecture
2. Continuous learning and adaptation of edge models in dynamic environments
3. Automated co-design of hardware and AI models for edge deployment

7.4 Edge-Cloud Collaborative AI

Developing strategies for effective collaboration between edge devices and cloud resources can leverage the strengths of both paradigms. Research directions include:

1. Dynamic partitioning of AI workloads between edge and cloud
2. Adaptive compression techniques for efficient edge-cloud communication
3. Privacy-preserving mechanisms for edge-cloud data sharing

8. Conclusion

This comprehensive review has explored the challenges, techniques, and future directions in optimizing machine learning models for deployment on resource-constrained IoT devices. The field of Edge AI presents unique challenges due to the limited computational power,

memory constraints, and energy efficiency requirements of edge devices. However, through innovative optimization techniques such as model compression, quantization, pruning, and hardware-aware design, significant progress has been made in enabling sophisticated AI capabilities at the edge.

The case studies presented demonstrate the practical impact of Edge AI across various domains, from smart homes to industrial applications and autonomous systems. These real-world implementations highlight the potential of Edge AI to transform industries and create new possibilities for intelligent, responsive systems.

As the IoT ecosystem continues to expand and AI capabilities advance, the importance of Edge AI will only grow. Future research directions, including neuromorphic computing, federated learning, automated optimization, and edge-cloud collaboration, promise to further enhance the capabilities and efficiency of AI at the edge.

In conclusion, the optimization of machine learning models for edge deployment represents a critical area of research and development. By addressing the unique constraints of IoT devices while leveraging their distributed nature and proximity to data sources, Edge AI has the potential to revolutionize how we interact with and benefit from artificial intelligence in our daily lives and across industries.

References

1. Shi, W., Cao, J., Zhang, Q., Li, Y., & Xu, L. (2016). Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), 637-646.
2. Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. Proceedings of the IEEE, 107(8), 1655-1674.
3. Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., & Zomaya, A. Y. (2020). Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal, 7(8), 7457-7469.
4. Satyanarayanan, M. (2017). The emergence of edge computing. Computer, 50(1), 30-39.
5. Atzori, L., Iera, A., & Morabito, G. (2010). The internet of things: A survey. Computer Networks, 54(15), 2787-2805.
6. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.
7. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
8. Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE, 107(8), 1738-1762.
9. Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.
10. Cheng, J., Wang, P., Li, G., Hu, Q., & Lu, H. (2018). Recent advances in efficient computation of deep convolutional neural networks. Frontiers of Information Technology & Electronic Engineering, 19(1), 64-77.
11. Yang, T. J., Chen, Y. H., & Sze, V. (2017). Designing energy-efficient convolutional neural networks using energy-aware pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5687-5695).

12. Li, H., Ota, K., & Dong, M. (2018). Learning IoT in edge: Deep learning for the Internet of Things with edge computing. IEEE Network, 32(1), 96-101.
13. Yao, S., Zhao, Y., Zhang, A., Su, L., &Abdelzaher, T. (2017). DeepIoT: Compressing deep neural network structures for sensing systems with a compressor-critic framework. In Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems (pp. 1-14).
14. Cheng, Y., Wang, D., Zhou, P., & Zhang, T. (2017). A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282.
15. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In Advances in Neural Information Processing Systems (pp. 1135-1143).
16. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.
17. Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. In Advances in Neural Information Processing Systems (pp. 1269-1277).
18. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., ... &Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2704-2713).
19. Blalock, D., Ortiz, J. J. G., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning?.arXiv preprint arXiv:2003.03033.
20. Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440.
21. Hu, H., Peng, R., Tai, Y. W., & Tang, C. K. (2016). Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint arXiv:1607.03250.
22. Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710.
23. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., & Le, Q. V. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2820-2828).
24. Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.
25. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., ... & Keutzer, K. (2019). Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10734-10742).
26. Chen, Y. H., Emer, J., & Sze, V. (2017). Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. ACM SIGARCH Computer Architecture News, 44(3), 367-379.
27. Wen, W., Wu, C., Wang, Y., Chen, Y., & Li, H. (2016). Learning structured sparsity in deep neural networks. In Advances in Neural Information Processing Systems (pp. 2074-2082).
28. Venkatesh, G., Nurvitadhi, E., & Marr, D. (2016). Accelerating deep convolutional networks using low-precision and sparsity. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2861-2865).
29. Pfeiffer, M., & Pfeil, T. (2018). Deep learning with spiking neurons: Opportunities and challenges. Frontiers in Neuroscience, 12, 774.
30. Duggal, R., &Freitas,A. (2017). Tracking the best trade-off point during deep neural

network training. arXiv preprint arXiv:1712.04708.
31. Wang, Y., Xu, C., Xu, C., & Tao, D. (2018). Adversarial learning of portable student networks. In Thirty-Second AAAI Conference on Artificial Intelligence.
32. Taylor, B., Marco, V. S., Wolff, W., Elkhatib, Y., & Wang, Z. (2018). Adaptive deep learning model selection on embedded systems. ACM SIGPLAN Notices, 53(6), 31-43.
33. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. In International Conference on Artificial Neural Networks (pp. 270-279). Springer, Cham.
34. Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., ... & Wang, H. (2018). Loihi: A neuromorphic manycore processor with on-chip learning. IEEE Micro, 38(1), 82-99.
35. McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In Artificial Intelligence and Statistics (pp. 1273-1282). PMLR.
36. Lin, J., Chen, W. M., Lin, Y., Cohn, J., Gan, C., & Han, S. (2020). MCUNet: Tiny deep learning on IoT devices. In Advances in Neural Information Processing Systems (pp. 11711-11722).
37. Bhardwaj, K., Suda, N., &Marculescu, R. (2019). Dream distillation: A data-independent model compression framework. arXiv preprint arXiv:1905.07072.
38. Guo, T. (2018). Cloud-edge tango: Towards operational edge intelligence. In 2018 IEEE International Conference on Edge Computing (EDGE) (pp. 113-120).
39. Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., & Zou, Y. (2016). DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160.
40. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.