14 January 2025, Preprint v1 · CC-BY 4.0 Qeios PREPRINT

Research Article

Optimizing Edge AI: A Comprehensive Survey on Data, Model, and System Strategies

Xubin Wang[1,2,3], **Weijia Jia**[2,3]

1. Hong Kong Baptist University, Hong Kong; 2. BNU-HKBU United International College; 3. Beijing Normal University, China

The emergence of 5G and edge computing hardware has brought about a significant shift in artificial intelligence, with edge AI becoming a crucial technology for enabling intelligent applications. With the growing amount of data generated and stored on edge devices, deploying AI models for local processing and inference has become increasingly necessary. However, deploying state-of-the-art AI models on resource-constrained edge devices faces significant challenges that must be addressed. This paper presents an optimization triad for efficient and reliable edge AI deployment, including data, model, and system optimization. First, we discuss optimizing data through data cleaning, compression, and augmentation to make it more suitable for edge deployment. Second, we explore model design and compression methods at the model level, such as pruning, quantization, and knowledge distillation. Finally, we introduce system optimization techniques like framework support and hardware acceleration to accelerate edge AI workflows. Based on an in-depth analysis of various application scenarios and deployment challenges of edge AI, this paper proposes an optimization paradigm based on the data-model-system triad to enable a whole set of solutions to effectively transfer ML models, which are initially trained in the cloud, to various edge devices for supporting multiple scenarios.

Corresponding author: Weijia Jia, jiawj@bnu.edu.cn

I. Introduction

From AlphaGo to ChatGPT, the rapid progress of AI technology in recent years makes us marvel at its huge potential. Simultaneously, media coverage of the threat of artificial intelligence challenging human intelligence is increasing. In fact, while the potential of AI has been shown, there are still some gaps between AI applications and the real world. To achieve better performance, larger datasets and more parameters are often used to train AI models, which usually requires a large amount of training consumption and also makes the model complex. A typical example is the large-scale language model GPT-3, which has 175 billion parameters and requires about 800GB of storage[1]. Unfortunately, the high cost of training makes it out of reach for the average person, and only exists in labs like OpenAI with a deep accumulation. Meanwhile, these cumbersome models are difficult to deploy to small, resource-constrained models such as edge devices, which are widely distributed in life. Therefore, there is an urgent need to design efficient AI models and frameworks for use on small devices.

With the development of communication technologies such as 5G and the Internet of Things, the edge of the network can reach a variety of devices in multiple settings, including schools, hospitals, factories, shops and homes[2]. These widely distributed edge devices generate huge amounts of data. According to Gartner, by 2025, about 75% of enterprise-generated data will not come from traditional data centers or the cloud, but from edge devices[3]. Storing and processing these large amounts of data in the traditional cloud will bring great system overhead and transmission bandwidth requirements. Meanwhile, processing data at the edge is an important requirement for some applications (eg. Smart Cities[4], Autonomous driving[5]), even as rapid advances in network technologies such as 5G bring increased communication capabilities. Edge computing is a kind of computing mode which is close to data source and strives to reduce transmission delay through local computation[6]. By putting computing on the edge, it relieves the real-time requirements that cloud computing cannot meet in some scenarios[7].

Edge AI refers to AI algorithms deployed on edge devices for local processing, which can process data without a network connection. As more and more devices cannot rely on the cloud to process data, the emergence and development of edge AI can help alleviate such problems[8]. Especially with the advent of the era of big data, the use of artificial intelligence technology to improve the level of automatic processing equipment is particularly important. For example, a prominent feature of Industry 4.0 is smart automation, where industrial robots need to process data at high speed with minimal latency[9]. With the help of AI, industrial robots can process and infer a large amount of multi-modal data from mobile devices, sensors and Internet of Things platforms with an efficiency beyond the reach of human beings, so as to find potential risks and deal with them in a timely and effective manner, thus improving the intelligence of factories[10]. In recent years, deep learning has brought about breakthroughs in artificial intelligence technology[11]. However, these models usually need to consider a large number of parameters during training and rely on high-performance multi-core processing devices such as GPUs. For AI models to cover real-life scenario problems, it is essential to deploy them on devices with limited resources. At the same time, the deployment of the model on the local device can also avoid data leakage during the transmission process to the server, so as to meet the increasing importance of security and privacy needs of people. Therefore, it is necessary and practical to study efficient models that can be deployed to resource-constrained edge devices.

Paper	Contributions	Data	Model	System	Applications	Repository
Liu et al.[12]	Edge AI for communication between edge devices.			✓		
Chen et al. [13]	Deep learning applications deployed at the edge of networks.		✓	✓	✓	
Letaief et al. [14]	Edge AI for 6G.			✓	✓	
Xu et al.[15]	Introduce edge intelligence from: caching, training, inference, and offloading.		✓	✓	✓	
Deng et al. [2]	Introduce edge intelligence from: edge for AI and edge on AI.		✓	✓		
Yao et al.[16]	Introduce cloud AI and edge AI from the aspect of edge-cloud collaboration.		✓	✓	✓	
Murshed et al.[17]	Deploying machine learning systems at the edge of the network.		✓	✓	✓	
Park et al. [18]	Explore the key building blocks of wireless network intelligence.	✓	✓			
Wang et al. [19]	Introduce the benefits of edge intelligence and intelligent edge.		✓	✓	✓	
Zhou et al. [8]	Introduce edge intelligence from: architecture, framework, and key technologies.		✓	✓		
Ours	Explain how to learn an efficient model from the data, model, and system perspectives.	✓	✓	✓	✓	✓

Table I. Related Surveys and Their Contributions

A. Related Surveys

Our main work in this survey is to provide a comprehensive overview of the current state-of-the-art techniques and approaches for developing efficient models for resource-constrained devices. Compared with the previous edge AI survey, Shi et al.[12] discussed from the perspective of efficient communication, Dai et al.[20] introduced from the perspective of computation offloading, Zhang et al. talked[21] mobile edge AI for the Internet of Vehicles, Park et al.[18] provided an overview for wireless network intelligence, and Letaief et al.[14] discussed edge AI for 6G. While these surveys are essential and cover various aspects of edge AI, they do not provide a comprehensive discussion of deploying AI models on edge devices.

Deng et al.[2] discussed edge AI from the perspectives of AI on edge and AI for edge, while Zhou et al. [8] provided a comprehensive overview of relevant frameworks, technologies, and structures for deep learning models involved in training and reasoning on the network's edge. Similarly, Xu et al. [22] conducted an extensive review of edge intelligence, including edge inference, edge training, edge caching, and edge offloading. On the other hand, Hua et al.[23] introduced their survey from the viewpoint of AI-assisted edge computing, while Murshed et al.[17] reviewed edge AI from the perspective of practical applications. Additionally, surveys[13] and[19] also touched on edge inference and model deployment. Finally, survey[24] and[25] covered the topics of model compression and acceleration. Although some of these surveys have briefly discussed the deployment of AI models on edge devices, none has provided a comprehensive discussion of this crucial aspect. Therefore, this survey aims to fill this gap and provide a detailed and in-depth analysis of AI model deployment on edge devices. Table 1 presents a comparison of our survey with the existing and significant surveys on edge AI from the data, model, system, applications, and repository aspects.

B. Our Contributions

In this survey, our focus is to provide an academic response to the following research questions (RQs):

- RQ1: What are the data challenges for building and deploying machine learning (ML) models on edge devices and how can we address them?
- RQ2: How can we optimize ML models for efficient edge deployment without significantly compromising accuracy?

- RQ3: What system infrastructure and tools can best support edge AI workflows and seamless model deployment on heterogeneous edge hardware?
- RQ4: What are the applications of edge AI in daily life?
- RQ5: What are the challenges of edge AI and how can they be mitigated and addressed?
- RQ6: What are the future trends of edge AI?

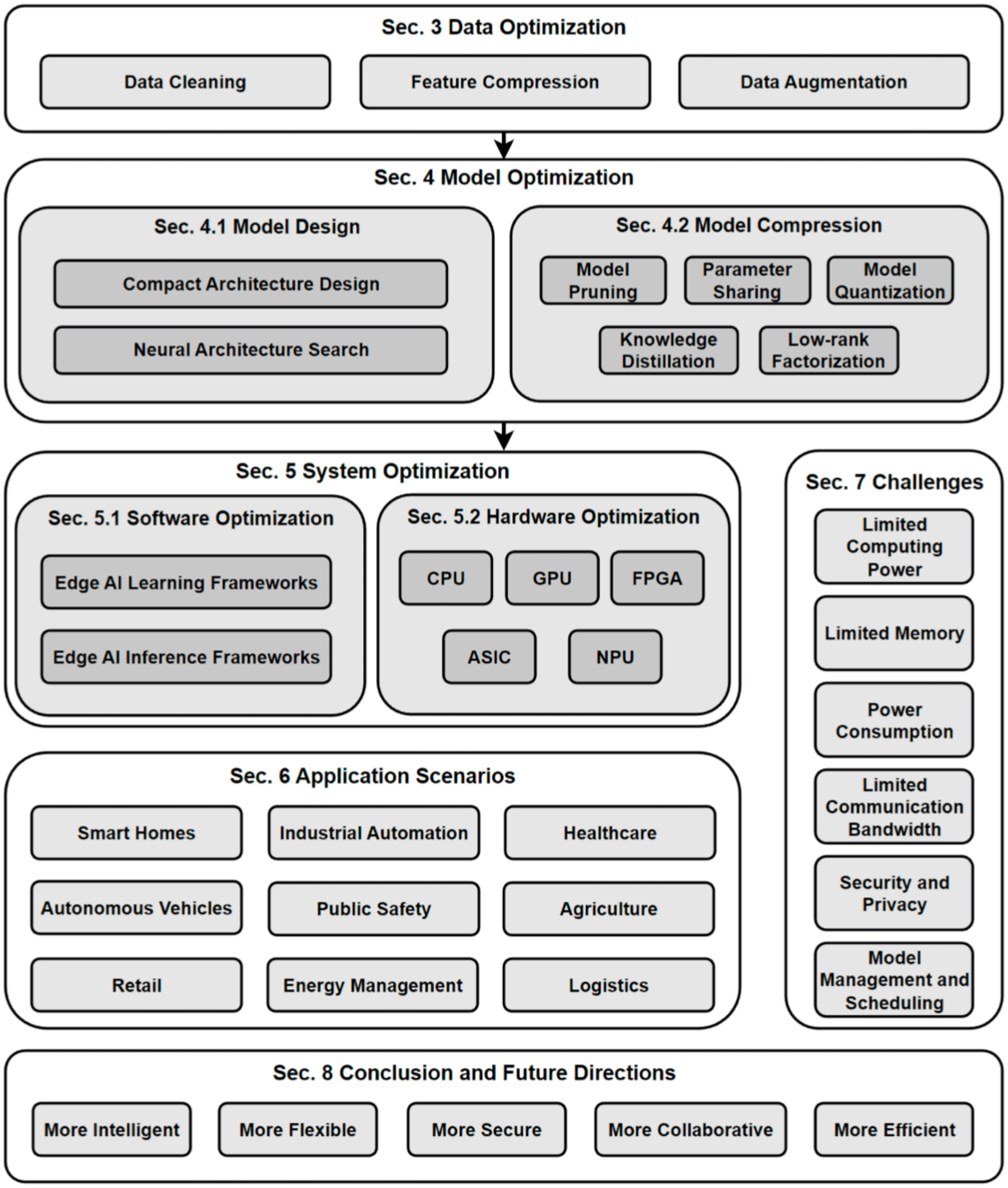

Figure 1. The taxonomy of the discussed topics in this survey.

Acronym	Explanation
5G	5^{th} Generation Mobile Networks
6G	6^{th} Generation Mobile Networks
AI	Artificial Intelligence
ASIC	Application Specific Integrated Circuit
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DNN	Deep Neural Network
DSP	Digital Signal Processor
EC	Edge Computing
Edge AI	Edge Artificial Intelligence
FLOP	Floating-point Operations Per Second
FPGA	Field Programmable Gate Array
GAN	Generative Adversarial Network
GIGO	Garbage in, Garbage out
GPU	Graphics Processing Unit
IoT	Internet of Things
KD	Knowledge Distillation
ML	Machine Learning
NAS	Neural Architecture Search
NLP	Natural Language Processing
NPU	Neural Processing Unit
RNN	Recurrent Neural Network
SLAM	Simultaneous Localization And Mapping
VPU	Vision Processing Unit

Table II. List of abbreviations

By answering the aforementioned research questions, our contributions can be summarized into the following six points:

1. **Novel Taxonomy:** We propose a novel taxonomy for optimizing the deployment of ML models to edge devices based on three dimensions: data, model, and system. This “optimization triad” provides a systematic perspective on the requirements, challenges and solutions for enabling AI at the edge. The proposed triad framework provides a conceptual advance in guiding integrated edge AI solutions. It aims to inspire further development of unified standards, tools, benchmarks and best practices to accelerate progress in this space.
2. **Comprehensive Review:** We present a comprehensive review of techniques for optimizing the deployment of ML models to resource-constrained edge environments. Enabling sophisticated AI on endpoint devices demands overcoming constraints around data, computing, and infrastructure through customized solutions. Our analysis provides an integrated perspective on capabilities and open challenges at each layer of the ML-to-edge pipeline.
3. **Potential Applications:** We present an in-depth analysis of potential edge AI applications that could enhance daily life through enhanced connectivity and intelligent personalization. By categorizing use cases based on technical and experiential attributes, this review aims to systematically uncover value propositions to motivate further development of customized edge AI solutions.
4. **Challenges and Mitigation Strategies:** We analyze various technological and social challenges confronting edge AI that must be navigated to fulfill its promise of improving daily life. Our review explains the limitations and risks of edge AI, from data to models and systems, proposing customized solutions and controls where relevant.
5. **Future Trends:** We analyze emerging trends anticipated to shape the continued progress of edge AI, guiding responsible development and maximizing benefits. In the future, edge AI is expected to have broader applications and become more intelligent, flexible, secure, collaborative, and efficient. Advancements in AI chip technology, edge computing capabilities, and new technologies like blockchain will enable this evolution. With improved processing power, privacy

protection, and security measures, edge devices will be able to handle increasingly complex tasks, ensure user data privacy, provide effective industry-specific solutions, and unlock the full potential of edge AI.

6. **Abundant Resources:** We compile a comprehensive set of resources on edge AI spanning backgrounds, literature, and open source codes into an open access Github repository available at https://github.com/wangxb96/Awesome-AI-on-the-edge to provide researchers and developers a foundation to build upon. By organizing and annotating key materials in this space, our curated collection aims to chart a roadmap of edge AI.

C. Organization

In this survey, we aim to explain how to learn efficient models for edge deployment and edge inference from three perspectives: *data*, *model*, and *system*. At the data level, we focus on data preprocessing and improving the quality of the trained model by removing noisy data and extracting key features. This enables the model to effectively learn from the data and produce accurate predictions. At the model level, we focus on the design of the model architecture and a series of model compression methods to further compact the model. By reducing the model's size, we can achieve faster inference and reduce the computational resources required to run the model. This is particularly important for edge AI, where resource-constrained devices require efficient models. At the system level, we explore the software and hardware level to accelerate model training and inference methods. By leveraging these techniques, we can achieve faster and more efficient model training and inference, allowing for faster deployment and more responsive edge AI systems.

The remaining sections of this paper are as follows: Section 2 reviews the fundamental concepts of edge computing and edge AI. In Sections 3, 4, and 5, we explore optimizing ML for resource-constrained edge environments from data, model and system levels respectively (The work pipeline is shown in Figure 2.). Section 6 shows the application scenarios of edge AI. In Section 7, we discuss the challenges of edge AI. Finally, we conclude the article and show the potential trends of edge AI in Section 8. Specifically, we summarize the taxonomy of the discussed topics of this paper in Figure 1, and the discussed edge AI optimization triad is shown in Figure 3[1].

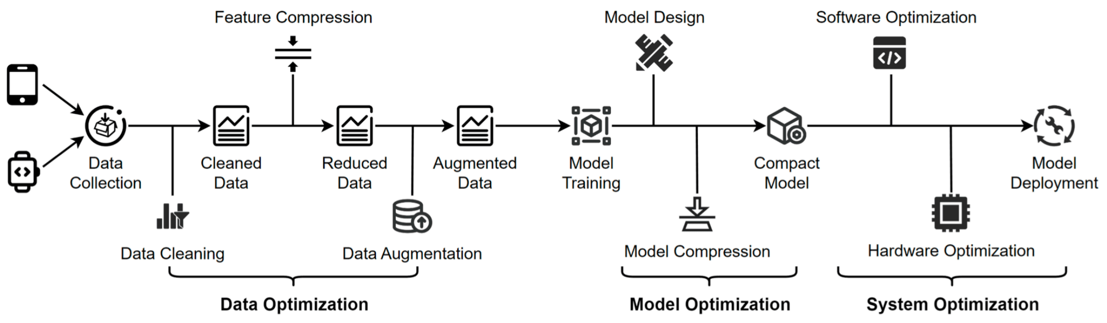

Figure 2. An overview of edge deployment. The figure shows a general pipeline from the three aspects of data, model and system. Note that not all steps are necessary in real applications.

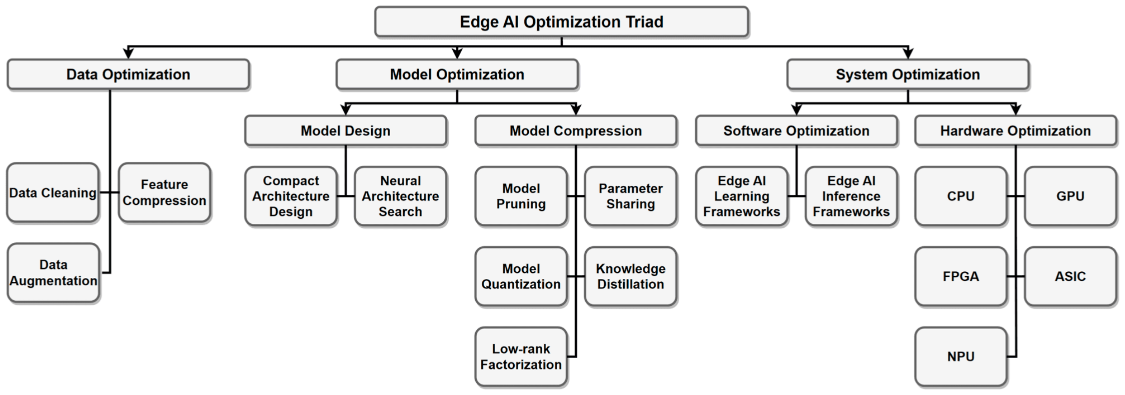

Figure 3. Edge AI Optimization Triad.

II. Fundamental Concepts

This section provides the fundamental concepts of edge computing and edge artificial intelligence, and Figure 4 shows the intersection between edge computing and artificial intelligence and the focus of this survey.

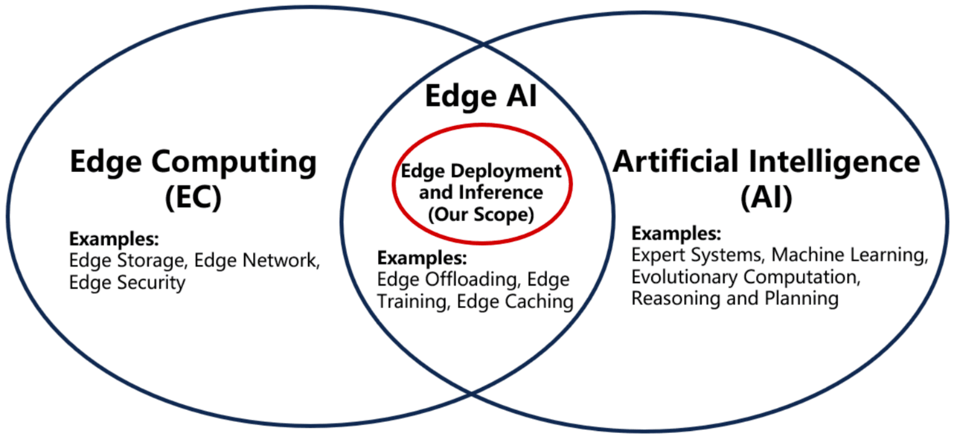
Edge AI
Edge Computing (EC)
Examples: Edge Storage, Edge Network, Edge Security
Edge Deployment and Inference (Our Scope)
Examples: Edge Offloading, Edge Training, Edge Caching
Artificial Intelligence (AI)
Examples: Expert Systems, Machine Learning, Evolutionary Computation, Reasoning and Planning

Figure 4. The intersection of edge computing and artificial intelligence. This survey focuses on edge deployment and inference in edge AI.

A. Edge Computing

Cloud computing offers many benefits, including flexibility, scalability, enhanced collaboration, and reduced costs for modern enterprises[26]. However, cloud computing systems are completely dependent on the Internet, and without a valid Internet connection, users will not be able to access services. Additionally, since the cloud infrastructure is provided by the cloud provider, the cloud user has limited control over applications, resources, and services. The risk of user data being leaked in the cloud and during transmission is also noteworthy[27]. Despite the many advantages of cloud computing, when edge devices have real-time requirements for data processing, the response time of modes that transport output from edge to cloud for processing and then return may be too long, especially in the case of an unstable network. Edge computing, a distributed computing architecture, has been proposed to address this issue. It moves data processing to the edge node where the data is generated, addressing the issue of slow response and high delay that can occur in cloud processing[7]. Figure 5 shows the difference between cloud computing and edge computing.

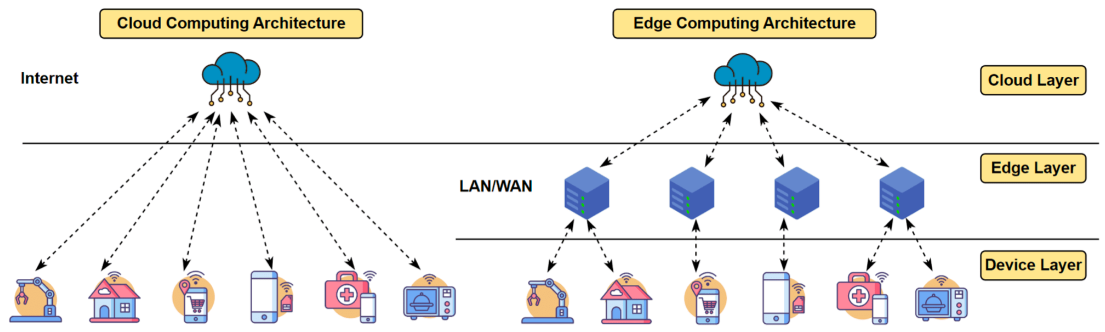

Figure 5. Cloud computing concentrates resources in centralized places like data centers, while edge computing places compute and storage resources closer to the data source.

Edge application services reduce the transfer of data and aim to keep processing locally, which alleviates problems such as network latency and transmission overhead. Since the data is stored and processed locally, the user has absolute ownership of the data, which also avoids the risk of data leakage during transmission between edge nodes and servers[28]. Edge computing brings computing closer to the end-user and speeds up the response time of services, which is necessary and essential in services such as autonomous driving[5]. When local resources are limited, the local device transmits data to the edge network server instead of to the cloud server, which can avoid long-distance transmission and response and thus improve efficiency[29]. Moreover, the deployment and access of edge devices and their ability to continue service even when communication is slow or temporarily interrupted ensure the scalability and reliability of edge computing[7]. The application of edge computing has been greatly successful in many aspects, for example, IoT[30], autonomous driving[5], smart cities[4], robotics[31], and so on.

B. Edge Artificial Intelligence

Edge artificial intelligence, or edge AI, is a combination of edge computing and artificial intelligence. With the proliferation of IoT devices, a large amount of multi-modal data (such as audio, video, pictures, etc.) is continuously generated. Advances in edge computing allow data on these edge devices to be processed locally in real-time without being sent back to the cloud, reducing latency and providing more efficient and timely responses[7]. Artificial intelligence is an automated technology that quickly analyzes large amounts of data to extract information for further prediction and decision-making, which makes it suitable for application on edge devices in many scenarios[15]. As the computing power of edge devices improves without a significant increase in hardware costs and advances in algorithm-optimization techniques enable computationally demanding AI models to run on edge devices, the generation and development of edge AI technology that meets the requirements of real-time response is made possible[19].

As shown in Figure 6, edge AI allows data to be processed at the local level, which greatly reduces latency between cloud and local data processing. With less data being transferred, the system's bandwidth requirements and cost are reduced. More importantly, because data is stored and processed locally, data security is improved, and there is less risk of data leakage. Along with the application of AI technology, this increases the level of automation of tasks handled by edge devices. Furthermore, edge AI enables model training and reasoning on edge devices, which allows real-time decisions to be made. It also enables local decision making, which is independent of network quality and cloud systems, further improving the reliability of edge task execution. Edge AI is used in a wide range of applications, including autonomous cars, virtual reality games, smart factories, security cameras, and wearable healthcare devices[32]. Enabled by AI technology, the automation and intelligence level of edge equipment is enhanced.

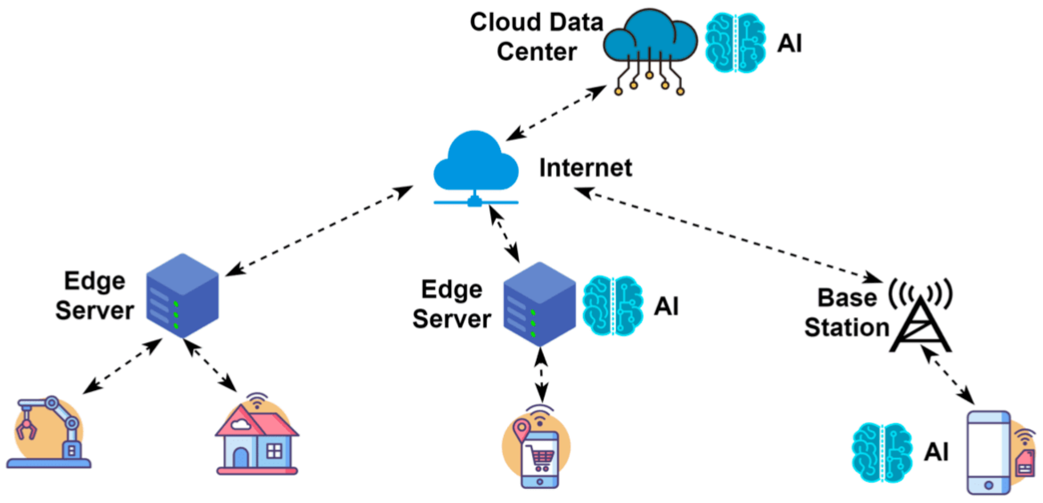

Figure 6. Edge AI is the application of AI algorithms and technologies to edge computing devices to achieve faster and real-time data processing and applications.

III. Data Optimization for Edge AI Deployment

Garbage in, garbage out (GIGO) is a commonly used idiom in the computer world, which implies that poor quality data entering a computer system will produce poor results. In industry, it is widely recognized that data and features determine the upper limit of ML, and data preprocessing methods, such as feature engineering, play a crucial role in industrial processes. An example of this is the GPT series, which shares a similar basic model architecture but has seen improvements in both the scale and quality of training data. As a result, the performance of these models has been significantly enhanced[33]. To improve the performance of the model in resource-constrained devices, data preprocessing is often an essential step. Additionally, feature compression techniques can significantly reduce the data size, thereby reducing the model's size and resource requirements. By effectively preprocessing data, we can remove noisy or irrelevant data, extract relevant features, and normalize the data to achieve better model performance. Common data preprocessing techniques include data cleaning, feature selection, and feature extraction. These techniques can be applied to various types of data, including text, image, and time-series data. In this section, we will introduce common data preprocessing methods and their applications in resource-constrained environments. Figure 7 shows the operations of data optimization.

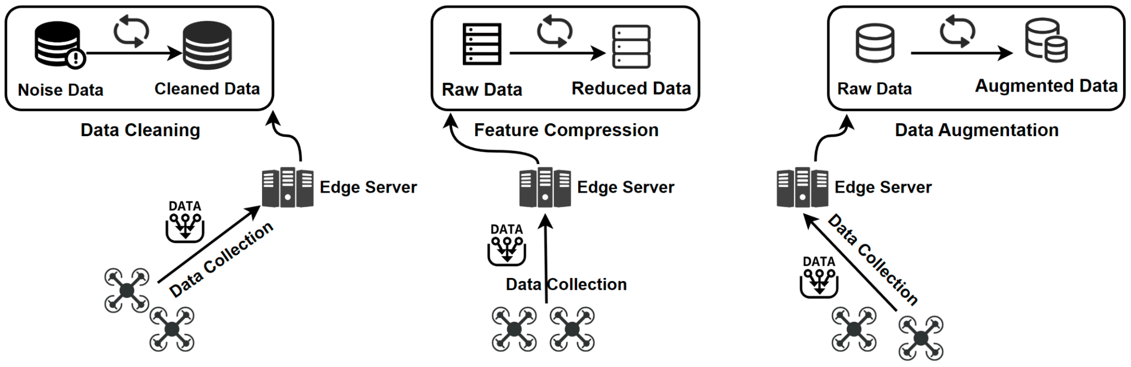

Figure 7. An overview of data optimization operations. Data cleaning improves data quality by removing errors and inconsistencies in the raw data. Feature compression is used to eliminate irrelevant and redundant features. For scarce data, data augmentation is employed to increase the data size.

A. Data Cleaning

Data cleaning is a crucial step in data preprocessing that involves removing or correcting noisy or incorrect data and removing irrelevant or redundant observations. In ML, the presence of label noise in data can significantly impact the accuracy of the trained model. However, re-labeling large datasets accurately can be a challenging task particularly in situations where resources are limited. To address this issue, recent research has proposed innovative approaches, such as active label cleaning, which identifies and prioritizes visibly mislabeled samples to clean up the noisy data[34]. Mishra et al. [35] have developed an ensemble method based on three deep learning models to handle noise labels of different concentrations of human movement activities collected by smartphones, which can alleviate the problem of label noise arising from crowdsourcing or rapid labeling on the Internet.

With the proliferation of smart sensors in the IoT, vast amounts of data are being collected. However, the harsh sensor environment tends to introduce noise into the collected data. To mitigate the problem that traditional sensor nodes are not enough to handle big data, researchers have proposed various innovative approaches. For instance, Wang et al.[36] proposed a method of data cleaning during data collection and optimized the model through online learning. Ma et al.[37] proposed a federated data cleaning approach for future edge-based distributed IoT applications while protecting data privacy. Sun et al.[38] developed a data stream cleaning system with the help of both the cloud servers and edge devices. Additionally, Sun et al.[39] also proposed an adaptive data cleaning method based on intelligent data collaboration for filtering noise data. The work of Gupta et al.[40] proposed a ProtoNN compression approach to reduce the model size further by learning a small number of prototypes to represent the training set to enable deployment on resource-scarce devices. These approaches offer promising solutions for cleaning data and enabling efficient processing of big data in resource-constrained environments.

Discussion: While innovative approaches like active label cleaning and ensemble methods based on deep learning models can alleviate the problem of label noise, they may have some disadvantages. For instance, active label cleaning depends on the availability of a small set of labeled samples, and the performance of the approach may suffer if the labeled samples are not representative of the entire dataset. Similarly, ensemble methods can be computationally expensive and may increase the risk of overfitting. In addition, some of the proposed approaches for data cleaning in IoT environments, such as data cleaning during data collection and federated data cleaning, may require significant computational resources and may not be feasible in resource-constrained environments. Furthermore, intelligent data collaboration for filtering noise data may require significant communication overhead, which can be a challenge in IoT environments. Therefore, while these approaches offer promising solutions for cleaning data and enabling efficient processing of big data in resource-constrained environments, they also have limitations that need to be carefully considered.

B. Feature Compression

Feature compression is a common technique used in ML to reduce the dimensionality of high-dimensional feature space. Two popular methods of feature compression are feature selection and feature extraction, which aim to remove redundant and irrelevant features while retaining the necessary information[41]. Feature selection involves choosing a subset of relevant features from the original set while maintaining maximum usefulness, resulting in improved model accuracy, reduced complexity, and enhanced interpretability[42]. In contrast, feature extraction creates new features based on the functions of the original ones, ensuring that the newly created features contain useful information while being non-redundant[43]. By leveraging these techniques, researchers can compress the feature space and improve the efficiency and performance of their models.

With the increasing popularity and growth of computationally constrained devices such as smartphones, wearables, and IoT devices, there is a growing need to develop efficient and effective ML algorithms for on-device analysis on these platforms. Feature selection has emerged as a popular technique for reducing the dimensionality of high-dimensional feature spaces and improving the efficiency and accuracy of ML models. In recent years, researchers have applied feature selection methods to various resource-constrained applications. For example, Do et al.[44] proposed an accessible melanoma detection method using smartphones, where they designed a feature selection module to select the most discriminative features for classification. Similarly, Fasih et al.[45] adopted feature selection methods to reduce memory and computational requirements in their Active Feature Selection approach for emotion recognition. Summerville et al.[46] designed an ultra-lightweight deep approach based on feature selection for anomaly detection in IoT devices. Sudhakar et al.[47] proposed ActID, a framework for user identification based on activity sensors, where the feature selection method is used to evaluate and select discriminative high-quality features, thus reducing the complexity of the algorithm and making it better adapt to the requirements of resource-limited devices. Laddha et al.[48] proposed a method for selecting features with high invariance and robustness based on descriptor score to achieve the required pose precision for real-time simultaneous localization and mapping (SLAM) on resource-constrained platforms.

In addition, feature selection has also been applied in other edge environments to enhance the performance and efficiency of ML algorithms. Several studies have demonstrated the usefulness of this technique in various edge-based applications, such as Parkinson's disease classification[49], atrial fibrillation recognition[50], data dimensionality reduction[51], fault diagnosis on the edge of IoT[52], COVID-19 detection[53], and more. For instance, swarm intelligence-based methods[51], pre-training models[53], and social learning particle swarm optimization[53] have been employed to select the most informative features. Feature selection appears to be a promising approach to reducing the computational complexity of ML algorithms, enhancing their accuracy, and enabling real-time analysis on resource-constrained devices. Therefore, its broader adoption is expected to facilitate on-device analysis in constrained environments and accelerate the development of efficient and effective edge-based ML solutions.

The increasing demand for intelligent sensing and analysis on edge has led to a growing need for efficient and effective methods to reduce the energy and memory cost of deep learning algorithms in resource-constrained edge computing systems. To address this challenge, researchers have proposed various feature compression methods. For example, Matsubara et al.[54] proposed a supervised feature compression method based on KD, while Chen et al.[55] proposed a sparse projection method for face recognition. To better perform intelligent sensing at the front end, Chen et al.[56] proposed an intermediate-layer deep learning feature compression method. Liu et al.[57] developed a method for compressing features along the spatiotemporal dimensions for action recognition, while Shao et al. [58] designed a lightweight encoder-decoder structure to reduce the size of corresponding features. Further, Abdellatif et al.[59] proposed a lightweight classification mechanism for detecting seizures with high accuracy and low computational requirements at the edge of the network through feature extraction of vital signs. Zhou et al.[60] applied image preprocessing techniques to vision sensing and designed an industrial wireless camera system to reduce energy consumption. Similarly, Abdellatif et al.[61] designed a multi-modal data compression and edge-based feature extraction method for event detection. Moreno-Rodenas et al.[62] proposed a method for monitoring wastewater pumping stations using in-camera image processing, while Guo et al.[63] designed an adaptive region-of-interest-based image compression scheme for target detection.

Discussion

These feature compression methods have shown great potential in reducing the energy and memory cost of deep learning algorithms in resource-constrained edge computing systems. By compressing features, these methods enable efficient and effective analysis on edge with limited resources, making it possible to perform intelligent sensing and analysis in a wide range of applications. While feature compression methods are promising for edge-based applications, there is a trade-off between the size of the compressed features and the accuracy of the resulting model. Thus, it is essential to carefully balance the compression ratio and the accuracy of the model to ensure the best performance on resource-constrained devices. Moreover, further research is needed to develop more sophisticated feature compression methods that can better adapt to different applications and scenarios.

C. Data Augmentation

Data augmentation is a commonly used technique in ML to increase the amount of a dataset by generating new data through slight modifications of existing data. This technique can be particularly useful when dealing with smaller datasets and can help alleviate overfitting problems. In the field of image processing, data augmentation can be achieved through various techniques such as rotation, edge enhancement, denoising, and scaling of images[64]. By applying these modifications to existing images, new and diverse images can be generated, thereby increasing the size of the dataset and improving the performance of the model. In natural language processing (NLP) tasks, data augmentation can be realized by various techniques such as randomly adding or deleting words, adjusting the order of words, auxiliary task utilization[65], translating samples into a second language and then translating back to form new samples, among others[66]. These techniques help in generating new and diverse data, which can be used to train better models and improve the performance of the model on the test data.

To address the challenge of limited data availability in edge devices, researchers have proposed various data augmentation methods that generate new and diverse data for training ML models. For instance, Wang et al.[67] designed a traffic prediction method based on a 5G cellular infrastructure that incorporates data augmentation to alleviate data shortages and privacy issues on edge devices. Similarly, Liao et al.[68] proposed three data augmentation methods to accelerate the creation of a multi-user enhanced PHY layer authentication system model. Another example is the work of Liu et al.

[69], who improved the prediction accuracy of a KITTI road detection model by introducing appropriate data augmentation strategies, such as adding road edge labels to small training samples. In addition, data augmentation has been employed by researchers to improve the generalization of images in complex scenes, as demonstrated by Jiao et al.[70] in their method for litchi monitoring. Gu et al.[71] proposed a line segment detection method named M-LSD that leverages data augmentation to provide auxiliary line data for the training process. Furthermore, Liu et al.[72] utilized the data augmentation technique to improve the performance of intrusion detection systems in the industrial IoT by addressing the problem of data imbalance. Pan et al.[73] expanded the amount of training data for 1D tracking through the use of data augmentation, which reduced the pressure of collecting more data from users.

Discussion

Although data augmentation is a powerful technique for generating new and diverse data and improving the performance of ML models, it has some limitations and disadvantages. One major disadvantage is that data augmentation may introduce bias or unrealistic assumptions into the training data, which can negatively affect the performance of the model on the test data. Moreover, the effectiveness of data augmentation depends on the choice of augmentation techniques and the specific application domain. For instance, some augmentation techniques may not be suitable for certain types of data, such as medical images, where introducing artificial modifications can be risky. Additionally, data augmentation can be computationally expensive, especially for large datasets and complex models. Therefore, while data augmentation is a valuable technique for improving the performance of ML models, it is important to carefully consider its limitations and potential drawbacks in different application scenarios.

IV. Model Optimization for Edge AI Deployment

Model optimization is a critical step in deploying ML models to edge devices where computational resources are limited. There are two main approaches to model optimization: model design and model compression (as shown in Figure 8). The former involves developing compact model architectures and using automated neural architecture search techniques to achieve superior performance while minimizing the computational burden and number of model parameters. The latter involves using methods such as pruning, parameter sharing, quantization, knowledge distillation, and low-rank factorization to shrink the size of deep learning models without significantly affecting their accuracy or performance. These techniques are crucial for deploying complicated models on devices with limited resources or in large-scale distributed systems with constrained processing, memory, and storage.

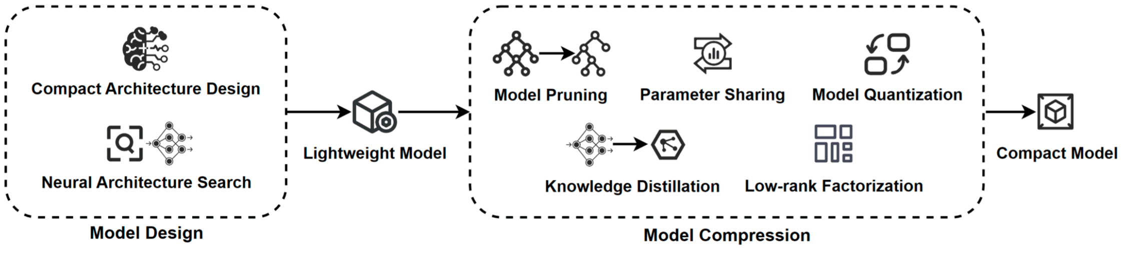

Figure 8. An overview of model optimization operations. Model design involves creating lightweight models through manual and automated techniques, including architecture selection, parameter tuning, and regularization. Model compression involves using various techniques, such as pruning, quantization, and knowledge distillation, to reduce the size of the model and obtain a compact model that requires fewer resources while maintaining high accuracy.

A. Model Design

Developing optimal model architectures is critical for achieving superior performance across a range of ML applications. In this section, we will explore two approaches for addressing this challenge: the design of compact model structures and the use of automated neural architecture search (NAS) techniques. These strategies aim to achieve superior model performance while minimizing the computational burden and number of model parameters, enabling practical deployment on various computational devices.

1. Compact Architecture Design

Compact neural network architectures are typically characterized by their lower requirement for computing resources and fewer parameters. Due to the limited computing power of edge devices, it is increasingly important to develop neural network models that are both efficient and compact. Therefore, in this section, we will introduce some of the noteworthy lightweight neural network models that have been proposed in the literature.

The rise of areas such as the IoT and edge computing, which require processing huge amounts of data and the ability to perform real-time analysis on edge devices, has boosted the development of lightweight neural networks. These lightweight neural networks typically use techniques such as convolutional grouping, depth-separable convolution, width-separable convolution, channel pruning, network pruning, and others to compact the network architecture[74], resulting in higher computational efficiency and lower memory consumption. For example, the MobileNets series[75][76][77] is a collection of lightweight neural networks built for mobile vision applications. These networks were developed by Google researchers and have gained a lot of traction in the computer vision world due to their high accuracy and minimal computing complexity, making them perfect for usage on mobile and embedded devices with limited resources. Moreover, Zhou et al.[78] proposed to invert the structure and introduce a novel bottleneck design, referred to as the ”sandglass block,” which conducts identity mapping and spatial transformation at higher dimensions, thereby reducing information loss and gradient confusion more effectively. In Tan et al.[79]'s research, they introduced an automated approach for mobile NAS that incorporates model latency as a crucial optimization objective to solve the difficulty of manual solution for so many architecture possibilities in CNN.

ShuffleNets series is a lightweight CNN proposed by MegVII, which aims to solve the balance problem between the accuracy and efficiency of lightweight neural networks. The core idea of ShuffleNets is to enhance the information flow of the network and improve its accuracy by performing channel shuffling within groups. In ShuffleNetV1[80], channel shuffling is introduced, which divides the input group into multiple sub-groups along the channel dimension and performs convolution operations on each sub-group. The results are then concatenated along the channel dimension. Through this operation, ShuffleNetV1 can effectively reduce computational complexity while improving accuracy. Based on ShuffleNetV1, ShuffleNetV2[81] employs a novel ShuffleNetV2 unit structure, which incorporates designs such as channel shuffling and pointwise convolutions. This unit structure significantly improves information flow, thereby further enhancing the accuracy of the network. In OneShot proposed by Guo et al.[82], it alleviates the weight adaptive problem by building a simplified super network in which all architectures are single paths.

SqueezeNet[83] achieves efficient information transfer with few parameters by introducing a component named ”Fire module,” which consists of a 1 x 1 convolutional layer called squeeze layer and a 1 x 1 and 3 x 3 convolutional layer called expand layer. Squeeze layer compresses the number of channels in the input feature graph, and expand layer increases the number of channels in the compressed feature graph. The subsequent version of SqueezeNet, SqueezeNext, using hardware simulation results of power consumption and inference speed on embedded systems, showed that compared to SqueezeNet, the model is 2.59x faster, 2.25x more energy-efficient, and without any accuracy degradation[84]. Han et al.[85] proposed a plug-and-play Ghost module, which tends to generate more feature graphs through low-cost operations to enhance feature extraction, and at the same time, it uses a Ghost bottleneck structure to enhance the representation ability of models.

The EfficientNet series[86][87][88], proposed by Tan et al. of the Google Brain Group, are also famous efficient CNNs. EfficientNet employs a technique called "compound scaling," which adjusts not only the depth, width, and resolution of the network when scaling it up but also the interdependent relationships between these parameters. This results in a more efficient and accurate network[86]. EfficientNetV2 is an upgraded version of EfficientNet, which further improves the performance of the network by using more efficient network structure design and optimized training strategies, and proposes an improved progressive learning method to adaptively adjust the learning strategy[87]. EfficientDet is based on EfficientNet as the backbone network and achieves higher detection accuracy and faster inference speed through innovative designs such as the introduction of the BiFPN structure, carefully designed feature network hierarchy and feature fusion mechanism, as well as optimized loss function[88].

Huang et al.[89] proposed CondenseNet, an efficient CNN architecture that encourages feature reuse through dense connectivity and prunes filters associated with redundant feature reuse through learned group convolutions. The pruned network can be efficiently converted into a network with regular group convolutions for efficient inference, which can be easily implemented with limited computational costs during training. Yang et al.[90] proposed an alternative scheme named CondenseNetV2 to improve the reuse efficiency of features. In this approach, each layer has the capability to selectively utilize a specific set of highly significant features from the previous layer while concurrently updating a set of earlier features to enhance their relevance to subsequent layers. Mehta et al. proposed ESPNet[91] and ESPNetV2[92], where ESPNet reduces computation and learns representations with large receptive fields by using point-wise convolutions and spatial pyramid of dilated convolutions. ESPNetV2 is an extension of ESPNet that uses depth-separable convolution and outperforms ESPNet by 4-5%. FBNets (Facebook-Berkeley-Nets), a series of lightweight networks created by Facebook and UC Berkeley, FBNet[93] uses a differentiable NAS framework to optimize neural architecture using a gradient-based method, while the second version FBNetV2[94] focuses on the small DNAS search space. The third version FBNetV3[95], takes into account the fact that the other approaches overlooked a better architecture-recipe combination. PeleeNet, unlike recent lightweight networks that heavily rely on depthwise separable convolutions, utilizes conventional convolutions and is primarily designed for deployment on mobile devices[96].

The Inception series is also a classic network proposed by Google. The idea is to use multiple convolution kernels of different sizes to process input data in parallel and then concatenate their outputs along the channel dimension to form the network output[97]. InceptionV2 uses batch normalization and replaces large convolution kernels with small convolution kernels[98]. In InceptionV3, the factorization into smaller convolutions is introduced, the larger two-dimensional convolution is split into smaller one-dimensional convolutions, and the Inception module structure is optimized[99]. InceptionV4 introduced stem modules and reduction Blocks[100]. Xception primarily achieved complete separation of learning spatial correlation and learning inter-channel correlation tasks through the introduction of depthwise separable convolutions[101].

Mehta et al.[102] proposed MobileViT to learn the global representation of networks, which combines the advantages of CNNs and ViTs and is superior to both. Wu et al.[103] designed a lightweight NLP architecture, Lite-Transformer, where the key is a Long Short Range of Attention, with one set of heads focused on local context modeling and another on long-distance relationship modeling. Recently, lightweight networks based on attention have been proposed. For instance, Hou et al. [104] took into account that some channel attention studies ignored location information and embedded it into channel attention to enhance network performance. To avoid the complexity of the model caused by the sophisticated attention module, Wang et al.[105] designed an Efficient Channel Attention (ECA) module that can bring clear performance improvement with only a handful parameters involved. In attention mechanisms, there are two types: spatial attention and channel attention. Combining the two can improve performance, but it inevitably leads to increased model complexity. To address this, Zhang et al.[106] designed the Shuffle Attention (SA) module, which combines the advantages of both types of attention while avoiding excessive model complexity. Misra et al.[107] proposed triplet attention, a novel method for efficient attention weight calculation by using a three-branch structure to capture cross-dimensional interactions. In the study by Zhang et al.[108], they designed a Split-Attention block for use in ResNet, which allows attention to span feature groups while ensuring the simplicity and ease of use of ResNet. In addition, to help readers better understand these lightweight networks, we summarized their characteristics in Table 3.

Family	Model	Highlights
MobileNets	MobileNets[75]	Depth-separable convolution reduces computation while maintaining model accuracy.
	MobileNetV2[76]	Introduce a novel inverted residual module with a linear bottleneck.
	MobileNetsV3[77]	Explore how automatic search algorithms and network design can work together.
	MobileNeXt[78]	Propose a new bottleneck design named sandglass block.
MnasNet	MnasNet[79]	Propose an automated mobile NAS method to solve the difficulty of manual solution.
ShuffleNets	ShuffleNetV1[80]	Utilize pointwise group convolution and channel shuffle.
	ShuffleNetV2[81]	It introduces a novel unit structure, the ShuffleNetV2 unit.
	OneShot[82]	It alleviates the weight co-adaption problem by constructing a simplified super network.
SqueezeNets	SqueezeNet[83]	It proposes an extremely compressed network structure design.
	SqueezeNext[84]	It introduces low rank filters, bottleneck module and fully connected layer.
GhostNet	GhostNet[85]	The ghost module and ghost bottleneck are designed to reduce the computation.
EfficientNets	EfficientNet[86]	Propose a new scaling method that uses a simple and efficient compound coefficient.
	EfficientNetV2[87]	Combine training perceptual neural architecture searching and scaling.
	EfficientDet[88]	Introduce innovative designs such as the BiFPN structure, elaborately designed feature network hierarchy and feature fusion mechanism, as well as optimized loss functions.
CondenseNets	CondenseNet[89]	Encourage feature reuse and realize an efficient convolutional network architecture.
	CondensenetV2[90]	An alternative method called sparse feature reactivation is proposed to improve feature reuse.

Family	Model	Highlights
ESPNets	ESPNet[91]	Propose an efficient spatial pyramid (ESP) module.
	ESPNetV2[92]	It's an extension of ESPNet and uses depth-wise separable convolutions.
FBNets	FBNet[93]	Use gradient-based methods to optimize the architecture of a ConvNet.
	FBNetV2[94]	Propose DMaskingNAS to alleviate the problem of small search space of DNAS.
	FBNetV3[95]	It takes into account the neglect of better architecture-recipe combinations in previous approaches.
PeleeNet	PeleeNet[96]	This is a variation of DenseNet, designed for mobile devices.
Inception	InceptionV1[97]	It improves the utilization rate of computing resources in the network.
	InceptionV2[98]	It proposes batch normalization method.
	InceptionV3[99]	Use convolution decomposition to improve efficiency and an efficient feature map to reduce dimension.
	InceptionV4[100]	It introduces stem modules and reduction blocks.
	Xception[101]	Propose the use of depthwise separable convolutions.
Transformer	MobileViT[102]	Combine the advantages of CNNs and ViTs.
	Lite-Transformer[103]	Introduce a lightweight NLP architecture with Long-Short Range Attention.
Attention Based	CANet[104]	Embed positional information into channel attention.
	ECANet[105]	Propose an Efficient Channel Attention module.
	SANet[106]	Feature grouping and channel attention information replacement are introduced.
	Triplet attention[107]	Propose triplet attention for cross-dimensional interaction.
	ResNeSt[108]	A modular Split-Attention block is proposed to enable attention across feature groups.

Table III. Classical Lightweight Neural Network Models

Discussion

The advantages of compact architecture design are that it produces efficient and compact neural network models with higher computational efficiency, lower memory consumption, and improved accuracy. These models are suitable for deployment on edge devices with limited resources, making them ideal for IoT and edge computing applications. However, designing optimal model architectures can be a time-consuming and resource-intensive process. Additionally, some model design techniques, such as depthwise separable convolutions and pointwise convolutions, may be less effective in capturing complex features compared to traditional convolutional layers, which may negatively impact the model's accuracy.

2. Neural Architecture Search

NAS aims to automate the process of designing neural network architectures, which can be a time-consuming and resource-intensive process when done manually. NAS typically employs different optimization methods such as evolutionary algorithms, reinforcement learning, or gradient-based optimization to search the space of neural architectures and identify the one that performs best on a specific task while satisfying certain computational constraints. Recently, with the rise of IoT and AI of Things (AIoT), there has been a growing demand for intelligent devices with low energy consumption, high computing efficiency, and low resource usage. NAS has emerged as a promising approach to design efficient and lightweight neural networks that can be deployed on edge devices. In this section, we will discuss various recent studies that have used NAS to design efficient neural networks for edge computing.

One notable subject area in current studies is the employment of advanced search algorithms and multi-objective optimization approaches to identify neural network architectures that optimize different performance metrics such as accuracy, resource consumption, and power efficiency. This work has become increasingly important in recent years as edge computing applications require efficient yet accurate models. To this end, researchers have proposed various approaches for multi-objective architecture search. Lu et al.[109] and Lyu et al.[110] are two such studies that employed multi-objective optimization to identify efficient and accurate neural network architectures for edge computing applications. Similarly, Chen et al.[111] used performance-based strategies to search efficiently for architectures that are optimal with regard to multiple objectives. These studies underscore the significance of considering different objectives during the architecture search process to ensure that the neural architectures are both accurate and efficient. By using advanced search algorithms and multi-objective optimization techniques, researchers can design models that are effective in resource-constrained environments while still maintaining high accuracy, which is crucial for edge computing applications.

The innovative techniques and algorithms to improve the efficiency and effectiveness of NAS are often employed in the research. For example, Mendis et al.[112] incorporated intermittent execution behavior into the search process to find accurate network architectures that can safely and efficiently execute under intermittent power, which is a common challenge in edge computing applications. Similarly, Ning et al.[113] identified factors that affect the fault resilience capability of neural network models and used this knowledge to design fault-tolerant CNN architectures for edge devices. These studies demonstrate the importance of considering unique challenges and constraints of edge computing applications when designing neural networks through NAS. By incorporating innovative techniques and algorithms, researchers can develop architectures that are not only efficient and accurate but also robust and reliable.

Furthermore, several studies proposed hardware-efficient primitives and frameworks to optimize the search process and improve the performance of the resulting networks. For instance, Liu et al. [114] proposed the Point-Voxel Convolution (PVConv) primitive, which combines the best of point-based and voxel-based models for efficient NAS. This hardware-efficient primitive achieves state-of-the-art performance with significant speedup and has been successfully deployed in real-world edge computing scenarios, such as an autonomous racing vehicle. Donegan et al.[115] proposed the use of a differentiable NAS method to find efficient CNNs for Intel Movidius Vision Processing Unit (VPU), achieving state-of-the-art classification accuracy on ImageNet. These studies highlight the importance of considering hardware efficiency when designing neural networks for edge computing applications. By leveraging hardware-efficient primitives and frameworks, researchers can not only optimize the search process but also develop neural architectures that are efficient and effective in resource-constrained environments.

Moreover, some studies propose novel NAS approaches that strictly adhere to resource constraints, while others focus on addressing non-i.i.d. data distribution in federated learning scenarios. For instance, Nayman et al.[116] introduced HardCoRe-NAS, which is a constrained NAS approach that strictly adheres to multiple resource constraints such as latency, energy, and memory, without compromising the accuracy of the resulting networks. Similarly, MemNAS[117] is a framework that optimizes both the performance and memory usage of NAS by considering memory usage as an optimization objective during the search process. On the other hand, Zhang et al.[118] focused on addressing non-i.i.d. data distribution in federated learning scenarios by proposing the Federated Direct NAS (FDNAS) and Cluster Federated Direct NAS (CFDNAS) frameworks. These frameworks leverage advanced proxylessNAS and meta-learning techniques to achieve device-aware NAS, tailored to the particular data distribution and hardware constraints of each device. These studies demonstrate the importance of considering various constraints and challenges in the design of neural networks for edge computing applications.

Discussion

While NAS has shown promise in designing efficient and lightweight neural networks for edge computing applications, there are still some disadvantages to this approach. One major limitation is that NAS can be computationally expensive, especially when searching through a large space of possible architectures. This can make it challenging to deploy NAS-based models in resource-constrained environments, particularly those with limited computational power. Additionally, while NAS can optimize for multiple objectives, it can be difficult to find a balance between accuracy and efficiency, especially when dealing with complex tasks or non-i.i.d. data distribution. Furthermore, the resulting neural architectures may not be easily interpretable, making it challenging to understand how they work or explain their decisions.

B. Model Compression

Model compression is a group of methods (such as pruning, parameter sharing, quantization, knowledge distillation and low-rank factorization) for shrinking the size of deep learning models without significantly affecting their accuracy or performance by eliminating extraneous components, such as duplicated parameters or layers. Due to deep learning models' high computational and storage needs, model compression has become more and more crucial. These methods are created to make it possible to deploy complicated models on devices with limited resources or in large-scale distributed systems with constrained processing, memory, and storage. To enhance the efficacy and efficiency of deep learning models in various applications, it is possible to employ these techniques either in isolation or in conjunction. For instance, a classic example of combining multiple techniques is Deep Compression, which combines techniques such as pruning, quantization, and Huffman coding to achieve significant compression of deep neural networks (DNN)[119].

1) Model Pruning

DNN models typically consist of numerous parameters and hierarchies, making them computationally and storage-intensive. Due to their frequent application in scenarios with limited resources, such as mobile devices, it is imperative that these models are smaller in size and require less computational power to perform optimally. Pruning is a prevalent model compression technique that reduces the model's size by eliminating extraneous layers or parameters, thereby enhancing its efficiency and reasoning speed. Additionally, pruning helps to prevent overfitting and bolsters the model's ability to generalize.

In recent years, there has been a growing interest in developing pruning techniques for AI models to reduce their size and improve their efficiency, particularly for deployment in resource-constrained environments. Various research efforts have been undertaken to address this challenge. For instance, Xu et al.[120] developed a framework named DiReCtX, which includes improved CNN model pruning and accuracy tuning strategies to achieve fast model reconfiguration in real-time, resulting in significant computation acceleration, memory reduction, and energy savings. Ahmad et al. [121] proposed SuperSlash, which utilizes a pruning technique guided by a ranking function to significantly reduce off-chip memory access volume compared to existing methods. DropNet[122] is an iterative pruning method that reduces the complexity of DNNs by removing nodes/filters with the lowest average post-activation value across all training samples. The findings suggest that DropNet can achieve significant pruning (up to 90% of nodes/filters) without sacrificing accuracy, and the pruned network remains effective even after weight and bias reinitialization. Interestingly, Li et al. [123] demonstrated in their study that heavily compressed large models achieve higher accuracy than lightly compressed small models. Ma et al.[124] proposed gradient-based pruning strategies to improve performances across languages.

Structural pruning is a prominent technique in pruning that involves removing entire structures or modules from a neural network. One approach to structural pruning is the method developed by Gao et al.[125], which uses an efficient discrete optimization method to directly optimize channel-wise differentiable discrete gates under resource constraints while freezing all other model parameters. This results in a compact CNN with strong discriminative power. Wu et al.[126] proposed MOBC, a pruned reinforcement learning-based approach with a structured pruning method that adaptively reduces block migrations and foreground segment cleanings in log-structured file systems, resulting in improved storage endurance and reduced latency with lower overheads. Furthermore, structural pruning has also been utilized in the development of federated learning frameworks. NestFL[127] is a learning-efficient federated learning framework that improves training efficiency and achieves personalization by assigning sparse-structured subnetworks to edge devices through progressively structured pruning.

Dynamic pruning is a commonly used technique in neural networks that involves pruning during the training process. The technique evaluates the importance of neural network weights or neurons and selectively removes unimportant weights or neurons. Geng et al.[128] developed an out-of-order architecture called O3BNN-R that uses dynamic pruning to significantly reduce the size of Binarized Neural Networks, enabling efficient computation on cost- and power-restricted edge devices. This technique improves the efficiency of neural network tasks on edge devices, making them more practical. Another novel dynamic pruning technique is FuPruner[129]. The approach optimizes both parametric and nonparametric operators to accelerate neural networks on resource-constrained edge devices. The approach uses an aggressive fusion method to transform the model and a dynamic filter pruning method to reduce computational costs while retaining accuracy. Gu et al.[130] demonstrated that their proposed mixed-training strategy, which combines two-level sparsity and power-aware dynamic pruning, achieves superior optimization stability, higher efficiency, and significant power savings compared to existing methods. Some researchers have focused on pruning after training, such as Kwon et al.[131], who proposed a framework for pruning Transformers after training, achieving significant reduction in FLOPs and inference latency while maintaining accuracy.

Several researchers have proposed jointly performing pruning and other model compression methods to improve the efficiency of neural networks. Lin et al.[132] introduced HRank, a filter pruning method that prunes filters with low-rank feature maps, resulting in significant reductions in FLOPs and parameters while maintaining similar accuracies. Li et al.[133] proposed a novel pruning technique called kernel granularity decomposition, which combines low-rank approximation with redundancy exploitation to achieve model compactness and hardware efficiency simultaneously. Tung et al. [134] developed CLIP-Q, a novel approach that combines network pruning and weight quantization in a single learning framework. Fedorov et al.[135] designed a Sparse Architecture Search method that combines NAS with pruning to automatically design CNNs that can fit onto memory-limited MCUs while maintaining high prediction accuracy. Khaleghi et al.[136] proposed a quantization and pruning technique for hyperdimensional computing to achieve privacy-preserving training and inference while obfuscating information and enabling efficient hardware implementation. These techniques demonstrate the effectiveness of jointly performing pruning and other model compression methods to improve the efficiency of neural networks while maintaining their accuracy.

Other approaches have incorporated context-aware pruning, such as Huang et al.[137] with DeepAdapter, which improves inference accuracy with a smaller and faster model, and Liu et al.[138], who devised a novel content-aware channel pruning approach for unconditional GANs that significantly reduces the FLOPs of StyleGAN2 by 11x with visually negligible image quality loss compared to the full-size model. In addition, researchers have explored the use of pruning for specific applications, such as radio frequency fingerprinting tasks[139], super-resolution networks[140], and privacy preserved hyperdimensional computing[136]. Moreover, some researchers have proposed hardware-software co-design approaches for pruning, such as Sun et al.[141], who presented a hardware-aware pruning technique for a novel 3D network called R(2+1)D that achieves high accuracy and significant computation acceleration on FPGA.

Discussion

While model pruning can improve the efficiency and speed of neural networks, there are also some disadvantages to this approach. One major limitation is that pruning can result in a loss of model accuracy, especially when a significant number of parameters or layers are removed. Additionally, pruning can be computationally expensive, particularly when searching for the optimal set of parameters to prune. Pruning can also result in a less interpretable model, as the removed parameters or layers may have played a critical role in the network's decision-making process. Finally, some pruning methods may not be compatible with certain hardware or software configurations, limiting their practicality.

2. *Parameter Sharing*

Parameter sharing is a model compression technique that involves sharing the weights of a neural network among multiple layers. This approach results in a significant reduction in the number of parameters required to represent the model, thus reducing its computational and memory requirements. Consequently, the model can be better suited for deployment on resource-constrained devices. This technique has been successfully applied to various deep learning architectures, including CNNs and RNNs.

Several studies have explored different parameter sharing techniques to achieve high compression rates with minimal or no loss of accuracy. For instance, Wu et al.[142] proposed a novel scheme for compressing CNNs by applying k-means clustering on the weights to achieve parameter sharing. The proposed method includes a spectrally relaxed k-means regularization to make hard assignments of convolutional layer weights to learned cluster centers during re-training, and is evaluated across several CNN models demonstrating promising results in terms of compression ratio and energy consumption reduction without incurring accuracy loss. Obukhove et al.[143] designed T-Basis, a compact representation of a set of tensors modeled using Tensor Rings for efficient neural network weight compression while allowing for weight sharing. T-Basis achieves high compression rates with acceptable performance drops and is well-suited for resource-constrained devices. In Ullrich et al. [144]'s research, they employed a version of soft weight-sharing to realize competitive compression rates. You et al.[145] proposed ShiftAddNAS, a NAS method for hybrid neural networks that integrate both powerful multiplication-based and efficient multiplication-free operators. It highlights a novel weight sharing strategy that effectively shares parameters among different operators with heterogeneous distributions, leading to a largely reduced supernet size and better searched networks. These methods have shown promising results in achieving high compression rates with minimal or no loss of accuracy. In research[146], the proposed methods incorporate parameter sharing among candidate architectures to enable efficient search over a large design space and consistently outperform manual design and random search approaches, achieving up to 1.0% absolute word error rate reduction and 28% relative model size reduction on the Switchboard corpus.

Moreover, many studies have applied parameter sharing methods to practical applications. For example, EfficientTDNN[147] is a novel speaker embedding architecture search framework that employs weight-sharing subnets to efficiently search for TDNN architectures, achieving a favorable trade-off between accuracy and efficiency with parameter sharing, as demonstrated on the VoxCeleb dataset. CpRec[148] is a compressed sequential recommendation framework that employs block-wise adaptive decomposition and layer-wise parameter sharing schemes to reduce the number of parameters in sandwich-structured DNNs used in sequential recommender systems. Sindhwani et al.

[149] proposed a unified framework for learning structured parameter matrices with low displacement rank, enabling a rich range of parameter sharing configurations that offer superior accuracy-compactness-speed tradeoffs for mobile deep learning.

Discussion: While parameter sharing can lead to significant reductions in the number of parameters required to represent a neural network, there are also several disadvantages to this approach. One major limitation is that parameter sharing can result in a loss of model accuracy, particularly when the shared parameters are not well-suited for the task at hand. Additionally, parameter sharing may not be compatible with certain types of neural networks or architectures, limiting its applicability. Another potential issue is that parameter sharing can result in a less interpretable model, as it may be more difficult to understand how the shared parameters contribute to the network's decision-making process. Finally, some parameter sharing methods may be computationally expensive or require significant computational resources, particularly when searching for the optimal set of shared parameters.

3. Model Quantization

The quantization technique has become increasingly important for optimizing DNN models for deployment on resource-constrained devices. It offers multiple benefits, including improved computational efficiency, reduced memory and storage usage, and lower power consumption. By reducing the precision of model parameters and activations, quantization enables a significant reduction in model size while minimizing the impact on task accuracy. It is a widely adopted technique in the field of deep learning and has been shown to offer significant improvements in model efficiency and performance on a range of hardware platforms.

Recently, there has been a growing interest in applying quantization techniques to design lightweight models for edge devices. This trend is driven by the need to improve the efficiency and performance of ML on edge devices, which often have limited resources. One such technique is progressive fractional quantization and dynamic fractional quantization, which were proposed in FracTrain by Fu et al.[150]. This method reduces the computational cost and energy/latency of DNN training while maintaining comparable accuracy. Similarly, edgeBERT[151] is a hardware system that employs a combination of adaptive attention span, selective network pruning, and floating-point quantization to alleviate computation and memory footprint overheads on resource-constrained edge platforms. PNMQ is a data-free method for network compression using Parametric Non-uniform Mixed precision

Quantization, which allows efficient implementations for network compression on edge devices without requiring any model retraining or expensive calculations[152]. Another technique for quantized deep neural networks (QDNNs) is SPEQ[153], a novel stochastic precision ensemble training method that employs KD without the need for a separate teacher network. Moreover, Cui et al. [154] proposed a quantization-based approach for reducing the storage and memory consumption of deep ensemble models, using a differentiable and parallelizable bit sharing scheme that allows the members to share less significant bits of parameters, while maintaining accuracy, and an efficient encoding-decoding scheme for real deployment on edge devices.

Some studies have also explored the quantization of other neural networks. For example, Capsule Networks (CapsNets) have been shown to outperform traditional CNNs in image classification, but they are computationally intensive and challenging to deploy on resource-constrained edge devices. A specialized quantization framework for CapsNets has been developed to enable efficient edge implementations, which can reduce the memory footprint by 6.2x with minimal accuracy loss[155]. Moreover, FSpiNN is a memory and energy-efficient framework for spiking neural networks (SNNs) that incorporates fixed-point quantization while maintaining accuracy, making it suitable for deployment on edge devices with unsupervised learning capability[156].

Hardware-based quantization design are also a recent research hotspot. Zhou et al.[157] proposed an INT8 training method implemented in Octo, a lightweight cross-platform system that achieves higher training efficiency and memory reduction over full-precision training on commercial AI chips. Similarly, Wang et al.[158] introduced the Hardware-Aware Automated Quantization (HAQ) framework, which determines the optimal quantization policy for each layer in a DNN based on the hardware architecture, resulting in reduced latency and energy consumption without significant accuracy loss. Li et al.[159] devised a novel quantization framework, RaQu, that combines information about neural network models and hardware structures to improve resource utilization and computation efficiency on resistive-memory-based processing-in-memory (RRAM-based PIM) for edge devices. Additionally, a hardware/software co-design solution is proposed via an inexact multiplier and a retraining strategy to quantize neural network weights to Fibonacci encoded values for computationally demanding algorithms on resource-constrained edge devices[160].

Discussion

Model quantization is a useful technique for improving the efficiency and performance of deep neural networks on resource-constrained devices. However, it also has some drawbacks. One major limitation is that quantization can result in a loss of model accuracy, especially if the precision of model parameters and activations is reduced too much. Additionally, finding the optimal set of precision levels can be computationally expensive. Reduced precision can also make the model less interpretable, making it difficult to comprehend how the model is making its decisions. Moreover, certain hardware or software configurations may not be compatible with some quantization methods, limiting their practicality.

4. Knowledge Distillation

KD is a model compression technique that aims to reduce the size and computational cost of DNNs by transferring knowledge from a large and complex teacher model to a smaller and simpler student model. It works by softening the teacher's output into a probability distribution and using it to train the student model to mimic the teacher's behavior. This technique allows for the compression of complex models while still maintaining their performance, making them more efficient and practical for deployment in real-world applications.

KD, introduced by Geoffrey Hinton et al.[161], has been effectively employed in various domains. One approach is the self-distillation framework proposed by Zhang et al.[162] that enhances the performance of CNNs by compressing the knowledge within the network. This framework improves accuracy while providing depth-wise scalable inference on resource-limited edge devices. Another approach is the DynaBERT model introduced by Hou et al.[163], which is a dynamically adjustable BERT model that can adapt to the requirements of different edge devices by selecting adaptive width and depth. The training process involves KD from the full-sized model to small sub-networks, resulting in superior performance compared to existing BERT compression methods. Zhang et al.[164] proposed the SCAN framework, which divides DNNs into multiple sections and constructs shallow classifiers using attention modules and KD. This framework's threshold-controlled scalable inference mechanism allows for sample-specific inference, resulting in significant performance gains on CIFAR100 and ImageNet. In addition, Zhang et al.[165] proposed the dynamic knowledge distillation (DKD) framework for deep CNNs, which leverages a dynamic global distillation module for multiscale features imitation and a dynamic instance selection distillation module for self-judgment. The framework also tailors a training-status-aware loss to handle hard samples in regression, enabling the deployment of models on low computation edge devices such as satellites and unmanned aerial vehicles. Hao et al.[166] designed the CDFKD-MFS framework, which compresses multiple pretrained models into a tiny model for resource-limited edge devices without requiring the original dataset. This framework utilizes a multi-header student module, an asymmetric adversarial data-free KD module, and an attention-based aggregation module. Additionally, Hao et al.[167] proposed a fine-grained manifold distillation method for transformer-based networks to compress the architecture of vision transformers into compact students, achieving high accuracy with lower computational costs. Zhang et al.[168] used a comparable model to teach lexical knowledge to its counterpart model, achieving significant performance gain. Furthermore, Shen et al.[169] used two models to teach each other and reach the trade-off between the two models. Their results show the teacher models in KD don't necessarily have to be larger or much stronger models.

In addition, various other KD methods have been proposed for different applications. For example, Liu et al.[138] proposed novel content-aware KD and effective channel pruning schemes specialized for unconditional GANs, achieving a substantial improvement over the state-of-the-art compression method. Ni et al.[170] introduced an end-to-end Vision-to-Sensor KD (VSKD) framework for human activity recognition (HAR) based on a multi-modal approach, which reduces computational demands on edge devices and produces a learning model that closely matches the performance of the computational expensive approach, using only time-series data. Jin et al.[171] presented a Personalized Federated Learning (PFL) framework for edge devices, named pFedSD, which utilizes self-KD to train models that perform well for individual clients. By distilling knowledge from previous personalized models, pFedSD accelerates the process of recalling personalized knowledge and provides an implicit ensemble of local models. Li et al.[172] developed an instance-specific multi-teacher KD model (IsMt-KD) for distracted driver posture classification on embedded systems with limited memory space and computing resources. This model utilizes an instance-specific teacher grading module to dynamically assign weights to teacher models based on individual instances, achieving high accuracy and real-time inference on edge hardware platforms.

KD methods can be combined with other model compression approaches to further improve the performance of DNNs on edge devices. For instance, Boo et al.[153] proposed a stochastic precision ensemble training scheme for QDNNs that utilizes KD with a continuously changing teacher model formed by sharing the student network's parameters. This allows for improved performance in various edge device tasks without the need for cumbersome teacher networks. Xia et al.[173] presented an on-device recommender system for a session-based recommendation that uses ultra-compact models and a self-supervised KD framework to address the challenges of limited memory and computing resources. The compressed model achieves a 30x size reduction with almost no accuracy loss and even outperforms its uncompressed counterpart. Xu et al.[174] devised a lightweight Identity-aware Dynamic Network (IDN) for subject-agnostic face swapping on edge devices, which utilizes an efficient Identity Injection Module (IIM) and a KD-based method for stable training. Moreover, the proposed lightweight SegFormer model in[175] for efficient semantic segmentation on edge devices utilizes a dynamic gated linear layer to prune uninformative neurons based on input instance and a two-stage KD to transfer knowledge from the original teacher to the pruned student network, achieving more than 60% computation savings with a minimal drop in mIoU.

Discussion

KD is a powerful method for compressing complex models while maintaining their performance, enabling efficient deployment on resource-limited edge devices. However, it can result in a loss of model accuracy if the precision level is not appropriately chosen, and it can be computationally expensive, particularly when dealing with large datasets or complex models. Additionally, while there are many successful applications of KD, its effectiveness can vary depending on the specific domain and task, and it may require careful tuning and experimentation to achieve optimal results.

5. Low-rank Factorization

DNNs often require high memory consumption and large computational loads, which limits their deployment on edge or mobile devices. Low-rank factorization is a method that can help by approximating weight matrices with low-rank matrices, finding a lower-dimensional representation of the data that retains the most important information. For example, SVD training[176] is a new method that achieves low-rank DNNs during training without applying SVD on every step, using sparsity-inducing regularizers on singular values. This method achieves a higher reduction in computation load under the same accuracy compared to previous factorization and filter pruning methods. Efforts have also been made to apply low-rank factorization on resource-constrained edge devices. MicroNet[177] is an efficient CNN designed for edge devices, achieving low computational cost by using Micro-Factorized convolution that factorizes pointwise and depthwise convolutions into low-rank matrices. The network compensates for network depth reduction with the introduction of the Dynamic Shift-Max activation function. MicroNet-M1 achieves 61.1% top-1 accuracy on ImageNet classification with 12 MFLOPs, outperforming MobileNetV3 by 11.3%.

Discussion

Low-rank factorization is a promising approach for reducing the computational and memory requirements of DNNs, making them more practical for deployment on resource-limited edge devices. However, implementing low-rank factorization can be challenging due to the high computational cost of the factorization operation, and the need for extensive retraining to achieve convergence.

V. System Optimization for Edge AI Deployment

As the demand for real-time performance and resource-efficient deep learning models increases, system optimization has become a crucial area of research. To address the need for deploying deep learning models on edge devices, it is necessary to optimize their computational efficiency. In this section, we present frameworks for lightweight model training and inference from a software perspective, as well as methods for accelerating models using hardware-based approaches. The workflow of system optimization is shown in Figure 9.

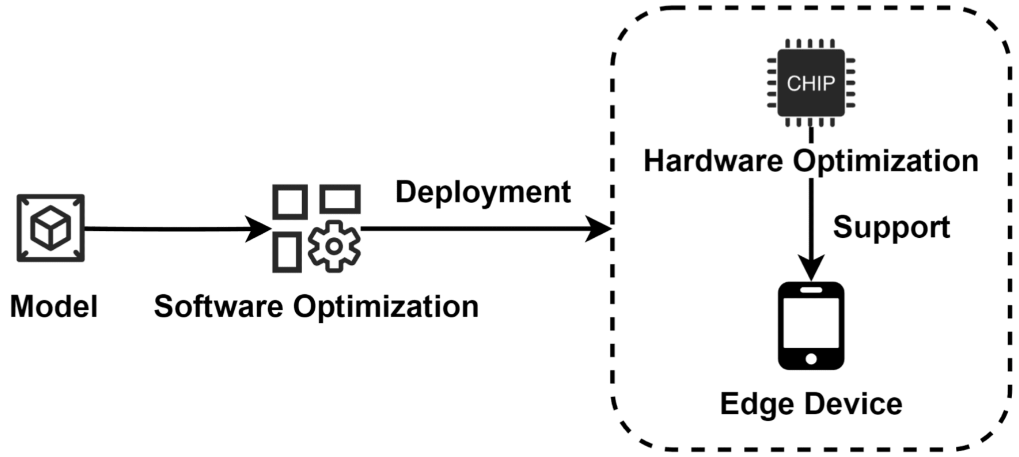

Figure 9. An overview of system optimization operations. Software optimization involves developing frameworks for lightweight model training and inference, while hardware optimization focuses on accelerating models using hardware-based approaches to improve computational efficiency on edge devices.

A. Software Optimization

1. Edge AI Learning Frameworks

PyTorch and TensorFlow are widely-used deep learning frameworks, but they may not be suitable for mobile applications due to their relative heaviness and third-party dependencies, which can be problematic for mobile devices. However, with the evolution and development of the AI ecosystem, these frameworks have been specifically designed for mobile deep learning through TensorFlow Lite and PyTorch Mobile, allowing for efficient training and deployment on mobile devices.

Both TensorFlow Lite and PyTorch Mobile are lightweight deep learning frameworks designed specifically for mobile applications. They provide a more streamlined development process and facilitate effective model training and deployment on mobile devices. The restricted computing and memory capabilities of mobile devices have been taken into account in the optimization of these frameworks for them. This optimization makes the models suitable for edge computing scenarios as they can be trained and deployed on mobile devices with less resource consumption. TensorFlow Lite and PyTorch Mobile have a range of features that are specifically designed for mobile applications. Specifically, some features of these two frameworks are highlighted in Table 4. Moreover, TensorFlow Lite is found to perform better with lightweight deep learning models and is suitable for edge computing applications, especially for mobile devices[178].

Framework	Producer	Highlights
TensorFlow Lite	Google	• Optimization of on-device ML by addressing: latency, privacy, connectivity, size, and power consumption • Support for multiple platforms including Android, iOS, embedded Linux, and microcontrollers • Support for multiple programming languages, including Java, Swift, Objective-C, C++, and Python • High-performance with hardware acceleration and model optimization support • Examples for common ML tasks on various platforms, including image/object/text classification
Pytorch Mobile	Facebook	• Works on iOS, Android, and Linux platforms • Provides APIs for common preprocessing and integration tasks • Supports tracing and scripting via TorchScript IR • Offers XNNPACK floating point and QNNPACK 8-bit quantized kernels for Arm CPUs • Provides an efficient mobile interpreter, build level optimization, and streamline model optimization via optimize__for__mobile

Table IV. Edge AI Learning Frameworks

2. *Edge AI Inference Frameworks*

Lightweight model inference is becoming increasingly important for applications at the edge, where computational resources are often limited. To address this, a number of software frameworks have emerged that allow for efficient inference of lightweight models, such as NCNN, OpenVINO and ONNX Runtime. These frameworks typically provide optimized implementations of common operations and architectures, and can run on a variety of hardware platforms, including IoT devices, mobile devices, and edge servers. In Table 5, we have presented a list of commonly used AI inference frameworks, including their manufacturers, supported hardware, advantages, and limitations.

Framework	Producer	Supported Hardware	Advantages	Limitations
ONNX Runtime[179]	Microsoft	CPU, GPU, etc	• It has built-in optimizations that can boost inferencing speed up to 17 times and training speed up to 1.4 times • It supports multiple frameworks, operating systems, and hardware platforms • High performance, and low latency	• Limited support for non-ONNX models • No support for some hardware backends
OpenVINO[180]	Intel	CPU, GPU, VPU, FPGA, etc	• It optimizes deep learning pipelines for high performance and throughput • Support for advanced functions such as FP16, INT8 quantization • It supports multiple deep learning frameworks and multiple operating systems	• Only Intel hardware products are supported • Deploying and integrating models still requires some technical knowledge and experience
NCNN[181]	Tencent	CPU, GPU, etc	• High performance and low memory usage • Supports a variety of hardware devices and model formats • Supports 8-bit quantization and ARM NEON optimization	• Limited support for non-NCNN models
Arm NN[182]	Arm	CPU, GPU, etc	• Cross platform • Supports a variety of hardware devices and model formats • Existing software can automatically take advantage of new hardware features	• Limited support for operators and network structures

Framework	Producer	Supported Hardware	Advantages	Limitations
			• Support for ARM Compute Library	
MNN[183]	Alibaba	CPU, GPU, NPU	• MNN is a lightweight, device-optimized framework with quantization support • MNN is versatile, supporting various neural networks and models, multiple inputs/outputs, and hybrid computing on multiple devices • MNN achieves high performance through optimized assembly, GPU inference, and efficient convolution algorithms. • MNN is easy to use, with support for numerical calculation, image processing, and Python API	• Limited community support • Technical expertise required
TensorRT[184]	NVIDIA	CPU, GPU	• Maximize throughput by quantifying the model to INT8 while maintaining high accuracy • Optimize GPU video memory and bandwidth usage by merging nodes in the kernel • Select the best data layer and algorithm based on the target GPU platform • Minimize video memory footprint and efficiently reuses memory for tensors • An extensible design for processing multiple input streams in parallel	• It only runs on NVIDIA graphics cards • It does not open source the kernel

Framework	Producer	Supported Hardware	Advantages	Limitations
TVM[185]	Apache	CPU, GPU, DSP, etc	• Compilation and minimal runtimes optimize ML workloads on existing hardware for better performance • Supports a variety of hardware devices and model formats • TVM's design enables flexibility for block sparsity, quantization, classical ML, memory planning, etc	• Deploying and integrating models still requires some technical knowledge and experience

Table V. Edge AI Inference Frameworks

Efficient AI inference frameworks for edge devices have seen rapid progress in academic research recently, with a focus on deploying CNN models in resource-constrained scenarios. Xia et al. [186] introduced a lightweight neural network architecture called SparkNet, which reduces weight parameters and computation demands for CNN feedforward inference on edge devices. Memsqueezer, an on-chip memory architecture for deep CNN inference acceleration on mobile/embedded devices, was designed by Wang et al.[187], achieving 2x performance improvement and 80% energy reduction with compression and redundancy detection. Wang et al.[188] presented an end-to-end solution for CNN model inference on integrated GPUs at the edge, using a unified IR and ML-based scheduling search schemes. Pipe-it, a pipelined framework limiting parallelization of convolution kernels to assigned clusters, was proposed by Wang et al.[189] for CNN inference on ARM big.LITTLE architecture in edge devices. SCA, a secure CNN accelerator using stochastic computing to protect models and weights during both training and inference phases, was devised by Zhao et al.[190], achieving significant speedup and energy reduction over non-secure and inference-only secure baselines. Hou et al.[191] developed NeuLens, a dynamic CNN acceleration framework for mobile and edge platforms that achieves significant latency reduction and accuracy improvement using a novel dynamic inference mechanism and operation reduction compatible with hardware-level accelerations.

Studies have also focused on deploying recurrent neural network (RNN) models. For example, Srivastava et al.[192] proposed an efficient RNN compression method for human action recognition, which uses a Variational Information Bottleneck theory-based pruning approach and a group-lasso regularization technique to significantly reduce model parameters and memory footprint. EdgeDRNN was designed by Gao et al.[193] for edge RNN inference with a batch size of 1, adopting the delta network algorithm to exploit temporal sparsity in RNNs to realize high performance and power efficiency. Wen et al.[194] developed a structured pruning method through neuron selection to reduce the overall storage and computation costs of RNNs, achieving significant practical speedup during inference without performance loss. Zhang et al.[195] proposed DirNet, a model compression approach for RNNs that dynamically adjusts the compression rate and sparsity of sparse codes across hierarchical layers while maintaining minimum accuracy loss.

Numerous works have explored the deployment of DNNs on edge devices to optimize performance and resource utilization. For example, Ding et al.[196] proposed a new task-mapping programming paradigm to embed scheduling into tensor programs for efficient DNN inference and introduced the Hidet deep learning compiler. Liu et al.[197] devised edgeEye, a high-level API for real-time intelligent video analytics applications, allowing developers to focus on application logic, while Wang et al. [158] developed the HAQ framework, which utilizes reinforcement learning to determine the optimal quantization policy for each layer of a DNN, taking into account the specific hardware architecture and resource constraints. Xie et al.[198] proposed GRACE, a DNN-aware compression algorithm that optimizes the compression strategy for a target DNN model, enabling efficient source compression at IoT devices without disturbing the inference performance. Huang et al.[199] designed LcDNN, a lightweight collaborative DNN for the mobile web that executes a lightweight binary neural network (BNN) branch on the mobile device to reduce the model size, accelerates inference, and reduces energy cost. Farhadi et al.[200] proposed a system-level design to improve the energy consumption of object detection on resource-limited user-end devices by deploying a shallow neural network (SHNN) and implementing a knowledge transfer mechanism to update the SHNN model using DNN knowledge from a powerful edge device. Yang et al.[201] designed DA3, a memory-efficient on-device multi-domain learning approach that reduces activation memory usage during training on resource-limited edge devices while maintaining high accuracy performance. Additionally, Kosta et al.[202] proposed RAPID-RL, an architecture for efficient deep reinforcement learning that allows conditional activation of DNN layers based on input difficulty level, dynamically adjusting computational effort during inference while maintaining performance. Table 6 summarizes the optimized models, goals, and performance of these studies. Moreover, some studies have explored other models, such as Weightless Neural Networks (WNN)[203], Binarized Neural Networks (BNN)[128][204], Long Short-Term Memory (LSTM)[205], transformer[206] and Graph Neural Networks (GNN)[207][208][209].

Method	Model	Goal	Performance
SparkNet[186]	CNN	• To reduce parameters and computation demands	• Compress CNN by a factor of 150x • Performance: 337.2 GOP/s • Energy efficiency: 44.48 GOP/s/w
Memsqueezer[187]	CNN	• To enable CNN inference on edge devices	• 2x performance improvement • 80% energy consumption reduction
UGIO[188]	CNN	• To propose an end-to-end solution for CNN inference on edge devices	• Achieves similar, or even better (up to 1.62x), performance
ACG-Engine[210]	CNN	• To address the performance bottleneck of instance normalization in generative networks on edge devices	• Speed: 4.56x • Power efficiency: 29x
Pipe-it[189]	CNN	• To perform CNN inference on ARM big.LITTLE architecture in edge devices	• Achieve a 39% higher throughput
SCA[190]	CNN	• Enable IP protection when deploying CNN on edge devices	• 4.8x speedup over a non-secure baseline • 34.2x speedup over an inference-only secure baseline • Reduce 84.3% over a non-secure baseline • Reduce 98.5% speedup over an inference-only secure baseline
NeuLens[191]	CNN	• To achieve latency reduction and accuracy improvement in CNN acceleration on edge systems	• Reduce up to 58% latency • Improve up to 67.9% accuracy improvement

Method	Model	Goal	Performance
TS-VIB-LSTM[192]	RNN	• To compress RNN for human action recognition on edge devices	• More than 70x compression rate
edgeDRNN[193]	RNN	• To enable a low-latency, low-power RNN accelerator for real-time applications on edge devices	• It updates a 5M 2-layer GRU-RNN in 0.5ms for low-latency edge inference • 5x faster than commercial edge AI platforms • It achieves 20.2GOp/s throughput and 4x higher power efficiency than commercial platforms
SP-RNN[194]	RNN	• To enable the deployment of RNN models on edge devices through network pruning techniques	• Nearly 20x speedup • Without performance loss
DirNet[195]	RNN	• To introduce a model compression approach for RNNs to make it can be deployed on resource constrained devices	• Reduce 8x model size • With negligible performance loss
Hidet[196]	DNN	• To design efficient tensor programs for deep learning operators	• Outperform by up to 1.48x (1.22x on average) • Reduce tuning time by 11x
edgeEye[196]	DNN	• For real-time intelligent video analytics applications	• Achieve efficient and high-performance deep learning inference
HAQ[158]	DNN	• Optimize specialized neural network for a particular hardware architecture	• Reduce latency by 1.4 - 1.95x • Reduce energy consumption by 1.9x

Method	Model	Goal	Performance
			• With the negligible loss of accuracy
GRACE[198]	DNN	• To enable efficient source compression at IoT devices without disturbing the inference performance	• Reduce a source size by 23% • Achieves 7.5% higher inference accuracy • Reduce 90% bandwidth consumption
LcDNN[199]	DNN	• To enable deep learning on the mobile web	• Reduce the model size by 16x - 29x • Reduce the end-to-end latency • Reduce the mobile energy cost
DA3[201]	DNN	• To enable on-device multi-domain learning	• Reduce training memory by 5x - 37x • Reduce training time by 2x
RAPID-RL[202]	DNN	• To enable deep RL systems to be deployed on edge devices	• Incur 0.34x (0.25x)'s operations while maintaining 0.88x (0.91x)'s performance

Table VI. Summary of Model Inference Methods from the Literature

B. Hardware Optimization

In addition to software optimizations, hardware acceleration is crucial for achieving high performance for lightweight models on edge devices. Various approaches to hardware acceleration include using specialized processors such as Central Processing Unit (CPU), Graphic Processing Unit (GPU), Field Programmable Gate Array (FPGA), Application Specific Integrated Circuit (ASIC), and Neural Processing Unit (NPU), as well as implementing custom hardware designs for specific models. Table 7 provides information on common processors for edge devices, including basic information, examples, and features.

Hardware	Basic Information	Examples	Advantages	Limitations
CPU	A kind of universal computing equipment	• ARM Cortex-M55 • Intel Atom x7-E3950 • Qualcomm Snapdragon 865 • Apple A14 Bionic • MediaTek Helio P90	• It has a wide range of application scenarios and versatility • Stable computing performance • Rich hardware and software ecosystem and support	• CPU performance and efficiency may not be as good as dedicated GPU and ASIC
GPU	Can be used to accelerate deep learning algorithms	• Qualcomm Adreno 640 • Imagination Technologies PowerVR Series9XE • Intel Iris Plus Graphics G7 • NVIDIA Tegra X1 • ARM Mali-G76	• High parallelism • Flexibility • Wide application support	• High power consumption
FPGA	To accelerate deep learning algorithms through customized hardware logic	• Lattice sensAI • QuickLogic EOS S3 • Xilinx Zynq UltraScale+ • Intel Movidius Neural Compute Stick	• Highly flexible • Low power consumption • Customizable	• High development and deployment costs

Hardware	Basic Information	Examples	Advantages	Limitations
ASIC	To achieve high performance and power efficiency through hardware optimization	• Google edge TPU • Horizon Robotics Sunrise • MediaTek NeuroPilot • Cambricon MLU100	• Highly optimized hardware structure • Low power consumption • High performance	• High development and production costs
NPU	Designed to accelerate deep learning algorithms	• Qualcomm Hexagon 680 • Apple Neural Engine • Huawei Kirin 990 • MediaTek APU • NVIDIA Jetson Nano	• High efficiency • Low power consumption • Highly optimized hardware structure	• Limited support for some models

Table VII. Edge AI Model Accelerator

In recent years, there has been a growing interest in using CPU-based accelerators for model inference on edge devices due to their versatility and stable computing performance. REDUCT[211] is a solution that bypasses traditional CPU resources to enable efficient data parallel DNN inference workloads, achieving significant performance/Watt improvements and raw performance scaling on multi-core CPUs. By maximizing the utilization of existing bandwidth resources and distributing lightweight tensor compute near all caches, REDUCT achieves performance similar to or better than state-of-the-art Domain Specific Accelerators (DSA) for DNN inference. Similarly, Zhu et al.[212] designed the Neural CPU (NCPU) architecture to optimize the end-to-end performance of low-cost embedded systems. It combines a binary neural network accelerator with an in-order RISC-V CPU pipeline, achieving energy savings and area reduction compared to conventional heterogeneous architectures, while supporting flexible programmability and local data storage to avoid costly core-to-core data transfer.

GPUs are recognized as powerful hardware accelerators for deep learning, thanks to their highly parallel architecture and programmability. Capodieci et al.[213] presented a prototype real-time scheduler for GPU activities on an embedded SoC featuring an NVIDIA GPU architecture, with preemptive EDF scheduling and bandwidth isolations using a Constant Bandwidth Server. FPGAs are also increasingly used as hardware accelerators for deep learning due to their highly flexible and customizable hardware logic. For example, Xia et al.[186] proposed an FPGA-based accelerator architecture specifically built for SparkNet, achieving high performance and energy efficiency with a fully pipelined CNN hardware accelerator. Choudhury et al.[214] proposed an FPGA overlay for efficient CNN processing that exploits all forms of parallelism inside a convolution operation and can be scaled based on available compute and memory resources. Yu et al.[215] explored FPGA-based overlay processors for an efficient acceleration of lightweight operations in LW-CNNs, utilizing a corresponding compilation flow. Overall, GPUs and FPGAs offer different advantages for hardware acceleration, with GPUs providing powerful parallel computing capabilities and FPGAs offering highly customizable hardware logic for efficient and scalable acceleration.

ASICs are increasingly popular as dedicated hardware accelerators for deep learning, owing to their highly optimized hardware structure and low power consumption. For example, Tambe et al. [151] proposed edgeBERT for multi-task NLP inference, which adopts algorithm-hardware co-design and early exit prediction based on entropy. Roohi et al.[216] designed ApGAN to alleviate the computationally intensive GAN problem, especially for running on resource-constrained edge devices. NPUs can be considered ASICs designed specifically for AI, and they are also gaining popularity as dedicated hardware accelerators for deep learning due to their high efficiency and low power consumption. Some studies aim to improve task performance by enhancing the NPU. For example, Kouris et al.[217] introduced two new dimensions to the NPU hardware architecture design space, Fluid Batching, and Stackable Processing Elements, to improve NPU utilization. Yang et al.[218] proposed BitSystolic, a 2b-8b NPU that supports mixed-precision DNN models with configurable numerical precision and configurable data flows. Multiple measures are adopted to improve hardware utilization, such as in PL-NPU[219], where a posit-based logarithm-domain processing element, a reconfigurable inter-intra-channel-reuse dataflow, and a pointed-stake-shaped codec unit are employed. Some studies have also used acceleration methods that combine multiple processors, such as FPGA +

GPU[220] and CPU + GPU[221]. Overall, ASICs and NPUs offer highly optimized hardware structures and low power consumption, making them ideal for efficient and high-performance deep learning on edge devices. Specifically, the basic information on these hardware acceleration methods is listed in Table 8.

Method	Hardware	Model	Strategy	Performance
REDUCT[211]	CPU	DNN	• It bypasses CPU resources to optimize DNN inference	• 2.3x increase in convolution performance/Watt • 2x to 3.94x scaling in raw performance • 1.8x increase in inner-product performance/Watt • 2.8x scaling in performance
NCPU[212]	CPU	BNN	• Propose a unified architecture	• Achieved 35% area reduction and 12% energy saving compared to conventional heterogeneous architecture • Implemented two-core NCPU SoC achieves an end-to-end performance speed-up of 43% or an equivalent 74% energy saving
Prototype[213]	GPU	DNN	• The schedulability of repeated real-time GPU tasks is significantly improved	• Achieved 35% area reduction and 12% energy saving compared to conventional heterogeneous architecture • Implemented two-core NCPU SoC achieves an end-to-end performance speed-up of 43% or an equivalent 74% energy saving
SparkNoC[186]	FPGA	CNN	• Simultaneous pipelined work	• Performance: 337.2 GOP/s • Energy efficiency: 44.48 GOP/s/w

Method	Hardware	Model	Strategy	Performance
FPGA Overlay[214]	FPGA	CNN	• It exploits all forms of parallelism inside a convolution operation	• An improvement of 1.2x to 5x in maximum throughput • An improvement of 1.3x to 4x in performance density
Light-OPU[215]	FPGA	CNN	• With a corresponding compilation flow	• Achievement of 5.5x better latency and 3.0x higher power efficiency on average compared with NVIDIA Jetson TX2 • Achievement of 1.3x to 8.4x better power efficiency compared with previous customized FPGA accelerators
edgeBert[151]	ASIC	Transformer	• It employs entropy-based early exit predication	• The energy savings are up to 7x, 2.5x, and 53x compared to conventional inference without early stopping, latency-unbounded early exit approach
ApGAN[216]	ASIC	GAN	• By binarizing weights and using a hardware-configurable in-memory addition scheme	• Achieve energy efficiency improvements of up to 28.6x • Achieve a 35-fold speedup
Fluid Batching[217]	NPU	DNN	• Fluid Batching and Stackable Processing Elements are introduced	• 1.97x improvement in average latency • 6.7x improvement in tail latency SLO satisfaction

Method	Hardware	Model	Strategy	Performance
BitSystolic[218]	NPU	DNN	• Based on a systolic array structure	• It achieves high power efficiency of up to 26.7 TOPS/W with 17.8 mW peak power consumption
PL-NPU[219]	NPU	DNN	• A posit-based logarithm-domain processing element, a reconfigurable inter-intra-channel-reuse dataflow, and a pointed-stake-shaped codec unit are employed	• 3.75x higher energy efficiency • 1.68x speedup
FARNN[220]	FPGA + GPU	RNN	• To separate RNN computations into different tasks that are suitable for GPU or FPGA	• Improve by up to 4.2x
DART[221]	CPU + GPU	DNN	• It offers deterministic response time to real-time tasks and increased throughput to best-effort tasks	• Response time was reduced by up to 98.5% • Achieve up to 17.9% higher throughput

Table VIII. Summary of edge AI Model Accelerator from the Literature

VI. Application Scenarios

Edge AI is a technology that deploys AI algorithms and models on edge devices such as smartphones, cameras, sensors, and robots, enabling data processing and decision-making at the device end. This approach avoids data transmission delays and privacy issues while improving response speed and security. Edge AI has a wide range of application scenarios, including smart homes, industrial automation, healthcare, and many others. With edge AI, devices can perform advanced tasks such as object detection, face recognition, and action recognition without relying on the cloud or other external computing resources.

A. Smart Homes

In traditional smart home systems, sensors, cameras, and other devices collect home environment data and transmit it to the cloud for processing and decision-making. However, data transmission delays and privacy issues can cause real-time and security problems in the system, greatly reducing the quality of the user experience. Edge AI technology enables the deployment of AI algorithms and models on smart home devices, allowing for data processing and decision-making to occur at the device level. This approach can enable many aspects of home life, such as electricity demand forecasting[222], human activity prediction[223], and energy management[224]. In particular, edge AI techniques effectively circumvent issues related to data transmission delay and privacy, resulting in improved security and response times. For example,[225] identifies new vulnerabilities and attacks in widely-used smart home platforms by analyzing the complex interactions among the participating entities. With edge AI, smart home devices can become more intelligent and autonomous, providing benefits such as energy efficiency, improved security, and enhanced user experience.

B. Industrial Automation

Smart industry is a technological approach that aims to improve the efficiency and quality of production by deploying AI algorithms at the device level[226]. This is achieved through the implementation of industrial automation, which involves the use of advanced technologies and systems to control and monitor industrial processes. By deploying AI algorithms on industrial devices, smart industry enables machines to learn from data, adapt to changes in the production environment, and optimize their own operations in real-time. This results in improved efficiency, reduced waste, and enhanced product quality, which ultimately leads to increased profitability for industrial businesses[227]. The application of smart industry has been shown to have a significant impact on various aspects of industrial operations, including maintenance, quality control, and supply chain management. By leveraging AI-powered analytics and predictive maintenance, industrial companies can reduce downtime and increase the lifespan of their equipment while also improving product quality through real-time monitoring and control. Overall, the smart industry represents a major technological advancement in the field of industrial automation, providing businesses with the tools and capabilities needed to optimize their production processes and remain competitive in today's fast-paced business environment.

C. Healthcare

Edge AI technology can provide faster and more accurate disease diagnosis and treatment for intelligent healthcare systems, thereby improving medical efficacy and efficiency. Recent research in healthcare has explored various aspects of the field, such as heart patient analysis[228], voice disorder detection[229], COVID-19 analysis[230] [231], and pathology detection[232] [233]. In[234], a privacy-preserving Faster R-CNN framework is proposed for object detection in medical images using additive secret sharing techniques and edge computing. This approach ensures the privacy and security of medical data while also enabling accurate and efficient diagnosis. The use of edge AI in healthcare also allows for real-time monitoring of patient vitals and early detection of potential health issues, leading to improved patient outcomes. Overall, edge AI holds great promise for the healthcare industry, providing opportunities for improved medical diagnosis, treatment, and patient care.

D. Autonomous Vehicles

In the field of autonomous driving, real-time performance is critical for ensuring safe and efficient operation. Previous research has extensively studied the collaborative work between cloud/edge and edge computing, such as data scheduling[235], resource allocation[236], perception information sharing[237], object detection[238], and computation offloading[239][240]. By utilizing edge algorithms on vehicles, it is possible to decrease the dependence on cloud computing resources and enhance response speed. For example, in[241], a compiler-aware pruning search framework was proposed, which enables real-time 3D object detection on resource-limited mobile devices for autonomous driving. This approach allows for faster and more accurate detection of objects in the environment, improving the safety and efficiency of autonomous vehicles. In the future, we can expect to see more edge AI technologies developed and applied to empower autonomous driving on local devices.

E. Public Safety

The introduction of edge AI technology into public safety has significant potential for providing faster, more accurate, and reliable responses to public safety incidents. For example, Wang et al.

[242] proposed the Surveiledge system, which enables real-time queries of large-scale surveillance video streams through a collaborative cloud-edge approach, balancing load among different computing nodes and achieving a latency-accuracy tradeoff. Experiments show that Surveiledge significantly reduces bandwidth costs and query response times compared to cloud-only solutions while improving query accuracy and speed compared to edge-only approaches. Trinh et al. [243] investigated techniques for detecting urban anomalies by utilizing mobile traffic monitoring, where they proposed the use of the mobile network as an additional sensing platform to monitor crowded events for potential risks to public safety. In addition, methods for improving performance by combining various techniques have also been proposed. For instance, in[244], the proposed Metropolitan Intelligent Surveillance System (MISS) combines IoT, cloud computing, edge computing, and big data to enable a unified approach for implementing an ISS at an urban scale, increasing performance and security levels. These applications of edge AI in public safety demonstrate the potential for improving the response time and accuracy of public safety incidents, enhancing situational awareness, and ultimately improving public safety.

F. Agriculture

Edge AI technology has shown promising potential for applications in the agriculture sector. For example, Menshchikov et al.[245] presented an approach for fast and accurate detection of the harmful and fast-growing Hogweed of Sosnowskyi using an unmanned aerial vehicle with an embedded system running Fully CNNs. The proposed approach achieves high accuracy in segmentation and processing speed for individual plants and leaves, which can provide comprehensive and relevant data to control the expansion of this plant. In addition, there are interesting applications in the agricultural sector, such as real-time strawberry detection[246] and pest management[247]. These applications leverage edge AI technology to provide real-time data analysis, enabling farmers to make informed decisions and optimize their crop yields. For example, real-time strawberry detection can help farmers quickly identify and harvest ripe fruit, while pest management can help to reduce the use of harmful chemicals by targeting pests more precisely.

G. Retail

Smart retail is a new retail mode that improves the efficiency and experience of retail through intelligent technology and data analysis. Retailers are increasingly leveraging edge AI to analyze data from in-store cameras and sensors to create intelligent stores[248]. One practical application of edge AI in smart retail is to alert store employees when shelf inventory levels are low, reducing the impact of stockouts. Another use case is the automated checkout system in unmanned supermarkets that utilizes edge AI algorithms and sensors to identify user and product information for automatic checkout. With the help of edge AI, intelligent customer service can also be achieved to better serve customers. In addition, edge AI can be utilized to analyze inventory loss caused by theft, errors, fraud, waste, and damage. By analyzing data from cameras and sensors, retailers can identify patterns and potential issues, allowing them to take proactive measures to reduce losses and improve profitability. Overall, the use of edge AI in smart retail has the potential to revolutionize the way we approach retail, providing retailers with the tools and capabilities needed to optimize their operations, reduce losses, and enhance customer experience.

H. Energy Management

Edge AI has been widely applied in the field of energy management[249], including energy monitoring, intelligent control, energy optimization, and smart device management. By real-time monitoring energy usage and collecting data, edge AI can assist energy managers in understanding energy consumption status and providing energy efficiency improvement suggestions through data analysis and prediction[250]. Furthermore, edge AI can achieve automated energy control, optimize energy consumption, improve energy efficiency, and reduce energy costs. For example, an edge AI-based intelligent control system can adjust the HVAC system automatically based on occupancy patterns and environmental conditions to reduce energy waste. Similarly, edge AI can optimize the operation of distributed energy resources, such as solar panels and wind turbines, to maximize energy production and minimize costs. In addition, edge AI can enable smart device management, allowing energy managers to remotely monitor and control energy consumption in real-time. By using edge AI to analyze data from smart devices, energy managers can identify potential issues and take proactive measures to prevent energy waste and reduce costs.

I. Logistics

Logistics automation has the potential to significantly boost production capacity. By incorporating intelligent systems, productivity can be improved, human error reduced, and work efficiency enhanced. In particular, edge AI has emerged as a key technology for enabling intelligent logistics systems. Through the deployment of edge AI at the unit-level of logistics terminals, such as intelligent sorting robots, unmanned aerial vehicles for express delivery, and logistics distribution robots, these terminals can be imbued with higher levels of intelligence. By utilizing edge AI algorithms, these intelligent terminals can learn from data, adapt to changes in their environment, and optimize their operations in real-time. For example, edge AI can enable intelligent sorting robots to rapidly sort packages based on weight, size, and destination, reducing the time required for manual sorting and increasing efficiency. In addition, edge AI can enable unmanned aerial vehicles for express delivery to autonomously navigate through complex environments, such as urban areas, and identify the optimal delivery routes, reducing delivery times and costs. Furthermore, edge AI-powered logistics distribution robots can autonomously navigate through warehouses and distribution centers, reducing the need for human intervention and increasing productivity.

VII. Challenges

Compared to server devices, edge devices are resource-constrained in multiple aspects, including their computational power, storage capacity, energy resources, and communication bandwidth. These limitations pose significant challenges for deploying AI models on edge devices.

A. Limited Computing Power

Edge devices typically deploy low-power processors with limited computational capabilities, often unable to handle the large and complex computations required by AI models. To enable effective AI on resource-constrained edge devices, several optimization techniques can be employed. One approach is to optimize algorithms to reduce unnecessary computation and improve computational efficiency. This can be achieved by designing lightweight models (such as MobileNets series[75][76][77]) or through NAS[111][113], model compression (pruning[137][125], quantization[150][157], parameter sharing[142][143], knowledge distillation[162][165], etc.), which reduce the size and complexity of AI models without compromising their accuracy and performance.

Another approach is to enhance the hardware performance of edge devices. This can be achieved through the use of specialized hardware, such as GPUs[213], FPGAs[186][214][215] or ASICs[151][216][219], which can accelerate computational tasks and improve performance. In addition, cloud-edge/edge-edge collaboration can be used to augment the computational capabilities of edge devices[251][252][253][16].

Overall, a combination of optimization techniques, hardware enhancements, and cloud-edge/edge-edge collaboration can be used to overcome the limitations of edge devices and enable more sophisticated AI applications on these devices. By leveraging these techniques, edge devices can perform complex computational tasks while conserving energy and maintaining computational accuracy and performance.

B. Limited Memory

Edge AI aims to deploy AI models directly on resource-constrained edge devices rather than in the cloud. However, edge devices have highly limited storage capacity compared to the cloud, posing challenges for edge AI implementation. For instance, state-of-the-art deep learning models can require hundreds of megabytes to over a gigabyte of storage, which exceeds the storage limits of many edge devices. As such, minimizing storage demands and avoiding large model and data storage requirements is critical for making edge AI feasible. To address this challenge, several methods have been adopted. As previously mentioned, model compression is an effective way to reduce the model size and storage demands in edge AI[137][153][157]. This technique reduces the storage requirements of AI models, enabling them to be deployed on edge devices with limited storage capacity. In addition, incremental learning can be used to dynamically update models on edge devices by learning from new data, avoiding the need to store large amounts of historical data[254]. This approach reduces the storage requirements of edge devices by only storing the most recent data and models. Furthermore, storing data and models in a distributed manner across multiple edge devices can alleviate the limited storage capacity of individual devices[255]. This approach enables edge devices to leverage the storage capacity of other devices, reducing the storage requirements of individual devices. Edge caching technology can also be utilized to cache data and models between edge devices and the cloud, reducing storage demands and communication costs[256][257][258]. This approach enables frequently used data and models to be stored locally on edge devices, reducing the need to access the cloud for every request. Overall, these techniques enable the deployment of AI on edge devices with limited storage capacity.

C. Power Consumption

Edge devices are often battery-powered with limited energy budgets, while AI models can have high computational demands. This mismatch between the energy limitations of edge devices and the intensive computations of AI models poses a key challenge for practical edge AI deployment. Balancing computational efficiency and energy consumption presents a challenging task, necessitating the adoption of energy-efficient algorithms and hardware that can effectively minimize energy consumption while preserving computational accuracy and performance. One solution to address energy limitations is software design, as demonstrated by the development of energy-efficient algorithms such as PhiNets[259]. These algorithms are designed to minimize computational requirements, enabling them to run efficiently on edge devices with limited energy resources. Hardware design is another approach to address energy limitations. Researchers have developed energy-efficient hardware, such as[260][186][261], that reduces energy consumption and improves computational efficiency. In addition, hardware and software co-design, such as[262], can optimize both hardware and software for energy efficiency. Energy management is another solution to address energy limitations. Researchers have proposed energy management techniques, such as the use of AI-based controllers[263][264], to optimize energy consumption in edge devices.

Overall, the adoption of energy-efficient algorithms and hardware, as well as energy management techniques, can effectively minimize energy consumption while preserving computational accuracy and performance in edge devices. As the technology continues to advance, we can expect to see even more innovative solutions for addressing the energy limitations of edge devices.

D. Limited Communication Bandwidth

Compared to servers, edge devices typically have limited communication bandwidth, making it challenging to transfer large amounts of data between the edge device and the cloud. The limited connectivity poses challenges for transmitting large volumes of data between devices and the cloud, which many AI models require. To minimize communication costs, it is necessary to reduce the amount of data transmitted. One approach to reducing data transmission is through data preprocessing algorithms, which reduce the amount of data that needs to be transmitted during communication[36][38]. These algorithms can be used to filter and compress data, enabling only relevant information to be transmitted. Edge caching technology is another approach that can be utilized to store data and models on edge devices, reducing the frequency of communication with the cloud and minimizing the amount of data transmitted[265]. This approach enables frequently used data and models to be stored locally on edge devices, reducing the need to access the cloud for every request. On-device computation is another technique that can be used to enable real-time response at the edge[266][267][268]. By performing computation on the edge device, only relevant data needs to be transmitted to the cloud, reducing communication costs and enabling faster response times. Overall, these techniques can be used to minimize the amount of data transmitted between edge devices and the cloud, enabling efficient communication and reducing communication costs.

E. Security and Privacy

Deploying AI models on edge devices poses significant security and privacy challenges due to the distributed nature of edge computing and the processing of sensitive data on devices outside the control of data owners. To address these challenges, several techniques have been proposed, including data anonymization[269], trusted execution environment technology[270][271], homomorphic encryption[272][273] and secure multi-party computation[274].

Federated learning has become a research hotspot, enabling AI models to be trained on a distributed network of edge devices while preserving data privacy and security[275][276][277][278]. Recently, hybrid approaches have been proposed that combine multiple techniques to address the challenges of edge computing. For example, StarFL is a new hybrid Federated Learning architecture that combines a trusted execution environment, secure multi-party computation, and beidou satellites to address communication-heavy, high-frequent, and asynchronous data in urban computing scenarios[279].

F. Model Management and Scheduling

Due to the limited resources in edge computing and the mobility of users and devices, it is necessary to perform efficient model management and scheduling when deploying AI models to the edge. Model scheduling can primarily be divided into model placement, model migration, and elastic scaling of models.

During model placement, the first challenge to address is designing effective feature extraction methods to extract features from the edge environment and user tasks due to the heterogeneity of AI model requests and the edge environment[280]. Second, given the complex request and restriction relationships between user tasks and models, considering various restriction conditions such as task dependency, deadline restrictions, bandwidth limitations, etc., to carry out model placement is the second challenge[281][282][283]. Lastly, in light of the latency requirements for edge AI model deployment, another issue to solve is how to schedule by further mining the dependency relationships between the model's image layers to reduce the cold start time of the AI model[284][285][286].

After model placement, due to the mobility of users or devices, the AI model needs to be further migrated[287]. Firstly, how to migrate the AI model requested by the user to the appropriate new edge node to achieve better Quality of Service (QoS) considering user mobility is the first challenge[288][289][290]. Secondly, during the migration process, considering the storage structure characteristics of the AI model and the limited computing resources in edge computing to further optimize the migration cost is another pressing challenge to solve[291][292].

Finally, to cope with the common scenarios of a sudden large number of AI model requests or peak period requests in the edge environment, in the phase of elastic scaling of the AI model, the first challenge is how to efficiently and accurately predict the resource utilization rate of different edge nodes and reasonably price the resources[293][294][295]. Secondly, in view of the geographic distribution characteristics of edge computing, designing innovative elastic scaling strategies to meet the requests of users in different regions is another challenge to address[296][297].

VIII. Conclusion and Future Directions

In conclusion, enabling AI on the edge has numerous benefits, including faster inference, improved privacy, and reduced latency. However, this approach also presents numerous challenges, such as limited computational resources, memory, and power availability. To address these challenges, a range of techniques can be used to optimize AI on the edge, including data optimization, model optimization, and system optimization. These techniques have been successfully applied in a range of use cases, including smart homes, industrial automation, autonomous vehicles, agriculture, retail and healthcare and medical applications. As the field of AI on the edge continues to evolve, new hardware and software developments will also play an important role, as well as integration with 5G networks and edge-to-cloud orchestration. By leveraging these techniques and advancements, we can continue to enable AI on the edge and unlock the potential benefits of this approach to computing.

In the future, edge AI is expected to have even broader applications, and several key development trends can be identified:

- **More intelligent**: With the advancement of AI chip technology and edge computing capabilities, the development of more intelligent edge devices is expected. These devices will be capable of processing increasingly complex data and tasks, resulting in the deployment of more complex and high-performance AI models on edge devices. This will enable them to perform more sophisticated data processing and decision-making tasks, providing users with more personalized and efficient services. Furthermore, the level of automated processing and adaptability of edge AI for different tasks will also be improved, leading to even more efficient and effective processing of data at the edge.

- **More flexible:** As the computing capabilities of edge AI devices improve and edge AI algorithms continue to develop, edge AI technology will evolve from serving specific scenarios to becoming more universal and flexible. This will enable edge AI to better adapt to different application scenarios, providing more versatile and adaptable solutions to a wider range of use cases. This increased flexibility will allow edge AI to be deployed in various industries and domains, making it a more ubiquitous and reliable technology for processing and analyzing data at the edge.
- **More secure:** In addition to enhancing efficiency and enabling autonomous decision-making, edge AI has an important role in improving user privacy and data security. In the future, edge AI is expected to continue to enhance security by adopting technologies such as blockchain to enable secure and decentralized data sharing and analysis. By using blockchain, edge AI can ensure that data remains secure and private, avoiding issues such as user data leakage. As edge AI becomes more widespread, ensuring user privacy and data security will become increasingly important, and the adoption of technologies such as blockchain will be critical in achieving this goal.
- **More collaborative:** As individual edge devices have limited resources, more complex tasks will be achieved through collaboration between edge devices and through cloud-edge collaboration. This collaboration will enable edge devices to work together to process and analyze data, providing more robust and efficient solutions. Meanwhile, scenarios such as intelligent transportation urgently require improved collaboration and greater intelligence in edge AI to ensure the safety and efficiency of transportation systems. By using collaboration and leveraging the strengths of both edge and cloud computing, edge AI can be used to address increasingly complex problems and provide more effective solutions in a wide range of applications.
- **More efficient:** Efficiency is a critical factor in edge AI, and more efficient algorithms and hardware, along with optimization for different scenarios, will help edge AI to have more efficient processing capabilities. By continually improving algorithms and hardware, edge AI devices can process data more quickly and accurately, while using fewer resources. Additionally, optimizing edge AI for different scenarios will enable it to be more effective in specific use cases, such as industrial automation or healthcare. These improvements in efficiency will enable edge AI to

become more widely adopted and provide more efficient and effective solutions for processing and analyzing data at the edge.

Overall, the future of edge AI is promising, and continued advancements in hardware, software, and collaboration will enable it to unlock its full potential in a wide range of applications.

Footnotes

[1] We list the explanations for the main abbreviations used throughout the paper in Table 2.

References

1. ^*Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, Neelakantan A, Shyam et al. (2020). "Language models are few-shot learners." Advances in Neural Information Processing Systems.* **33***: 1877--1901.*
2. a, b, c*Deng S, Zhao H, Fang W, Yin J, Dustdar S, Zomaya AY (2020). "Edge intelligence: The confluence of edge computing and artificial intelligence". IEEE Internet of Things Journal.* **7** *(8): 7457–7469.*
3. ^*Gill B, Rao S. Technology insight: Edge computing in support of the internet of things. Gartner Research Report. 2017.*
4. a, b*Taleb T, Dutta S, Ksentini A, Iqbal M, Flinck H (2017). "Mobile edge computing potential in making cities smarter". IEEE Communications Magazine.* **55** *(3): 38--43.*
5. a, b, c*Liu S, Liu L, Tang J, Yu B, Wang Y, Shi W (2019). "Edge computing for autonomous driving: Opportunities and challenges." Proceedings of the IEEE.* **107** *(8): 1697–1716.*
6. ^*Garcia Lopez P, Montresor A, Epema D, Datta A, Higashino T, Iamnitchi A, Barcellos M, Felber P, Riviere E (2015). "Edge-centric computing: Vision and challenges." ACM SIGCOMM Computer Communication Review.* **45** *(5): 37--42.*
7. a, b, c, d*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.*
8. a, b, c*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. ^*Bai C, Dallasega P, Orzes G, Sarkis J (2020). "Industry 4.0 technologies assessment: A sustainability perspective". International Journal of Production Economics.* **229***: 107776.*

10. ^Wang W, Li R, Chen Y, Diekel ZM, Jia Y (2018). "Facilitating human--robot collaborative tasks by teaching-learning-collaboration from human demonstrations." IEEE Transactions on Automation Science and Engineering. **16**(2): 640--653.

11. ^LeCun Y, Bengio Y, Hinton G (2015). "Deep learning". Nature. **521** (7553): 436–444.

12. a, bShi Y, Yang K, Jiang T, Zhang J, Letaief KB (2020). "Communication-efficient edge AI: Algorithms and systems". IEEE Communications Surveys & Tutorials. **22** (4): 2167–2191.

13. a, bChen J, Ran X (2019). "Deep learning with edge computing: A review." Proceedings of the IEEE. **107** (8): 1655–1674.

14. a, bLetaief KB, Shi Y, Lu J, Lu J (2021). "Edge artificial intelligence for 6G: Vision, enabling technologies, and applications." IEEE Journal on Selected Areas in Communications. **40** (1): 5–36.

15. a, bXu D, Li T, Li Y, Su X, Tarkoma S, Jiang T, Crowcroft J, Hui P (2020). "Edge intelligence: Architectures, challenges, and applications." arXiv preprint arXiv:2003.12172.

16. a, bYao J, Zhang S, Yao Y, et al. "Edge-cloud polarization and collaboration: A comprehensive survey for AI." IEEE Transactions on Knowledge and Data Engineering. 2022.

17. a, bMurshed MG, Murphy C, Hou D, Khan N, Ananthanarayanan G, Hussain F (2021). "Machine learning at the network edge: A survey". ACM Computing Surveys (CSUR). **54** (8): 1–37.

18. a, bPark J, Samarakoon S, Bennis M, Debbah M (2019). "Wireless network intelligence at the edge". Proceedings of the IEEE. **107** (11): 2204–2239.

19. a, b, cWang X, Han Y, Leung VC, Niyato D, Yan X, Chen X (2020). "Convergence of edge computing and deep learning: A comprehensive survey." IEEE Communications Surveys & Tutorials. **22** (2): 869--904.

20. ^Dai Y, Zhang K, Maharjan S, Zhang Y (2020). "Edge intelligence for energy-efficient computation offloading and resource allocation in 5G beyond". IEEE Transactions on Vehicular Technology. **69** (10): 12175–12186.

21. ^Zhang J, Letaief KB (2019). "Mobile edge intelligence and computing for the internet of vehicles". Proceedings of the IEEE. **108** (2): 246–261.

22. ^Xu D, Li T, Li Y, Su X, Tarkoma S, Jiang T, Crowcroft J, Hui P (2021). "Edge intelligence: Empowering intelligence to the edge of network". Proceedings of the IEEE. **109** (11): 1778–1837.

23. ^Hua H, Li Y, Wang T, Dong N, Li W, Cao J (2023). "Edge Computing with Artificial Intelligence: A Machine Learning Perspective." ACM Computing Surveys. **55** (9): 1–35.

24. ^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. Available from: https://arxiv.org/abs/1710.09282.

25. ^Deng L, Li G, Han S, Shi L, Xie Y (2020). "Model compression and hardware acceleration for neural net works: A comprehensive survey". Proceedings of the IEEE. **108** (4): 485--532.
26. ^Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, et al. (2010). "A view of cloud computing." Communications of the ACM. **53** (4): 50–58.
27. ^Jamsa K. Cloud computing. Burlington, MA: Jones & Bartlett Learning; 2022.
28. ^Gai K, Wu Y, Zhu L, Xu L, Zhang Y (2019). "Permissioned blockchain and edge computing empowered privacy-preserving smart grid networks." IEEE Internet of Things Journal. **6**(5): 7992–8004.
29. ^Satyanarayanan M. "The emergence of edge computing." Computer. **50**(1): 30–39, 2017.
30. ^Premsankar G, Di Francesco M, Taleb T (2018). "Edge computing for the Internet of Things: A case stud y". IEEE Internet of Things Journal. **5** (2): 1275--1284.
31. ^Huang P, Zeng L, Chen X, Luo K, Zhou Z, Yu S (2022). "Edge Robotics: Edge-Computing-Accelerated Multi-Robot Simultaneous Localization and Mapping". IEEE Internet of Things Journal. 2022.
32. ^Ding AY, Peltonen E, Meuser T, Aral A, Becker C, Dustdar S, Hiessl T, Kranzlm\uoofcller D, Liyanage M, Maghsudi S, et al. "Roadmap for edge AI: a Dagstuhl perspective." ACM SIGCOMM Computer Communic ation Review. **52** (1): 28--33, 2022. ACM New York, NY, USA.
33. ^Zha D, Bhat ZP, Lai KH, Yang F, Jiang Z, Zhong S, Hu X (2023). "Data-centric artificial intelligence: A s urvey". arXiv preprint arXiv:2303.10158. Available from: https://arxiv.org/abs/2303.10158.
34. ^Bernhardt M, Castro DC, Tanno R, Schwaighofer A, Tezcan KC, Monteiro M, Bannur S, Lungren et al. A ctive label cleaning for improved dataset quality under resource constraints. Nature Communications. **1 3**(1):1161, 2022.
35. ^Mishra R, Gupta A, Gupta HP (2021). "Locomotion mode recognition using sensory data with noisy lab els: A deep learning approach". IEEE Transactions on Mobile Computing. **2021**.
36. a, bWang T, Ke H, Zheng X, Wang K, Sangaiah AK, Liu A (2019). "Big data cleaning based on mobile edg e computing in industrial sensor-cloud". IEEE Transactions on Industrial Informatics. **16** (2): 1321–132 9.
37. ^Ma L, Pei Q, Zhou L, Zhu H, Wang L, Ji Y (2020). "Federated data cleaning: Collaborative and privacy-preserving data cleaning for edge intelligence". IEEE Internet of Things Journal. **8** (8): 6757–6770.
38. a, bSun D, Xue S, Wu H, Wu J (2021). "A data stream cleaning system using edge intelligence for smart cit y industrial environments". IEEE Transactions on Industrial Informatics. **18** (2): 1165–1174.
39. ^Sun D, Wu J, Yang J, Wu H (2021). "Intelligent data collaboration in heterogeneous-device IoT platfor ms". ACM Transactions on Sensor Networks (TOSN). **17** (3): 1–17.

40. ^Gupta C, Suggala AS, Goyal A, Simhadri HV, Paranjape B, Kumar A, Goyal S, Udupa R, Varma M, Jain P. "Protonn: Compressed and accurate knn for resource-scarce devices." In: International Conference on Machine Learning. PMLR; 2017. p. 1331-1340.

41. ^Van Der Maaten L, Postma E, Van den Herik J, et al. Dimensionality reduction: a comparative. J Mach L earn Res. **10**(66-71):13, 2009.

42. ^Kratsios A, Hyndman C (2021). "Neu: A meta-algorithm for universal uap-invariant feature represent ation". The Journal of Machine Learning Research. **22** (1): 4102–4152.

43. ^Cunningham JP, Ghahramani Z (2015). "Linear dimensionality reduction: Survey, insights, and genera lizations." The Journal of Machine Learning Research. **16** (1): 2859--2900.

44. ^Do TT, Hoang T, Pomponiu V, Zhou Y, Chen Z, Cheung NM, et al. (2018). "Accessible melanoma detecti on using smartphones and mobile image analysis". IEEE Transactions on Multimedia. **20** (10): 2849–2 864.

45. ^Haider F, Pollak S, Albert P, Luz S (2021). "Emotion recognition in low-resource settings: An evaluatio n of automatic feature selection methods". Computer Speech & Language. **65**: 101119.

46. ^Summerville DH, Zach KM, Chen Y. "Ultra-lightweight deep packet anomaly detection for Internet of T hings devices." In: 2015 IEEE 34th international performance computing and communications conferen ce (IPCCC). IEEE; 2015. p. 1-8.

47. ^Sudhakar SRV, Kayastha N, Sha K (2021). "ActID: An efficient framework for activity sensor based user identification". Computers & Security. **108**: 102319.

48. ^Laddha P, Omer OJ, Kalsi GS, Mandal DK, Subramoney S. "Descriptor Scoring for Feature Selection in R eal-Time Visual Slam." In: 2020 IEEE International Conference on Image Processing (ICIP). IEEE; 2020. p. 2601--2605.

49. ^Masud M, Singh P, Gaba GS, Kaur A, Alroobaea R, Alrashoud M, Alqahtani SA (2021). "CROWD: crow s earch and deep learning based feature extractor for classification of Parkinson's disease." ACM Transact ions on Internet Technology (TOIT). **21** (3): 1–18.

50. ^Chen J, Zheng Y, Liang Y, Zhan Z, Jiang M, Zhang X, da Silva DS, Wu W, de Albuquerque VHC. "Edge2A nalysis: a novel AIoT platform for atrial fibrillation recognition and detection." IEEE Journal of Biomedi cal and Health Informatics. **26** (12): 5772–5782, 2022.

51. a, bLi T, Fong S, Li X, Lu Z, Gandomi AH (2019). "Swarm decision table and ensemble search methods in fog computing environment: case of day-ahead prediction of building energy demands using IoT sensor s". IEEE Internet of Things Journal. **7** (3): 2321–2342.

52. ^Marino R, Wisultschew C, Otero A, Lanza-Gutierrez JM, Portilla J, de la Torre E (2020). "A machine-learning-based distributed system for fault diagnosis with scalable detection quality in industrial IoT". IEEE Internet of Things Journal. 8 (6): 4339--4352.

53. a, b, cShen C, Zhang K, Tang J (2021). "A covid-19 detection algorithm using deep features and discrete social learning particle swarm optimization for edge computing devices". ACM Transactions on Internet Technology (TOIT). 22 (3): 1–17.

54. ^Matsubara Y, Yang R, Levorato M, Mandt S. "Supervised compression for resource-constrained edge computing systems." In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision; 2022. p. 2685-2695.

55. ^Chen D, Cao X, Wen F, Sun J. "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2013. p. 3025--3032.

56. ^Chen Z, Fan K, Wang S, Duan L, Lin W, Kot AC (2019). "Toward intelligent sensing: Intermediate deep feature compression". IEEE Transactions on Image Processing. 29: 2230–2243.

57. ^Liu C, Li X, Chen H, Modolo D, Tighe J (2021). "Selective feature compression for efficient activity recognition inference." In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021. p. 13628-13637.

58. ^Shao J, Zhang J (2020). "Communication-computation trade-off in resource-constrained edge inference". IEEE Communications Magazine. 58 (12): 20–26.

59. ^Abdellatif AA, Emam A, Chiasserini CF, et al. "Edge-based compression and classification for smart healthcare systems: Concept, implementation and evaluation." Expert Systems with Applications. 117: 1–14, 2019.

60. ^Zhou S, Van Le D, Yang JQ, Tan R, Ho D. "EFCam: Configuration-adaptive fog-assisted wireless cameras with reinforcement learning." In: 2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON). IEEE; 2021. p. 1-9.

61. ^Abdellatif AA, Mohamed A, Chiasserini CF, Tlili M, Erbad A (2019). "Edge computing for smart health: Context-aware approaches, opportunities, and challenges." IEEE Network. 33 (3): 196--203.

62. ^Moreno-Rodenas AM, Duinmeijer A, Clemens FH (2021). "Deep-learning based monitoring of FOG layer dynamics in wastewater pumping stations". Water Research. 202: 117482.

63. ^Guo Y, Zou B, Ren J, Liu Q, Zhang D, Zhang Y (2019). "Distributed and efficient object detection via interactions among devices, edge, and cloud". IEEE Transactions on Multimedia. 21 (11): 2903–2915.

64. ^Shorten C, Khoshgoftaar TM (2019). "A survey on image data augmentation for deep learning". *Journal of Big Data.* **6** (1): 1–48.

65. ^Ma W, Lou R, Zhang K, Wang L, Vosoughi S (2021). "GradTS: A Gradient-Based Automatic Auxiliary Task Selection Method Based on Transformer Networks". In: Proceedings of EMNLP 2021. pp. 5621–5632.

66. ^Feng SY, Gangal V, Wei J, Chandar S, Vosoughi S, Mitamura T, Hovy E (2021). "A survey of data augmentation approaches for NLP". arXiv preprint arXiv:2105.03075. **2021**.

67. ^Wang Z, Hu J, Min G, Zhao Z, Wang J (2020). "Data-augmentation-based cellular traffic prediction in edge-computing-enabled smart city". IEEE Transactions on Industrial Informatics. **17** (6): 4179–4187.

68. ^Liao RF, Wen H, Chen S, Xie F, Pan F, Tang J, Song H (2019). "Multiuser physical layer authentication in internet of things with data augmentation". IEEE Internet of Things Journal. **7** (3): 2077–2088.

69. ^Liu X, Deng Z (2018). "Segmentation of drivable road using deep fully convolutional residual network with pyramid pooling". Cognitive Computation. **10**: 272–281.

70. ^Jiao Z, Huang K, Jia G, Lei H, Cai Y, Zhong Z (2022). "An effective litchi detection method based on edge devices in a complex scene". Biosystems Engineering. **222**: 15–28.

71. ^Gu G, Ko B, Go S, Lee S-H, Lee J, Shin M (2022). "Towards light-weight and real-time line segment detection." In: Proceedings of the AAAI Conference on Artificial Intelligence. **36**(1): 726–734.

72. ^Liu C, Antypenko R, Sushko I, Zakharchenko O (2022). "Intrusion Detection System After Data Augmentation Schemes Based on the VAE and CVAE". IEEE Transactions on Reliability. **71** (2): 1000--1010.

73. ^Pan H, Chen Y-C, Ye Q, Xue G (2021). "Magicinput: Training-free multi-lingual finger input system using data augmentation based on mnists." In: Proceedings of the 20th International Conference on Information Processing in Sensor Networks. 2021. pp. 119–131.

74. ^Zhou Y, Chen S, Wang Y, Huan W. "Review of research on lightweight convolutional neural networks." In: 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC). IEEE; 2020. p. 1713-1720.

75. a, b, cHoward AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017). "Mobilenets: Efficient convolutional neural networks for mobile vision applications". arXiv preprint arXiv:1704.04861.

76. a, b, cSandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. "Mobilenetv2: Inverted residuals and linear bottlenecks." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2018. p. 4510-4520.

77. a, b, cHoward A, Sandler M, Chu G, Chen L-C, Chen B, Tan M, Wang W, Zhu Y, et al. Searching for mobile netv3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019. p. 1314–1324.

78. a, bZhou D, Hou Q, Chen Y, Feng J, Yan S. "Rethinking bottleneck structure for efficient mobile network design." In: Computer Vision--ECCV 2020: 16th European Conference, Glasgow, UK, August 23--28, 2020, Proceedings, Part III 16. Springer; 2020. p. 680--697.

79. a, bTan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV. "Mnasnet: Platform-aware neural architecture search for mobile." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 2820–2828.

80. a, bZhang X, Zhou X, Lin M, Sun J (2018). "Shufflenet: An extremely efficient convolutional neural network for mobile devices." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6848–6856.

81. a, bMa N, Zhang X, Zheng HT, Sun J (2018). "Shufflenet v2: Practical guidelines for efficient cnn architecture design." In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 116–131.

82. a, bGuo Z, Zhang X, Mu H, et al. Single path one-shot neural architecture search with uniform sampling. In: Computer Vision--ECCV 2020: 16th European Conference, Glasgow, UK, August 23--28, 2020, Proceedings, Part XVI 16. Springer; 2020. p. 544--560.

83. a, bIandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size". arXiv preprint arXiv:1602.07360.

84. a, bGholami A, Kwon K, Wu B, Tai Z, Yue X, Jin P, et al. "Squeezenext: Hardware-aware neural network design." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops; 2018. p. 1638–1647.

85. a, bHan K, Wang Y, Tian Q, Guo J, Xu C, Xu C (2020). "Ghostnet: More features from cheap operations." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020. pp. 1580–1589.

86. a, b, cTan M, Le Q (2019). "Efficientnet: Rethinking model scaling for convolutional neural networks." In: International Conference on Machine Learning. PMLR. pp. 6105–6114.

87. a, b, cTan M, Le Q (2021). "EfficientNetV2: Smaller Models and Faster Training." In: International Conference on Machine Learning. PMLR. pp. 10096–10106.

88. a, b, cTan M, Pang R, Le QV (2020). "Efficientdet: Scalable and efficient object detection." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10781–10790.

89. a, bHuang G, Liu S, Van der Maaten L, Weinberger KQ (2018). "Condensenet: An efficient densenet using learned group convolutions." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Re cognition. 2018. p. 2752–2761.

90. a, bYang L, Jiang H, Cai R, Wang Y, Song S, Huang G, Tian Q. "Condensenet v2: Sparse feature reactivatio n for deep networks." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Reco gnition; 2021. p. 3569–3578.

91. a, bMehta S, Rastegari M, Caspi A, Shapiro L, Hajishirzi H. "Espnet: Efficient spatial pyramid of dilated c onvolutions for semantic segmentation." In: Proceedings of the European Conference on Computer Visio n (ECCV); 2018. p. 552–568.

92. a, bMehta S, Rastegari M, Shapiro L, et al. "Espnetv2: A light-weight, power efficient, and general purpo se convolutional neural network." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2019. p. 9190–9200.

93. a, bWu B, Dai X, Zhang P, et al. "Fbnet: Hardware-aware efficient convnet design via differentiable neur al architecture search." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Rec ognition. 2019. p. 10734-10742.

94. a, bWan A, Dai X, Zhang P, et al. Fbnetv2: Differentiable neural architecture search for spatial and chann el dimensions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognitio n. 2020. p. 12965-12974.

95. a, bDai X, Wan A, Zhang P, Wu B, He Z, Wei Z, et al. Fbnetv3: Joint architecture-recipe search using predi ctor pretraining. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recogniti on. 2021. p. 16276-16285.

96. a, bWang RJ, Li X, Ling CX (2018). "Pelee: A real-time object detection system on mobile devices". Advan ces in Neural Information Processing Systems. **31**.

97. a, bSzegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. "Going deeper with convolutions." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2015. p. 1–9.

98. a, bIoffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal cov ariate shift. In: International Conference on Machine Learning. pmlr; 2015. p. 448–456.

99. a, bSzegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016). "Rethinking the inception architecture for computer vision." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826.

100. ^a, b^ *Szegedy C, Ioffe S, Vanhoucke V, Alemi A (2017). "Inception-v4, inception-resnet and the impact of residual connections on learning." In: Proceedings of the AAAI conference on Artificial Intelligence. **31(1)**.*

101. ^a, b^ *Chollet F. "Xception: Deep learning with depthwise separable convolutions." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. p. 1251-1258.*

102. ^a, b^ *Mehta S, Rastegari M (2021). "Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer." International Conference on Learning Representations.*

103. ^a, b^ *Wu Z, Liu Z, Lin J, Lin Y, Han S (2020). "Lite transformer with long-short range attention." International Conference on Learning Representations.*

104. ^a, b^ *Hou Q, Zhou D, Feng J (2021). "Coordinate attention for efficient mobile network design." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 13713–13722.*

105. ^a, b^ *Wang Q, Wu B, Zhu P, Li P, Zuo W, Hu Q (2020). "ECA-Net: Efficient channel attention for deep convolutional neural networks." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11534–11542.*

106. ^a, b^ *Zhang QL, Yang YB. "Sa-net: Shuffle attention for deep convolutional neural networks." In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2021. p. 2235-2239.*

107. ^a, b^ *Misra D, Nalamada T, Arasanipalai AU, Hou Q (2021). "Rotate to attend: Convolutional triplet attention module." In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3139--3148.*

108. ^a, b^ *Zhang H, Wu C, Zhang Z, Zhu Y, Lin H, Zhang Z, Sun Y, He T, Mueller J, Manmatha R, et al. "Resnest: Split-attention networks." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2022. p. 2736-2746.*

109. ^ *Lu H, Du M, He X, Qian K, Chen J, Sun Y, Wang K (2021). "An adaptive neural architecture search design for collaborative edge-cloud computing". IEEE Network. **35** (5): 83–89.*

110. ^ *Lyu B, Wen S, Shi K, Huang T (2021). "Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing". IEEE Transactions on Cybernetics. **2021**. Published by IEEE.*

111. ^a, b^ *Chen H, Zhuo L, Zhang B, Zheng X, Liu J, Ji R, Doermann D, Guo G (2021). "Binarized neural architecture search for efficient object recognition". International Journal of Computer Vision. **129**: 501–516.*

112. ^ *Mendis HR, Kang C-K, Hsiu P-c (2021). "Intermittent-aware neural architecture search". ACM Transactions on Embedded Computing Systems (TECS). **20** (5s): 1–27.*

113. a, b*Ning X, Ge G, Li W, Zhu Z, Zheng Y, Chen X, et al. FTT-NAS: Discovering fault-tolerant convolutional neural architecture. ACM Transactions on Design Automation of Electronic Systems (TODAES). 26(6):1–24, 2021.*

114. ^*Liu Z, Tang H, Zhao S, Shao K, Han S (2021). "Pvnas: 3D neural architecture search with point-voxel convolution". IEEE Transactions on Pattern Analysis and Machine Intelligence. 44 (11): 8552--8568.*

115. ^*Donegan C, Yous H, Sinha S, Byrne J. "VPU specific CNNs through neural architecture search." In: 2020 25th International Conference on Pattern Recognition (ICPR). IEEE; 2021. p. 9772–9779.*

116. ^*Nayman N, Aflalo Y, Noy A, Zelnik L (2021). "Hardcore-nas: Hard constrained differentiable neural architecture search." In: International Conference on Machine Learning. PMLR. pp. 7979–7990.*

117. ^*Liu P, Wu B, Ma H, Seok M. "MemNAS: Memory-efficient neural architecture search with grow-trim learning." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020. p. 2108–2116.*

118. ^*Zhang C, Yuan X, Zhang Q, Zhu G, Cheng L, Zhang N (2022). "Toward tailored models on private aiot devices: Federated direct neural architecture search". IEEE Internet of Things Journal. 9 (18): 17309–17322.*

119. ^*Han S, Mao H, Dally WJ (2016). "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding". International Conference on Learning Representations (ICLR).*

120. ^*Xu Z, Yu F, Qin Z, et al. Directx: Dynamic resource-aware cnn reconfiguration framework for real-time mobile applications. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2020;40(2):246–259.*

121. ^*Ahmad H, Arif T, Hanif MA, Hafiz R, Shafique M (2020). "SuperSlash: A unified design space exploration and model compression methodology for design of deep learning accelerators with reduced off-chip memory access volume". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 39 (11): 4191--4204.*

122. ^*Tan CMJ, Motani M. "Dropnet: Reducing neural network complexity via iterative pruning." In: International Conference on Machine Learning. PMLR; 2020. p. 9356-9366.*

123. ^*Li Z, Wallace E, Shen S, Lin K, Keutzer K, Klein D, Gonzalez J (2020). "Train big, then compress: Rethinking model size for efficient training and inference of transformers." In: International Conference on Machine Learning. PMLR. pp. 5958–5968.*

124. ^Ma W, Zhang K, Lou R, Wang L, Vosoughi S. Contributions of Transformer Attention Heads in Multi- and Cross-lingual Tasks. In: Proceedings of ACL-IJCNLP 2021; 2021.

125. a, bGao S, Huang F, Pei J, Huang H (2020). "Discrete model compression with resource constraint for deep neural networks." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1899--1908.

126. ^Wu C, Cui Y, Ji C, Kuo TW, Xue CJ (2020). "Pruning deep reinforcement learning for dual user experience and storage lifetime improvement on mobile devices". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 39 (11): 3993–4005.

127. ^Zhou X, Jia Q, et al. "NestFL: efficient federated learning through progressive model pruning in heterogeneous edge computing." In: Proceedings of the 28th Annual International Conference on Mobile Computing And Networking, 2022, pp. 817–819.

128. a, bGeng T, Li A, Wang T, Wu C, et al. O3BNN-R: An out-of-order architecture for high-performance and regularized BNN inference. IEEE Transactions on Parallel and Distributed Systems. 32(1):199–213, 2020.

129. ^Li G, Ma X, Wang X, Liu L, et al. (2020). "Fusion-catalyzed pruning for optimizing deep learning on intelligent edge devices." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 39 (11): 3614–3626.

130. ^Gu J, Feng C, Zhao Z, Ying Z, et al. Efficient on-chip learning for optical neural networks through power-aware sparse zeroth-order optimization. Proceedings of the AAAI Conference on Artificial Intelligence. 2021; 35(9): 7583–7591.

131. ^Kwon W, Kim S, Mahoney MW, Hassoun J, Keutzer K, Gholami A (2022). "A Fast Post-Training Pruning Framework for Transformers." Advances in Neural Information Processing Systems. 2022.

132. ^Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, Shao L. "Hrank: Filter pruning using high-rank feature map." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2020. p. 1529–1538.

133. ^Li S, Hanson E, Li H, Chen Y (2020). "Penni: Pruned kernel sharing for efficient CNN inference." In: International Conference on Machine Learning. PMLR. pp. 5863–5873.

134. ^Tung F, Mori G (2018). "Clip-q: Deep network compression learning by in-parallel pruning-quantization." In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7873–7882.

135. ^*Fedorov I, Adams RP, Mattina M, Whatmough P (2019). "Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers". Advances in Neural Information Processing Systems. 32.*

136. a, b*Khaleghi B, Imani M, Rosing T. "Prive-hd: Privacy-preserved hyperdimensional computing." In: 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE; 2020. p. 1-6.*

137. a, b, c*Huang Y, Qiao X, Tang J, Ren P, et al. Deepadapter: A collaborative deep learning framework for the mobile web using context-aware network pruning. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE; 2020. p. 834–843.*

138. a, b*Liu Y, Shu Z, Li Y, Lin Z, Perazzi F, Kung S-Y. "Content-aware gan compression." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 12156-12166.*

139. ^*Jian T, Gong Y, Zhan Z, Shi R, Soltani N, Wang Z, Dy J, Chowdhury K, Wang Y, Ioannidis S (2021). "Radio frequency fingerprinting on the edge". IEEE Transactions on Mobile Computing. 21 (11): 4078–4093.*

140. ^*Wang L, Dong X, Wang Y, Ying X, et al. "Exploring sparsity in image super-resolution for efficient inference." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021. p. 4917–4926.*

141. ^*Sun M, Zhao P, Gungor M, Pedram M, Leeser M, Lin X. "3D CNN acceleration on FPGA using hardware-aware pruning." In: 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE; 2020. p. 1-6.*

142. a, b*Wu J, Wang Y, Wu Z, et al. "Deep k-means: Re-training and parameter sharing with harder cluster assignments for compressing deep convolutions." In: International Conference on Machine Learning. PMLR; 2018. p. 5363–5372.*

143. a, b*Obukhov A, Rakhuba M, Georgoulis S, Kanakis M, Dai D, Van Gool L. "T-basis: a compact representation for neural networks." In: International Conference on Machine Learning. PMLR; 2020. p. 7392–7404.*

144. ^*Ullrich K, Meeds E, Welling M (2017). "Soft weight-sharing for neural network compression". International Conference on Learning Representations.*

145. ^*You H, Li B, Huihong S, Fu Y, Lin Y. "ShiftAddNAS: Hardware-inspired search for more accurate and efficient neural networks." In: International Conference on Machine Learning. PMLR; 2022. p. 25566-25580.*

146. ^*Hu S, Xie X, Cui M, Deng J, Liu S, Yu J, et al. "Neural architecture search for LF-MMI trained time delay neural networks." IEEE/ACM Transactions on Audio, Speech, and Language Processing. 30: 1093–1107, 2022.*

147. ^Wang R, Wei Z, Duan H, Ji S, Long Y, Hong Z (2022). "EfficientTDNN: Efficient architecture search for speaker recognition". IEEE/ACM Transactions on Audio, Speech, and Language Processing. **30**: 2267–2279.

148. ^Sun Y, Yuan F, Yang M, Wei G, et al. "A generic network compression framework for sequential recommender systems." In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval; 2020. p. 1299-1308.

149. ^Sindhwani V, Sainath T, Kumar S (2015). "Structured transforms for small-footprint deep learning". Advances in Neural Information Processing Systems. **28**.

150. a, bFu Y, You H, Zhao Y, Wang Y, Li C, et al. (2020). "Fractrain: Fractionally squeezing bit savings both temporally and spatially for efficient dnn training." Advances in Neural Information Processing Systems. **33**: 12127–12139.

151. a, b, c, dTambe T, Hooper C, Pentecost L, et al. "Edgebert: Sentence-level energy optimizations for latency-aware multi-task nlp inference." In: MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture; 2021. p. 830-844.

152. ^Chikin V, Antiukh M (2022). "Data-Free Network Compression via Parametric Non-uniform Mixed Precision Quantization." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 450–459.

153. a, b, cBoo Y, Shin S, Choi J, Sung W (2021). "Stochastic precision ensemble: self-knowledge distillation for quantized deep neural networks." In: Proceedings of the AAAI Conference on Artificial Intelligence. **35** (8): 6794–6802.

154. ^Cui Y, Wu S, Li Q, Chan AB, Kuo TW, Xue CJ (2022). "Bits-Ensemble: Toward Light-Weight Robust Deep Ensemble by Bits-Sharing". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. **41** (11): 4397–4408.

155. ^Marchisio A, Bussolino B, Colucci A, Martina M, Masera G, Shafique M. "Q-capsnets: A specialized framework for quantizing capsule networks." In: 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE; 2020. p. 1-6.

156. ^Putra RVW, et al. (2020). "Fspinn: An optimization framework for memory-efficient and energy-efficient spiking neural networks." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. **39** (11): 3601–3613.

157. a, b, cZhou Q, Guo S, Qu Z, et al. "Octo: INT8 Training with Loss-aware Compensation and Backward Quantization for Tiny On-device Learning." In: USENIX Annual Technical Conference, 2021, pp. 177–191.

158. a, b, c*Wang K, Liu Z, Lin Y, Lin J, Han S (2019). "Haq: Hardware-aware automated quantization with mixed precision." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8612--8620.*

159. ^*Li B, Qu S, Wang Y (2021). "An automated quantization framework for high-utilization rram-based pim". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.* **41** *(3): 583--596.*

160. ^*Simon WA, Ray V, Levisse A, Ansaloni G, Zapater M, Atienza D. "Exact neural networks from inexact multipliers via fibonacci weight encoding." In: 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE; 2021. p. 805-810.*

161. ^*Hinton G, Vinyals O, Dean J (2015). "Distilling the knowledge in a neural network". arXiv preprint arXiv:1503.02531. Available from: https://arxiv.org/abs/1503.02531.*

162. a, b*Zhang L, Song J, Gao A, et al. (2019). "Be your own teacher: Improve the performance of convolutional neural networks via self distillation." In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3713–3722.*

163. ^*Hou L, Huang Z, Shang L, Jiang X, Chen X, Liu Q (2020). "Dynabert: Dynamic bert with adaptive width and depth". Advances in Neural Information Processing Systems.* **33**: *9782–9793.*

164. ^*Zhang L, Tan Z, Song J, Chen J, Bao C, Ma K (2019). "Scan: A scalable neural networks framework towards compact and efficient models". Advances in Neural Information Processing Systems. 32.*

165. a, b*Zhang Y, Yan Z, Sun X, Diao W, Fu K, Wang L (2021). "Learning efficient and accurate detectors with dynamic knowledge distillation in remote sensing imagery". IEEE Transactions on Geoscience and Remote Sensing.* **60**: *1–19.*

166. ^*Hao Z, Luo Y, Wang Z, Hu H, An J (2022). "CDFKD-MFS: Collaborative Data-Free Knowledge Distillation via Multi-Level Feature Sharing". IEEE Transactions on Multimedia.* **24**: *4262–4274.*

167. ^*Hao Z, Guo J, Jia D, Han K, Tang Y, Zhang C, Hu H, Wang Y (2022). "Learning Efficient Vision Transformers via Fine-Grained Manifold Distillation". Advances in Neural Information Processing Systems.* **35**: *9164–9175.*

168. ^*Zhang K, Tao C, Shen T, Xu C, Geng X, Jiao B, Jiang D (2023). "LED: Lexicon-Enlightened Dense Retriever for Large-Scale Retrieval." In: Proceedings of WWW 2023. doi:10.1145/3543507.3583294.*

169. ^*Shen T, Geng X, Tao C, Xu C, Long G, Zhang K, Jiang D (2023). "UnifieR: A Unified Retriever for Large-Scale Retrieval". arXiv. Available from: https://arxiv.org/abs/2205.11194.*

170. ^*Ni J, Sarbajna R, Liu Y, et al. "Cross-modal knowledge distillation for vision-to-sensor action recognition." In: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing*

(ICASSP). IEEE; 2022. p. 4448-4452.

171. ^*Jin H, Bai D, Yao D, Dai Y, Gu L, Yu C, Sun L (2022). "Personalized edge intelligence via federated self-knowledge distillation". IEEE Transactions on Parallel and Distributed Systems.* **34** *(2): 567–580.*
172. ^*Li W, Wang J, Ren T, Li F, Zhang J, Wu Z (2022). "Learning Accurate, Speedy, Lightweight CNNs via Ins tance-Specific Multi-Teacher Knowledge Distillation for Distracted Driver Posture Identification". IEEE Transactions on Intelligent Transportation Systems.* **23** *(10): 17922–17935.*
173. ^*Xia X, Yin H, Yu J, et al. "On-Device Next-Item Recommendation with Self-Supervised Knowledge Dist illation." In: Proceedings of the 45th International ACM SIGIR Conference on Research and Developmen t in Information Retrieval, 2022, pp. 546–555.*
174. ^*Xu Z, Hong Z, Ding C, Zhu Z, Han J, Liu J, Ding E (2022). "Mobilefaceswap: A lightweight framework fo r video face swapping." Proceedings of the AAAI Conference on Artificial Intelligence.* **36** *(3): 2973–298 1.*
175. ^*Bai H, Mao H, Nair D. "Dynamically pruning segformer for efficient semantic segmentation." In: ICASS P 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE; 2022. p. 3298-3302.*
176. ^*Yang H, et al. "Learning low-rank deep neural networks via singular vector orthogonality regularizati on and singular value sparsification." In: Proceedings of the IEEE/CVF Conference on Computer Vision a nd Pattern Recognition Workshops, 2020. p. 678–679.*
177. ^*Li Y, Chen Y, Dai X, Chen D, Liu M, Yuan L, Liu Z, Zhang L, Vasconcelos N (2020). "MicroNet: Towards i mage recognition with extremely low FLOPs". arXiv preprint arXiv:2011.12289.* **2020***. Available from: ht tps://arxiv.org/abs/2011.12289.*
178. ^*Verma G, Gupta Y, Malik AM, Chapman B. "Performance evaluation of deep learning compilers for edg e inference." In: 2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IP DPSW). IEEE; 2021. p. 858–865.*
179. ^*Microsoft (2019). "ONNX Runtime: cross-platform, high performance ML inferencing and training acc elerator." GitHub repository. Available from: https://github.com/microsoft/onnxruntime.*
180. ^*Intel (2018). "$OpenVINO^{TM}$ Toolkit repository." GitHub repository. Available from: https://githu b.com/openvinotoolkit/openvino.*
181. ^*Tencent (2017). "NCNN is a high-performance neural network inference framework optimized for the mobile platform." GitHub repository. Available from: https://github.com/Tencent/ncnn.*
182. ^*Arm (2018). "Arm NN ML Software. https://github.com/ARM-software/armnn". GitHub repository.*

183. ^*Alibaba (2018). "MNN is a blazing fast, lightweight deep learning framework". GitHub repository. Available from: https://github.com/alibaba/MNN.*

184. ^*NVIDIA (2017). "$NVIDIA^{\circledR}$ $TensorRT^{TM}$, an SDK for high-performance deep learning inference". GitHub repository. Available from: https://github.com/NVIDIA/TensorRT.*

185. ^*Apache (2018). "Open deep learning compiler stack for cpu, gpu and specialized accelerators." GitHub repository. Available from: https://github.com/apache/tvm.*

186. a, b, c, d, e, f*Xia M, Huang Z, Tian L, Wang H, Chang V, Zhu Y, Feng S (2021). "SparkNoC: An energy-efficiency FPGA-based accelerator using optimized lightweight CNN for edge computing". Journal of Systems Architecture.* ***115****: 101991.*

187. a, b*Wang Y, Li H, Li X. "Re-architecting the on-chip memory sub-system of machine-learning accelerator for embedded devices." In: 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE; 2016. p. 1–6.*

188. a, b*Wang L, Chen Z, Liu Y, Wang Y, Zheng L, Li M, Wang Y. "A unified optimization approach for cnn model inference on integrated gpus." In: Proceedings of the 48th International Conference on Parallel Processing; 2019. p. 1-10.*

189. a, b*Wang S, Ananthanarayanan G, Zeng Y, et al. High-throughput cnn inference on embedded arm big. little multicore processors. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.* ***39****(10):2254–2267, 2019.*

190. a, b*Zhao L, Zhang Y, Yang J. "SCA: a secure CNN accelerator for both training and inference." In: 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE; 2020. p. 1-6.*

191. a, b*Hou X, Guan Y, Han T (2022). "NeuLens: spatial-based dynamic acceleration of convolutional neural networks on edge." In: Proceedings of the 28th Annual International Conference on Mobile Computing And Networking. pp. 186--199.*

192. a, b*Srivastava A, Dutta O, Gupta J, et al. "A variational information bottleneck based method to compress sequential networks for human action recognition." In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision; 2021. p. 2745-2754.*

193. a, b*Gao C, Rios-Navarro A, Chen X, Liu S-C, Delbruck T (2020). "EdgeDRNN: Recurrent neural network accelerator for edge inference". IEEE Journal on Emerging and Selected Topics in Circuits and Systems.* ***10*** *(4): 419--432.*

194. a, b*Wen L, Zhang X, Bai H, Xu Z (2020). "Structured pruning of recurrent neural networks through neuron selection". Neural Networks.* ***123****: 134–141.*

195. ^ a, b *Zhang J, Wang X, Li D, Wang Y. "Dynamically hierarchy revolution: dirnet for compressing recurrent neural network on mobile devices." In: Proceedings of the 27th International Joint Conference on Artificial Intelligence; 2018. p. 3089-3096.*

196. ^ a, b, c *Ding Y, Yu CH, et al. "Hidet: Task-mapping programming paradigm for deep learning tensor programs." In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2. 2023. p. 370–384.*

197. ^ *Liu P, Qi B, Banerjee S. "Edgeeye: An edge service framework for real-time intelligent video analytics." In: Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking; 2018. p. 1–6.*

198. ^ a, b *Xie X, Kim KH. "Source compression with bounded dnn perception loss for iot edge computer vision." In: The 25th Annual International Conference on Mobile Computing and Networking; 2019. p. 1-16.*

199. ^ a, b *Huang Y, Qiao X, Ren P, Liu L, Pu C, Dustdar S, Chen J (2020). "A lightweight collaborative deep neural network for the mobile web in edge cloud". IEEE Transactions on Mobile Computing. **21** (7): 2289–2305.*

200. ^ *Farhadi M, Ghasemi M, Vrudhula S, Yang Y (2020). "Enabling incremental knowledge transfer for object detection at the edge." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 396–397.*

201. ^ a, b *Yang L, Rakin AS, Fan D (2022). "DA3: Dynamic Additive Attention Adaption for Memory-Efficient On-Device Multi-Domain Learning." In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 2619–2627.*

202. ^ a, b *Kosta AK, Anwar MA, et al. "RAPID-RL: A Reconfigurable Architecture with Preemptive-Exits for Efficient Deep-Reinforcement Learning." In: 2022 International Conference on Robotics and Automation (ICRA). IEEE; 2022. p. 7492–7498.*

203. ^ *Susskind Z, Arora A, Miranda ID, Villon LA, Katopodis RF, de Araújo LS, Dutra DL, Lima PM, França FM, Breternitz Jr M, et al. "Weightless neural networks for efficient edge inference." In: Proceedings of the International Conference on Parallel Architectures and Compilation Techniques; 2022. p. 279–290.*

204. ^ *He S, Meng H, Zhou Z, Liu Y, Huang K, Chen G (2021). "An efficient GPU-accelerated inference engine for binary neural network on mobile phones". Journal of Systems Architecture. **117**: 102156.*

205. ^ *Zhang X, Jiang W, Hu J (2020). "Achieving full parallelism in LSTM via a unified accelerator design." In: 2020 IEEE 38th International Conference on Computer Design (ICCD). IEEE. pp. 469–477.*

206. ^Zhou Z, Liu J, Gu Z, Sun G (2022). "Energon: Toward Efficient Acceleration of Transformers Using Dynamic Sparse Attention". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 42 (1): 136–149.

207. ^Zhou Z, Shi B, Zhang Z, Guan Y, Sun G, Luo G. "Blockgnn: Towards efficient gnn acceleration using block-circulant weight matrices." In: 2021 58th ACM/IEEE Design Automation Conference (DAC). IEEE; 2021. p. 1009-1014.

208. ^Kwon M, Gouk D, et al. Hardware/Software Co-Programmable Framework for Computational SSDs to Accelerate Deep Learning Service on Large-Scale Graphs. In: 20th USENIX Conference on File and Storage Technologies (FAST 22); 2022. p. 147-164.

209. ^Yang T, Li D, Ma F, Song Z, Zhao Y, Zhang J, Liu F, Jiang L (2022). "Pasgcn: An reram-based pim design for gcn with adaptively sparsified graphs." IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 42 (1): 150--163.

210. ^Xu H, Wang Y, Wang Y, Li J, Liu B, Han Y (2019). "ACG-engine: An inference accelerator for content generative neural networks." In: 2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). IEEE. pp. 1–7.

211. a, bNori AV, Bera R, et al. "Reduct: Keep it close, keep it cool!: Efficient scaling of dnn inference on multi-core cpus with near-cache compute." In: 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE; 2021. p. 167–180.

212. a, bJia T, Ju Y, et al. "Ncpu: An embedded neural cpu architecture on resource-constrained low power devices for real-time end-to-end performance." In: 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE; 2020. p. 1097–1109.

213. a, b, cCapodieci N, Cavicchioli R, Bertogna M, Paramakuru A. "Deadline-based scheduling for GPU with preemption support." In: 2018 IEEE Real-Time Systems Symposium (RTSS). IEEE; 2018. p. 119–130.

214. a, b, cChoudhury Z, Shrivastava S, Ramapantulu L, Purini S (2022). "An FPGA overlay for CNN inference with fine-grained flexible parallelism". ACM Transactions on Architecture and Code Optimization (TACO). 19 (3): 1–26.

215. a, b, cYu Y, Zhao T, Wang K, He L. "Light-OPU: An FPGA-based overlay processor for lightweight convolutional neural networks." In: Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays; 2020. p. 122--132.

216. a, b, cRoohi A, Sheikhfaal S, Angizi S, Fan D, DeMara RF (2019). "Apgan: Approximate gan for robust low energy learning from imprecise components". IEEE Transactions on Computers. 69 (3): 349--360.

217. ^a, b^ *Kouris A, Venieris SI, Laskaridis S, Lane ND (2022). "Fluid Batching: Exit-Aware Preemptive Serving of Early-Exit Neural Networks on Edge NPUs". arXiv preprint arXiv:2209.13443. 2022.*

218. ^a, b^ *Yang Q, Li H (2020). "BitSystolic: A 26.7 TOPS/W 2b~ 8b NPU with configurable data flows for edge devices". IEEE Transactions on Circuits and Systems I: Regular Papers. **68** (3): 1134--1145.*

219. ^a, b, c^ *Wang Y, Deng D, Liu L, et al. (2022). "PL-NPU: An Energy-Efficient Edge-Device DNN Training Processor With Posit-Based Logarithm-Domain Computing". IEEE Transactions on Circuits and Systems I: Regular Papers. **69** (10): 4042–4055.*

220. ^a, b^ *Cho H, Lee J, Lee J (2021). "FARNN: FPGA-GPU hybrid acceleration platform for recurrent neural networks". IEEE Transactions on Parallel and Distributed Systems. **33** (7): 1725–1738.*

221. ^a, b^ *Xiang Y, Kim H. "Pipelined data-parallel CPU/GPU scheduling for multi-DNN real-time inference." In: 2019 IEEE Real-Time Systems Symposium (RTSS). IEEE; 2019. p. 392–405.*

222. ^ *Zhou S, Zhang L. "Smart home electricity demand forecasting system based on edge computing." In: 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS). IEEE; 2018. p. 164–167.*

223. ^ *Zhan Y, Haddadi H (2019). "Towards automating smart homes: Contextual and temporal dynamics of activity prediction." In: Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers. 2019. p. 413–417.*

224. ^ *Han T, Muhammad K, Hussain T, Lloret J, Baik SW (2020). "An efficient deep learning framework for intelligent energy management in IoT networks". IEEE Internet of Things Journal. **8** (5): 3170–3179.*

225. ^ *Zhou W, Jia Y, Yao Y, Zhu L, Guan L, Mao Y, Liu P, Zhang Y (2019). "Discovering and understanding the security hazards in the interactions between iot devices, mobile apps, and clouds on smart home platforms." Proceedings of the 28th USENIX Conference on Security Symposium. pp. 1133–1150.*

226. ^ *Li L, Ota K, Dong M (2018). "Deep learning for smart industry: Efficient manufacture inspection system with fog computing". IEEE Transactions on Industrial Informatics. **14** (10): 4665–4673.*

227. ^ *Chu Y, Feng D, Liu Z, Zhang L, Zhao Z, Wang Z, Feng Z, Xia XG. "A Fine-Grained Attention Model for High Accuracy Operational Robot Guidance." IEEE Internet of Things Journal. 2022.*

228. ^ *Tuli S, Basumatary N, Gill SS, et al. HealthFog: An ensemble deep learning based Smart Healthcare System for Automatic Diagnosis of Heart Diseases in integrated IoT and fog computing environments. Future Generation Computer Systems. **104**: 187--200, 2020.*

229. ^Verde L, Brancati N, De Pietro G, Frucci M, Sannino G (2021). "A deep learning approach for voice disorder detection for smart connected living environments". ACM Transactions on Internet Technology (TOIT). 22 (1): 1–16.

230. ^Rahman MA, Hossain MS, Alrajeh NA, Guizani N (2020). "B5G and explainable deep learning assisted healthcare vertical at the edge: COVID-I9 perspective". IEEE Network. 34 (4): 98--105.

231. ^Kong X, Wang K, Wang S, Wang X, Jiang X, Guo Y, Shen G, Chen X, Ni Q (2021). "Real-time mask identification for COVID-19: An edge-computing-based deep learning framework". IEEE Internet of Things Journal. 8 (21): 15929–15938.

232. ^Hossain MS, Muhammad G (2020). "Deep learning based pathology detection for smart connected healthcare". IEEE Network. 34 (6): 120–125.

233. ^Muhammad G, Alhamid MF, Long X (2019). "Computing and processing on the edge: Smart pathology detection for connected healthcare". IEEE Network. 33 (6): 44–49.

234. ^Liu Y, Ma Z, Liu X, Ma S, Ren K (2019). "Privacy-preserving object detection for medical images with faster R-CNN." IEEE Transactions on Information Forensics and Security. 17: 69–84.

235. ^Luo Q, Li C, Luan TH, Shi W (2020). "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning." IEEE Internet of Things Journal. 7 (10): 9637--9650.

236. ^Jiang X, Yu FR, Song T, Leung VC. "Intelligent resource allocation for video analytics in blockchain-enabled internet of autonomous vehicles with edge computing." IEEE Internet of Things Journal. 9(16):14260-14272, 2020.

237. ^Liu Q, Han T, Xie JL, Kim B. "Livemap: Real-time dynamic map in automotive edge computing." In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications. IEEE; 2021. p. 1-10.

238. ^Liang S, Wu H, Zhen L, et al. (2022). "Edge YOLO: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles". IEEE Transactions on Intelligent Transportation Systems. 23 (12): 25345–25360.

239. ^Malawade A, Odema M, Lajeunesse-DeGroot S, Al Faruque MA (2021). "Sage: A split-architecture methodology for efficient end-to-end autonomous vehicle control". ACM Transactions on Embedded Computing Systems (TECS). 20 (5s): 1–22.

240. ^Wang J, Ke H, Liu X, Wang H (2022). "Optimization for computational offloading in multi-access edge computing: A deep reinforcement learning scheme". Computer Networks. 204: 108690.

241. ^Zhao P, Niu W, Yuan G, et al. "Brief industry paper: Towards real-time 3D object detection for autonomous vehicles with pruning search." In: 2021 IEEE 27th Real-Time and Embedded Technology and Appl

ications Symposium (RTAS). IEEE; 2021. p. 425–428.

242. ^Wang S, Yang S, Zhao C. *"SurveilEdge: Real-time video query based on collaborative cloud-edge deep learning." In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE; 2020. p. 251 9-2528.*
243. ^Trinh HD, Giupponi L, Dini P (2019). *"Urban anomaly detection by processing mobile traffic traces wit h LSTM neural networks." In: 2019 16th Annual IEEE International Conference on Sensing, Communica tion, and Networking (SECON). IEEE. pp. 1–8.*
244. ^Dautov R, Distefano S, Bruneo D, Longo F, Merlino G, Puliafito A, Buyya R (2018). *"Metropolitan intelli gent surveillance systems for urban areas by harnessing IoT and edge computing paradigms". Software: Practice and experience.* **48** *(8): 1475--1492.*
245. ^Menshchikov A, Shadrin D, Prutyanov V, Lopatkin D, Sosnin S, et al. *"Real-time detection of hogweed: UAV platform empowered by deep learning." IEEE Transactions on Computers.* **70** *(8): 1175–1188, 2021.*
246. ^Zhang Y, Yu J, Chen Y, Yang W, Zhang W, He Y (2022). *"Real-time strawberry detection using deep neu ral networks on embedded system (rtsd-net): An edge AI application". Computers and Electronics in Ag riculture.* **192***: 106586.*
247. ^Rustia DJA, Chiu L-Y, Lu C-Y, Wu Y-F, Chen S-K, et al. (2022). *"Towards intelligent and integrated pe st management through an AIoT-based monitoring system". Pest Management Science.* **78** *(10): 4288–4302.*
248. ^Liu X, Jiang Y, Jain P, Kim KH (2018). *"TAR: Enabling fine-grained targeted advertising in retail store s." In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Se rvices. pp. 323–336.*
249. ^Liu Y, Yang C, Jiang L, Xie S, Zhang Y (2019). *"Intelligent edge computing for IoT-based energy manag ement in smart cities". IEEE Network.* **33** *(2): 111–117.*
250. ^Alsalemi A, Himeur Y, Bensaali F, Amira A (2022). *"An innovative edge-based internet of energy soluti on for promoting energy saving in buildings". Sustainable Cities and Society.* **78***: 103571.*
251. ^Li E, Zeng L, Zhou Z, Chen X (2019). *"Edge AI: On-demand accelerating deep neural network inference via edge computing". IEEE Transactions on Wireless Communications.* **19** *(1): 447–457.*
252. ^Shlezinger N, Farhan E, Morgenstern H, Eldar YC. *"Collaborative inference via ensembles on the edge." In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASS P). IEEE; 2021. p. 8478-8482.*

253. ^Banitalebi-Dehkordi A, Vedula N, Pei J, Xia F, Wang L, Zhang Y (2021). "Auto-split: a general framework of collaborative edge-cloud ai." In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2021. pp. 2543–2553.

254. ^Luo Y, Yu S (2021). "AILC: Accelerate on-chip incremental learning with compute-in-memory technology". IEEE Transactions on Computers. **70** (8): 1225–1238.

255. ^Lv Z, Qiao L, Verma S (2021). "AI-enabled IoT-edge data analytics for connected living". ACM Transactions on Internet Technology. **21** (4): 1–20.

256. ^Tran TX, Le DV, Yue G, Pompili D (2018). "Cooperative hierarchical caching and request scheduling in a cloud radio access network". IEEE Transactions on Mobile Computing. **17** (12): 2729–2743.

257. ^Ning Z, Zhang K, Wang X, Guo L, et al. (2020). "Intelligent edge computing in internet of vehicles: a joint computation offloading and caching solution." IEEE Transactions on Intelligent Transportation Systems. **22** (4): 2212–2225.

258. ^Ren Q, Abbasi O, Kurt GK, et al. Caching and computation offloading in high altitude platform station (HAPS) assisted intelligent transportation systems. IEEE Transactions on Wireless Communications. **21** (11):9010-9024, 2022.

259. ^Paissan F, Ancilotto A, Farella E (2022). "PhiNets: a scalable backbone for low-power AI at the edge". ACM Transactions on Embedded Computing Systems. **21** (5): 1–18.

260. ^Moran A, Frasser CF, Roca M, Rossello JL (2019). "Energy-efficient pattern recognition hardware with elementary cellular automata". IEEE Transactions on Computers. **69** (3): 392--401.

261. ^Nunez-Yanez J, Howard N (2021). "Energy-efficient neural networks with near-threshold processors and hardware accelerators". Journal of Systems Architecture. **116**: 102062.

262. ^Jayakodi NK, Doppa JR, Pande PP. "A general hardware and software co-design framework for energy-efficient edge AI." In: 2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE; 2021. p. 1-7.

263. ^Sodhro AH, Pirbhulal S, De Albuquerque VHC (2019). "Artificial intelligence-driven mechanism for edge computing-based industrial applications". IEEE Transactions on Industrial Informatics. **15** (7): 4235–4243.

264. ^Tambe T, Yang E-Y, Ko GG, et al. A 16-nm SoC for Noise-Robust Speech and NLP Edge AI Inference With Bayesian Sound Source Separation and Attention-Based DNNs. IEEE Journal of Solid-State Circuits. 2022.

265. ^*Hao Y, Miao Y, Hu L, Hossain MS, Muhammad G, Amin SU (2019). "Smart-Edge-CoCaCo: AI-enabled smart edge with joint computation, caching, and communication in heterogeneous IoT". IEEE Network.* **33** *(2): 58–64.*

266. ^*Ham M, Moon J, Lim G, et al. "NNStreamer: Efficient and Agile Development of On-Device AI System s." In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE; 2021. p. 198-207.*

267. ^*Ham M, Woo S, Jung J, Song W, et al. Toward among-device AI from on-device AI with stream pipeline s. In: Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice; 2022. p. 285-294.*

268. ^*Chen W-H, Dou C, Li K-X, Lin W-Y, Li P-Y, Huang J-H, Wang J-H et al. (2019). "CMOS-integrated me mristive non-volatile computing-in-memory for AI edge processors." Nature Electronics.* **2** *(9): 420–4 28.*

269. ^*Xiong J, Zhao M, Bhuiyan MZA, Chen L, Tian Y (2019). "An AI-enabled three-party game framework f or guaranteed data privacy in mobile edge crowdsensing of IoT". IEEE Transactions on Industrial Infor matics.* **17** *(2): 922--933.*

270. ^*Zhang Q, Zhong H, Shi W, Liu L (2021). "A trusted and collaborative framework for deep learning in Io T". Computer Networks.* **193***: 108055.*

271. ^*Li Q, Ren J, Pan X, et al. "ENIGMA: Low-Latency and Privacy-Preserving Edge Inference on Heterogen eous Neural Network Accelerators." In: 2022 IEEE 42nd International Conference on Distributed Compu ting Systems (ICDCS). IEEE; 2022. p. 458–469.*

272. ^*Sinha S, Saha S, Alam M, et al. (2022). "Exploring Bitslicing Architectures for Enabling FHE-Assisted Machine Learning". IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.* **4 1** *(11): 4004–4015.*

273. ^*Rahman MS, Khalil I, et al. (2020). "Towards privacy preserving AI based composition framework in e dge networks using fully homomorphic encryption". Engineering Applications of Artificial Intelligence.* **94***: 103737.*

274. ^*Wang K, Xu SP, Chen CM, Islam SH, Hassan MM, et al. (2021). "A trusted consensus scheme for collabo rative learning in the edge AI computing domain". IEEE Network.* **35** *(1): 204–210.*

275. ^*Song M, Wang Z, Zhang Z, Song Y, Wang Q, Ren J, Qi H (2020). "Analyzing user-level privacy attack ag ainst federated learning". IEEE Journal on Selected Areas in Communications.* **38** *(10): 2430--2444.*

276. ^Lim WYB, Ng JS, Xiong Z, Jin J, et al. Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning. IEEE Transactions on Parallel and Distributed Systems. 33 (3):536–550, 2021.

277. ^Yu S, Nguyen P, Abebe W, et al. SPATL: salient parameter aggregation and transfer learning for heterogeneous federated learning. In: 2022 SC22: International Conference for High Performance Computing, Networking, Storage and Analysis (SC). IEEE Computer Society; 2022. p. 495-508.

278. ^Guo J, Wu J, Liu A, Xiong NN (2021). "LightFed: An efficient and secure federated edge learning system on model splitting". IEEE Transactions on Parallel and Distributed Systems. 33 (11): 2701–2713.

279. ^Huang A, Liu Y, Chen T, Zhou Y, Sun Q, Chai H, Yang Q (2021). "Starfl: Hybrid federated learning architecture for smart urban computing." ACM Transactions on Intelligent Systems and Technology (TIST). 12 (4): 1–23.

280. ^Tang Z, Jia W, Zhou X, Yang W, You Y (2020). "Representation and reinforcement learning for task scheduling in edge computing". IEEE Transactions on Big Data. 8 (3): 795–808.

281. ^Lou J, Tang Z, Jia W, Zhao W, Li J (2023). "Startup-aware dependent task scheduling with bandwidth constraints in edge computing". IEEE Transactions on Mobile Computing. 2023.

282. ^Lou J, Tang Z, Zhang S, Jia W, Zhao W, Li J (2022). "Cost-Effective Scheduling for Dependent Tasks With Tight Deadline Constraints in Mobile Edge Computing". IEEE Transactions on Mobile Computing. IEEE.

283. ^Zhang S, Jia W, Tang Z, Lou J, Zhao W (2022). "Efficient instance reuse approach for service function chain placement in mobile edge computing". Computer Networks. 211: 109010.

284. ^Lou J, Luo H, Tang Z, Jia W, Zhao W (2022). "Efficient container assignment and layer sequencing in edge computing". IEEE Transactions on Services Computing. 2022.

285. ^Tang Z, Lou J, Jia W (2022). "Layer Dependency-aware Learning Scheduling Algorithms for Containers in Mobile Edge Computing". IEEE Transactions on Mobile Computing. 2022.

286. ^Gu L, Zeng D, Hu J, Jin H, Guo S, Zomaya AY. "Exploring layered container structure for cost efficient microservice deployment." In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications. IEEE; 2021. p. 1-9.

287. ^Lou J, Tang Z, Jia W (2022). "Energy-efficient Joint Task Assignment and Migration in Data Centers: A Deep Reinforcement Learning Approach". IEEE Transactions on Network and Service Management. 2022. Published by IEEE.

288. ^Tang Z, Zhou X, Zhang F, Jia W, Zhao W (2018). "Migration modeling and learning algorithms for containers in fog computing". IEEE Transactions on Services Computing. 12 (5): 712–725.

289. ^Wang S, Urgaonkar R, Zafer M, He T, Chan K, Leung KK (2019). "Dynamic service migration in mobile edge computing based on Markov decision process". IEEE/ACM Transactions on Networking. 27 (3): 1272–1288.

290. ^Ma Y, Liang W, Li J, Jia X, Guo S (2020). "Mobility-aware and delay-sensitive service provisioning in mobile edge-cloud networks". IEEE Transactions on Mobile Computing. 21 (1): 196–210.

291. ^Ma L, Yi S, Carter N, Li Q (2018). "Efficient live migration of edge services leveraging container layered storage". IEEE Transactions on Mobile Computing. 18 (9): 2020–2033.

292. ^Benjaponpitak T, Karakate M, Sripanidkulchai K. "Enabling live migration of containerized applications across clouds." In: Proceedings of 2020 IEEE Conference on Computer Communications (INFOCOM). IEEE; 2020. p. 2529–2538.

293. ^Tang Z, Zhang F, Zhou X, Jia W, Zhao W (2022). "Pricing model for dynamic resource overbooking in edge computing". IEEE Transactions on Cloud Computing. IEEE.

294. ^Hu S, Shi W, Li G (2022). "CEC: A containerized edge computing framework for dynamic resource provisioning". IEEE Transactions on Mobile Computing. Published by IEEE.

295. ^Luo Z, Wu C, Li Z, Zhou W (2019). "Scaling geo-distributed network function chains: A prediction and learning framework". IEEE Journal on Selected Areas in Communications. 37 (8): 1838–1850.

296. ^Wang S, Ding Z, Jiang C (2020). "Elastic scheduling for microservice applications in clouds". IEEE Transactions on Parallel and Distributed Systems. 32 (1): 98–115.

297. ^Lv W, Wang Q, Yang P, Ding Y, Yi B, Wang Z, Lin C (2022). "Microservice deployment in edge computing based on deep q learning". IEEE Transactions on Parallel and Distributed Systems. 33 (11): 2968–2978.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.