Onboard Optimization and Learning: A Survey

Monirul Islam Pavel¹, Siyi Hu^{1*}, Mahardhika Pratama¹, Ryszard Kowalczyk¹

^{1*}STEM, University of South Australia, Mawson Lakes, 5095, South Australia, Australia.

*Corresponding author(s). E-mail(s): siyi.hu@unisa.edu.au; Contributing authors: monirul_islam.pavel@mymail.unisa.edu.au; dhika.pratama@unisa.edu.au; ryszard.kowalczyk@unisa.edu.au;

Abstract

Onboard learning is a transformative approach in edge AI, enabling real-time data processing, decision-making, and adaptive model training directly on resource-constrained devices without relying on centralized servers. This paradigm is crucial for applications demanding low latency, enhanced privacy, and energy efficiency. However, onboard learning faces challenges such as limited computational resources, high inference costs, and security vulnerabilities. This survey explores a comprehensive range of methodologies that address these challenges, focusing on techniques that optimize model efficiency, accelerate inference, and support collaborative learning across distributed devices. Approaches for reducing model complexity, improving inference speed, and ensuring privacy-preserving computation are examined alongside emerging strategies that enhance scalability and adaptability in dynamic environments. By bridging advancements in hardware-software co-design, model compression, and decentralized learning, this survey provides insights into the current state of onboard learning to enable robust, efficient, and secure AI deployment at the edge.

Keywords: onboard, on-device, learning, training, inference, optimization, Edge AI

1 Introduction

The increasing adoption of edge artificial intelligence (AI) has driven a paradigm shift toward onboard optimization and learning, enabling devices to process data, make decisions, and update models locally without relying on centralized servers [1]. Traditional cloud-based AI architectures face significant limitations, including high latency, bandwidth constraints, and security risks associated with data transmission [2, 3]. These issues are particularly critical for applications such as autonomous vehicles, industrial IoT, and smart infrastructure, where real-time decision-making and responsiveness are essential. Onboard learning ¹ addresses these challenges by performing computations directly on resource-constrained devices, ensuring faster, more efficient, and privacy-preserving AI inference and adaptation [4–6].

Despite its advantages, onboard learning introduces several challenges. Edge devices typically have limited computational power, memory, and energy resources, making it difficult to deploy and run complex AI models efficiently [7, 8]. Furthermore, optimizing inference for real-time applications requires balancing model accuracy, latency, and power consumption. Additionally, decentralized learning frameworks must facilitate model training and updates across multiple devices without compromising privacy or requiring frequent cloud synchronization [2, 3]. Finally, ensuring security and robustness against adversarial threats is essential for onboard AI systems operating in dynamic and often untrusted environments.

This survey systematically categorizes onboard learning methodologies into five fundamental aspects (summarized in Table 1. The first aspect, Model Compression, focuses on reducing the size and complexity of AI models to fit within the computational constraints of edge devices. Techniques such as pruning, quantization, knowledge distillation, and neural architecture search are explored to optimize model efficiency without significantly degrading performance [5, 6]. The second aspect, Efficient Inference, examines strategies that minimize latency and energy consumption while maintaining real-time AI responsiveness. Methods such as model partitioning, early exit mechanisms, and computation offloading are discussed to improve inference efficiency [7-10]. The third aspect, Decentralized Learning, enables collaborative training and model adaptation across multiple devices without relying on a centralized server. Federated learning, continual learning (CL), and adaptive techniques are analyzed to improve scalability and maintain privacy while adapting to dynamic data distributions [2–5]. The fourth aspect, Security and Privacy, addresses the risks associated with deploying AI models in untrusted environments. Methods such as differential privacy, adversarial robustness, and encrypted model updates are discussed to safeguard onboard AI systems against potential threats [3, 7]. The final aspect Advanced Topics discusses research directions that enhance the effectiveness, scalability, and resilience of optimized onboard learning. This includes hardware-software co-design to optimize AI, bridging model compression with CL to increase flexibility while preserving efficiency, scalability, and standardization for efficient deployment across heterogeneous edge platforms [1, 11, 12].

By structuring onboard learning around these core aspects, this survey provides a comprehensive review of the methodologies optimizing AI for edge computing. The remainder of this paper is organized as follows. Section 2 explores techniques for model compression and efficiency. Section 3 discusses inference optimization strategies.

¹Terms such as on-device learning, onboard learning, and edge-based learning or edge AI are often used interchangeably in the existing literature. In this paper, we will use 'onboard learning' for standardization.

Table 1: Taxonomy of key techniques and challenges

Topic	Techniques	Performance Constraint				Latency & Comm		Hardware Adaptability		Optimization Complexity		
		HCC	MSC	HPC	LI	CO	OC	HC	HEE	TDB	PT	OC
	Pruning	✓	✓	-	✓	-	-	✓	-	✓	-	-
Model Compression	Quantization	✓	✓	✓	✓	-	-	✓	-	-	-	-
	Knowledge Distillation	✓	✓	-	✓	-	-	✓	-	-	-	-
	Neural Architecture Search	✓	✓	-	✓	-	✓	✓	✓	✓	✓	✓
Efficient Inference	Computation Offloading	√	-	✓	√	√	✓	-	✓	-	-	✓
	Model Partitioning	✓	-	-	✓	-	-	✓	✓	-	-	✓
	Early Exit Strategies	✓	-	✓	✓	-	-	-	✓	-	-	✓
Decentralized Learning	Federated Learning	-	-	-	-	√	✓	-	✓	✓	-	✓
	Continual Learning $+$ DL	-	-	-	-	-	✓	-	✓	✓	-	✓
	Adaptive Learning $+$ DL	✓	✓	-	-	✓	✓	-	-	✓	-	-
Security & Privacy	Privacy Protection	√	√	-	-	-	-	-	✓	-	-	✓
	Secure Model Execution	✓	✓	✓	-	-	-	✓	✓	-	-	✓
	Explainable AI	-	-	-	-	-	-	-	✓	-	-	✓
Advanced Topics	Model Compression + CL	-	-	-	-	-	✓	✓	-	✓	✓	✓
	Scalability	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Hardware Co-Design	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Performance Constraints: High Computational Cost (HCC), Memory & Storage Constraints (MSC), High Power Con-

Section 4 covers decentralized and adaptive learning methods. Section 5 examines security and privacy considerations. Section 6 presents emerging advanced topics for optimized onboard learning. The survey concludes with an overview and discussion of future directions for onboard AI.

2 Model Compression

Deploying deep learning models on resource-constrained devices requires efficient compression techniques to reduce computational cost, memory footprint, and power consumption while maintaining optimal performance. This section explores key model compression methods (summarized in Table 2, including pruning, knowledge distillation, quantization, weight sharing, and Neural Architecture Search (NAS), with a focus on their role in optimizing the onboard learning process. These techniques enable efficient and lightweight models by eliminating redundancy, reducing precision, and optimizing architectures, ensuring deep learning models remain practical for real-world applications.

2.1 Efficient Parameter Reduction: Pruning

Pruning is a fundamental technique for onboard learning that reduces the number of parameters in a neural network, enabling deep learning models to operate within the memory, computational, and energy constraints of resource-limited devices [13–15]. By eliminating less important weights, pruning reduces model size while preserving accuracy, making it highly suitable for real-time applications. The pruning process assigns importance scores to parameters, removing those with the lowest values based on a

[•] Latency and Communication: Latency Issues (LI), Communication Overhead (CO)
• Hardware Adaptability: Hardware Compatibility (HC), Heterogeneous Edge Environments (HEE)
• Optimization Complexity: Training & Deployment Bottlenecks (TDB), Performance Trade-off (PT), Optimization Complexity (OC)

Table 2: Summary of model compression methods for optimized onboard learning.

Technique	Category	Reference	Key Highlights
Pruning	Unstructured vs. Structured Pruning at Initialization Pruning During Training	[16–18] [19–22] [23–25]	Sparse vs. hardware-friendly pruning. Early pruning for efficiency. Dynamic real-time pruning.
Quantization	Post-Training Quantization Quantization-Aware Training Mixed Precision Quantization	[26, 27] [28, 29] [30, 31]	Converts FP32 to INT8. Simulates quantization in training. Variable bit-width layers.
Knowledge Distillation	Teacher-Student Learning KD for Edge Devices Federated KD	[32, 33] [34–36] [37–40]	Model compression via knowledge transfer. Reduces FLOPs and memory. Distills global to local models.
Neural Architecture Search	Hardware-Aware NAS NAS Compression Challenges Few-Shot NAS	[41–43] [44, 45] [46, 47]	Auto-optimized for edge devices. High search cost, memory use. Fast, memory-efficient architecture search.

predefined pruning rate. Selecting an optimal pruning rate is critical to maintaining accuracy while minimizing computational overhead.

Unstructured vs. Structured Pruning. Pruning can be broadly classified into unstructured and structured approaches [16]. Unstructured pruning removes individual weights across the network, leading to sparse weight matrices that can improve efficiency when supported by dedicated accelerators. However, since standard deep learning frameworks and hardware do not efficiently support sparse computation, unstructured pruning often requires specialized hardware or software libraries for performance gains. In contrast, structured pruning removes entire groups of weights, such as neurons, channels, or filters, reducing model dimensions in a way that aligns well with conventional deep learning frameworks. Structured pruning is generally more suitable for onboard learning, as it produces a compact, hardware-friendly model compatible with edge devices [17, 18].

Pruning at Different Stages. Pruning can be applied at various points in the model lifecycle: before training (pruning at initialization), during training (dynamic pruning), or after training (post-training pruning). Each stage offers distinct trade-offs in terms of computational efficiency and model accuracy.

Pruning at Initialization. Pruning at initialization removes unnecessary parameters before training begins, reducing model complexity and enabling efficient training on resource-constrained devices [17, 48–50]. A common approach is layerwise pruning, where entire layers or channels are removed based on their contribution to model performance [17, 18, 51, 52].

Several methods have been proposed to enhance pruning at initialization. SynFlow prevents layer collapse by selectively pruning while maintaining model trainability [19]. Single-shot network pruning (SNIP) assigns sensitivity scores to each weight and removes unimportant ones before training, ensuring critical connections are preserved [20, 21]. CPSCA, introduced by Liu et al. [53], employs spatial and channel attention to identify and prune unimportant channels while preserving inference accuracy. However, attention mechanisms in CPSCA introduce computational overhead, which may be prohibitive for highly constrained devices.

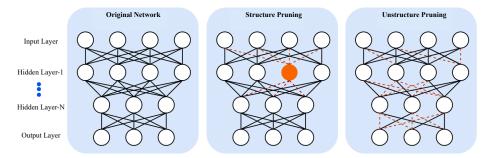


Fig. 1: General architecture of structured and unstructured pruning. The dotted orange lines and circles denote neurons or connections to be pruned.

Another promising method is channel clustering pruning, proposed by Chang et al. [22]. This method groups channels based on feature similarity before pruning, optimizing the pruned network through knowledge transfer. Such approaches facilitate quicker performance recovery post-pruning and improve model efficiency.

Unstructured pruning at initialization focuses on weight-level sparsity, where individual parameters are removed based on significance [54, 55]. While computationally efficient, single-shot pruning may lead to accuracy degradation if not followed by iterative fine-tuning [56]. Together, these pruning techniques enable flexible and efficient model compression for onboard AI systems.

Pruning During Training. Dynamic pruning modifies the model structure during training, selectively removing parameters based on real-time performance metrics. This technique can significantly reduce computational costs, with applications on edge devices demonstrating up to a 94% reduction in multiply-accumulate (MAC) operations [25].

Adaptive pruning, such as PruneFL [23], dynamically resizes models during federated learning to balance communication and computation overhead while maintaining accuracy. Online pruning, introduced by Guo et al. [24], integrates real-time pruning into the training process, eliminating near-zero parameters to reduce computational load and improve inference efficiency. These techniques enable onboard AI to dynamically optimize model complexity without significant performance degradation, making them ideal for resource-limited environments.

2.2 Precision Optimization: Quantization

Quantization is a model compression technique that reduces the precision of weights and activations, typically converting floating-point values to lower-bit fixed-point representations. This reduction in precision decreases memory usage and computational costs, making deep learning models more efficient for resource-constrained devices [57–59]. Quantization enables faster inference and lowers energy consumption while maintaining model performance, making it a practical solution for real-time applications [60–62].

Quantization can be represented as a mapping function $C_Q = C_{Q,\text{deq}} \circ C_{Q,\text{int}}$, where $C_{Q,\text{int}}: \mathbb{R} \to \mathbb{Z}$ maps real values to integers, and $C_{Q,\text{deq}}: \mathbb{Z} \to \mathbb{R}$ is an affine function that recovers original values. This process defines a quantization grid, with a step size s and zero point z aligning the quantizer with the input range: $C_Q(v) = C_{Q,\text{deq}}(C_{Q,\text{int}}(v)) = s \cdot \left(\left\lfloor \frac{v_{\text{int}}+z}{s} \right\rfloor\right)$. Here, v_{int} is the quantized integer value, with $s = \frac{q-p}{2nk-1}$ determining the step size for an n_k -bit representation within the range [p,q].

Post-Training Quantization (PTQ). PTQ applies quantization to a trained model without additional retraining, converting FP32 tensors to INT8 while maintaining accuracy [26, 27]. A single scaling parameter adjusts the quantization range per layer or channel, ensuring adaptability to different data distributions. Fine-tuning based on calibration samples further refines accuracy. However, PTQ may struggle to generalize across unseen data distributions, leading to potential precision loss. Online dynamic schemes mitigate this by recalibrating tensor ranges during inference but introduce computational overhead [63].

Quantization-Aware Training (QAT). Unlike PTQ, QAT incorporates quantization into the training process to enhance performance on quantized hardware. During forward propagation, weights and activations are quantized, simulating deployment conditions [28, 29]. Backpropagation updates FP32 weights, minimizing quantization errors while training. This dual-format approach improves model robustness but introduces hyperparameter tuning complexities [64].

Mixed Precision Quantization (MPQ). MPQ mitigates accuracy loss in ultra-low precision models by assigning different bit widths to different network components [30]. While traditional INT8 quantization preserves accuracy in critical layers, MPQ improves compression and efficiency for onboard deployment. The challenge lies in optimally selecting bit widths, prompting techniques like OneShot Mixed-Precision Search [30], which improved model correlation by 41% between sampled child models and standalone fine-tuned models on ImageNet. Hessian Aware Quantization (HAWQ) introduced second-order MPQ, achieving an 8× compression on ResNet-20 with 1% higher accuracy on CIFAR-10 and 68% top-1 accuracy on ImageNet with a 1MB model [31]. MPQ offers a balance between accuracy and efficiency, making it highly suitable for onboard learning.

By integrating PTQ, QAT, and MPQ, onboard systems optimize model efficiency while maintaining performance, enabling real-time inference under stringent resource constraints.

2.3 Knowledge Transfer for Compact Models: Knowledge Distillation

Deploying deep learning models on resource-constrained edge devices presents significant challenges due to limited computational power, memory, and energy efficiency [65, 66]. Knowledge distillation (KD) addresses these limitations by transferring knowledge from a high-capacity teacher model to a lightweight student model, enabling efficient onboard deployment [32, 33, 67]. The student model learns to approximate the teacher's predictive capabilities by leveraging the teacher's probability distribution over output classes, commonly referred to as soft labels [68]. This technique allows

smaller models to retain high accuracy while reducing memory footprint and inference latency.

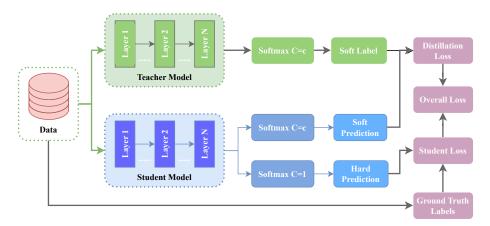


Fig. 2: General architecture of knowledge distillation

During training, the student model minimizes a loss function that integrates both hard labels (ground truth) and soft labels from the teacher. Soft labels encode interclass relationships, providing richer supervision and enabling the student model to generalize better. However, since well-trained teacher models tend to produce sparse probability distributions with high confidence in the correct class, a temperature parameter C is introduced in the softmax function to soften the distribution and enhance knowledge transfer.

Mathematically, given input data **X** with associated hard labels **y** for K classes, the soft labels from the teacher model $\mathbf{q} = \{q_i\}_{i=1}^K$ are computed as: $q_i = \frac{\exp(z_i/C)}{\sum_{j=1}^K \exp(z_j/C)}$, where z_i represents the teacher's logit for class i and C is the temperature parameter. As C increases, \mathbf{q} approaches a uniform distribution, enhancing inter-class information.

The student model is optimized using a combined loss function that balances hard label loss L_{Hard} and soft label loss L_{Soft} :

$$L_{\text{KD}} = \frac{1}{R} \sum_{\mathbf{X}, \mathbf{y}} \left((1 - \gamma) L_{\text{Hard}}(\mathbf{X}, \mathbf{y}) + \gamma L_{\text{Soft}}(\mathbf{X}, \mathbf{q}) \right), \tag{1}$$

where γ controls the trade-off between the two loss terms, and R is the total number of input samples. The individual loss functions are defined as: $L_{\text{Hard}}(\mathbf{X}, \mathbf{y}) = -\sum_{i=1}^{K} y_i \log p_i(\mathbf{X})$ or $L_{\text{Soft}}(\mathbf{X}, \mathbf{q}) = -\sum_{i=1}^{K} q_i \log p_i(\mathbf{X})$, where $p_i(\mathbf{X})$ represents the student model's predicted probability for class i. Hard labels use C = 1, whereas soft labels utilize the teacher's temperature.

Knowledge distillation is widely applied in onboard learning to improve memory efficiency and reduce computational demands [69–72]. Sepahvand et al. [34] introduced a tensor decomposition-based KD approach, achieving a $265.67 \times$ compression rate

for ResNet-18 with minimal accuracy loss. In [35], a method named Moonshine is developed with similar student-teacher architecture to reduce resource and memory utilization, which resulted 40% smaller model with 5.02% less error than baseline after compression. Xiao et al. [36] introduced DQN-KD, applying knowledge distillation with reinforcement learning to minimize memory utilization where it achieved 50.4% fewer FLOPs (flops full form floating point operations per second) than baseline with 47.5% parameter reduction.

To reduce the communication overhead during onboard deployment, Itahara et al., [37] demonstrated semi-supervised federated learning with knowledge distillation that deducted 99% communication cost while maintaining similar accuracy compared to benchmark via onboard local models' outputs exchange between heterogeneous devices and optimizes local model. This proposed approach adapted KD for onboard deployment by transferring global knowledge from a teacher model to client models using soft labels, allowing clients to learn generalized patterns despite non-IID data. Moreover, Qu et al. [38] demonstrated an adaptive quantized federated knowledge distillation approach for edge devices to address the bottleneck of communication cost and speed up model training while maintaining accuracy with 2% of the original model size. Luo et al. [39] proposed KeedEdge with deep neural network (DNN) with 5.86% improvement in the student model through knowledge distillation with 14% model reduction aiming to lower complexity and latency for UAV positioning.

To address the challenges of heterogeneous computation, Qi et al. [40] introduced bidirectional knowledge distillation to facilitate efficient and scalable onboard deployment for modulation classification in IoT-edge systems in federated learning framework. The proposed knowledge distillation approach of multi-teacher knowledge distillation, where the global network was regarded as a student network that unifies the heterogeneous knowledge from multiple teacher networks, enabled lightweight model updates by transferring essential insights between local and global models, minimizing communication costs and reducing memory and computational demands on devices.

Overall, knowledge distillation is crucial for onboard deployment as it creates lightweight models that approximate the accuracy of larger counterparts while reducing memory, computational demands, and inference latency. By transferring "soft" labels from a teacher model, it encodes rich inter-class relationships, enabling efficient deployment in resource-constrained environments. This is especially vital for real-time applications like autonomous vehicles, where fast, localized decision-making is essential. Furthermore, distillation allows flexible deployment across diverse devices by optimizing student models for specific capabilities, enhancing performance in heterogeneous environments. It also supports continual learning by enabling incremental updates without full retraining, reducing computational overhead. In summary, knowledge distillation ensures efficient, adaptive, and high-performing onboard models that are adaptable to the constraints of edge and embedded systems.

2.4 Automated Architecture Optimization: Neural Architecture Search

Deploying deep learning models on edge devices presents challenges due to diverse hardware architectures, real-time processing demands, and stringent resource constraints. Unlike traditional deep learning models optimized for high-performance GPUs or cloud environments, onboard AI must balance accuracy, latency, and energy efficiency [73]. Neural Architecture Search (NAS) has emerged as a powerful model compression approach, automating the design of efficient deep learning architectures tailored to memory-constrained embedded devices [12]. By dynamically optimizing models for specific hardware, NAS accelerates the design process and enhances adaptability, making it a key enabler of next-generation edge AI.

Traditional deep learning models require manual adaptation for resource-limited environments, which is labor-intensive and suboptimal for real-time applications. To address this, Lyu et al. [41] proposed a multi-objective resource-constrained NAS framework that directly searches for optimal architectures on edge devices (e.g., Jetson Nano, Raspberry Pi) without proxy tasks. Their approach achieved Pareto-optimal architectures, demonstrating significant improvements in precision, memory consumption, and latency [42, 43]. By eliminating the need for human-designed architectures, hardware-aware NAS significantly enhances onboard deployment efficiency.

Despite its advantages, NAS-based compression for onboard processing is computationally expensive, as it requires transferring new models to remote embedded devices, executing them, and gathering performance data to compare architectures [44]. While differential architecture search techniques like DARTS [45] reduce search time, they introduce high memory overhead due to the need to store intermediate outputs of all candidate operations.

To balance search time and optimal performance, few-shot NAS frameworks such as MetaNAS leverage meta-learning to optimize architectures efficiently [46]. Additionally, On-NAS [47] addresses memory constraints in embedded devices by employing an expectation-based operation selection method and an edge-pair search strategy. This approach reduces memory consumption while maintaining high performance, making it a promising solution for onboard deep learning deployment.

In conclusion, NAS plays a crucial role in compressing and optimizing deep learning models for onboard AI, enabling automated adaptation to onboard hardware constraints. By leveraging hardware-aware and few-shot NAS methods, onboard systems can achieve efficient, high-performance compressed model design while minimizing memory usage and computational overhead.

3 Efficient Inference

To achieve efficient inference on resource-constrained edge devices, the combination of optimization strategies plays vital role. In onboard setup with low processing capability, computation offloading enables efficient inference through computationally intensive tasks delegated to external resources, reducing the burden onboard. While offloading is limited by network constraints or privacy concerns, model partitioning

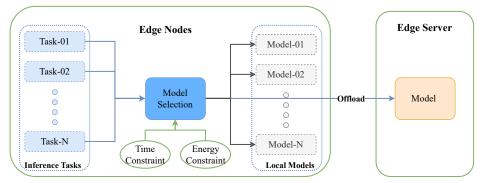


Fig. 3: General architecture of edge-based computational offloading.

offers a viable alternative by distributing computations between the device and external systems to enhance inference for execution. Moreover, in onboard execution, where responsiveness and minimal energy consumption are required, early exit strategies further optimize inference by allowing models to terminate processing once a confident prediction is reached. This section highlights these techniques in detail by discussing their benefits and trade-offs in achieving efficient inference for optimized onboard processing.

3.1 Distributed Computation for Onboard AI: Computation Offloading

Computation offloading enables resource-constrained devices to execute deep learning tasks efficiently by delegating computationally intensive operations to nearby edge servers. This approach reduces energy consumption and improves inference speed, which is critical for real-time applications such as autonomous navigation and surveillance [74–79]. While network latency and security risks remain concerns, offloading enhances scalability and optimizes resource utilization across distributed systems [80, 81].

To improve inference efficiency, DNNs can leverage offloading strategies that selectively assign tasks between local devices and edge servers. Zhou et al. [82] proposed a fused layer-based model parallelism approach that dynamically partitions DNN layers in multi-access edge computing (MEC) environments using Particle Swarm Optimization, achieving a 12.75× inference speedup. Similarly, Teerapittayanon et al. [83] introduced a distributed DNN framework where edge devices execute partial inference, aggregating intermediate outputs to minimize communication overhead. To further optimize offloading, Nikoloska et al. [84] developed a data selection scheme that transmits only uncertain samples, while Fresa et al. [85] employed LP-relaxation-based dynamic programming for optimal task scheduling.

Several techniques improve offloading efficiency by reducing computational load and bandwidth requirements. Deep compressive offloading, introduced by Yao et al. [86], applies compressive sensing to reduce transmitted data size, achieving a $2\times-4\times$ reduction in latency with minimal accuracy loss. Gao et al. [87] proposed entropy-based

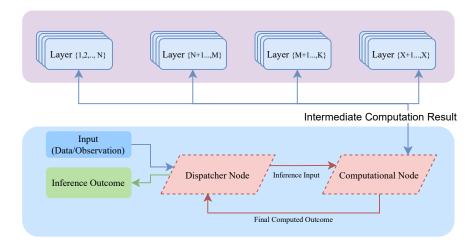


Fig. 4: Model partitioning for deep learning inference at the edge.

pruning to selectively remove redundant features, further improving efficiency. Additionally, runtime management strategies dynamically adjust computational priorities, bypassing less critical computations to minimize latency [88].

When a single device lacks sufficient resources, collaborative edge computing enables multiple devices to function as a virtual edge server, distributing inference tasks dynamically [89]. This cooperative framework balances computational load reduces latency, and enhances real-time responsiveness by leveraging shared computational resources across edge devices.

3.2 Adaptive Execution Strategies: Model Partitioning

Model partitioning optimizes onboard inference by distributing computation across multiple devices, ensuring low latency, energy efficiency, and real-time responsiveness [90, 91]. Instead of fully offloading computations, partitioning allows selective execution of neural network layers, retaining early-stage processing onboard while delegating high-complexity computations to external resources [92]. This technique balances computational demand while minimizing data transfer overhead.

Layer-wise partitioning assigns computations at specific layer boundaries, enabling lightweight feature extraction locally while deeper layers refine representations on an edge server. Operator-based partitioning further decomposes models into computational units, dynamically distributing tasks for efficient execution. To enhance adaptability, adaptive partitioning adjusts assignments in real-time based on network conditions, processing power, and latency constraints [91].

To further optimize partitioned inference, right-sizing through early exit mechanisms prevents unnecessary computations by terminating inference at intermediate layers once a confident prediction is reached [88, 93]. Confidence-based early exit stops

execution when the model reaches a high-certainty prediction, reducing energy consumption. Alternatively, threshold-based adaptive exit dynamically adjusts exit points based on input complexity, ensuring efficient processing [94, 95].

Despite its advantages, model partitioning introduces trade-offs between latency, accuracy, and energy consumption. While offloading reduces onboard computation, unstable network conditions may introduce latency overhead. Additionally, frequent inter-device communication can impact efficiency, necessitating optimized scheduling strategies. By integrating intelligent partitioning with adaptive exit mechanisms, onboard AI achieves a balance between computational efficiency and inference performance, making it a critical approach for real-time onboard learning.

3.3 Dynamic Stopping for Faster Inference: Early Exit Strategies

Early exit mechanisms optimize onboard inference by allowing deep learning models to terminate computation once a confident prediction is reached, reducing latency and energy consumption [96, 97]. This technique is particularly beneficial for resource-constrained devices, where real-time responsiveness is critical. By skipping unnecessary layers, early exits enhance efficiency without sacrificing accuracy, making them essential for fast and adaptable onboard/edge inference.

Early exit strategies enable models to process input data through initial layers and stop inference if confidence exceeds a predefined threshold [98]. If confidence remains low, additional layers may be executed locally or offloaded to an edge or cloud server for further refinement [99, 100]. This flexibility minimizes computational overhead while maintaining predictive reliability.

When integrated with computation offloading and model partitioning, early exits further optimize resource allocation. Initial layers process data onboard, and if a confident prediction is achieved, offloading is bypassed, reducing latency and bandwidth usage [99, 101, 102]. Otherwise, deeper layers are computed remotely to refine predictions.

Furthermore, adaptive approaches enhance early exits by dynamically selecting the optimal exit layer based on real-time conditions. Reinforcement learning (RL) improves efficiency by learning optimal exit points based on past inference outcomes [103, 104]. An RL agent determines when to stop computation, balancing speed and accuracy to match dynamic workloads. Federated learning (FL) further refines early exit models by enabling collaborative training across multiple edge devices without sharing raw data, preserving privacy while improving robustness [105–107].

By integrating early exits with offloading, RL, and FL, onboard inference becomes more efficient, reducing computation, minimizing energy consumption, and improving adaptability to dynamic workloads while maintaining data privacy.

4 Decentralized Learning

Decentralized learning is crucial for onboard AI, enabling edge devices to train and update models collaboratively without relying on centralized servers. By distributing learning across multiple devices it enhances privacy, reduces latency, and improves

adaptability in dynamic environments. However, deploying decentralized learning on resource-constrained devices introduces several challenges, including data heterogeneity, model adaptation over time, and computational efficiency under dynamic resource constraints. This section systematically examines decentralized learning in onboard AI, categorizing existing methods based on the key challenges they address.

4.1 Privacy-Preserving Model Updates: Federated Learning

Federated Learning (FL) has emerged as a privacy-preserving decentralized learning framework, allowing devices to collaboratively train models while keeping data local [108]. Instead of transferring raw data to a central server, FL shares only model updates, reducing privacy risks and communication costs. However, FL faces significant challenges, including heterogeneous data distributions, security vulnerabilities, and optimization constraints.

Handling Data Heterogeneity in FL. One major limitation of FL is non-identically distributed (non-IID) data across edge devices, which can degrade model performance. To address this, meta-learning-based learning-to-learn techniques improve generalization across different data distributions [109, 110]. Additionally, personalized FL techniques train models that adapt to specific device distributions, enhancing performance in heterogeneous settings [111, 112].

Security and Robustness in FL. Federated learning remains vulnerable to adversarial attacks, particularly Byzantine failures, where malicious nodes send corrupted updates. Tao et al. [113] introduced Byzantine-resilient FL, which integrates distributed gradient descent with gradient compression to mitigate adversarial influences while ensuring convergence. These techniques enhance security without compromising learning efficiency.

Optimizing Local Model Updates. FL's communication overhead can be reduced through gradient compression and adaptive update strategies. Wang et al. [114] proposed an optimization algorithm that balances local update frequency and global aggregation, improving computational efficiency. Additionally, Split Federated Learning (SFL) [115] partitions models between devices and servers to reduce memory and processing burdens, achieving a 43% reduction in completion time while maintaining model accuracy.

Federated learning serves as the foundation for decentralized AI, but it assumes a relatively stable learning environment. In real-world applications, onboard AI must continuously adapt to evolving data distributions, requiring continual learning mechanisms.

4.2 Continual Adaptation in Decentralized Systems: Continual Learning

Continual Learning (CL) enables AI models to incrementally acquire new knowledge while preserving past information, making it critical for real-time onboard learning. Unlike traditional FL, which assumes fixed training tasks, CL allows models to evolve dynamically. However, CL faces two key challenges: catastrophic forgetting and computational efficiency in resource-limited environments.

Mitigating Catastrophic Forgetting. Standard CL methods suffer from catastrophic forgetting, where previously learned knowledge deteriorates as new tasks are introduced. To address this, LightCL [116, 117] freezes the lower and middle layers of a pre-trained model while updating only higher layers, reducing memory consumption while preserving past knowledge. Additionally, task-aware dynamic masking (TDM) selectively retains critical parameters, balancing adaptation and retention.

Decentralized Continual Learning. Continual learning in decentralized settings introduces additional challenges due to device heterogeneity and intermittent connectivity. Decoupled replay mechanisms, such as Chameleon [118], maintain separate short-term and long-term memory buffers, optimizing data retention while minimizing resource usage. Further improvements are achieved through hardware-accelerated optimizations, as demonstrated in Chameleon-Accel [119, 120], which achieves up to $2.1\times$ speedup over SLDA and $3.5\times$ over Latent Replay, making CL feasible for onboard deployment.

While CL enhances adaptability, it still operates under the assumption of relatively stable resource availability. However, in highly dynamic environments, onboard AI must continuously adjust its learning strategy based on fluctuating constraints, requiring adaptive learning techniques.

4.3 Dynamic Adaptation Under Resource Constraints: Adaptive Learning

Adaptive learning extends continual learning by dynamically adjusting model behavior based on real-time changes in computational resources, task complexity, and environmental variations. In edge AI scenarios, where resources fluctuate, learning strategies must be both efficient and responsive.

Real-Time Model Adjustment. IMPALA [121] and SEED RL [122] employ asynchronous updates and parallel experience collection, enabling models to efficiently learn from decentralized environments while reducing synchronization bottlenecks. Importance-weighted corrections ensure stability by mitigating inconsistencies in decentralized updates [123].

Resource-Aware Learning Strategies. Jin et al. [124] proposed a resource-aware optimization framework that formulates a non-linear mixed-integer program to dynamically allocate computational resources for learning tasks. Their method reduces communication overhead and optimizes onboard switching between edge devices, ensuring efficient adaptation in dynamic environments.

Scalability and Fault Tolerance in Decentralized Systems. By prioritizing localized learning and reducing reliance on central servers, decentralized approaches enhance scalability, improve latency efficiency, and strengthen fault tolerance. These properties make decentralized learning particularly suitable for autonomous edge systems, IoT networks, and federated AI applications. Through dynamic resource allocation and distributed model updates, adaptive learning ensures robust and efficient AI decision-making in continuously evolving real-world settings.

5 Security and Privacy

The deployment of deep learning models on edge devices introduces significant security and privacy challenges [125]. Unlike centralized cloud-based AI, onboard learning operates in resource-constrained environments, making it susceptible to adversarial attacks, data breaches, and inefficiencies in privacy-preserving mechanisms [126]. These vulnerabilities enhance risks to model integrity, confidentiality, and robustness, necessitating dedicated security measures. This section systematically examines key security threats and mitigation strategies, focusing on three fundamental aspects: privacy protection in decentralized learning, secure model execution, and trustworthy AI for onboard learning (summarized in Table 3.

5.1 Privacy Protection in Decentralized Learning

Federated learning (FL) enables collaborative training across edge devices without requiring raw data transmission, mitigating privacy risks [127, 128]. However, despite its decentralized nature, FL remains vulnerable to inference attacks, where adversaries exploit model updates to extract sensitive information. Differential privacy (DP) addresses these risks by adding controlled noise to model updates, preventing malicious parties from reconstructing training data [129].

While DP improves privacy, its implementation in onboard scenarios presents challenges. Excessive noise injection degrades model accuracy, and the heterogeneous nature of edge devices complicates uniform privacy protection. Furthermore, DP introduces computational overhead, which can hinder real-time learning in resource-constrained environments. To optimize trade-offs between privacy and utility, adaptive DP techniques dynamically adjust noise levels based on model sensitivity and device constraints. Recent approaches, such as Hierarchical Split Federated Learning, implement local differential privacy (LDP) at both client and server levels, ensuring stronger privacy guarantees while maintaining computational efficiency.

Beyond DP, federated learning remains susceptible to membership inference attacks (MIA) [130, 131] and adversarial model inversion [132], where attackers infer whether specific data points were used for training. These threats necessitate additional defense mechanisms, such as randomized response techniques and secure aggregation protocols, which obfuscate individual contributions while preserving collaborative learning benefits. Ensuring privacy in FL requires a careful balance between model performance, security overhead, and computational efficiency, making it an ongoing research challenge.

5.2 Secure Model Execution and Adversarial Defenses

While privacy-preserving techniques safeguard user data, onboard learning also demands secure execution environments to protect models from adversarial threats and unauthorized access. Hardware-based security solutions, such as Trusted Execution Environments (TEEs), provide isolation for sensitive computations, preventing attackers from extracting model parameters or manipulating inference processes. TEEs such as Intel SGX, ARM TrustZone, and AMD SEV enable encrypted execution, mitigating adversarial model extraction risks [148–151]. Frameworks like TEE-ML leverage

Table 3: Summary of security and privacy approaches for Onboard AI

	Domain	Reference	Key Highlights
Challenges	Threats in onboard AI Computational constraints	[125] [126]	Adversarial attacks and data leaks. Limited resources hinder strong privacy mechanisms.
Differential Privacy	Privacy risks in FL DP overhead Adaptive DP Techniques	[127, 128] [129] [130, 131]	Exposes sensitive info via model updates. Degrades accuracy and increases computation load. Tunes noise dynamically for efficiency.
Secured Hardware	Security Architecture Privacy-Preserving Inference	[133, 134] [135–139]	Layer-wise computations and encrypted model execution. TEE-secured environments and external accelerators.
Trustworthy Onboard AI	Efficient XAI Compression for XAI Security-Aware Optimization	[140, 141] [142, 143] [144–147]	Lightweight XAI methods make onboard learing interpretable. Entropy-based pruning efficiency. Minimal computational overhead for reliability.

these environments to perform encrypted model inference with minimal performance trade-offs [152].

Despite their benefits, hardware accelerators introduce vulnerabilities that adversaries can exploit through side-channel attacks (SCAs). Power analysis and cache timing attacks allow attackers to infer neural network parameters by analyzing execution traces [133]. Research has demonstrated that power analysis-based model inversion can reconstruct deployed models by observing FPGA power consumption patterns [134]. Similarly, cache-based timing attacks expose layer-wise computations during DNN inference, enabling attackers to reconstruct intermediate activations and infer sensitive inputs [153, 154].

To counter these risks, secure model execution strategies involve randomized memory access patterns, encrypted computation pipelines, and lightweight obfuscation techniques for edge devices [155, 156]. Additionally, hardware-aware adversarial defenses, such as quantization-based perturbation masking and dynamic model encryption, can further harden onboard learning against inference-time attacks.

5.3 Privacy-Preserving Inference

Beyond securing model execution, onboard learning must also protect inference data from potential breaches, particularly in sensitive applications such as healthcare and finance. Privacy-preserving inference techniques allow deep learning models to process encrypted data without exposing raw inputs, ensuring confidentiality while maintaining utility. Cryptographic methods such as Fully Homomorphic Encryption (FHE) and Secure Multi-Party Computation (SMPC) provide robust security guarantees by enabling computations on encrypted inputs [135–137]. However, their excessive computational cost limits practical deployment on resource-constrained edge devices.

To address these challenges, hybrid privacy-preserving inference frameworks such as ShadowNet and MirrorNet have been developed. ShadowNet leverages TEE-secured computation while offloading non-sensitive tasks to untrusted hardware accelerators, optimizing performance while maintaining model confidentiality [138]. Similarly, MirrorNet partitions neural network layers between TEE-protected environments and external accelerators, minimizing latency while ensuring privacy-preserving inference [139]. By combining cryptographic techniques with hardware-enforced security,

these frameworks enable real-time secure inference without excessive computational overhead.

5.4 Ensuring Trust and Transparency: Explainable AI

As deep learning models increasingly operate in decentralized, autonomous edge environments, ensuring transparency and trust becomes essential. However, the black-box nature of DNNs makes it difficult to interpret decision-making processes, raising concerns about fairness, accountability, and adversarial robustness [125, 140]. Unlike centralized AI systems, which leverage large-scale computational resources for interpretability, onboard learning must incorporate lightweight explainability mechanisms to ensure real-time efficiency while maintaining transparency.

Traditional Explainable AI (XAI) methods, such as SHAP and LIME, introduce significant computational overhead, making them unsuitable for onboard deployment [157]. However, recent advancements in approximate Shapley values and model-agnostic interpretability techniques enable low-cost XAI integration in edge learning systems [143]. Additionally, Federated Explainable AI (FED-XAI) extends federated learning by incorporating interpretability constraints directly into decentralized training, allowing local models to produce human-interpretable decisions while preserving data privacy [141, 158].

Beyond model interpretability, XAI plays a crucial role in adversarial robustness. By embedding explainability into pruning strategies, models can dynamically remove redundant or high-risk parameters, improving both efficiency and resilience against adversarial attacks [146, 147]. Security-aware pruning ensures that only the most informative features contribute to decision-making, reducing susceptibility to model drift and adversarial exploitation [144, 145]. Furthermore, explainability-driven adversarial detection mechanisms allow onboard AI to identify and respond to attacks in real-time, strengthening trust in edge-based AI applications.

In summary, integrating explainability into onboard AI security strategies bridges the gap between efficiency, scalability, and trustworthiness. Lightweight interpretability techniques, federated XAI, and adversarial-aware model pruning collectively enable real-time, privacy-preserving, and transparent AI systems. These advancements enhance the security and reliability of onboard learning, making it a viable solution for mission-critical edge applications.

6 Advanced Topics

Onboard learning continues to evolve, requiring advanced techniques to balance efficiency, adaptability, and robustness in resource-constrained environments. This section explores three critical areas: (1) bridging model compression with continual learning to improve adaptability while preserving efficiency, (2) enhancing scalability and standardization to enable seamless deployment across heterogeneous edge platforms, and (3) leveraging hardware-software co-design to optimize AI for next-generation edge devices.

6.1 Bridging Model Compression with Continual Learning

Continual Learning (CL) enables AI models to incrementally learn new tasks without catastrophic forgetting, making it crucial for long-term deployment in dynamic environments. However, CL is constrained by high memory requirements, computational overhead, and knowledge retention issues—challenges that are especially critical for onboard AI. Model compression techniques such as pruning, knowledge distillation, and quantization offer solutions by optimizing memory usage and inference speed while preserving task knowledge.

Pruning for Continual Learning. Integrating pruning into CL helps onboard models adapt efficiently while maintaining a compact structure. Sparse Continual Learning (SparCL) [159] employs weight sparsity [160], gradient sparsity [161], and dynamic masking to selectively retain critical weights and prune redundant ones, minimizing memory footprint. Task-aware dynamic masking (TDM) further enhances retention by ensuring that essential parameters are preserved. Alternative methods such as gradient pruning [162] and continual prune-and-select (CP&S) [163] further improve memory allocation by structuring models into reusable subnetworks optimized for incremental learning.

Knowledge Distillation for Continual Learning. Traditional experience replay is often infeasible for onboard AI due to memory constraints and privacy concerns. Knowledge Distillation (KD) mitigates catastrophic forgetting by transferring past knowledge to new models without requiring explicit data storage [164, 165]. Learning without Forgetting (LwF) [164] preserves past knowledge by constraining parameter updates, while deep model consolidation [165] further refines inter-task representations. Prototype-sample relation distillation [166] and batch-level distillation [167] improve adaptation in non-stationary environments by optimizing feature transfer between tasks.

Quantization for Continual Learning. Quantization reduces model size while preserving past knowledge, ensuring efficient adaptation in resource-limited environments. Adaptive quantization modules (AQM) [168] dynamically adjust compression rates based on data complexity, while Bit-Level Information Preserving (BLIP) [169] applies weight quantization to maintain knowledge retention. Vector Quantization Prompting (VQ-Prompt) [170] enhances continual learning by improving task abstraction and reducing memory footprint.

By integrating pruning, knowledge distillation, and quantization, continual learning models can efficiently adapt to new tasks while operating within strict resource constraints, enhancing onboard AI scalability and robustness.

6.2 Scalability and Standardization in Onboard AI

To ensure widespread deployment, onboard learning must efficiently scale across various hardware configurations while maintaining performance consistency and inter-operability. Scalability ensures AI models can operate across different edge platforms while standardization fosters cross-platform compatibility.

Scalability through Model Optimization. Onboard AI must adapt to resource-limited environments without sacrificing performance. Model compression techniques

(discussed in Section 2) reduce inference complexity, enabling deployment across heterogeneous edge devices [171, 172]. Additionally, Hardware-Aware Neural Architecture Search (HW-NAS) automates model design for specific hardware constraints, ensuring optimized performance across various platforms [11, 12].

Standardization for Onboard AI Deployment. Standardized AI frameworks improve interoperability, portability, and deployment efficiency. ONNX (Open Neural Network Exchange) allows models trained in one framework (e.g., PyTorch) to be executed on various hardware backends [173]. Multi-Level Intermediate Representation (MLIR) facilitates the efficient compilation of AI models for different edge architectures, ensuring cross-platform consistency [174, 175]. Benchmarking tools such as MLPerf Tiny provide standardized evaluations for assessing model performance on constrained devices [176, 177].

By integrating scalability-driven model optimization and standardized AI frameworks, onboard learning can achieve efficient, flexible, and interoperable deployment across diverse edge environments.

6.3 Hardware Co-Design for Next-Generation onboard learning

Meeting the stringent performance and energy efficiency requirements of onboard AI requires a hardware-software co-design approach. Traditional hardware architectures struggle with the computational intensity of deep learning, necessitating custom AI accelerators, and co-optimized learning algorithms.

Application-Specific Hardware for Onboard AI. Application-Specific Integrated Circuits (ASICs) and System-on-Chip (SoC) solutions optimize AI inference for resource-constrained environments [178]. Unlike general-purpose processors, these specialized architectures enhance power efficiency and throughput.

Holistic Model-Hardware Co-Design. Efficient onboard AI requires co-optimizing model topology, compression techniques, and hardware configurations. Topology-aware NAS refines network structures to align with hardware constraints, while techniques such as quantization, pruning, and weight sharing optimize computational efficiency [179].

The general architecture of Hardware Co-Design for onboard learning (Figure 5) integrates model-level, hardware-level, and software-level optimizations into a unified framework. At the model level, Neural Architecture Search (NAS) optimizes network topology, conditional computing, and training to align with hardware constraints. The hardware level focuses on customized CPU/NPU acceleration, enhancing efficiency by optimizing platform composition and memory access. Meanwhile, the software level streamlines onboard DNN deployment through optimized code generation, kernel libraries, and memory management. By integrating hardware-aware AI model design, onboard learning can achieve scalable, high-performance deployment across next-generation edge devices.

7 Conclusion

This survey systematically examined methodologies for enabling efficient onboard learning across four key aspects: model compression, which reduces model size while

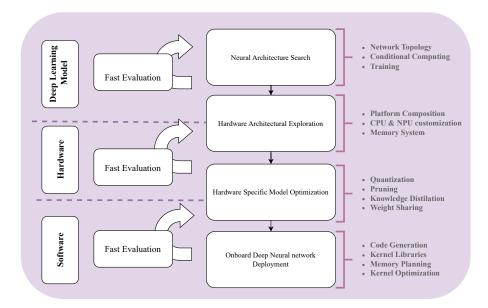


Fig. 5: General architecture of Hardware Co-Design for onboard learning

preserving accuracy; efficient inference, which minimizes latency and energy consumption; decentralized learning, which enables collaborative model updates without centralized dependencies; and security and privacy, which safeguards onboard AI from adversarial threats and data breaches. Despite advancements, significant challenges remain. The trade-off between compression and continual learning requires new strategies that optimize model adaptation without performance degradation. Balancing inference efficiency with secure learning remains an open problem, as privacy-preserving mechanisms often introduce latency overheads. Additionally, ensuring scalability and standardization across heterogeneous hardware platforms necessitates continued research into co-design strategies that integrate AI models with specialized edge accelerators. Moving forward, optimized onboard learning must evolve towards adaptive, energy-efficient, and privacy-preserving AI solutions that can operate autonomously in dynamic environments. By addressing these challenges, future research can drive innovations in autonomous systems, IoT, healthcare, and beyond, paving the way for the next generation of scalable and robust onboard learning.

Acknowledgements. This work is fully supported by the Australian Government SmartSat CRC (Cooperative Research Centre) Program, under the SCARLET- α lab: SpaceCraft Autonomy and Onboard AI for Next Generation Space Systems project (P2.52).

References

- [1] Dhar, S., Guo, J., Liu, J., Tripathi, S., Kurup, U., Shah, M.: A survey of ondevice machine learning: An algorithms and learning theory perspective. ACM Transactions on Internet of Things 2(3), 1–49 (2021)
- [2] Bourechak, A., Zedadra, O., Kouahla, M.N., Guerrieri, A., Seridi, H., Fortino, G.: At the confluence of artificial intelligence and edge computing in iot-based applications: A review and new perspectives. Sensors 23(3), 1639 (2023)
- [3] Zhu, X., Ma, F., Ding, F., Guo, Z., Yang, J., Yu, K.: A low-latency edge computation offloading scheme for trust evaluation in finance-level artificial intelligence of things. IEEE Internet of Things Journal (2023)
- [4] Cai, H., Gan, C., Zhu, L., Han, S.: Tinytl: Reduce memory, not parameters for efficient on-device learning. Advances in Neural Information Processing Systems 33, 11285–11297 (2020)
- [5] Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., Peste, A.: Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. Journal of Machine Learning Research 22(241), 1–124 (2021)
- [6] Imteaj, A., Thakker, U., Wang, S., Li, J., Amini, M.H.: A survey on federated learning for resource-constrained iot devices. IEEE Internet of Things Journal 9(1), 1–24 (2021)
- [7] Wang, X., Han, Y., Leung, V.C., Niyato, D., Yan, X., Chen, X.: Convergence of edge computing and deep learning: A comprehensive survey. IEEE Communications Surveys & Tutorials 22(2), 869–904 (2020)
- [8] Deng, S., Zhao, H., Fang, W., Yin, J., Dustdar, S., Zomaya, A.Y.: Edge intelligence: The confluence of edge computing and artificial intelligence. IEEE Internet of Things Journal 7(8), 7457–7469 (2020)
- [9] Lin, J., Zhu, L., Chen, W.-M., Wang, W.-C., Gan, C., Han, S.: On-device training under 256kb memory. Advances in Neural Information Processing Systems 35, 22941–22954 (2022)
- [10] Zhu, S., Voigt, T., Rahimian, F., Ko, J.: On-device training: A first overview on existing systems. ACM Transactions on Sensor Networks **20**(6), 1–39 (2024)
- [11] Luo, X., Liu, D., Kong, H., Huai, S., Chen, H., Liu, W.: Lightnas: On lightweight and scalable neural architecture search for embedded platforms. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 42(6), 1784–1797 (2022)
- [12] Chitty-Venkata, K.T., Somani, A.K.: Neural architecture search survey: A

- hardware perspective. ACM Computing Surveys 55(4), 1–36 (2022)
- [13] Cheng, H., Zhang, M., Shi, J.Q.: A survey on deep neural network pruning: Taxonomy, comparison, analysis, and recommendations. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
- [14] Li, G., Ma, X., Wang, X., Yue, H., Li, J., Liu, L., Feng, X., Xue, J.: Optimizing deep neural networks on intelligent edge accelerators via flexible-rate filter pruning. Journal of Systems Architecture 124, 102431 (2022)
- [15] Liu, D., Kong, H., Luo, X., Liu, W., Subramaniam, R.: Bringing ai to edge: From deep learning's perspective. Neurocomputing 485, 297–320 (2022)
- [16] Xia, M., Zhong, Z., Chen, D.: Structured pruning learns compact and accurate models. arXiv preprint arXiv:2204.00408 (2022)
- [17] Yu, F., Cui, L., Wang, P., Han, C., Huang, R., Huang, X.: Easiedge: A novel global deep neural networks pruning method for efficient edge computing. IEEE Internet of Things Journal 8(3), 1259–1271 (2020)
- [18] Ro, Y., Choi, J.Y.: Autolr: Layer-wise pruning and auto-tuning of learning rates in fine-tuning of deep networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 2486–2494 (2021)
- [19] Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. Advances in neural information processing systems **33**, 6377–6389 (2020)
- [20] Frantar, E., Alistarh, D.: Sparsegpt: Massive language models can be accurately pruned in one-shot. In: International Conference on Machine Learning, pp. 10323–10337 (2023). PMLR
- [21] Kohama, H., Minoura, H., Hirakawa, T., Yamashita, T., Fujiyoshi, H.: Single-shot pruning for pre-trained models: Rethinking the importance of magnitude pruning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 1433–1442 (2023)
- [22] Chang, J., Lu, Y., Xue, P., Xu, Y., Wei, Z.: Iterative clustering pruning for convolutional neural networks. Knowledge-Based Systems 265, 110386 (2023)
- [23] Jiang, Z., Xu, Y., Xu, H., Wang, Z., Liu, J., Chen, Q., Qiao, C.: Computation and communication efficient federated learning with adaptive model pruning. IEEE Transactions on Mobile Computing 23(3), 2003–2021 (2023)
- [24] Guo, X., Wang, W.-S., Zhang, J., Gong, L.-S.: An online growing-and-pruning algorithm of a feedforward neural network for nonlinear systems modeling. IEEE Transactions on Automation Science and Engineering (2024)

- [25] Jelčicová, Z., Verhelst, M.: Delta keyword transformer: Bringing transformers to the edge through dynamically pruned multi-head self-attention. arXiv preprint arXiv:2204.03479 (2022)
- [26] Jeon, Y., Lee, C., Cho, E., Ro, Y.: Mr. biq: Post-training non-uniform quantization based on minimizing the reconstruction error. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12329–12338 (2022)
- [27] Kim, J., Park, K., Lee, C., Kim, H.-y., Kim, J., Jeon, Y.: Towards next-level post-training quantization of hyper-scale transformers. arXiv preprint arXiv:2402.08958 (2024)
- [28] Rasch, M.J., Mackin, C., Le Gallo, M., Chen, A., Fasoli, A., Odermatt, F., Li, N., Nandakumar, S., Narayanan, P., Tsai, H., et al.: Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators. Nature communications 14(1), 5282 (2023)
- [29] Shen, M., Liang, F., Gong, R., Li, Y., Li, C., Lin, C., Yu, F., Yan, J., Ouyang, W.: Once quantization-aware training: High performance extremely low-bit architecture search. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5340–5349 (2021)
- [30] Koryakovskiy, I., Yakovleva, A., Buchnev, V., Isaev, T., Odinokikh, G.: One-shot model for mixed-precision quantization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7939–7949 (2023)
- [31] Dong, Z., Yao, Z., Gholami, A., Mahoney, M.W., Keutzer, K.: Hawq: Hessian aware quantization of neural networks with mixed-precision. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 293–302 (2019)
- [32] Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. International Journal of Computer Vision 129(6), 1789–1819 (2021)
- [33] Wang, L., Yoon, K.-J.: Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. IEEE transactions on pattern analysis and machine intelligence 44(6), 3048–3068 (2021)
- [34] Sepahvand, M., Abdali-Mohammadi, F., Taherkordi, A.: Teacher-student knowledge distillation based on decomposed deep feature representation for intelligent mobile applications. Expert Systems with Applications 202, 117474 (2022)
- [35] Crowley, E.J., Gray, G., Storkey, A.J.: Moonshine: Distilling with cheap convolutions. Advances in Neural Information Processing Systems **31** (2018)

- [36] Xiao, H., Fu, L., Shang, C., Bao, X., Xu, X.: A knowledge distillation compression algorithm for ship speed and energy coordinated optimal scheduling model based on deep reinforcement learning. IEEE Transactions on Transportation Electrification (2024)
- [37] Itahara, S., Nishio, T., Koda, Y., Morikura, M., Yamamoto, K.: Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. IEEE Transactions on Mobile Computing 22(1), 191–205 (2021)
- [38] Qu, X., Wang, J., Xiao, J.: Quantization and knowledge distillation for efficient federated learning on edge devices. In: 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 967–972 (2020). IEEE
- [39] Luo, H., Chen, T., Li, X., Li, S., Zhang, C., Zhao, G., Liu, X.: Keepedge: A knowledge distillation empowered edge intelligence framework for visual assisted positioning in uav delivery. IEEE Transactions on Mobile Computing 22(8), 4729–4741 (2022)
- [40] Qi, P., Zhou, X., Ding, Y., Zhang, Z., Zheng, S., Li, Z.: Fedbkd: Heterogenous federated learning via bidirectional knowledge distillation for modulation classification in iot-edge system. IEEE Journal of Selected Topics in Signal Processing 17(1), 189–204 (2022)
- [41] Lyu, B., Yuan, H., Lu, L., Zhang, Y.: Resource-constrained neural architecture search on edge devices. IEEE Transactions on Network Science and Engineering 9(1), 134–142 (2021)
- [42] Lee, H., Lee, S., Chong, S., Hwang, S.J.: Hardware-adaptive efficient latency prediction for nas via meta-learning. Advances in Neural Information Processing Systems 34, 27016–27028 (2021)
- [43] Zhou, A., Yang, J., Qi, Y., Qiao, T., Shi, Y., Duan, C., Zhao, W., Hu, C.: Hgnas: Hardware-aware graph neural architecture search for edge devices. IEEE Transactions on Computers (2024)
- [44] Liu, S., Zhang, H., Jin, Y.: A survey on computationally efficient neural architecture search. Journal of Automation and Intelligence 1(1), 100002 (2022)
- [45] Liu, H., Simonyan, K., Yang, Y.: Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
- [46] Elsken, T., Staffler, B., Metzen, J.H., Hutter, F.: Meta-learning of neural architectures for few-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12365–12375 (2020)

- [47] Kim, B., Lee, S.: On-nas: On-device neural architecture search on memory-constrained intelligent embedded systems. In: Proceedings of the 21st ACM Conference on Embedded Networked Sensor Systems, pp. 152–166 (2023)
- [48] Eccles, B.J., Rodgers, P., Kilpatrick, P., Spence, I., Varghese, B.: Dnnshifter: An efficient dnn pruning system for edge computing. Future Generation Computer Systems **152**, 43–54 (2024)
- [49] Hayou, S., Ton, J.-F., Doucet, A., Teh, Y.W.: Robust pruning at initialization. arXiv preprint arXiv:2002.08797 (2020)
- [50] Jiang, Y., Wang, S., Valls, V., Ko, B.J., Lee, W.-H., Leung, K.K., Tassiulas, L.: Model pruning enables efficient federated learning on edge devices. IEEE Transactions on Neural Networks and Learning Systems 34(12), 10374–10386 (2022)
- [51] Zhu, Z., Shi, Y., Luo, J., Wang, F., Peng, C., Fan, P., Letaief, K.B.: Fedlp: Layer-wise pruning mechanism for communication-computation efficient federated learning. In: ICC 2023-IEEE International Conference on Communications, pp. 1250–1255 (2023). IEEE
- [52] Li, H., Yue, X., Wang, Z., Chai, Z., Wang, W., Tomiyama, H., Meng, L.: Optimizing the deep neural networks by layer-wise refined pruning and the acceleration on fpga. Computational Intelligence and Neuroscience 2022(1), 8039281 (2022)
- [53] Liu, M., Fang, W., Ma, X., Xu, W., Xiong, N., Ding, Y.: Channel pruning guided by spatial and channel attention for dnns in intelligent edge computing. Applied Soft Computing 110, 107636 (2021)
- [54] Wang, Y., Qin, Y., Deng, D., Wei, J., Chen, T., Lin, X., Liu, L., Wei, S., Yin, S.: Trainer: An energy-efficient edge-device training processor supporting dynamic weight pruning. IEEE Journal of Solid-State Circuits 57(10), 3164–3178 (2022)
- [55] Luo, J.-H., Wu, J.: Neural network pruning with residual-connections and limited-data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1458–1467 (2020)
- [56] Xu, K., Wang, Z., Geng, X., Wu, M., Li, X., Lin, W.: Efficient joint optimization of layer-adaptive weight pruning in deep neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 17447–17457 (2023)
- [57] Hong, C., Kim, H., Baik, S., Oh, J., Lee, K.M.: Daq: Channel-wise distribution-aware quantization for deep image super-resolution networks. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2675–2684 (2022)

- [58] Motetti, B.A., Risso, M., Burrello, A., Macii, E., Poncino, M., Pagliari, D.J.: Joint pruning and channel-wise mixed-precision quantization for efficient deep neural networks. IEEE Transactions on Computers (2024)
- [59] Zhu, Y., Peng, H., Fu, A., Yang, W., Ma, H., Al-Sarawi, S.F., Abbott, D., Gao, Y.: Towards robustness evaluation of backdoor defense on quantized deep learning models. Expert Systems with Applications 255, 124599 (2024)
- [60] Zhang, Q., Zhang, M., Wang, M., Sui, W., Meng, C., Yang, J., Kong, W., Cui, X., Lin, W.: Efficient deep learning inference based on model compression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 1695–1702 (2018)
- [61] Fan, A., Stock, P., Graham, B., Grave, E., Gribonval, R., Jegou, H., Joulin, A.: Training with quantization noise for extreme model compression. arXiv preprint arXiv:2004.07320 (2020)
- [62] Wang, Z., Luo, T., Goh, R.S.M., Zhou, J.T.: Edcompress: Energy-aware model compression for dataflows. IEEE Transactions on Neural Networks and Learning Systems 35(1), 208–220 (2022)
- [63] Rokh, B., Azarpeyvand, A., Khanteymoori, A.: A comprehensive survey on model quantization for deep neural networks in image classification. ACM Transactions on Intelligent Systems and Technology 14(6), 1–50 (2023)
- [64] Nagel, M., Fournarakis, M., Bondarenko, Y., Blankevoort, T.: Overcoming oscillations in quantization-aware training. In: International Conference on Machine Learning, pp. 16318–16330 (2022). PMLR
- [65] Huang, T., You, S., Wang, F., Qian, C., Xu, C.: Knowledge distillation from a stronger teacher. Advances in Neural Information Processing Systems 35, 33716– 33727 (2022)
- [66] Yang, C., Zhu, Y., Lu, W., Wang, Y., Chen, Q., Gao, C., Yan, B., Chen, Y.: Survey on knowledge distillation for large language models: Methods, evaluation, and application. ACM Transactions on Intelligent Systems and Technology (2024)
- [67] Zhu, Y., Wang, Y.: Student customized knowledge distillation: Bridging the gap between student and teacher. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5057–5066 (2021)
- [68] Park, D.Y., Cha, M.-H., Kim, D., Han, B., et al.: Learning student-friendly teacher networks for knowledge distillation. Advances in neural information processing systems 34, 13292–13303 (2021)
- [69] Jang, I., Kim, H., Lee, D., Son, Y.-S., Kim, S.: Knowledge transfer for on-device

- deep reinforcement learning in resource constrained edge computing systems. IEEE Access 8, 146588–146597 (2020)
- [70] Fang, W., Xue, F., Ding, Y., Xiong, N., Leung, V.C.: Edgeke: an on-demand deep learning iot system for cognitive big data on industrial edge devices. IEEE Transactions on Industrial Informatics 17(9), 6144–6152 (2020)
- [71] Mishra, R., Gupta, H.P.: Designing and training of lightweight neural networks on edge devices using early halting in knowledge distillation. IEEE Transactions on Mobile Computing (2023)
- [72] Blakeney, C., Li, X., Yan, Y., Zong, Z.: Parallel blockwise knowledge distillation for deep neural network compression. IEEE Transactions on Parallel and Distributed Systems 32(7), 1765–1776 (2020)
- [73] Ye, Z., Gao, W., Hu, Q., Sun, P., Wang, X., Luo, Y., Zhang, T., Wen, Y.: Deep learning workload scheduling in gpu datacenters: A survey. ACM Computing Surveys **56**(6), 1–38 (2024)
- [74] Duan, Y., Wu, J.: Computation offloading scheduling for deep neural network inference in mobile computing. In: 2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), pp. 1–10 (2021). IEEE
- [75] Jeong, H.-J., Jeong, I., Lee, H.-J., Moon, S.-M.: Computation offloading for machine learning web apps in the edge server environment. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp. 1492–1499 (2018). IEEE
- [76] Huda, S.A., Moh, S.: Deep reinforcement learning-based computation offloading in uav swarm-enabled edge computing for surveillance applications. IEEE Access (2023)
- [77] Kim, B., Min, H., Heo, J., Jung, J.: Dynamic computation offloading scheme for drone-based surveillance systems. Sensors 18(9), 2982 (2018)
- [78] Xu, X., Wu, Q., Qi, L., Dou, W., Tsai, S.-B., Bhuiyan, M.Z.A.: Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles. IEEE Transactions on Intelligent Transportation Systems 22(3), 1787– 1796 (2020)
- [79] Shakarami, A., Shahidinejad, A., Ghobaei-Arani, M.: An autonomous computation offloading strategy in mobile edge computing: A deep learning-based hybrid approach. Journal of Network and Computer Applications 178, 102974 (2021)
- [80] Abbas, Z.H., Ali, Z., Abbas, G., Jiao, L., Bilal, M., Suh, D.-Y., Piran, M.J.: Computational offloading in mobile edge with comprehensive and energy efficient cost function: a deep learning approach. Sensors **21**(10), 3523 (2021)

- [81] Dey, S., Mondal, J., Mukherjee, A.: Offloaded execution of deep learning inference at edge: Challenges and insights. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pp. 855–861 (2019). IEEE
- [82] Zhou, H., Li, M., Wang, N., Min, G., Wu, J.: Accelerating deep learning inference via model parallelism and partial computation offloading. IEEE Transactions on Parallel and Distributed Systems **34**(2), 475–488 (2022)
- [83] Teerapittayanon, S., McDanel, B., Kung, H.-T.: Distributed deep neural networks over the cloud, the edge and end devices. In: 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 328–339 (2017). IEEE
- [84] Nikoloska, I., Zlatanov, N.: Data selection scheme for energy efficient supervised learning at iot nodes. IEEE Communications Letters **25**(3), 859–863 (2020)
- [85] Fresa, A., Champati, J.P.: Offloading algorithms for maximizing inference accuracy on edge device under a time constraint. arXiv preprint arXiv:2112.11413 (2021)
- [86] Yao, S., Li, J., Liu, D., Wang, T., Liu, S., Shao, H., Abdelzaher, T.: Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency. In: Proceedings of the 18th Conference on Embedded Networked Sensor Systems, pp. 476–488 (2020)
- [87] Gao, D., He, X., Zhou, Z., Tong, Y., Xu, K., Thiele, L.: Rethinking pruning for accelerating deep inference at the edge. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 155–164 (2020)
- [88] Li, E., Zeng, L., Zhou, Z., Chen, X.: Edge ai: On-demand accelerating deep neural network inference via edge computing. IEEE Transactions on Wireless Communications 19(1), 447–457 (2019)
- [89] Xue, F., Fang, W., Xu, W., Wang, Q., Ma, X., Ding, Y.: Edgeld: Locally distributed deep learning inference on edge device clusters. In: 2020 IEEE 22nd International Conference on High Performance Computing and Communications; IEEE 18th International Conference on Smart City; IEEE 6th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), pp. 613–619 (2020). IEEE
- [90] Feltin, T., Marchó, L., Cordero-Fuertes, J.-A., Brockners, F., Clausen, T.H.: Dnn partitioning for inference throughput acceleration at the edge. IEEE Access 11, 52236–52249 (2023)
- [91] Chow, M., Jahanshahi, A., Wong, D.: Krisp: Enabling kernel-wise right-sizing

- for spatial partitioned gpu inference servers. In: 2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 624–637 (2023). IEEE
- [92] Liang, H., Sang, Q., Hu, C., Cheng, D., Zhou, X., Wang, D., Bao, W., Wang, Y.: Dnn surgery: Accelerating dnn inference on the edge through layer partitioning. IEEE transactions on Cloud Computing 11(3), 3111–3125 (2023)
- [93] Chen, Y., Luo, T., Fang, W., Xiong, N.N.: Edgeci: Distributed workload assignment and model partitioning for cnn inference on edge clusters. ACM Transactions on Internet Technology **24**(2), 1–24 (2024)
- [94] Lin, S., Zhou, Z., Zhang, Z., Chen, X., Zhang, J.: On-demand accelerating deep neural network inference via edge computing. In: Edge Intelligence in the Making: Optimization, Deep Learning, and Applications, pp. 151–168. Springer, ??? (2021)
- [95] Zeng, L., Chen, X., Zhou, Z., Yang, L., Zhang, J.: Coedge: Cooperative dnn inference with adaptive workload partitioning over heterogeneous edge devices. IEEE/ACM Transactions on Networking 29(2), 595–608 (2020)
- [96] Laskaridis, S., Kouris, A., Lane, N.D.: Adaptive inference through early-exit networks: Design, challenges and directions. In: Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning, pp. 1–6 (2021)
- [97] Teerapittayanon, S., McDanel, B., Kung, H.-T.: Branchynet: Fast inference via early exiting from deep neural networks. In: 2016 23rd International Conference on Pattern Recognition (ICPR), pp. 2464–2469 (2016). IEEE
- [98] Passalis, N., Raitoharju, J., Tefas, A., Gabbouj, M.: Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. Pattern Recognition 105, 107346 (2020)
- [99] Li, X., Lou, C., Chen, Y., Zhu, Z., Shen, Y., Ma, Y., Zou, A.: Predictive exit: Prediction of fine-grained early exits for computation-and energy-efficient inference. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 8657–8665 (2023)
- [100] Dong, R., Mao, Y., Zhang, J.: Resource-constrained edge ai with early exit prediction. Journal of Communications and Information Networks **7**(2), 122–134 (2022)
- [101] Pacheco, R.G., Bochie, K., Gilbert, M.S., Couto, R.S., Campista, M.E.M.: Towards edge computing using early-exit convolutional neural networks. Information 12(10), 431 (2021)
- [102] Jo, J., Kim, G., Kim, S., Park, J.: Locoexnet: Low-cost early exit network for

- energy efficient cnn accelerator design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **42**(12), 4909–4921 (2023)
- [103] She, Y., Shi, T., Wang, J., Liu, B.: Dynamic batching and early-exiting for accurate and timely edge inference. In: 2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring), pp. 1–6 (2024). IEEE
- [104] Wołczyk, M., Wójcik, B., Bałazy, K., Podolak, I.T., Tabor, J., Śmieja, M., Trzcinski, T.: Zero time waste: Recycling predictions in early exit neural networks. Advances in Neural Information Processing Systems 34, 2516–2528 (2021)
- [105] Douch, S., Abid, M.R., Zine-Dine, K., Bouzidi, D., Bourhnane, S., Benhaddou, D.: Early exists federated learning (eefl): Brining training to the edge. In: 2024 Sixth International Conference on Intelligent Computing in Data Sciences (ICDS), pp. 1–8 (2024). IEEE
- [106] Ilhan, F., Su, G., Liu, L.: Scalefl: Resource-adaptive federated learning with heterogeneous clients. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 24532–24541 (2023)
- [107] Zhong, Z., Bao, W., Wang, J., Zhu, X., Zhang, X.: Flee: A hierarchical federated learning framework for distributed deep neural network over cloud, edge, and end device. ACM Transactions on Intelligent Systems and Technology (TIST) 13(5), 1–24 (2022)
- [108] Lim, W.Y.B., Luong, N.C., Hoang, D.T., Jiao, Y., Liang, Y.-C., Yang, Q., Niyato, D., Miao, C.: Federated learning in mobile edge networks: A comprehensive survey. IEEE communications surveys & tutorials **22**(3), 2031–2063 (2020)
- [109] Ye, Y., Li, S., Liu, F., Tang, Y., Hu, W.: Edgefed: Optimized federated learning based on edge computing. IEEE Access 8, 209191–209198 (2020)
- [110] Su, Z., Wang, Y., Luan, T.H., Zhang, N., Li, F., Chen, T., Cao, H.: Secure and efficient federated learning for smart grid with edge-cloud collaboration. IEEE Transactions on Industrial Informatics 18(2), 1333–1344 (2021)
- [111] Wang, Z., Xu, H., Xu, Y., Jiang, Z., Liu, J.: Coopfl: Accelerating federated learning with dnn partitioning and offloading in heterogeneous edge computing. Computer Networks 220, 109490 (2023)
- [112] Kim, Y.G., Wu, C.-J.: Autofl: Enabling heterogeneity-aware energy efficient federated learning. In: MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 183–198 (2021)
- [113] Tao, Y., Cui, S., Xu, W., Yin, H., Yu, D., Liang, W., Cheng, X.: Byzantine-resilient federated learning at edge. IEEE Transactions on Computers 72(9),

- [114] Wang, S., Tuor, T., Salonidis, T., Leung, K.K., Makaya, C., He, T., Chan, K.: Adaptive federated learning in resource constrained edge computing systems. IEEE journal on selected areas in communications 37(6), 1205–1221 (2019)
- [115] Liao, Y., Xu, Y., Xu, H., Yao, Z., Wang, L., Qiao, C.: Accelerating federated learning with data and model parallelism in edge computing. IEEE/ACM Transactions on Networking (2023)
- [116] Wang, L., Zhang, X., Su, H., Zhu, J.: A comprehensive survey of continual learning: theory, method and application. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024)
- [117] Buzzega, P., Boschini, M., Porrello, A., Abati, D., Calderara, S.: Dark experience for general continual learning: a strong, simple baseline. Advances in neural information processing systems **33**, 15920–15930 (2020)
- [118] Aggarwal, S., Binici, K., Mitra, T.: Chameleon: Dual memory replay for online continual learning on edge devices. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2023)
- [119] Pellegrini, L., Graffieti, G., Lomonaco, V., Maltoni, D.: Latent replay for real-time continual learning. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 10203–10209 (2020). IEEE
- [120] Ravaglia, L., Rusci, M., Nadalini, D., Capotondi, A., Conti, F., Benini, L.: A tinyml platform for on-device continual learning with quantized latent replays. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 11(4), 789–802 (2021)
- [121] Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al.: Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In: International Conference on Machine Learning, pp. 1407–1416 (2018). PMLR
- [122] Espeholt, L., Marinier, R., Stanczyk, P., Wang, K., Michalski, M.: Seed rl: Scalable and efficient deep-rl with accelerated central inference. arXiv preprint arXiv:1910.06591 (2019)
- [123] Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., Madry, A.: Implementation matters in deep policy gradients: A case study on ppo and trpo. In: International Conference on Learning Representations (2020)
- [124] Jin, Y., Jiao, L., Qian, Z., Zhang, S., Lu, S.: Learning for learning: Predictive online control of federated learning with edge provisioning. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications, pp. 1–10 (2021). IEEE

- [125] Chen, J., Ran, X.: Deep learning with edge computing: A review. Proceedings of the IEEE 107(8), 1655–1674 (2019)
- [126] Gill, S.S., Golec, M., Hu, J., Xu, M., Du, J., Wu, H., Walia, G.K., Murugesan, S.S., Ali, B., Kumar, M., et al.: Edge ai: A taxonomy, systematic review and future directions. Cluster Computing 28(1), 1–53 (2025)
- [127] Wu, X., Zhang, Y., Shi, M., Li, P., Li, R., Xiong, N.N.: An adaptive federated learning scheme with differential privacy preserving. Future Generation Computer Systems 127, 362–372 (2022)
- [128] Chen, S., Yu, D., Zou, Y., Yu, J., Cheng, X.: Decentralized wireless federated learning with differential privacy. IEEE Transactions on Industrial Informatics 18(9), 6273–6282 (2022)
- [129] El Ouadrhiri, A., Abdelhadi, A.: Differential privacy for deep and federated learning: A survey. IEEE access 10, 22359–22380 (2022)
- [130] Wang, K., Hu, Z., Ai, Q., Liu, Q., Chen, M., Liu, K., Cong, Y.: Membership inference attack with multi-grade service models in edge intelligence. IEEE Network 35(1), 184–189 (2021)
- [131] Bai, L., Hu, H., Ye, Q., Li, H., Wang, L., Xu, J.: Membership inference attacks and defenses in federated learning: A survey. ACM Computing Surveys 57(4), 1–35 (2024)
- [132] Yuan, X., He, P., Zhu, Q., Li, X.: Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems **30**(9), 2805–2824 (2019)
- [133] Wang, H., Sayadi, H., Dinakarrao, S.M.P., Sasan, A., Rafatirad, S., Homayoun, H.: Enabling micro ai for securing edge devices at hardware level. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 11(4), 803–815 (2021)
- [134] Moini, S., Tian, S., Holcomb, D., Szefer, J., Tessier, R.: Power side-channel attacks on bnn accelerators in remote fpgas. IEEE Journal on Emerging and Selected Topics in Circuits and Systems 11(2), 357–370 (2021)
- [135] Gross, W.J., Meyer, B.H., Ardakani, A.: Hardware-aware design for edge intelligence. IEEE Open Journal of Circuits and Systems 2, 113–127 (2020)
- [136] Vedadi, E., Keshtkarjahromi, Y., Seferoglu, H.: Efficient coded multi-party computation at edge networks. IEEE Transactions on Information Forensics and Security (2023)
- [137] Dai, T., Duan, L., Jiang, Y., Li, Y., Mei, F., Sun, Y.: Force: Highly efficient four-party privacy-preserving machine learning on gpu. In: Nordic Conference

- on Secure IT Systems, pp. 330-349 (2023). Springer
- [138] Sun, Z., Sun, R., Liu, C., Chowdhury, A.R., Lu, L., Jha, S.: Shadownet: A secure and efficient on-device model inference system for convolutional neural networks. In: 2023 IEEE Symposium on Security and Privacy (SP), pp. 1596–1612 (2023). IEEE
- [139] Liu, Z., Luo, Y., Duan, S., Zhou, T., Xu, X.: Mirrornet: A tee-friendly framework for secure on-device dnn inference. In: 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 1–9 (2023). IEEE
- [140] Liang, Y., Li, S., Yan, C., Li, M., Jiang, C.: Explaining the black-box model: A survey of local interpretation methods for deep neural networks. Neurocomputing 419, 168–182 (2021)
- [141] Zhang, Y., Zeng, D., Luo, J., Fu, X., Chen, G., Xu, Z., King, I.: A survey of trust-worthy federated learning: Issues, solutions, and challenges. ACM Transactions on Intelligent Systems and Technology 15(6), 1–47 (2024)
- [142] Becking, D., Dreyer, M., Samek, W., Müller, K., Lapuschkin, S.: Ecq x: explainability-driven quantization for low-bit and sparse dnns. In: International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers, pp. 271–296 (2020). Springer
- [143] Boselli, R., D'Amico, S., Nobani, N.: explainable ai for word embeddings: A survey. Cognitive Computation 17(1), 1–24 (2025)
- [144] Mahfuz, T., Bhunia, S., Chakraborty, P.: X-dfs: Explainable artificial intelligence guided design-for-security solution space exploration. IEEE Transactions on Information Forensics and Security (2024)
- [145] Villar-Rodriguez, E., Pérez, M.A., Torre-Bastida, A.I., Senderos, C.R., López-de-Armentia, J.: Edge intelligence secure frameworks: Current state and future challenges. Computers & Security 130, 103278 (2023)
- [146] Cantini, R., Orsino, A., Talia, D.: Xai-driven knowledge distillation of large language models for efficient deployment on low-resource devices. Journal of Big Data 11(1), 63 (2024)
- [147] Yu, L., Xiang, W.: X-pruner: explainable pruning for vision transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 24355–24363 (2023)
- [148] Ohrimenko, O., Schuster, F., Fournet, C., Mehta, A., Nowozin, S., Vaswani, K., Costa, M.: Oblivious {Multi-Party} machine learning on trusted processors. In: 25th USENIX Security Symposium (USENIX Security 16), pp. 619–636 (2016)

- [149] Mofrad, S., Zhang, F., Lu, S., Shi, W.: A comparison study of intel sgx and amd memory encryption technology. In: Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy, pp. 1–8 (2018)
- [150] Manzoor, H.U., Shabbir, A., Chen, A., Flynn, D., Zoha, A.: A survey of security strategies in federated learning: Defending models, data, and privacy. Future Internet 16(10), 374 (2024)
- [151] Demigha, O., Larguet, R.: Hardware-based solutions for trusted cloud computing. Computers & Security 103, 102117 (2021)
- [152] Hesamifard, E., Takabi, H., Ghasemi, M., Wright, R.N.: Privacy-preserving machine learning as a service. Proceedings on Privacy Enhancing Technologies (2018)
- [153] Yan, M., Fletcher, C.W., Torrellas, J.: Cache telepathy: Leveraging shared resource attacks to learn {DNN} architectures. In: 29th USENIX Security Symposium (USENIX Security 20), pp. 2003–2020 (2020)
- [154] Yuan, J., Zhang, J., Qiu, P., Wei, X., Liu, D.: A survey of of side-channel attacks and mitigation for processor interconnects. Applied Sciences **14**(15), 6699 (2024)
- [155] Wang, H., Sayadi, H., Mohsenin, T., Zhao, L., Sasan, A., Rafatirad, S., Homayoun, H.: Mitigating cache-based side-channel attacks through randomization: A comprehensive system and architecture level analysis. In: 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1414–1419 (2020). IEEE
- [156] Baciu, V.-E., Braeken, A., Segers, L., Silva, B.d.: Secure tiny machine learning on edge devices: A lightweight dual attestation mechanism for machine learning. Future Internet 17(2), 85 (2025)
- [157] Shrotri, A.A., Narodytska, N., Ignatiev, A., Meel, K.S., Marques-Silva, J., Vardi, M.Y.: Constraint-driven explanations for black-box ml models. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 8304–8314 (2022)
- [158] Bechini, A., Daole, M., Ducange, P., Marcelloni, F., Renda, A.: An application for federated learning of xai models in edge computing environments. In: 2023 IEEE International Conference on Fuzzy Systems (FUZZ), pp. 1–7 (2023). IEEE
- [159] Wang, Z., Zhan, Z., Gong, Y., Yuan, G., Niu, W., Jian, T., Ren, B., Ioannidis, S., Wang, Y., Dy, J.: Sparcl: Sparse continual learning on the edge. Advances in Neural Information Processing Systems 35, 20366–20380 (2022)
- [160] Kurtz, M., Kopinsky, J., Gelashvili, R., Matveev, A., Carr, J., Goin, M., Leiserson, W., Moore, S., Shavit, N., Alistarh, D.: Inducing and exploiting activation

- sparsity for fast inference on deep neural networks. In: International Conference on Machine Learning, pp. 5533–5543 (2020). PMLR
- [161] Evci, U., Ioannou, Y., Keskin, C., Dauphin, Y.: Gradient flow in sparse neural networks and how lottery tickets win. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 6577–6586 (2022)
- [162] Hung, C.-Y., Tu, C.-H., Wu, C.-E., Chen, C.-H., Chan, Y.-M., Chen, C.-S.: Compacting, picking and growing for unforgetting continual learning. Advances in neural information processing systems **32** (2019)
- [163] Dekhovich, A., Tax, D.M., Sluiter, M.H., Bessa, M.A.: Continual pruneand-select: class-incremental learning with specialized subnetworks. Applied Intelligence 53(14), 17849–17864 (2023)
- [164] Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
- [165] Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., Kuo, C.-C.J.: Class-incremental learning via deep model consolidation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 1131–1140 (2020)
- [166] Asadi, N., Davari, M., Mudur, S., Aljundi, R., Belilovsky, E.: Prototype-sample relation distillation: towards replay-free continual learning. In: International Conference on Machine Learning, pp. 1093–1106 (2023). PMLR
- [167] Fini, E., Lathuiliere, S., Sangineto, E., Nabi, M., Ricci, E.: Online continual learning under extreme memory constraints. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16, pp. 720–735 (2020). Springer
- [168] Caccia, L., Belilovsky, E., Caccia, M., Pineau, J.: Online learned continual compression with adaptive quantization modules. In: International Conference on Machine Learning, pp. 1240–1250 (2020). PMLR
- [169] Shi, Y., Yuan, L., Chen, Y., Feng, J.: Continual learning via bit-level information preserving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16674–16683 (2021)
- [170] Jiao, L., Lai, Q., Li, Y., Xu, Q.: Vector quantization prompting for continual learning. arXiv preprint arXiv:2410.20444 (2024)
- [171] Mayer, R., Jacobsen, H.-A.: Scalable deep learning on distributed infrastructures: Challenges, techniques, and tools. ACM Computing Surveys (CSUR) 53(1), 1–37 (2020)

- [172] Tyagi, S., Swany, M.: Scadles: Scalable deep learning over streaming data at the edge. In: 2022 IEEE International Conference on Big Data (Big Data), pp. 2113–2122 (2022). IEEE
- [173] Zhou, M., Gao, X., Wu, J., Liu, K., Sun, H., Li, L.: Investigating white-box attacks for on-device models. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, pp. 1–12 (2024)
- [174] Lattner, C., Amini, M., Bondhugula, U., Cohen, A., Davis, A., Pienaar, J., Riddle, R., Shpeisman, T., Vasilache, N., Zinenko, O.: Mlir: Scaling compiler infrastructure for domain specific computation. In: 2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO), pp. 2–14 (2021). IEEE
- [175] Majumder, K., Bondhugula, U.: Hir: An mlir-based intermediate representation for hardware accelerator description. In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4, pp. 189–201 (2023)
- [176] Akkisetty, P.K.: An overview of ai platforms, frameworks, libraries, and processors. Model Optimization Methods for Efficient and Edge AI: Federated Learning Architectures, Frameworks and Applications, 43–55 (2025)
- [177] Tabanelli, E., Tagliavini, G., Benini, L.: Dnn is not all you need: Parallelizing non-neural ml algorithms on ultra-low-power iot processors. ACM Transactions on Embedded Computing Systems **22**(3), 1–33 (2023)
- [178] Lee, J., Kang, S., Lee, J., Shin, D., Han, D., Yoo, H.-J.: The hardware and algorithm co-design for energy-efficient dnn processor on edge/mobile devices. IEEE Transactions on Circuits and Systems I: Regular Papers **67**(10), 3458–3470 (2020)
- [179] Zhang, X., Li, Y., Pan, J., Chen, D.: Algorithm/accelerator co-design and cosearch for edge ai. IEEE Transactions on Circuits and Systems II: Express Briefs 69(7), 3064–3070 (2022)