Training Machine Learning Models at the Edge: A Survey

AYMEN RAYANE KHOUAS, Deakin University, AU
MOHAMED REDA BOUADJENEK, Deakin University, AU
HAKIM HACID, Technology Innovation Institute, UAE
SUNIL ARYAL, Deakin University, AU

Edge computing has gained significant traction in recent years, promising enhanced efficiency by integrating artificial intelligence capabilities at the edge. While the focus has primarily been on the deployment and inference of Machine Learning (ML) models at the edge, the training aspect remains less explored. This survey explores the concept of edge learning, specifically the optimization of ML model training at the edge. The objective is to comprehensively explore diverse approaches and methodologies in edge learning, synthesize existing knowledge, identify challenges, and highlight future trends. Utilizing Scopus and Web of Science advanced search, relevant literature on edge learning was identified, revealing a concentration of research efforts in distributed learning methods, particularly federated learning. This survey further provides a guideline for comparing techniques used to optimize ML for edge learning, along with an exploration of the different frameworks, libraries, and simulation tools available. In doing so, the paper contributes to a holistic understanding of the current landscape and future directions in the intersection of edge computing and machine learning, paving the way for informed comparisons between optimization methods and techniques designed for training on the edge.

CCS Concepts: • **Computing methodologies** → **Machine learning**; *Distributed artificial intelligence*; Distributed computing methodologies; • **General and reference** → **Surveys and overviews**; • **Computer systems organization** → *Embedded and cyber-physical systems.*

Additional Key Words and Phrases: Edge Learning; Edge Computing; Edge AI; On-Device Training; Edge intelligence; IoT.

ACM Reference Format:
Aymen Rayane Khouas, Mohamed Reda Bouadjenek, Hakim Hacid, and Sunil Aryal. 2026. Training Machine Learning Models at the Edge: A Survey. *ACM Trans. Intell. Syst. Technol.* 1, 1 (August 2026), 52 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent years, the fields of Artificial Intelligence (AI) and Machine Learning (ML) have witnessed significant growth, and have demonstrated remarkable success across various industrial applications [1]. ML's essence lies in the interplay between algorithmic models and large quantities of data, as the latter is often required to successfully train ML models. Traditionally, datasets have been collected in cloud storage, databases, and data lakes. These datasets are then processed in central cloud servers to train various ML models.

Authors' addresses: Aymen Rayane Khouas, a.khouas@research.deakin.edu.au, Deakin University, 75 Pigdons Road, Waurn Ponds, Geelong, Victoria, AU, 3216; Mohamed Reda Bouadjenek, reda.bouadjenek@deakin.edu.au, Deakin University, 75 Pigdons Road, Waurn Ponds, Geelong, Victoria, AU, 3216; Hakim Hacid, hakim.hacid@tii.ae, Technology Innovation Institute, Masdar City - SE45 01, Abu Dhabi, UAE; Sunil Aryal, sunil.aryal@deakin.edu.au, Deakin University, 75 Pigdons Road, Waurn Ponds, Geelong, Victoria, AU, 3216.

ACM 2157-6904/2026/8-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conversely, the rapid proliferation of smart devices and sensors in recent years has led to an explosion of data generation at the edge of the network. With edge devices generating vast quantities of data closer to the source, growing concerns about privacy and security, as well as the desire to optimize the bandwidth consumption on the increasing number of edge devices and reduce the computational load on cloud servers, have driven a paradigm shift towards edge computing. In this context, computational processes are decentralized and migrated to edge devices. This sets the stage for a novel intersection between ML and edge computing.

This shift has sparked growing interest in edge ML. A union between machine learning and edge computing, deploying ML models at the edge, closer to end devices, enabling inference or training to occur at the edge. Edge learning is a subset of Edge ML that involves training ML models directly at the edge. Traditionally, ML models have relied on cloud infrastructure for training and deployment. However, this approach poses several challenges. These include high latency, significant communication overheads, and concerns around data privacy and security. By processing data closer to its source, edge learning tackles these challenges while enabling real-time decision-making and reducing cloud resource usage. Furthermore, this enables innovative ML applications, such as privacy-aware recommendation systems and smart technologies. These applications span multiple industries, including healthcare, manufacturing, agriculture, and space exploration.

Training ML models at the edge poses unique challenges due to edge devices' limited computational power and memory. Moreover, despite the abundance of data at the edge, individual devices usually lack sufficient data to train ML models from scratch. To address these challenges, techniques like federated learning, knowledge distillation, and transfer learning have been proposed. These methods aim to optimize ML models to fit within the constraints of edge devices, rendering them suitable for training in resource-constrained environments, scenarios with low data availability, or through collaborative training across multiple edge devices that leverage their collective data.

This survey paper aims to provide an overview on edge learning, covering its methodologies, requirements, applications, challenges, and open research directions. We explore state-of-the-art techniques for training and optimizing ML models on edge devices, highlighting their advantages. Furthermore, we compare these approaches, providing a broad overview of their strengths and weaknesses. We also examine the various applications that benefit from edge learning, as well as the frameworks, libraries, and simulation tools that support and optimize it. Please note that a background in ML and deep learning is assumed for this survey, and readers without this knowledge may find it helpful to consult a general introduction to the field, such as the ones found in [2–5].

1.1 Comparison with existing surveys

There have been multiple surveys about Edge ML that attempt to define the field and present the different approaches that exist for AI in Edge. Most of these surveys focus on edge inference or on a single aspect of edge learning, such as federated learning [16, 17] or on-device training [12, 27].

As previously defined, this survey provides a comprehensive overview of training ML models on edge devices. This survey has multiple contributions that we will use to compare with the existing surveys. The contributions are presented in the following six points that we will label as "topics".

(1) **Explore techniques**: We examine various techniques for optimizing the training of ML models on edge devices.
(2) **Metrics for Edge Learning**: We define metrics to evaluate and compare edge learning approaches, and identify requirements for edge learning in real-world scenarios.
(3) **Compare techniques**: We compare the different edge learning techniques based on their performance, requirements, and popularity.

Table 1. Summary of Edge Machine Learning related surveys

Survey	Year	Explore techniques	Metrics for Edge Learning	Compare techniques	Explore Types of ML	Explore tools and libraries	Use-cases and applications
Chen et al. [6]	2019	•	✗	•	✗	•	•
Wang et al. [7]	2020	•	✗	•	✗	•	•
Shi et al. [8]	2020	•	✗	•	✗	✗	✗
Xu et al. [9]	2020	•	✗	•	✗	✗	✗
Lim et al. [10]	2020	○	✗	✗	✗	✗	✗
Leon Veas et al. [11]	2021	•	✗	✗	✗	✗	✗
Dhar et al. [12]	2021	○	✗	○	✗	✗	✗
Tak et al. [13]	2021	○	✗	✗	✗	✗	✗
Zhang et al. [14]	2021	•	✓	✗	✗	✓	✗
Murshed et al. [15]	2022	•	✗	✗	✗	•	•
Abreha et al. [16]	2022	○	✗	✗	✗	○	○
Boobalan et al. [17]	2022	○	✗	✗	✗	✗	○
Joshi et al. [18]	2022	✓	✓	✓	✗	✗	✗
Cai et al. [19]	2022	•	✗	•	✗	✗	✗
Cui et al. [20]	2022	○	✗	✗	✗	✗	○
Imteaj et al. [21]	2022	○	✗	○	✗	✗	○
Mendez et al. [22]	2022	✗	✗	✗	✗	•	✗
Filho et al. [23]	2022	•	✗	✗	✗	•	•
Ray et al. [24]	2022	✗	✗	✗	✗	•	✗
Li et al. [25]	2023	•	•	✓	✗	•	✗
Hua et al. [26]	2023	•	✗	✗	✗	✗	•
Zhu et al. [27]	2023	○	✓	○	✗	✗	✗
Wu et al. [28]	2023	○	✗	○	✗	✗	✗
Hoffpauir et al. [29]	2023	✗	✗	✗	✗	✗	•
Barbuto et al. [30]	2023	•	✗	✗	✗	✗	✗
Trindade et al. [31]	2024	○	✗	✗	✗	✗	✗
Grzesik et al. [32]	2024	✗	•	✗	✗	•	•
Jouini et al. [33]	2024	✗	✗	✗	✗	•	•
Tu et al. [34]	2025	○	○	○	✗	•	○
Piccialli et al. [35]	2025	○	○	✗	○	○	○
Lourenço et al. [36]	2025	○	✓	○	✓	✗	✗
Jia et al. [37]	2025	○	○	○	✗	✗	✗
Our survey	2026	✓	✓	✓	✓	✓	✓

✓: Fully covers the topic of Edge Learning; ○ : Partially covers the topic of Edge Learning;
•: Focuses on both Edge Learning and Inference; ✗: Does not cover edge learning at all;

(4) **Explore Types of ML**: We examine various types of ML, including unsupervised and reinforcement learning, in the context of edge learning.
(5) **Explore tools and libraries**: We survey tools and libraries for training ML models on edge devices, as well as simulations and emulators for edge learning.
(6) **Use-cases and applications**: We present various use cases and applications of edge learning explored in academic research.

Table 1 presents the relevant studies related to Edge ML, and compares them to our survey based on the aforementioned topics. The symbols used in the table convey the extent to which each study addresses the different topics of edge learning outlined previously.

(1) ✓ indicates that a survey comprehensively covers a particular topic in edge learning.
(2) ○ denotes partial coverage of the topic, where a study may focus on a specific subset of the topic. For example, the surveys [13, 16, 17] explore techniques for using/optimizing ML for the edge but only focus on federated learning.
(3) • signifies that a survey touches on the topic, but its primary emphasis lies on inference on the edge, rather than training, which often results in less comprehensive coverage of the training aspects.
(4) ✗ indicates that a study does not address the topic at all.

1.2 Structure of the survey

This survey is organized into six main sections, excluding this introduction and the conclusion (Section 8). First, Section 2 provides a detailed definition of edge computing, edge learning and edge devices, and presents the requirements and metrics for training ML models at the edge. In Section 3, we explore the techniques used to enable, optimize, and accelerate edge learning. A detailed comparison between these techniques is presented in Section 3.5. In Section 4, we discuss the integration of different types of ML (unsupervised learning, reinforcement learning, etc) on the edge, either for direct training or to optimize/enable the training of other models. In Section 5, we explore the use cases and current applications of edge learning. Then in Section 6, we present various tools, frameworks and libraries used to create simulations and train ML models at the edge. Finally, Section 7 identifies open challenges and discusses potential future trends and research directions in edge learning.

2 EDGE COMPUTING AND EDGE LEARNING

This section introduces the fundamental concepts of edge computing, edge machine learning, and edge learning. We will then examine the essential requirements of edge learning.

2.1 Edge Computing

Edge computing is a new computing paradigm that aims to address the limitations of traditional cloud computing models in handling large scale data generated by the increasing number of smart devices connected to the Internet. It involves performing calculations at the edge of the network, closer to the user and the source of the data. In contrast to traditional cloud computing models, edge computing emphasizes local, small-scale data storage and processing [38]. In this paper, we define edge devices as both edge servers and end devices. For a more thorough dive into edge computing, the reader can consult edge computing surveys such as [38, 39].

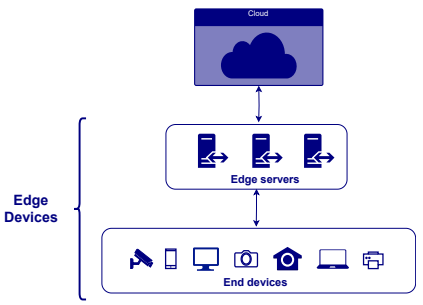

Fig. 1. A typical architecture for edge computing

Edge computing addresses several limitations of cloud computing and provides multiple benefits:

- **Reduced latency**: Edge computing brings data processing closer to the source, reducing the time it takes for data to travel to a centralized cloud server, thereby reducing latency and improving response time [40].

- **Bandwidth reduction**: Edge computing reduces the need for transmitting large amounts of data to centralized cloud servers, reducing bandwidth load and network congestion [38].
- **Improved data privacy**: Edge computing allows for local data processing, reducing the need to transmit sensitive data to centralized cloud servers, thereby minimizing the risk of data breaches and unauthorized access [38].
- **Operational resilience**: Edge computing enables applications to continue functioning even in disconnected or low-bandwidth environments, ensuring operational resilience and reducing dependency on centralized cloud infrastructures [40].

2.2 Edge Learning

A key advancement in edge computing is the integration of AI and ML. Edge ML enables the training and deployment of ML models directly on edge devices, which includes both edge learning and edge inference. Edge learning, also known as edge training, involves training ML models directly on edge devices, reducing the reliance on centralized cloud infrastructure. In contrast, edge inference focuses on facilitating the inference of ML models on resource-constrained edge devices, regardless of where the models were trained [41]. Another term commonly used in the literature is edge intelligence, which shares similarities with edge ML. However, it also includes data collection, caching, processing, and analysis at the edge, making it a broader concept than edge ML [9].

While most Edge ML research focuses on edge inference [42, 43], edge learning remains a promising approach. By enabling localized model training, edge learning can be tailored to the specific requirements and resource constraints of edge devices, making it ideal for applications that require privacy preservation and model customization for specific use cases.

Edge Learning employs various strategies, most of which are either categorized as distributed or collaborative learning methods such as federated or split learning; or on-device learning, which involves training ML models on individual edge devices, and may employ optimization or fine-tuning techniques as necessary. In this survey, we will explore both on-device learning and distributed learning on edge devices. Distributed learning is defined as the training of ML models collaboratively across multiple edge devices. In contrast, on-device learning refers to the training of ML models in a single edge device. To ensure clarity, our definition of edge devices also encompasses edge servers, network elements, and end devices. We comprehensively address ML model training across all these devices.

2.3 Requirements for edge learning

The successful training of ML models at the edge requires meeting specific requirements that dictate the efficiency and performance of these models. These requirements are essential for ensuring that the models perform optimally and efficiently within the resource-constrained environment inherent to edge devices. While no single metric fully captures training efficiency in edge environments [44], the requirements below can collectively serve as evaluation dimensions for assessing how well a method performs under resource constraints.

(1) **Computational Efficiency**: Computational Efficiency refers to the ability of an algorithm to achieve high performance with minimal computational cost. This is especially important in edge learning as edge devices often have limited computational resources [42], and ML models typically require high computational complexity for their training [45].
(2) **Memory Footprint Efficiency**: Similarly to computational complexity, edge devices often have low memory availability [46, 47], which contrast with the large memory requirements of ML models.

(3) **Fast Training Time**: Rapid convergence of model parameters is critical on edge devices, which typically lack the computational headroom to sustain long training runs. Beyond reducing the processing burden and energy cost of the training, quick convergence allows models to adapt to shifting data patterns and evolving user requirements, making the system more responsive to dynamic environments.
(4) **Optimized Bandwidth**: Reducing bandwidth usage involves minimizing data transfer between edge devices and improving communication efficiency among them. This is particularly important for distributed learning techniques and especially in bandwidth-limited systems, since these techniques require frequent sharing of the ML model across the network devices.
(5) **Low Energy Consumption**: Energy consumption is a crucial consideration for edge devices, especially in mobile edge computing. This is due to the limited energy available on such devices. Therefore, ML models trained on the edge must be energy-efficient to ensure better computing performance, longer battery life, and successful model training. Energy efficiency refers to the ability to perform tasks or functions using minimal energy. It involves reducing energy waste and optimizing energy consumption.
(6) **Robustness to Unlabelled Data**: Most edge-generated data is unlabelled [48]. Therefore, using ML techniques that can handle unlabeled data or Auto-labelling strategies may be beneficial in edge learning.
(7) **Task-Specific Metrics and Performance**: Since edge learning encompasses different ML tasks and use-cases, specific metrics and benchmarks are commonly used to evaluate a model's performance to assess its effectiveness in achieving its intended goals.

3 OVERVIEW OF EDGE LEARNING TECHNIQUES

In general, edge ML training is similar to traditional ML training, with the added requirements and constraints outlined in Section 2.3. The feasibility of training on the edge depends on the resource requirements of the model, and the device resources. Today, the increasing processing power, energy storage, and memory capacity of edge devices [49] enable small ML models to be trained on edge devices without requiring significant optimization or distribution. Examples include KMeans [50], Self-Organizing Maps [51], and SVMs [52]. However, the training of more complex models that require heavier resources, such as neural networks, is more challenging at the edge. Therefore, in this section, we will present an overview of techniques used to optimize the training of more complex ML models on the edge. Figure 2 shows a global view of edge learning techniques reviewed in this paper.

3.1 Distributed and Collaborative Techniques

In this section, we will explore distributed techniques to train ML models at the edge. They work by leveraging the computational capabilities of multiple edge devices, and aggregating their results, instead of relying on a single resource constrained device.

3.1.1 Federated Learning. Federated Learning (FL), offers a transformative approach to decentralized model training, where a shared model is trained across multiple clients without the need to transmit local data to a central server [53]. In edge learning, clients typically represent edge devices. As we will see in Section 3.5 FL is the most widely adopted technique for edge learning [16], with applications in various domains, including cyberattack [54, 55] or spam detection [56, 57], smart cities [58, 59], autonomous vehicles [60] and Recommendation systems [61].

The process of training an FL model typically involves three main operations. A client selection process, a local training phase, and a model aggregation phase [53]. These operations are then repeated for multiple rounds until model convergence. While the local training depends on the

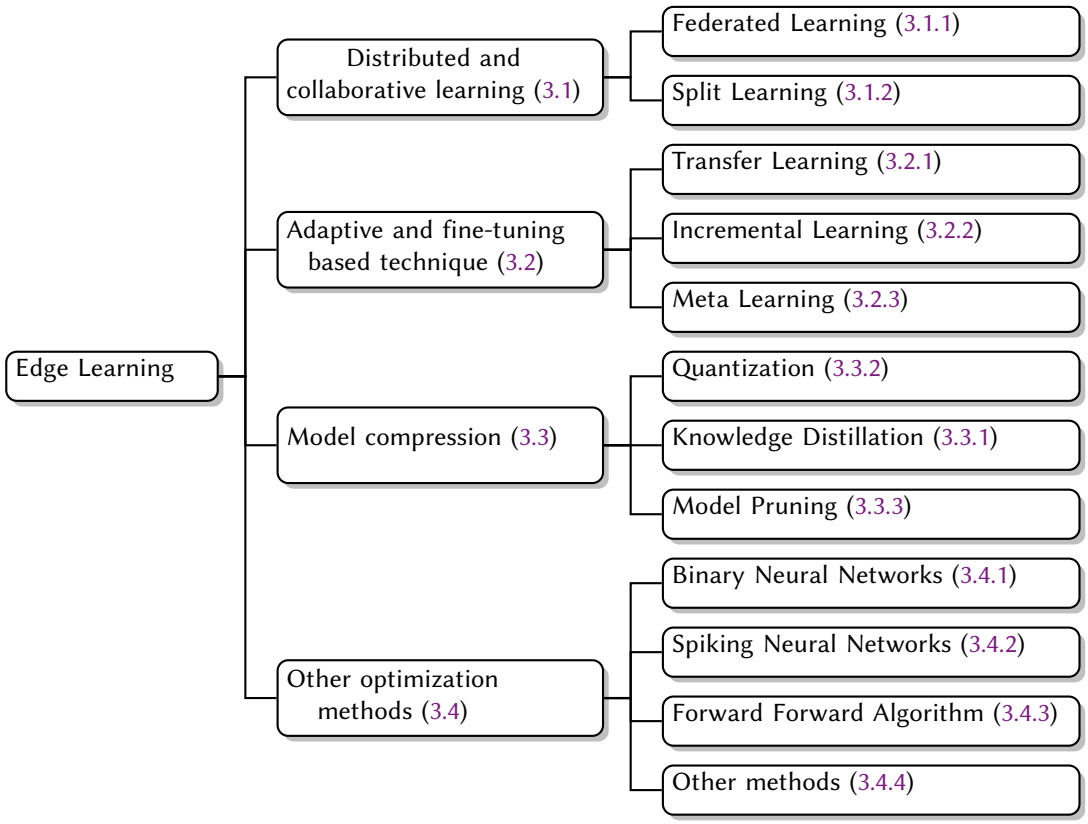

Fig. 2. A taxonomy of techniques used to enable and/or optimize edge learning

specific model architecture and task and isn't typically very different from traditional ML, multiple client selection and aggregation methods have been proposed over the years. Client Selection is an important step where a subset of clients is selected for local training in a given round [62]. While it is possible to select every client in the system at each round, especially if we only have a few clients, this approach can be expensive for local devices, suffer from bottlenecks such as the straggler effect [63] and yield suboptimal results with large number of clients because of larger conflicting updates and slower convergence [64]. The most common selection scheme is random client selection, in which clients are simply randomly selected. However, this approach can underperform compared to other more sophisticated selection methods in terms of accuracy and doesn't take into account the reliability and constraints of local clients. Various other methods ranging from gradient-based, reputation-based, to reinforcement learning based selection methods have been proposed to optimize the selection process for better performance, robustness to attacks, malicious clients and statistical challenges, increased fairness or optimization of local resources [62]. On the other hand, aggregation refers to the combination of multiple locally trained shared models into a global model on a central server, also referred to as an aggregation server. Most aggregation methods in the literature are based on a weighted averaging of the model parameters. The predominant method FedAvg (Federated Averaging) is the original aggregation method, in which each aggregation weight represents the number of instances in the local client's training data [53]. Since then, multiple other aggregation methods have been proposed to address various challenges in FL, such as

handling non-IID data, improving convergence speed, and enhancing robustness against malicious clients [65]. Some of these methods include Per-FedAvg [66], FedAtt [67], and FedMedian [68]. Beyond aggregation methods that focus on weighting client updates, a complementary line of work applies adaptive optimisation at the server level. FedAdam, FedYogi, and FedAdagrad [69] adapt standard adaptive optimisers to the federated server aggregation step, treating the average client update as a pseudo-gradient. Across heterogeneous data settings, these methods consistently improve convergence speed relative to FedAvg [53] without requiring modifications to client-side training procedures. This property is particularly relevant in edge environments, where reducing the number of communication rounds can directly translate into reduced energy consumption and bandwidth usage per device.

Despite its growing popularity and multiple benefits, FL models suffer from some limitations. For instance, non-IID (Non-Independent and Identically Distributed) data across clients often negatively impacts the performance of the global model [70]. To solve this issue, Hybrid FL approaches have been proposed [71], where very small amounts of data is shared between participants. Other approaches to solve this problem include FedNets [72] and FedDynamic [70]. FL is also vulnerable to malicious or low-quality users [73], with attacks such as byzantine attacks. Robust aggregation methods such as Krum [74], FedMedian [68], and others [65] have been proposed to mitigate the impact of such users. Solving the issue using client selection methods has also been investigated using reputation based selection [75] or anomaly detection [73, 76] for handling malicious clients, and backup [77] or asynchronous [78, 79] client selection for improved fault tolerance. Other issues in FL include emerging new classes with completely unseen data distributions whose data cannot be accessed by the global server or other users [80] as well as single node failure [81, 82], channel bandwidth bottlenecks [83], and scaling issues for increasing network size [81]. A deeper issue underlying non-IID degradation is client drift, because each client minimises its own local rather than the global objective, local gradient updates point in directions that may be inconsistent with the global optimum, causing the aggregated model to drift away from the centralised solution [84]. This objective inconsistency becomes more severe as the number of local training steps increases and as data distributions diverge more sharply across clients [85]. Methods such as SCAFFOLD [84] address this through variance reduction via control variates, explicitly correcting the drift in each client's local update direction. FedProx [86] takes a complementary approach, adding a proximal regularisation term to each client's local objective to limit the distance between local and global model parameters, thereby bounding the extent of drift without requiring explicit gradient correction. Together, these methods represent the two principal algorithmic strategies for addressing client drift and should be considered foundational references when evaluating FL convergence under heterogeneous data.

Multiple variations of FL have been introduced for different scenarios. Hierarchical FL methods help address scaling issues and reduce communication costs and training time by introducing multiple aggregation levels between clients and the central server in the form of edge servers [87, 88]. On the other hand, One shot FL aims to reduce the communication cost by training the model in a single round of communication between clients and the server often through the use of knowledge distillation or generated data [89]. While One shot FL offers a more scalable and efficient alternative, it also tends to have lower performance and higher vulnerability to poor updates sent by clients. Asynchronous FL methods allow clients to train and send updates to the server at different times, rather than waiting for all clients to finish their local training before aggregating the updates [78, 79]. This approach is particularly useful in edge environments where clients may have varying levels of computational power and network connectivity [90]. However, asynchronous FL can also lead to stale updates and slower convergence if not properly managed. Another family of variants is Decentralized FL, where the central aggregation server is removed and clients communicate directly

with each other to share model updates using decentralized protocols [91]. Different decentralized FL methods have been proposed, such as gossip-based [92] or swarm-based methods [90, 93], Peer-to-Peer FL [94], and blockchain-mediated FL [95]. Other variations include blind FL [96], over the air FL [97], modular federated learning [98], and clustered federated learning [99] which will be presented in more details in Section 4.2.

There is a growing interest in training language and multimedia models at the edge using FL, with several approaches being introduced in recent years. While the training of Large Language Models (LLMs) using FL is still experimental, some approaches have been tried such as FATE-LLM [100], and FwdLLM [101, 102]. Conversely, smaller language models like BERT [103] have been moderately explored in FL settings, with approaches such as FedBERT [104] that combine FL and SL approaches for pre-training BERT. FedSplitBERT [105] addresses the challenges of heterogeneous data and decreases the communication cost by splitting BERT encoder layers into two parts. A local part trained on the client-side and a global part trained by aggregating gradients of multiple clients. Another example is FedSPAM [56], which fine-tunes a federated distilBERT model [106] on mobile devices for SMS spam detection. For computer vision, FedVKD [107] was proposed as a federated knowledge distillation method for small CNN models on edge devices. Periodically, the knowledge from these models is transferred to a large server-side vision transformer encoder via knowledge distillation. On the other hand, [108] introduces an FL approach for visual classification with real-world data distribution. Finally, training audio models at the edge using FL is widely explored for tasks such as speech recognition [109, 110] or audio classification [80, 111].

One important concept in FL is differential privacy, a technique that involves adding artificial noise to protect individual privacy while maintaining model utility [112]. The survey [113] provides a detailed examination of differential privacy while [114] and [115] provides an exploration of differential privacy in the context of FL. Another privacy preserving method in FL is secure aggregation [116], which enables the server to have access to the aggregated sums of all transmitted parameters while ensuring that individual raw updates from clients are unreadable. This is typically achieved with pairwise masking [116], where random noise is added to each client in a way that it cancels out when combined between clients. Other secure aggregation protocols are also regularly employed to increase robustness of the secure aggregation to user dropout vulnerabilities, byzantine fault tolerance, or to reduce communication overhead [117]. Secure aggregation is practically important in edge deployments where the aggregation server may not be fully trusted [118], such as healthcare applications [119], UAV-assisted federated edge learning systems [120, 121], fault prediction in industrial equipment [122], or cross-device recommendation systems [123]. FL is also used alongside other techniques presented in this survey, such as split learning [124], meta learning [125, 126], knowledge distillation [28, 127, 128], and Quantization [105, 128, 129], etc.

3.1.2 split learning. Split learning offers an alternative to FL for collaborative edge training. Rather than training the full model locally, it divides the model into sections, one section is trained on the client, while the remaining sections are trained on a central server [130] or, less commonly, on other clients [131]. Instead of transmitting raw data, only the activations at the cut layer are passed to the server, and gradients are returned to the client for backpropagation. This reduces the computational load on individual edge devices and makes split learning particularly suitable for resource-constrained environments. However, split learning incurs higher communication overhead than FL due to its relay-based, sequential execution [124, 132], and the transmitted intermediate activations introduce privacy leakage risks [133]. In recent years, there has been a growing interest in split learning at the edge, as evident from the increasing number of studies in this area (see figure 3). For instance, SplitEasy [134] is a framework that enables the training of ML models on mobile devices using split learning. Another paper, [135] proposes a data protection approach for

split learning without compromising the model accuracy. Additionally, [136] proposes an online model splitting method with a resource provisioning game scheme that aims to minimize the total time cost of participating devices. Finally, VFLAIR-LLM [137] introduces a framework that allows privacy-preserving fine-tuning and inference of LLMs on resource constrained devices. Adaptive split learning is a branch of split learning that aims to overcome its shortcomings compared to FL. Specifically, it addresses these challenges by eliminating the transmission of gradients from the server to the client, resulting in a smaller payload and reduced communication cost, and allowing the client to update only sparse partitions of the server model, adapting to the variable resource budgets of different clients which decreases the computation cost and improves performance across heterogeneous clients [138, 139]. Split learning has often been combined with FL, in order to eliminate both techniques' inherent drawbacks [124, 140–142].

3.2 Adaptive and Fine-Tuning-Based Techniques

The techniques in this section share a common assumption: the most computationally expensive part of training has already been performed in the cloud or edge servers. What remains on the edge device is a lightweight adaptation step, which reduces computational overhead while preserving privacy and enabling personalisation.

3.2.1 transfer learning. Transfer learning is an ML technique which applies knowledge from a pre-trained model to a new, related task, instead of building models from scratch. By capitalizing on existing knowledge, transfer learning accelerates model training, improves generalization, and proves exceptionally useful in domains or scenarios where data scarcity and limited computational resources pose a challenge [143] such as on edge devices. For a more detailed understanding of transfer learning, readers are encouraged to review the surveys [143–145]. In the context of edge learning, transfer learning is a prominent technique used to fine-tune ML models based on local data in an edge device. This approach serves as a fully on-device alternative to collaborative learning methods that distribute the training across different devices [47]. Notable state-of-the-art methods for transfer learning in edge learning include TinyTL [46], which addresses the critical issue of memory efficiency in low-memory edge devices. This is achieved by freezing the weights of the model and only learning a memory-efficient bias module, thus removing the need to store the intermediate activations. Similarly, RepNet [146] proposes an intermediate feature re-programming of a pre-trained model with a tiny reprogramming network to develop memory-efficient on-device transfer learning. MobileTL [147] also proposes a memory and computationally efficient on-device transfer learning method, specifically designed for models built with inverted residual blocks. Additionally, [148] propose an edge CNN framework for 5G industrial edge networks, with the CNN model trained in advance in an edge server, which is further fine-tuned based on the limited datasets uploaded from the devices with the aid of transfer learning, and [149] proposes a runtime convergence monitor to achieve massive computational savings in the practical on-device training workloads. Multiple approaches also focus on combining transfer learning with FL, to create federated transfer learning algorithms [150], that aim to leverage FL for privacy preservation, and use transfer learning to train a well-performing local model despite users typically having insufficient data for this purpose. This is achieved by training the base model with a public dataset and passing it to the federated users to be fine-tuned for the target task [151, 152]. Finally, [153] proposes freeze and reconfigure, a transfer learning method for on-device training of a BERT model.

3.2.2 Incremental Learning. Incremental learning also known as continual learning or life-long learning, involves continuously updating and expanding a model's knowledge as new data becomes available. Unlike traditional batch learning, where models are trained from scratch on entire

datasets, incremental learning dynamically incorporates new information without discarding previously acquired knowledge [154], and can be used to address the well-known issue of catastrophic forgetting in deep neural networks [155–157]. Readers seeking a more thorough understanding of incremental learning are directed to [158]. There have been considerable attempts at implementing incremental learning in the context of edge learning. These include: learning with sharing [159] which aims to reduce the training complexity and memory requirements while achieving high accuracy during the incremental learning process and bypass the considerable memory requirements that can make incremental learning unsuited for edge devices; PILOTE [155] which trains an incremental learning model on edge devices for human activity recognition; [160] introduces an incremental algorithm based on transfer learning and k-nearest neighbor to support the on-device learning; RIANN [161] is an indexing and search system for a graph-based approximate nearest neighbor algorithm for mobile devices; and RILOD [162] which aims to incrementally train an existing object detection model to detect new object classes without losing its capability to detect old classes, to avoid catastrophic forgetting. RILOD distills three types of knowledge from the old model to mimic the old model's behaviour on object classification, bounding box regression and feature extraction, and it was implemented under both edge-cloud and edge-only setups. There are a variety of promising approaches and directions for incremental learning on the edge from combining it with other techniques (such as FL [163, 164], meta learning [164, 165] and compression methods [166–168]) to sparse [169] or distributed continual learning [167].

3.2.3 Meta-learning. Meta learning focuses on enhancing a model's ability to learn new tasks quickly and effectively. Unlike traditional learning paradigms that optimize for a specific task, meta learning trains models to learn from a diverse set of tasks, thereby enabling them to generalize knowledge and adapt rapidly to novel tasks with minimal data [170]. By exposing models to various learning scenarios, meta learning equips them with transferable skills, such as recognizing patterns and adapting to new contexts. For further insight into meta learning, we recommend consulting [171, 172]. In the context of edge learning, the application of meta learning can help address the challenges posed by limited data availability and resource constraints [173, 174]. For instance, [175] proposes adaptation-aware network pruning, a model pruning method designed to work with existing meta learning methods to achieve fast adaptation on edge devices, while [165] proposes a continual meta-learning approach with bayesian graph neural networks that mathematically formulates meta-learning as continual learning of a sequence of tasks. On the other hand, p-Meta [173] aims to achieve faster generalization to unseen tasks and enforces structure-wise partial parameter updates to support memory-efficient adaptation. Meta learning can also be used in combination with other techniques such as FL [164, 176], or FedMC [125] which integrates reinforcement learning models trained on edge devices into a generalized federated reinforcement learning framework based on a meta-learning method.

3.3 Model Compression-based Techniques

This section covers model compression techniques, which reduce the size and computational cost of ML models, making them more amenable to training under the resource constraints of edge devices. While they are traditionally used in edge inference, a notable shift is observed towards employing these techniques for the training in the edge.

3.3.1 Knowledge distillation. Knowledge distillation in deep learning is a process whereby a small or student neural network is trained to emulate the knowledge and predictive capabilities of a larger or teacher network. This technique serves as a means to transfer the expertise and generalization capabilities of a complex model to a simpler one. As a result, inference efficiency is enhanced, and computational demands are reduced. The underlying principle involves the student network

learning not only from ground truth labels but also from the soft, probabilistic outputs of the teacher network, thereby capturing finer details and nuances in the data [177]. For a deeper dive into knowledge distillation, readers can refer to the following survey [177]. In the context of edge learning, knowledge distillation is usually used to reduce the size and complexity of a large neural network, to simplify its training in limited resources devices. Therefore, knowledge distillation is well suited to be used in collaboration with other techniques such as federated learning [28, 127], split learning [178] or incremental learning [166]. However, distillation is also used independently of other techniques [179, 180]. The integration of knowledge distillation with FL on edge devices has shown promising results, with recent trends indicating great potential in combining the two techniques [28]. Several approaches have been proposed, including attack-resistant FL methods [181], speech recognition tasks [182] or keyword spotting [183]. Mix2FLD [184] is another method that combines knowledge distillation and FL. Meanwhile, [128] use both distillation and quantization to train FL models on edge devices. Several other hybrid approaches combining FL and knowledge distillation have been explored [185, 186]. These approaches are further discussed in Wu et al.'s survey on knowledge distillation in federated edge learning [28]. Knowledge distillation can also be applied to other distributed learning methods, such as [178], which introduces a spatio-temporal distillation method for split learning for a tiny server in order to alleviate the frequent communication costs that happen when communicating from the server to edge devices. [187] introduces a distributed distillation algorithm where devices communicate and learn from soft-decision outputs, which are inherently architecture-agnostic and scale only with the number of classes in order to alleviate the communication costs from transmitting model weights in the network and improve the inclusion of devices with different model architectures. Finally, knowledge distillation has been used as a standalone technique in an edge learning context, for recommendation systems [179, 188, 189], edge cardiac disease detection [190] and on-device deep reinforcement learning [191]. Knowledge distillation was, additionally, used with multiple variants, including dataset distillation techniques [192, 193] and knowledge transfer [180, 194].

3.3.2 Quantization. Quantization in deep learning refers to the process of reducing the precision of numerical values representing model parameters or activations, typically from floating-point to fixed-point or integer representations, in order to strike a balance between maintaining an acceptable level of model accuracy and reducing the memory and computational requirements [195]. This computational optimization technique is pivotal in mitigating the resource-intensive demands of deep neural networks, rendering them more amenable for resource-constrained hardware platforms, such as edge devices and embedded systems. There are two types of quantization: quantization-aware training and post-training quantization. In quantization-aware training, the quantized model is fine-tuned using training data in order to adjust parameters and recover accuracy degradation or perturbation introduced by the quantization. In contrast, post-training quantization is a less expensive approach, where the pretrained model is quantized and its weight adjusted without any fine tuning [195]. To gain a more complete understanding of quantization techniques, readers are advised to consult [195]. Quantization techniques are not only used to optimize machine learning models before deployment for edge inference [149, 196], but also in edge learning to simplify the fine-tuning of large models [47, 197]. Like knowledge distillation, quantization is frequently combined with other techniques, including FL [198], transfer Learning [47], incremental learning [199] or with other types of techniques [200]. Quantization is used with FL particularly extensively, including one-bit quantization [201–203] and hierarchical FL [129]. Other FL-based approaches that utilize quantization include [128, 204–206]. Other quantization-based methods for training ML models in the edge include quantization-aware scaling, which was proposed in [47]. This method automatically scales the gradient of tensors with different bit-precisions

without requiring any fine-tuning, and was used alongside a tiny training engine and sparse updates. Investigations in [207] showed that quantization helps further in reducing the resource requirements for the training for on-device few shot learning for audio classification. The Holmes optimizer [208] uses quantization to improve the accuracy by combining different quantization techniques, such as limiting quantization bits, fixed-point numbers, and logarithmic quantization.

3.3.3 Model Pruning. Model pruning is a technique used to reduce the size of ML models by removing certain parts of the model, such as model parameters, nodes in a decision tree [209] or weight matrices in transformer-based models [210]. Similarly to quantization, model pruning is commonly used in the edge to reduce the computational resources required for the inference of ML models [211]. Model pruning has also shown great potential in edge learning, where it reduces the size of ML model before fine-tuning on edge devices. This technique is particularly effective when used in conjunction with other methods, such as FL [212–215], incremental learning [160, 168] or meta-learning [175]. Similar to knowledge distillation and quantization, FL emerges as the most prominent technique when combined with model pruning for edge learning. Noteworthy is PruneFL [214], which reduces communication and computation overhead by adaptively adjusting model size throughout training, beginning with an initial pruning stage at a selected client before proceeding with FL rounds. Other approaches that combine model pruning with FL include [215] which introduces model pruning for wireless FL to scale down neural networks. Meanwhile, [213] employ an adaptive dynamic pruning approach to prevent overfitting by slimming the model through the dropout of unimportant parameters. In addition, several approaches use model pruning on edge devices. For instance, [216] uses model pruning in the context of on-device personalization for an activity recognition system, and Deeprec [217] leverages model pruning and embedding sparsity techniques to reduce computation and network overhead. Furthermore, OmniDRL [218], a deep reinforcement learning based approach on edge devices, incorporates weight pruning in each learning iteration to achieve a high weight compression ratio. Finally, [219] explores the reduction in memory footprint for further pruning during the training phase of BitTrain, a bitmap memory efficient compression technique for training on edge devices.

3.4 Optimization and Acceleration-Based Techniques

In this section, we explore some other techniques that don't fit neatly into a specific category and are used to optimize or provide more optimized alternatives to ML models for edge learning.

3.4.1 Binary neural networks. Binary Neural Networks (BNNs) are deep neural networks that use binary values (-1 or 1) instead of floating-point numbers for weights and activations. BNNs are attractive for resource-constrained devices because of their ability to compress deep neural networks [220]. BNNs share similarities with other techniques, such as quantization and model pruning, which are also considered good candidates for edge inference due to their extreme compute and memory savings over higher-precision alternatives [221]. However, BNNs' compute and memory efficiency can also be leveraged for edge learning. For example, by proposing a hybrid quantization of a continual learning model [222], or by developing a model based on an MRAM array with ternary gradients for both training and inference on the edge [223]. Other BNN based approaches for edge learning include [221, 224, 225].

3.4.2 Spiking neural networks. Spiking neural networks (SNNs) are another type of deep neural networks that are promising for the edge. SNNs communicate between neurons using events called spikes [226] and are known for their asynchronous and sparse computations. These properties result in decreased energy consumption [227, 228], which makes them well suited for energy limited devices [229]. Training ML models at the edge using SNNs has gained some attention in recent years.

For example, [230] proposes FL-SNN, a cooperative training through FL for networked on-device SNNs, while [231] presents a memristor spiking neuron and synaptic trace circuits for efficient on device learning. Other approaches include integrating meta-learning with SNNs for lifelong learning on a stream of tasks with local backpropagation-free nested updates [174], and using event-driven, power and memory-efficient local learning rules, such as spike-timing-dependent plasticity [232]. There are other approaches that leverage SNNs for edge learning, including [233–235].

3.4.3 Forward Forward Algorithm . The Forward-Forward (FF) algorithm [236] is an alternative to backpropagation that replaces the forward and backward passes with two forward passes operating on positive and negative data with opposite objectives. By training layers locally, FF eliminates the need to store intermediate activations across the full network, directly reducing the peak memory pressure that makes backpropagation impractical on MCU-class devices. This approach has been adapted for edge learning in several methods. μ-FF [237] demonstrates on-device FF training on microcontrollers, while PEPITA [238] offers a related backpropagation-free approach based on error-driven input modulation. Sparse-FF [239] further improves FF efficiency by selectively processing only the top-k most significant neuron activations during training, achieving up to 61× lower computational effort than backpropagation on certain datasets. FF-INT8 [240] combines FF's layer-wise strategy with INT8 quantization and a look-ahead scheme to improve accuracy. Application-oriented studies have also explored FF for camera trap image classification on the edge [241], structural crack detection [242], and embedded computer vision through a novel Contextual Convolution Block that addresses FF's known degradation on deeper networks and larger output spaces [243]. Finally, [244] integrates a sparse FF variant into FL settings to reduce both the computational load and communication overhead.

3.4.4 Other techniques. In this section, we will explore some other techniques used to optimize ML models for training in the edge. One such technique is MiniLearn [245], which enables re-training of deep neural networks on resource-constrained devices using IoT data collected during deployment. Another approach used in the tiny training engine [47] are sparse updates, which skip the gradient computation of less important layers and sub-tensors, to allow for a reduction in memory usage and computation. On the other hand, [246] introduces a novel reduced precision optimization technique for on-device learning primitives on MCU class devices with specialized shape transform operators and matrix multiplication kernels, which is accelerated with parallelization and loop unrolling for the backpropagation algorithm. Conversely, POET [247] allows for the training of large neural networks on memory-scarce and battery-operated edge devices, with integrated rematerialization and paging. POET reduces the memory consumption of backpropagation, allowing the fine-tuning of both ResNet-18 and BERT within edge devices' memory constraints. Other approaches include data booleanization [248], echo state networks [249], or tensorization [200].

3.5 Comparison of techniques used in edge learning

In this section we will compare the different families of techniques used to train ML models in the edge that we explored in the previous part of this paper based on two factors: (i) The number of academic contributions of each family and their evolution over the years; (ii) The techniques' potential of answering the different needs and requirements particular to edge learning.

3.5.1 Comparison of the usage of the different techniques. We will first start by comparing the different edge learning techniques over the years by analyzing their academic contributions. Figure 3 shows the number of papers per technique per year on a logarithmic scale. We used the advanced

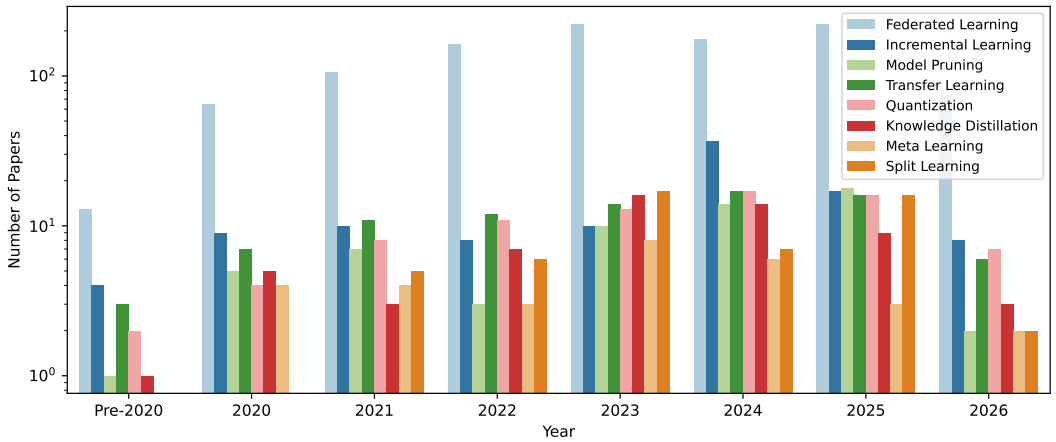

Fig. 3. Trend of techniques used to train ML models in the edge over the years

search features of Scopus[1] and Web of Science[2] to get the data. Using those search engines, we searched for the terms "edge learning", "Edge Intelligence", "training/learning on the edge/mobile devices", "on-device training/learning" and "on-device adaptation" as well as relevant keywords for each technique ("Federated Learning", "Split Learning", etc.), in the title, keywords and abstract. We excluded surveys, books, and notes, as we are only interested in technical contributions to optimizing ML model training on the edge using the aforementioned techniques. Additionally, we manually reviewed each paper and removed papers that either didn't provide a technical contribution, or weren't about training ML models on the edge, despite containing relevant keywords. Finally, we added a few papers found manually that were not indexed in both Scopus and Web of Science or did not contain the relevant keywords, but were still relevant to our analysis. We excluded the families of techniques that had fewer than 10 papers in total for better clarity. The excluded techniques are: forward-forward, BNNs and SNNs. This exclusion applies exclusively to the quantitative trend analysis in Figure 3, which requires a minimum corpus of data points to produce meaningful year-over-year comparisons. These three families are retained in the taxonomy (Figure 2) and in the qualitative comparison (Table 2) because they represent conceptually distinct and potentially significant directions for edge training, particularly given their potential advantages even if their current training-specific literature is insufficient to support trend claims. Additionally, since our analysis is specifically about methods for training on the edge, we ignore methods in FL that introduce FL theory or FL specific aggregation or selection methods that don't include discussions, or experiments related to edge learning. The final number of papers associated with the analysis was 1411 papers, some of which, were briefly covered in Section 3. Note that multiple techniques can be used in a single paper (see Figure 4), therefore the total count in the Figure 3 will exceed 1411. The cut-off date for the year 2026 is the 15th May 2026.

The analysis of Figure 3 reveals that FL is the dominant approach for training ML models in edge environments, given the resource constraints of edge devices. This dominance is expected, as distributed learning methods that capitalize on the collective computing power of multiple devices are deemed more practical and efficient in edge settings. Moreover, we anticipate that this

[1]Scopus: https://www.scopus.com/
[2]Web of Science: https://clarivate.com/products/scientific-and-academic-research/research-discovery-and-workflow-solutions/webofscience-platform/

trend will persist and expand to include split learning, another promising distributed learning technique. Other methods, including incremental learning, transfer learning, Model Compression Techniques (e.g., quantization, knowledge distillation), although consistently employed, lag behind FL in terms of popularity. A closer examination of Figure 3 unveils a remarkable surge in the number of publications focused on edge learning over the past six years. Notably, there has been a steady rise in the adoption of FL and split learning, aligning with our forecast of a trend favoring these two techniques. In contrast, the use of techniques like incremental learning and transfer learning has been more steady during the same period, and meta-learning despite being employed consistently in previous years received less attention in 2023. Finally, for the model compression techniques while quantization and knowledge distillation experienced a small rise in popularity over the years, model pruning has exhibited greater fluctuation during that period.

As previously discussed, various approaches have been proposed that integrate multiple techniques to mitigate the limitations of individual methods and capitalize on their respective strengths. Examples of such approaches include [124, 150, 164, 165]. To provide a clearer illustration of the relationships between these various techniques, a heatmap depicting the intersection of their usage is presented in Figure 4. This visual representation allows for a more comprehensive understanding of the synergies and overlap between different approaches. The heatmap includes all the technique families discussed in Section 3. We can note that FL has been combined the most with other techniques, which is expected considering the overwhelming number of FL contributions to the edge (see Figure 3). Furthermore, model compression techniques such as knowledge distillation, model pruning and quantization are also often used together with other techniques as shown in previous sections. Finally, we can observe other collaborations between the different families, and we expect this trend to continue in the future.

3.5.2 Comparison based on the requirements and needs of edge learning. As outlined in Section 2.3, there are several requirements that must be met for edge learning, which can function as imprecise measures for assessing the viability of edge learning methods. In this section, we use these requirements as rough evaluation dimensions to compare the families of techniques presented in Sections 3.1 through 3.4. Before discussing the results, we clarify the criteria underpinning our assessments and the scope of their validity. We exclude "Robustness to Unlabelled Data" from this comparison as it is more a function of the ML paradigm (supervised, semi-supervised, unsupervised) than of the optimisation technique itself, and is addressed separately in Section 4. Similarly, "task-specific metrics and performance" is replaced by a general "high performance" dimension that captures whether a technique tends to preserve, improve, or degrade model quality relative to its unoptimised counterpart. As a result, we use six distinct measures to evaluate the techniques:

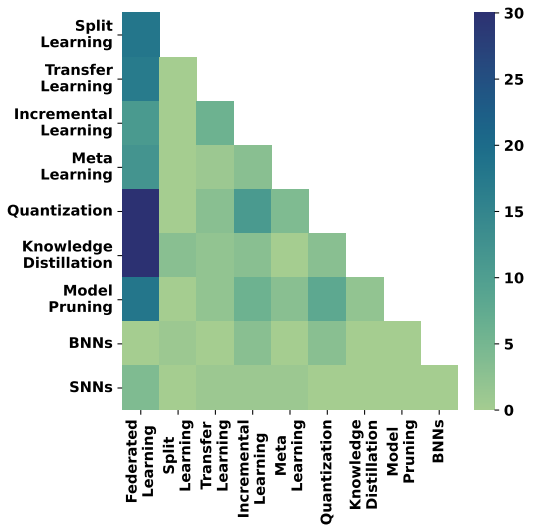

Fig. 4. Co-occurrence of techniques used together for training ML models at the edge. Color intensity represents the number of papers.

Table 2. Comparison between The different techniques that enable edge learning

Technique	Computational Efficiency	Memory footprint	Low energy consumption	Fast Training Time	Optimized Bandwidth	High Performance
Federated Learning	✓	✗	✓	✓	•	✗
Split Learning	✓	✓	✓	✗	✗	\
Transfer Learning	•	✗	•	•	✓	✓
Incremental Learning	•	✗	•	•	✓	✓
Meta-Learning	•	✗	•	•	✓	✓
Knowledge Distillation	✓	✓	✓	✓	✓	✗
Quantization	✓	✓	•	✓	✓	✗
Model Pruning	✓	✓	✓	✓	✓	✗
BNNs	✓	✓	✓	✓	✓	✗
SNNs	✓	\	✓	\	\	✗
Forward-Forward Algorithm	✓	✓	✓	✓	\	\

✓: Have a positive effect on the requirement;
✗: Have a negative effect on the requirement;
•: Have a neutral or uncertain effect based on specific conditions on the requirement;
\: There is not enough information and literature to estimate the effect on the requirement
Detailed justifications and supporting references for each mark are provided in the per-family discussion of Section 3.5.2 (Distributed Learning Methods through Other optimisation techniques).

computational efficiency, memory footprint, low energy consumption, quick training time, reduced bandwidth, and high performance.

The outcomes, as depicted in Table 2, should be interpreted as coarse-grained, family-level assessments rather than verdicts on any specific implementation. A checkmark ("✓") indicates that, in general and across the majority of studied implementations, a technique generally improves or is well-suited to a given requirement relative to a baseline of naive centralised training or unoptimised on-device training. A cross ("✗") indicates that, in general, the requirement represents a weakness of the technique in the edge learning context. It is important to note that techniques like FL, quantization, and others have various variants and specific approaches that influence how these methods align with the requirements. For instance, while quantization techniques may result in a minor decrease in performance in most scenarios [250], leading to an "✗" check in the "High Performance" column. Certain specific approaches to quantization may exceptionally yield no performance loss or even improvements, as demonstrated in studies such as [208]. Therefore, Table 2 offers a general overview informed by the literature rather than a definitive judgment on the suitability of each technique for every situation, as these and are subject to change depending on variants, implementation details, use cases, tasks, and hardware platforms.

Distributed Learning Methods. First we start by discussing and analyzing the distributed methods FL and split learning. Both methods achieve computational efficiency through distribution. Rather than requiring a single device to execute a full training pass, they spread the compute burden across multiple participants, which reduces the per-device load [53, 87]. However, they differ on several other dimensions, FL's most significant structural weakness is memory footprint. Standard FL requires each participating client to hold a complete copy of the global model in memory during local training [53], which is directly at odds with minimizing memory footprint. In other words, while FL spreads the computational burden across client, the model would still need to be loaded in full. On the other hand, split learning's memory footprint is substantially smaller than FL's for the same model architecture [124, 134] since only a portion of the model is executed and stored

on the client device. This makes it particularly attractive for very resource-constrained devices or large model architectures, as shown by frameworks like SplitEasy [134] and VFLAIR-LLM [137]. FL and split learning's energy consumption is generally favourable relative to non-distributed training, as a result of better computational efficiency. Training time per round on the other hand, is generally fast at the device level for FL, though wall-clock convergence can be slow when many rounds are required due to non-IID data [251, 252]. However, split learning's relay-based execution, in which each forward pass must traverse client and server sequentially, introduces significant communication overhead and slows training time compared to FL [124]. Bandwidth is an inherently mixed picture for FL: while raw data never leaves devices, model updates must be transmitted every round, and for large models this communication cost is substantial [83]. One-shot FL [89] and hierarchical FL [87, 88] reduce this cost meaningfully, but standard FedAvg does not. For split learning, every training step requires transmitting intermediate activations and gradients between the cut layer on the client and the remainder of the model on the server, which makes split learning poorly suited to high-latency or bandwidth-constrained networks [124, 136]. Finally, FL's performance relative to centralised training is a well-documented weakness [70, 251, 252]: the aggregation of locally trained models from non-IID data distributions systematically degrades global model quality, and while some methods [65, 68, 70, 74, 86] partially mitigate this, they do not eliminate it. Split learning's performance is harder to assess generally, since it depends heavily on where the model is split and whether the split is adaptive [138, 139]; we therefore mark it as unknown. It must also be noted that combining split learning with FL [124, 140–142], can partially recover the bandwidth and time disadvantages of pure split learning while retaining the memory efficiency advantage, and represents a promising direction discussed further in Section 7.

Adaptive and fine-tuning based techniques. Transfer learning, incremental learning, and meta-learning all assume that the heaviest part of training has already been performed, and that what remains to be done on the device is a comparatively lightweight adaptation step. This shared property makes all three broadly favourable for bandwidth efficiency (since only a fine-tuned or adapted model, not raw data or full model updates, needs to move between cloud and device in typical configurations [47, 151]) and for model performance (since fine-tuning from a strong pre-trained initialisation typically yields better results than training from scratch under data scarcity [143, 147]). However, these techniques generally do not reduce the memory footprint of the model during adaptation. Transfer learning in its standard form still requires loading the full model onto the device to fine-tune it [46, 147]. Notable exceptions exist. TinyTL [46] addresses this by freezing weights and learning only a small bias module, and MobileTL [147] proposes memory-efficient adaptation for inverted residual block architectures, but these are specialised solutions rather than properties of transfer learning in general. Incremental learning faces an even sharper memory challenge because it must retain sufficient information about prior tasks to avoid catastrophic forgetting [154–156], which typically implies maintaining either replay buffers, expanded model components, or regularisation terms that all carry memory costs [159, 166]. Meta-learning similarly requires loading the full model, and approaches such as p-Meta [173] that introduce partial parameter updates are still the exception rather than the norm. The effect of these techniques on computational efficiency and energy consumption is ambiguous and depends heavily on implementation. A transfer learning approach that freezes most layers [46, 153] requires far less compute than one that fine-tunes the full network [150, 151], but both fall under the same family label. We therefore mark these dimensions as mixed (•) for all three techniques, while noting that practitioners can meaningfully move them toward ✓ by adopting parameter-efficient fine-tuning strategies such as those in [46, 147, 153] or sparse update methods [47]. A critical distinction that Table 2 cannot fully capture is the dependency of these techniques on the availability of a

pre-trained model and, in the case of federated transfer learning [150–152], on access to a suitable public dataset or centrally collected data for base model training. When such resources are available, adaptive techniques can be good options for edge learning.

Model compression techniques. Knowledge distillation, quantization, and model pruning share the goal of reducing the size and computational cost of ML models, which translates directly and reliably into improvements across most of the resource-related requirements in Section 2.3. All three reduce memory footprint by definition [177, 195, 214], as a distilled student model, a quantized model, or a pruned model is smaller than its uncompressed counterpart. All three similarly reduce computational cost, which in turn reduces energy consumption and potential training time [47, 177, 195, 214]. Bandwidth is also improved as a side effect, smaller models require fewer bits to transmit when used in conjunction with FL or split learning [129, 178, 214]. The consistent weakness of all three compression techniques is performance degradation [214, 250]. Compressing a model inherently involves discarding information, soft labels in distillation allow some recovery [177], but the student model is still limited by its reduced capacity, quantization introduces rounding errors that accumulate through training [195]; and pruning removes parameters that may carry task-relevant information [214, 215]. Exceptions exist. For instance, the Holmes optimizer [208] uses a combination of logarithmic and fixed-point quantization that can avoid accuracy loss in specific settings, but these are not representative of the family in general. As such a trade-off should be expected in general with compression techniques, which are most appropriate when a small and predictable performance degradation is acceptable in exchange for substantially reduced resource requirements. In healthcare diagnostic applications, for instance, even a small accuracy drop may be clinically unacceptable [253], whereas in smart home or environmental monitoring applications, a modest degradation may be entirely tolerable [254, 255]. It must also be noted that as shown in Figure 4, these methods are the most commonly combined with decentralized [127, 128, 128, 129, 178, 181–184, 201–206], and adaptive [47, 160, 166, 168] methods.

Other optimisation techniques. Assessments for BNNs [220–222], SNNs [226, 230, 231], and the Forward-Forward Algorithm [236, 237, 247] are based on a literature base of fewer than 10 papers specifically addressing edge training. These should be treated as preliminary and directional rather than established assessments. Binary neural networks and spiking neural networks both offer theoretical advantages in computational efficiency and energy consumption by replacing floating-point arithmetic with binary or event-driven operations [220, 226, 228, 229]. BNNs in particular achieve extreme memory and compute compression [220, 221], and SNNs' asynchronous, sparse computation is well-suited to energy-limited devices [226, 228–230]. However, both families are assessed with significant uncertainty on several dimensions because the empirical literature on their use in edge learning remains limited. Most BNN and SNN work focuses on inference efficiency, with training-specific results sparser and less consistent [221–223, 230, 231]. The forward-forward algorithm [236] and related methods such as μ-FF [237] and PEPITA [238] avoid the backpropagation pass entirely, which in principle reduces peak memory usage and avoids the memory overhead of storing intermediate activations. These approaches are promising precisely because they target the training-specific memory pressure that compression techniques do not fully address, but they remain early-stage and their performance profiles are not yet well-characterised across diverse tasks and devices. Until this literature matures, these techniques should be considered experimental for training workloads, and practitioners should expect implementation complexity significantly above that of the more established families.

Overall Synthesis. Taken together, the analysis of Table 2 and the preceding discussion supports several actionable observations:

- No single technique family satisfies all requirements simultaneously, and the choice of technique is fundamentally a prioritisation decision over competing requirements. FL prioritises computational efficiency and energy while accepting memory and performance penalties. Split learning prioritises memory while accepting communication costs. Compression techniques prioritise resource reduction while accepting performance loss. Adaptive techniques prioritise performance and bandwidth while accepting that pre-trained models and cloud infrastructure must be available.
- Since the performance dimension is the most consistently adversely affected by techniques designed to reduce resource consumption, when performance is a primary constraint, practitioners should first assess whether a pre-trained model exists for their domain and whether adaptive fine-tuning on the edge is feasible, before considering distributed or compression-based approaches.
- Combining techniques can resolve trade-offs that no individual technique can address alone, and this is the direction in which the most active research is occurring, as Figure 4 illustrates. FL combined with quantization [128, 129] reduces FL's bandwidth cost without sacrificing its privacy properties. FL combined with knowledge distillation [28, 127, 184] can reduce both bandwidth and memory requirements. Transfer learning combined with FL [150–152] can improve performance under non-IID data by initialising from a strong shared base model. These combinations are not without cost. They can introduce additional implementation complexity, weaken existing strengths, and in some cases create additional privacy risks.
- The choice of technique should be made with reference to the constraints of the deployment scenario rather than from general preference. A device with very limited memory but reliable connectivity is poorly served by FL but can be well served by split learning. A device with intermittent connectivity but adequate compute is poorly served by split learning but may be well served by on-device transfer learning with sparse updates [47]. A device with only local data and no cloud access is the canonical setting for decentralised FL [91, 92] or gossip learning [92], not for any technique that assumes a central aggregation server. In deployments where the aggregation server itself is not fully trusted, secure aggregation [116] should be treated as a baseline requirement rather than an optional enhancement. Making these constraints explicit before selecting a technique is the most effective way to avoid deploying methods whose structural assumptions are violated by the deployment environment.

4 EDGE LEARNING FOR DIFFERENT TYPES OF MACHINE LEARNING

This section examines how different ML paradigms present specific challenges and opportunities when adapted to the edge. We focus on unsupervised, semi-supervised, self-supervised, and reinforcement learning, as these paradigms require dedicated treatment at the edge. Supervised learning, while included in the discussion, is not explored to the same depth as the others as it is the default training paradigm, and the vast majority of techniques explored in this survey are already formulated as supervised methods.

4.1 Supervised Learning

Supervised learning can be viewed as the default paradigm in edge learning. The techniques surveyed in Section 3 such as FL, split learning, transfer learning, and model compression, are usually in their standard formulations, applied to supervised objectives. Rather than repeating that

treatment here, this subsection identifies three supervised learning challenges that are specific to the edge context and that directly motivate the technique choices discussed in Section 3.

Label noise and annotation quality. In edge settings, supervised training data can be labelled under imperfect conditions, by non-expert users, by automatic labelling pipelines, or derived from noisy sensor readings. Label noise degrades supervised model quality in ways that are compounded in FL settings, where the central server cannot inspect local data to identify or correct annotation errors [256–258]. Auto-labelling systems such as Flame [259] represent one practical response to this challenge at the edge, while robust aggregation methods [65, 68, 74] partially mitigate the downstream effect of noisy local updates during FL training.

Class imbalance under non-IID distributions. In distributed supervised training, the class distribution across clients is rarely uniform. A device monitoring a specific environment or user will observe only the classes present in that local context, creating per-device class imbalance that translates directly into the objective inconsistency problem described in Section 3.1.1 [251, 260]. Clustered FL approaches (See more details in Section 4.2) and personalised aggregation methods [66, 86] are specifically designed to address this by grouping clients with similar class distributions or allowing local model personalisation.

Personalisation of supervised classifiers. A recurring theme across the application domains surveyed in Section 5, including human activity recognition [155, 261–263], recommendation systems [123, 264, 265], and healthcare [253, 266, 267], is the need for supervised models that are simultaneously globally coherent and locally personalised. This is structurally a supervised learning problem and can be addressed through federated transfer learning [150–152], meta-learning [173, 175, 176], and personalised aggregation methods [66, 86]. The tension between global and local objectives is arguably the central design challenge in supervised edge learning, and the techniques described in Section 3 can be understood as providing the principal engineering tools for navigating it.

4.2 Unsupervised Learning

Considering the vast amount of unlabeled data produced in edge and end devices [48], it is very promising to use unlabeled data to train ML models on the edge. However, unsupervised learning comes with multiple challenges and restrictions for edge learning, especially when it comes to collaborative learning techniques such as FL or split learning, which represent the vast majority of techniques used in the edge (see Figure 3). Unsupervised learning datasets may have a non-IID nature. Each node in a collaborative setting might have a different subset of the data, and the data distribution might vary across nodes. This non-IID property can make it difficult to effectively combine information from different nodes. Additionally, in the case of clustering algorithms, clusters may have varying sizes across nodes in a collaborative setting and clustering algorithms may need to adapt to changes in data distribution and cluster structures over time. Finally, since there are no available labels, and their assignment may differ between nodes (for example in a clustering algorithm), ensuring consistency across distributed nodes is difficult but crucial for aggregating meaningful global labels (clusters).

Figures 5, 6 and 7 represent the main different types of unsupervised learning approaches used for training ML models in the edge, (a) using unsupervised learning algorithms directly on-device with non-collaborative methods [51, 268] as shown in Figure 5; (b) using unsupervised learning methods to assist in the training of collaborative learning approaches, such as clustered federated learning [99, 269, 270] highlighted in Figure 6; and (c) training an unsupervised learning model on the edge collaboratively such as federated clustering [271] in Figure 7.

Unsupervised learning on a single edge device. Having an unsupervised learning model trained on a single edge device is possible if the device has enough computation power and/or the learning algorithm is lightweight and can be trained with low resources, the training with such models is usually no different from the training on the cloud or other devices, the only difference being the constraint of low resources and data available [51]. Examples of unsupervised learning algorithms trained with this approach include [50] which investigates the application of K-Means on mainstream controllers, and [272] that presents the first dedicated CycleGAN accelerator for energy-constrained mobile applications, achieving a higher throughput-to-area ratio and higher energy efficiency than a GPU. In [268], an unsupervised segmentation was proposed that can be executed on edge devices without the need of annotated data. While [273], proposes TMNet, an approach to solve unsupervised video object segmentation problem at the edge. Finally, [274] proposes an FPGA based architecture for a self-organization neural network capable of performing unsupervised learning on input features from a CNN by dynamically growing neurons and connections in order to perform class-incremental lifelong learning for object classification in the edge.

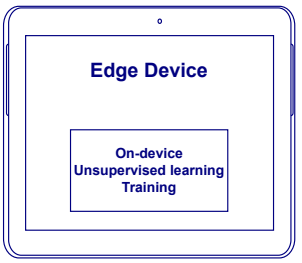

Fig. 5. Unsupervised learning with the training happening only on a single device, with all the required data hosted locally

Unsupervised learning to assist collaborative learning approaches. As seen in Section 3.1.1, FL faces multiple challenges, including non-IID data, uneven computing power [275] and sub-optimal results when the local clients' data distributions diverge [99]. To address these issues, the usage of clustering alongside FL has been explored to cluster devices with similar environmental data distributions [276]. One popular approach is Clustered Federated Learning (CFL) [99], which exploits geometric properties of the FL loss surface and group the client into clusters with jointly trainable data distributions. CFL has been widely adopted and has inspired other approaches, such as [269, 270, 275, 277]. Alternative approaches include hierarchical over-the-air FL [278], which utilizes intermediary servers to form clusters near mobile users, and HPFL-CN [276], a hierarchical FL method that clusters edge servers with similar environmental data distributions, then train personalized models for each cluster using a hierarchical architecture. Finally, [80] uses unsupervised learning methods for audio classification to distinguish between classes across different users, when new classes emerge with completely unseen data distributions on device in FL settings.

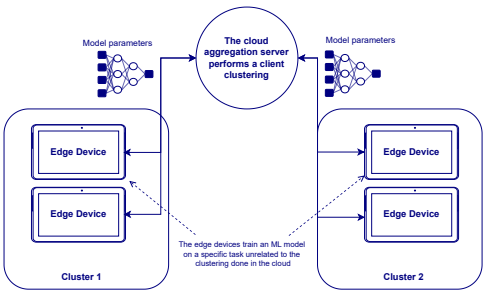

Fig. 6. Unsupervised learning to assist collaborative learning, a clustering is typically applied to edge devices to improve on the FL process (for example CFL)

Collaborative Unsupervised learning at the edge. Collaborative unsupervised learning methods are challenging to train for all the reasons explained previously. Nevertheless, several methodologies have surfaced. Among them is federated clustering, which aim to execute clustering on distributed data without sharing the data [271, 279, 280]. Federated clustering methods can be described as one-shot if they require only one round of transfer between clients and the servers [271, 281], or

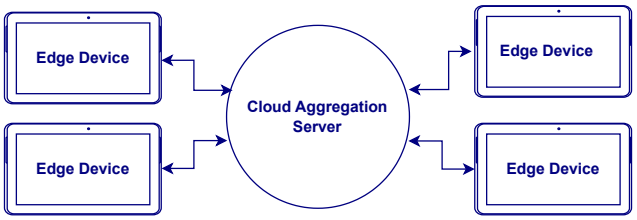

Fig. 7. Collaborative unsupervised learning, the goal being to apply unsupervised learning algorithms on completely distributed data

they can require multiple rounds of communication [282]. An alternative methodology, known as FedUReID, has been proposed by Zhuang et al. [283] as a person Re-identification system without the use of any labels, all the while ensuring the preservation of privacy. Finally, federation of unsupervised learning [284] proposes a method where unlabeled data is transformed to become surrogate labelled data for each client. Following this, the desired model is obtained by recovering it from a modified model which is trained using supervised FL.

4.3 Reinforcement learning

Reinforcement learning (RL) has been successfully applied in the past on different problems in areas such as robotics, recommendation systems, video games and automatic vehicles [285, 286], making RL a promising and interesting direction for edge learning. However, the training of RL models in resource-constrained environments is often limited by high compute and memory requirements from gradient computations [287], making the application of RL in the context of edge learning challenging. Despite these challenges, multiple RL approaches have been proposed for training ML models in the edge. Among them, federated RL [288, 289] is a promising approach that allow multiple RL agents to learn optimal control policies for a series of devices with slightly different dynamics [290]. Furthermore, it is employed to achieve diverse objectives including personalization [291, 292], IoT traffic management [293, 294], Autonomous Systems [290] and resource allocation for unmanned aerial vehicle (UAV) [295]. FedMC [125] integrates RL models trained by multiple edge devices into a general model based on a meta-learning approach. FedGPO is an RL-based aggregation technique for FL introduced in [296], and aims to optimize the energy-efficiency of FL while guaranteeing model convergence. On the other hand, [297] introduces DDQN-Trust, a trust-based double deep Q-learning-based selection algorithm for FL that takes into account the trust scores and energy levels of the IoT devices to make appropriate scheduling decisions and integrate it with the main FL aggregation techniques (FedAvg, FedProx and FedShare). Other RL approaches that don't rely on FL or other collaborative learning paradigms have been proposed, such as [298] which uses online sequential learning to achieve full on-device RL on an FPGA platform. In [299], a method combining supervised and reinforcement learning is proposed for adaptive video streaming on edge servers or on-device, and [300] introduces an on-device RL-based adaptive video transmission algorithm to predict heterogeneous network bandwidth. Finally, RL has also been employed in edge learning via shielding techniques [301], as proposed in [302] with a multiagent system that enables each edge node to schedule its own jobs using SROLE, a shielded RL technique used to check for action collisions that may occur because of the absence of coordination between the nodes, and provides alternative actions to avoid them.

4.4 Semi-supervised learning

Semi-supervised learning, is a paradigm that combines labeled and unlabeled data for specific learning tasks [303]. It can be described as a middle ground between supervised and unsupervised learning, and leverages the advantages of both approaches. By combining a small amount of labeled with a large amount of unlabeled data. This is particularly beneficial in an edge learning context, where vast amounts of unlabeled data are generated continuously by end devices [48], and where sending the data for labeling in the cloud is not always possible for privacy reasons. For a comprehensive overview of semi-supervised learning, refer to the following survey [303]. Semi-supervised learning presents an alternative mean to harness the vast amount of unlabeled data at the edge. Additionally, it offers other advantage on the edge, as training datasets are often incomplete before training and might need supplementation with real-time data [304]. Various methodologies have been developed to adapt semi-supervised learning to edge environments, including FL based approaches [305, 306], as well as other techniques [307–309].

4.5 Self-supervised learning

Self-supervised learning is an ML approach that allows models to learn from vast amounts of data without explicit labels [310]. It creates labels from the data itself by defining pretext tasks where the data provides its own supervision. For instance, in natural language processing, a common pretext task involves predicting the context surrounding a word or predicting a given word in a sentence given the previous word, also known as language modeling. In computer vision, a pretext task might involve predicting masked patches of an image. Self-supervised learning can be especially beneficial in domains where labelled data is scarce, or the specific task can not be known a priori [310]. For a deeper exploration on self-supervised learning we refer the reader to [310].

Because of its independence from labeled data, self-supervised learning is considered a promising approach for edge learning. By learning from data without explicit labels, it enables learning useful representations and skills that can be fine-tuned for specific tasks, such as recommendation systems [179, 188], speech and audio-related applications [311, 312], and others [313–316]. Moreover, self-supervised learning can be particularly effective in scenarios with data and concept drifting [313]. Several recent studies have proposed innovative self-supervised learning methods tailored for edge devices, such as contrastive learning [313, 315, 316].

5 EDGE LEARNING USE CASES AND APPLICATIONS

As explained in previous sections, edge learning offer multiple advantages that range from low latency, bandwidth efficiency to privacy preservation and improved reliability and robustness, it also allows more customization and personalization by adapting to user preferences and behavior without relying on centralized computing or needing to collect and store private user data in cloud servers. In this section, we explore some uses cases and applications for edge learning that have been researched and developed in the past years.

5.1 Healthcare and Remote Monitoring

The use of ML in healthcare has been under constant improvement over the last years [317]. However, cloud-based ML solutions still struggle to meet the sector's stringent security requirements [318], address privacy concerns [319], and satisfy low latency requirements [320]. Edge learning has emerged as a promising solution to address these challenges, gaining traction in the field, mostly by using federated edge learning [321–323]. However, while FL in healthcare is increasingly explored as a privacy-preserving approach, the training of these models is often done with large resource requirements [253, 324], making it hard to implement on edge devices.

Moreover, for various tasks, medical data is collected and managed directly by large organizations such as hospitals and medical facilities [325]. For such tasks FL with more powerful clients with access to GPUs may be preferred, since the privacy constraints in these scenarios often involves data not being shared outside the organization rather than the local device. Nevertheless, for tasks where the data shouldn't leave medical edge devices and wearables, such as sensors and ECG devices, edge learning remains a viable and promising direction [266, 326]. Research that use edge learning for healthcare spans most of the field, from atrial fibrillation recognition [327], preterm labor risk prediction [328], cardiac disease detection [190], breast ultrasound image classification [329] to dermatological disease [316] and COVID-19 diagnosis, leveraging techniques such as CFL [321] and federated transfer learning [330]. For a more comprehensive overview of FL and edge learning in healthcare, refer to the survey [253].

When it comes to monitoring for medical purposes, Human Activity Recognition (HAR) represents one of the most popular use cases. HAR refers to the automation of the identification and categorization of the various activities performed by humans and their interactions with the environment [331]. As personalization for HAR has been shown to improve the results and performance of these systems [267], training the model on the edge using incremental learning or meta learning approaches can help achieve that while increasing the privacy and reducing the bandwidth consumption. PILOTE [155] proposes an incremental learning-based approach for HAR, designed for edge devices with extremely limited resources and demonstrates reliable performance in mitigating catastrophic forgetting. In addition, [261] proposes a personalizable lightweight CNN model for HAR, as well as a training algorithm to find personalization-friendly parameters. With the objective of improving the accuracy after the personalization when dealing with a wide range of target users. ClusterFL [262], proposes a clustering-based FL approach for edge-based HAR. Finally, [263] presents an on-device deep learning approach for STM32 microcontrollers, which fine-tunes a CNN model for enhanced HAR personalization.

5.2 Smart Technologies

Edge learning has emerged as a pivotal technological advancement for smart technologies such as smart cities [332], smart agriculture [333], smart homes [254], etc. In this section, we will explore some of its applications in these settings. Smart cities are urban ecosystems designed using IoT technologies to solve urban life problems and improve the residents' quality of life [334]. In this context, edge learning has been proposed to solve different challenges, mainly for its ability to leverage ML capabilities while preserving network bandwidth and reducing the charge on cloud servers. In [335], a cloud-aided edge learning method based on knowledge fusion for smart lighting systems has been proposed. Another application of ML in the edge is in smart grid systems where ML is needed to improve demand forecasting and automated demand response, as well as to analyze data related to energy use and obtain energy consumption patterns [336], detect anomalies [337], improve communications [338] and security [339] in the system. Other applications in smart cities include the detection of abnormal and dangerous activities [340, 341], pedestrian detection [342], water consumption forecasting [58, 343] and reducing congestions in intelligent traffic systems [344, 345]. Smart farming is another domain where ML is increasingly used to enhance the production quality, crop selection, and mineral deficiency detection, as well as to increase farmers' earnings [346]. In [333] a TinyML based framework using deep neural networks and LSTM models for unmanned aerial vehicles assisted smart farming was proposed, which measure soil moisture and ambient environmental conditions. Smart farming remains a promising domain for edge learning, although further research is needed for effectively harnessing its potential. Finally, using edge learning in smart homes can also be promising, however, at the time of writing this article, only a few papers explore this area, including [254].

5.3 Autonomous vehicles

Autonomous vehicles are vehicles that can operate without human intervention. They utilize sensor technologies, AI, and networking to navigate and make decisions [347]. Autonomous vehicles include self-driving cars, trucks, buses, drones, Unmanned Aerial Vehicles (UAVs), and even small robots. As demonstrated in [348], offloading deep learning tasks to edge devices or servers can improve the inference accuracy while meeting the latency constraint, which makes edge learning perfectly suitable for this use-cases, and as expected there has been extensive research done in this area. UAVs are by far the most prominent use of edge learning for autonomous vehicles. In [349], a synchronous FL structure for multi-UAVs was proposed, that aims to resolve device privacy concerns that come from sending raw data to UAV servers, as well as UAVs' limited processing or communication resources. On the other hand, [350] propose a model-aided federated MARL algorithm to coordinate multiple UAVs on data harvesting missions with limited knowledge about the environment, significantly reducing the real-world training data demand. As mentioned previously in Section 5.2, [333] aims to assist farming operations using UAVs that measure soil moisture and ambient environmental conditions and [351] proposes a model to derive computation specifications for learning-based visual odometry from physical characteristics of UAVs. Edge learning based approaches for other autonomous vehicles are also constantly explored and involve multiple applications, they include:

- **Trajectory predictions** such as [352] that proposes a solution for trajectory prediction in the edge for both human-driven and autonomous vehicles by leveraging the capabilities of the 5G multi-access edge computing platform to collect and process measurements from vehicles and road infrastructure in edge servers and use an LSTM model to predict the vehicle trajectory with high accuracy.
- **Energy efficiency** for autonomous vehicles, where [353] proposes a rate-splitting multiple access (RSMA)-based Internet of Vehicles system for energy-efficient FL in autonomous driving, using non-orthogonal unicasting and multicasting transmission.

5.4 Recommendation systems and personalization

Recommender systems are intelligent applications that assist users in making decisions by providing advice on products or services they might be interested in [354]. However, recommender systems rely on user data and can pose threats to user privacy, such as the inadvertent leakage of data to untrusted parties or other users [355]. Furthermore, privacy-enhancing techniques may lead to decreased accuracy in the recommendations [356]. Edge learning, and especially collaborative learning approaches such as FL, have a big potential in solving these problems by allowing recommender models to be partially or completely trained on the edge, keeping user interactions on the device and using them to further personalize the system [357]. Different approaches using FL have been used for recommendation systems, in what we call federated recommendation. Overall, they can be divided into two main categories, cross-platform (or cross-organization) and cross-device (or cross-user) FL [61]. In cross-platform FL, each federated client can have multiple users, this is often the case when multiple organizations or institutions collaborate to build a shared recommendation system without sharing their data. On the other hand, in cross-device FL, each client corresponds to a single user, and the model is trained on the user's device using their local data. While cross-platform FL can be used in edge learning, it is more commonly used in cloud-based FL settings. Cross-device FL is more suitable for edge learning, as it allows for training on individual user devices with maximum privacy settings [123]. Most federated recommendation systems adapt existing centralized recommendation models to FL settings, such as matrix factorization [264], neural collaborative filtering [265], sequential recommendation models [358],

and graph based recommendation [359, 360]. Other methods propose new FL aggregation and selection techniques to improve the performance of federated recommendation systems, such as FedFast [361], which introduces both a selection and aggregation method to improve the efficiency of federated recommendation systems and increase its convergence speed. A promising paradigm, in edge learning federated recommendation is cross-domain recommendation. Which aims to leverage data from multiple domains to improve recommendation accuracy in a target domain. While cross-domain recommendation has been extensively studied in centralized settings [362], the conventional approach of sharing data between services in a cloud setting often proves impractical or impossible due to privacy and security concerns. This limitation emphasizes the appeal of edge learning as an interesting direction for cross-domain recommendation. By enabling the training of recommender systems on multi-domain data residing on edge devices, while still respecting users' privacy. Cross-domain recommendation in edge learning has been attempted in a few works, such as FedCT [152], FedGCDR[363] and FedCDR[364], however, this area remains largely unexplored and presents a promising avenue for future research. Another interesting direction for edge learning recommender systems is cloud-edge collaboration approaches, where the model is partially trained on the edge and partially on the cloud. This approach leverages the inherent need to transfer parts of user data to cloud servers for some applications. For example, some platforms may be required by law to centrally store some data, such as logs for purchases or transaction histories. However, other types of data, such as clicks, browsing history, and other interactions, can be kept on the device. Cloud-edge collaborative federated recommendation might be more adapted to such scenarios [365]. Although the most popular approach, FL isn't the only method to train recommender systems on the edge. On-device learning using transfer learning [217] or knowledge distillation [179] has also been explored in the literature. Finally, other distributed learning methods are also used, such as Split Learning [366] or full decentralized methods [367].

5.5 Others

There are multiple other applications and use cases of edge learning that we couldn't explore in this section, from keyword spotting [183, 368], spam detection [56, 57], IoT threats prediction [55], camera trap image classification [369], detecting defects in photovoltaic components [370], estimating air quality [255], to face spoof attack detection [371] and speech recognition [109, 110, 163, 372–375], another interesting potential application explored in [376] is the use of edge learning in lunar analog environments for future space missions. In general, any use case that benefits from personalization on private user data, or suffers from bandwidth limitations or privacy risks in the training might benefit from fully or partially using edge learning. Therefore, we expect the trend of edge learning to continue rising, expand into other fields and areas and grow beyond the current use cases and applications in both academia and the industry.

6 LIBRARIES, SIMULATORS AND TOOLS FOR EDGE LEARNING

As an emergent field, edge learning requires multiples tools to facilitate its usability, integration and implementation, ranging from emulators and simulators, used to train and test ML models on cloud servers before training on the edge, to libraries that allows the successful training of ML models on edge devices.

Although there has been significant work on creating libraries and frameworks for ML at the edge, most of these libraries focus on the deployment and inference of deep learning on edge devices [377]. Only a few libraries enable the training of ML models on edge devices. These include

Table 3. Comparison between The different Frameworks for edge learning

Framework Name	Support simulation	Allow training on edge device	Type	Language	Platform
TensorFlow Lite [380]	✗	✓	Deep Learning	Python	Android / iOS
ONNX Runtime	✗	✓	Deep Learning / ML	Multiple Languages	Android / iOS / Other Platforms
Flower [378]	✓	✓	FL	Python	Android / iOS
FedML [379]	✓	✓	FL	Python	Android / iOS
PySyft [381]	✓	✓	FL	Python, Kotlin, Swift	Android / iOS
FedERA [382]	✓	✓	FL	Python	Raspberry Pi / CPUs / GPUs
FedLab [383]	✓	✗	FL	Python	-
FedJax [384]	✓	✗	FL	Python, Jax	-
LEAF [385]	✓	✗	FL	Python	-
Flute [386]	✓	✗	FL	Python	-
PyVertical [387]	✓	✗	FL / Split Learning	Python	-
OpenFL [388]	✓	✗	FL	Python	-
EasyFL [389]	✓	✗	FL	Python	-
FL_PyTorch [390]	✓	✗	FL	Python	-
TensorFlow Federated	✓	✗	FL	Python	-
CoreML	✗	✓	ML	Swift, Objective-C	iOS
EdgeRL [391]	✗	✓	RL	C/C++	Embedded Platforms
PULP-TrainLib [392]	✗	✓	Deep Learning	C	RISC-V Multi-core MCUs
NodeRec+ [393]	✓	✗	Recommendation	Python	-

ONNX Runtime[3], TensorFlow Lite[4], and libraries that focus on distributed learning tasks such as Flower [378] or FedML [379]. Some tools, only allow for researching, prototyping and experimenting of FL methods and are designed for simulating FL methods on the cloud. While others allow for the training and deployment of these techniques in edge devices. Table 3 shows a list of frameworks intended for edge learning, or for running training simulations for edge learning.

PyTorch [394] and TensorFlow [380], the most popular frameworks for training deep learning models, have both developed edge ML libraries. However, while TensorFlow Lite allows for both inference and training on the edge, ExecuTorch[5], only supports inference at the time of writing. Note that PyTorch models can be trained using ONNX Runtime[6]. ONNX Runtime is a cross-platform ML accelerator with on-device training capabilities. It has deep integration with PyTorch, Hugging Face[7] and TensorFlow, enabling accelerated training and inference on multiple platforms, including mobile devices (Android, iOS) and various hardware accelerators and programming languages. Additionally, ONNX Runtime supports FL on edge devices through its on-device training capabilities.

Over the years, multiple tools and libraries have been proposed to train FL algorithms on edge devices, driven by the need for efficient and decentralized learning. As mentioned earlier, these

[3]https://cloudblogs.microsoft.com/opensource/2023/05/31/on-device-training-efficient-training-on-the-edge-with-onnx-runtime/
[4]https://blog.tensorflow.org/2021/11/on-device-training-in-tensorflow-lite.html
[5]https://docs.pytorch.org/executorch/stable/index.html
[6]https://onnxruntime.ai/blogs/pytorch-on-the-edge
[7]https://huggingface.co/blog/optimum-onnxruntime-training

tools can be categorized into two types. Those that only allow simulation of an edge learning environment in the cloud, and those that allow training on edge devices.

Simulation only FL tools. Simulation tools are extremely important in the context of edge computing. They are used to model the behavior of fog/edge infrastructures, allowing for the study of interoperability across different layers and protocols in edge-cloud environments [395]. In edge learning, simulators enable experimentation with ML models on cloud servers, facilitating rapid prototyping and experimentation. Many edge learning simulators focus on FL. Notable examples include FL_PyTorch [390] a PyTorch-based simulation tool for FL, and TensorFlow Federated[8] a similar tool for TensorFlow. Other notable FL simulators are: LEAF [385]; FedJax [384] a JAX-based open source library; Flute [386] an open source platform with multiple optimization, privacy, and communication strategies; And finally FedLab [383] a lightweight open-source framework that focus on algorithm effectiveness and communication efficiency, and allows customization on server optimization, client optimization, communication agreement, and communication compression.

FL Simulation tools, which allow the training on an edge device. In recent years, numerous libraries have emerged to support the experimentation, development, and deployment of FL algorithms on edge devices. These libraries enable researchers to develop and test FL algorithms on the cloud while facilitating the transition from simulation to real-world deployment on the edge. Notable examples include Flower [378], and FedML [379] which are both aimed toward the research and experimentation of FL algorithms, while allowing the execution of the algorithms on a variety of edge devices. PySyft [381] is another FL open-source library that was built as an extension of PyTorch, Keras, and TensorFlow, and can be run on mobile devices using KotlinSyft[9] for Android and SwiftSyft[10] for iOS. FedERA [382] is a similar library that includes a verification module to ensure the validation of local models and avoid aggregating malicious ones. Additionally, FedERA features a carbon emission tracker module to accurately estimate CO2 emissions during the local parameter update phase.

Other non-FL Frameworks for edge learning. Edge learning frameworks and libraries have been proposed to target specific platforms or hardware. These frameworks aim to simplify the training of ML models on various devices. Notable examples include CoreML, a Swift-based ML inference and training framework for iOS, designed to simplify ML model deployment and training on iOS devices. PULP-TrainLib [392] is another framework, proposed for on-device training on RISC-V multi-core microcontrollers. Additionally, EdgeRL [391] is a lightweight C/C++ framework for on-device reinforcement learning, designed to run on single-core processors typically found in resource-limited embedded platforms.

Comparison of Tools in Regards to Training on the Edge. While Table 3 characterises the available frameworks in terms of their type, language, and target platform, it does not directly connect them to the requirements defined in Section 2.3. Table 4 addresses this gap by mapping each on-device framework (that is, frameworks that allow actual training on edge devices rather than cloud simulation) to those requirements. As with Table 2, assessments reflect framework-level design decisions rather than the behaviour of any specific model or task, and the same legend applies. The "high performance" dimension is excluded because it reflects the quality of the trained model rather than a framework design property. On the other hand, the "Robustness to Unlabelled Data" dimension is retained. While it was excluded from Table 2 because it depends on the ML

[8]https://www.tensorflow.org/federated
[9]KotlinSyft: Syft worker for secure on-device machine learning for Android https://github.com/OpenMined/KotlinSyft
[10]SwiftSyft: Syft worker for secure on-device machine learning for iOS https://github.com/OpenMined/SwiftSyft

Table 4. Mapping of on-device frameworks to edge learning requirements (Section 2.3)

Framework	Comp. efficiency	Memory footprint	Low energy	Fast training	Optimized Bandwidth	Robustness to Unlabelled Data
TensorFlow Lite [380]	✓	✓	•	•	/	•
ONNX Runtime	✓	✓	•	✓	/	•
CoreML	✓	✓	•	•	/	•
PULP-TrainLib [392]	✓	✓	✓	✓	/	•
EdgeRL [391]	✓	✓	✓	•	/	✓
Flower [378]	•	✗	•	•	•	•
FedML [379]	•	✗	•	•	•	•
PySyft [381]	•	✗	•	•	✗	•
FedERA [382]	•	✗	✓	•	•	•

✓: Framework design explicitly targets this requirement;
✗: Structural weakness relative to this requirement;
•: Depends on technique and hardware choices 2;
/: Not applicable (bandwidth is not framework-level for single-device frameworks)

paradigm rather than the optimisation technique. It becomes meaningful at the framework level as some are architecturally tied to supervised objectives, others, such as EdgeRL, are inherently label-free, and FL frameworks such as Flower or FedML can support unsupervised paradigms such as federated clustering [271] depending on the technique configured by the practitioner.

Table 4 maps the on-device frameworks to the requirements defined in Section 2.3. Two observations emerge from this mapping. First, there is a clear split in how frameworks address the requirements. General-purpose on-device frameworks (TensorFlow Lite, ONNX Runtime, CoreML) primarily address computational efficiency and memory footprint through operator-level optimisation (quantised kernels, reduced-precision arithmetic, and memory-efficient graph execution), but leave energy and training-time behaviour largely to the application and hardware layers. Energy consumption and training time are marked neutral for most of these frameworks because they depend heavily on the model architecture and the underlying hardware accelerator rather than on framework design choices. Specialised frameworks such as PULP-TrainLib and EdgeRL go further by targeting the specific constraints of ultra-low-resource hardware (such as MCUs and single-core embedded platforms respectively) where energy consumption is an explicit design objective rather than a side effect. Second, FL frameworks that support on-device deployment (Flower, FedML, PySyft, FedERA) present a different profile. Their primary contribution is orchestration, managing client selection, aggregation, and deployment logistics, rather than resolving the device-level resource constraints of Section 2.3. Memory footprint in particular is marked negative for all four, because FL requires each participating client to hold a full copy of the global model during local training [53], a structural property of the paradigm that no framework-level design choice can eliminate. The exception is energy: FedERA explicitly incorporates a carbon emission tracker module to monitor energy consumption during local updates [382], making it the only FL framework that actively surfaces energy as an operational concern rather than treating it as a side effect. Regarding bandwidth, the FL frameworks differ in ways that are attributable to framework-level design rather than to the aggregation method alone. Flower [378] and FedML [379] are communication-agnostic by design, delegating compression and quantisation to the practitioner, and therefore neither improve nor worsen bandwidth relative to a baseline FL configuration. PySyft [381] incurs a framework-level bandwidth penalty through its secure aggregation protocol as a tradeoff to added security. Pairwise masking requires each client to exchange mask shares with every other participant, introducing a communication overhead that scales quadratically with

the number of clients and is independent of any technique choice [116]. FedERA [382] sits between these two, its model validation module introduces a modest additional round of communication to verify local models before aggregation, but this is bounded and does not scale with client count. On Robustness to Unlabelled Data, all four FL frameworks are marked neutral. They can in principle support unsupervised paradigms such as federated clustering [271], but this requires explicit configuration by the practitioner and is not a built-in capability of any of them.

7 OPEN ISSUES, RESEARCH DIRECTIONS AND FUTURE TRENDS

In this section, we dive into the challenges, emerging research paths, and future trends in edge learning. To provide a comprehensive overview, we will divide this section into two parts. The first part will examine the open issues and existing challenges in edge learning, highlighting the obstacles that need to be addressed. The second part will explore promising research directions and our predictions for future trends, shedding light on the opportunities that lie ahead.

7.1 Challenges and open issues

7.1.1 Resource constraints. As highlighted in previous parts of this survey, the resource constraints inherent to edge devices pose the biggest challenge for edge learning. In Section 3 we explored the different approaches designed to optimize and accelerate the training of ML models on the edge. However, despite current efforts, limitations in computation, memory, and sometimes energy continue to impede the training of the largest and most complex ML models on the edge. As recent tasks and use cases demand bigger and more complex ML models, the resource limitations of edge devices still pose a significant challenge and remain an open issue. Consequently, ongoing research efforts focus on optimizing ML models for resource constrained environments. Another promising idea, not explored in this survey, involves optimizing edge device hardware for ML [396–398].

7.1.2 Challenges in detecting data quality issues in the edge. Ensuring data quality is a crucial aspect of training ML models [256, 257]. However, this has proven challenging in the context of edge learning. Due to the decentralized nature of storage inherent in edge computing, detecting data quality issues such as missing or incorrect labels and noisy data is difficult. Additionally, edge devices can be prone to hardware failures, leading to missing or corrupted data [399, 400]. As such, data quality for ML on the edge remains an ongoing challenge [258], necessitating further research into developing new methods for detecting and fixing data quality issues, as well as designing ML models that are robust to these issues. The survey [258] explores the different challenges, constraints, potential solutions, and ongoing efforts related to data quality for ML on the edge.

7.1.3 Lack of labelled data availability. In the context of edge learning, a prominent challenge arises from the prevalence of unlabeled data on edge devices. This issue becomes particularly problematic as the majority of ML applications traditionally emphasize supervised learning paradigms, necessitating labeled datasets for effective training [48]. To address this challenge, it is imperative to explore no-label or few-label solutions. This includes a focus on unsupervised (Section 4.2), self-supervised (Section 4.5), or semi-supervised (Section 4.4) techniques as well as methods that can make the most of limited labeled instances such as few-shot learning. Moreover, the development of auto-labeling systems, exemplified by solutions like Flame [259], emerges as a promising solution to mitigate the impact of the labeled data scarcity on the edge.

7.1.4 Abundance of non-IID data on the edge. As described in Section 3.1, distributed learning methods, such as FL, stand out as pivotal techniques in edge learning. However, a significant body of literature on FL is done under the assumption of IID data [260], and very often this assumption doesn't reflect the data present on edge devices. While the effects of non-IID data on FL depends

widely on the type of FL method employed, it often negatively impacts the training [251, 252, 401]. And while multiple solutions have been proposed to handle non-IID data, they might come at the expense of privacy preservation and a clear benchmark of real non-IID performance is unclear considering the large diversity in FL methods [251]. More details on the impact of non-IID data on FL as well as the specific challenges it poses, and the different approaches proposed to tackle the issue can be found on the following surveys [251, 260].

7.1.5 Data leakage and privacy concerns. Although edge learning aims to provide a privacy-aware alternative to traditional machine learning, it is not immune to privacy and leakage risks during or as a result of the training phase. Notably, model vulnerabilities can be exploited to leak sensitive data, particularly in FL settings where the model is shared among clients [402, 403]. This vulnerability is exacerbated when the model is susceptible to data reconstruction attacks, increasing the risk of private data leakage [402]. To mitigate these risks, techniques like differential privacy [114, 115], encryption methods [404], and other approaches are often employed to ensure optimal privacy. However, further research is necessary to guarantee the preservation of private data and minimize leakage risks in edge learning.

7.2 Future trends and research directions

7.2.1 Hybridization of ML techniques in edge learning. In the exploration of edge learning techniques detailed in Section 3, various strategies have been employed to optimize the training of relatively large ML models on edge devices. However, each of these techniques, as outlined in Table 2, comes with distinct advantages and drawbacks. Recognizing the diversity in these methodologies, there has been a notable surge in approaches that advocate for a hybridization of multiple techniques. Figure 4 illustrates this promising trend, wherein researchers aim to maximize the advantages and mitigate the drawbacks of individual techniques by combining them. This emerging direction, signifies a deliberate effort to create comprehensive and robust solutions tailored to the unique challenges posed by edge devices. Given the promising results showcased by such hybrid approaches, the trajectory indicates a continued surge in interest and research efforts towards refining and expanding the applicability of hybridized techniques in the domain of edge learning.

7.2.2 Training large models at the edge. Large models, including Large Language Models (LLMs) [405–407], Diffusion models [408], and Audio Generation models [409, 410], etc., are increasingly prevalent and are steadily growing in popularity. However, despite a growing demand for personalization of these models [411, 412] and privacy concerns in collecting and using personal or private data on centralized cloud servers [413], the fine-tuning, training or personalization of such models at the edge is very challenging considering the limited resources available for edge devices [414]. Although contributions in this domain are currently limited, the predicted surge in interest prompts a need for proactive exploration. Notable approaches, such as FedLLM[11], VFLAIR-LLM [137], FwdLLM [101] and FATE-LLM [100], have emerged using FL and split learning to address the challenges of training LLMs at the edge. Looking ahead, the increasing popularity of LLMs and diffusion models anticipates a growing interest in adapting them for edge learning. Furthermore, innovative techniques such as LoRA [415] and Fnet [416] offer potential solutions for the resource constraints on edge devices, especially when integrated with complementary approaches like FL 3.1.1, split learning 3.1.2, or model compression techniques 3.3. The convergence of these methodologies holds promise for overcoming challenges associated with training large models at the edge in the foreseeable future.

7.2.3 Extension to privacy preserving applications. The escalating popularity of ML applications has brought forth heightened concerns regarding the vast amounts of private and personal data

[11]FedLLM is a platform to Build LLMs on proprietary data with FL https://doc.fedml.ai/federate/fedllm

required for effective model training, raising questions about various legal and ethical implications. As discussed in earlier sections, edge learning emerges as a potential solution to address these privacy concerns, as it enables the training of ML models directly on the edge device, eliminating the need for sensitive data to traverse external networks. As such, the usage of edge learning for privacy-preserving applications is expected to be a pivotal research direction for the field. Domains like healthcare (5.1) often had legal requirements as well as ethical concerns of using the data for training ML model [417]. And as explored in discussed in previous sections (5.4), recommendation systems also stand out as a promising avenue for exploration because the significant scrutiny faced for their reliance on private data during model training [355, 356]. Additionally, other applications that require model personalization and tuning on private data ranging from spam detection in SMS and emails to word suggestions in keyboards and personal assistant chatbots or HAR, can benefit from edge learning, fostering a paradigm shift towards more ethical and privacy-conscious ML applications.

7.2.4 Reducing energy consumption and carbon footprint. The recent surge in distributed learning methodologies has raised concerns regarding their environmental impact. The substantial energy requirements for training models and data transfer to/from centralized data centers contribute to a significant carbon footprint [418, 419]. Recent trends in ML underscore the critical need to estimate and minimize the environmental impact from the training processes. According to [420], the estimated carbon footprint associated with edge devices by 2027 will be between 22 and 562 MtCO2-eq/year. Therefore, as a pressing research direction, there is a growing emphasis on developing techniques for edge learning that consider energy efficiency. Pioneering works, such as [421], have initiated analyses to quantify the environmental impact of ML in edge devices. Notably, frameworks like FedERA [382], designed for training FL models at the edge, incorporate a dedicated carbon emission tracker module to precisely estimate CO2 emissions during the local parameter update phase.

7.2.5 Frameworks to implement training. Despite the proliferation of frameworks and libraries aimed at enabling ML on edge devices, the current landscape lacks robust support for on-device training. Existing tools, such as PyTorch Mobile and ExecuTorch, predominantly emphasize inference on mobile/edge devices, neglecting the essential backpropagation algorithms crucial for the training phase. This imbalance in focus between training and inference highlights a critical gap in the current ecosystem. Although some tools have been proposed to facilitate on-device training on the edge (see Section 6), there is a pressing need for the development of new libraries, frameworks, and tools explicitly designed for edge learning.

8 CONCLUSION

This survey aims to provide a comprehensive overview of the vast field of edge learning, which involves training and fine-tuning of ML models at the edge. We have defined edge learning and its associated metrics and requirements, and explored various techniques and methodologies for optimizing ML training at the edge. Additionally, we explored the growing integration of ML types such as unsupervised learning, reinforcement learning, etc. and the different applications and use cases of edge learning. We have also examined the tools, libraries and frameworks used for edge learning. Furthermore, we have identified key challenges in edge learning and attempted to predict future trends and research directions.

Our analysis has revealed that distributed learning methods, including Federated Learning (FL), are gaining popularity for edge training. We have assessed the benefits and drawbacks of various techniques used to optimize the training at the edge and presented them in Section 3.5.2 and Table 2. We concluded that distributed techniques such as FL and split learning show great potential for

democratizing edge learning. On the other hand, adaptive and fine-tuning based techniques should be considered when possible as they often greatly improve the performance or reduce the training time on the edge significantly. Furthermore, model compression techniques are a great choice when slight decreases in performances are acceptable for a reduced model size and computational requirements. At last, we identified a growing trend towards combining different techniques to mitigate their limitations and maximize their benefits.

This survey has provided a broad understanding of edge learning, its requirements, challenges, use cases, and trends, as well as an overview of principal optimization techniques and tools. While it does not provide an in-depth evaluation and comparison of specific task performances, it serves as a reference for developing a foundational understanding of edge learning and identifying areas for future research.

ACKNOWLEDGMENTS

This research is supported by the Technology Innovation Institute, UAE under the research contract number TII/DSRC/2022/3143.

REFERENCES

[1] Nestor Maslej, Loredana Fattorini, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Helen Ngo, Juan Carlos Niebles, Vanessa Parli, et al. Artificial intelligence index report 2023. *arXiv preprint arXiv:2310.03715*, 2023.

[2] Sebastian Raschka, Yuxi (Hayden) Liu, and Vahid Mirjalili. *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python.* Packt Publishing Ltd, February 2022. ISBN 978-1-80181-638-0. Google-Books-ID: SVxaEAAAQBAJ.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. Publisher: Nature Publishing Group.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* MIT Press, November 2016. ISBN 978-0-262-33737-3. Google-Books-ID: omivDQAAQBAJ.

[5] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning.* Cambridge University Press, April 2020. ISBN 978-1-108-47004-9. Google-Books-ID: pFjPDwAAQBAJ.

[6] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019. doi: 10.1109/JPROC.2019.2921977.

[7] Xiaofei Wang, Yiwen Han, Victor C. M. Leung, Dusit Niyato, Xueqiang Yan, and Xu Chen. Convergence of Edge Computing and Deep Learning: A Comprehensive Survey. *IEEE Communications Surveys & Tutorials*, 22:869–904, 2020. ISSN 1553-877X, 2373-745X.

[8] Yuanming Shi, Kai Yang, Tao Jiang, Jun Zhang, and Khaled B. Letaief. Communication-Efficient Edge AI: Algorithms and Systems. *IEEE Communications Surveys & Tutorials*, 22(4):2167–2191, 2020. ISSN 1553-877X.

[9] Dianlei Xu, Tong Li, Yong Li, Xiang Su, Sasu Tarkoma, Tao Jiang, Jon Crowcroft, and Pan Hui. Edge intelligence: Architectures, challenges, and applications. *arXiv preprint arXiv:2003.12172*, 2020.

[10] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. Federated Learning in Mobile Edge Networks: A Comprehensive Survey, February 2020. arXiv:1909.11875.

[11] Joffre Luis Leon Veas, Luis Bryan Cordero Solis, Galo Enrique Valverde Landivar, and Miguel Angel Quiroz Martinez. Deep learning for edge computing: A survey. In *Artificial Intelligence, Computer and Software Engineering Advances*, pages 79–93, Cham, 2021. Springer International Publishing. ISBN 978-3-030-68080-0.

[12] Sauptik Dhar, Junyao Guo, Jiayi (Jason) Liu, Samarth Tripathi, Unmesh Kurup, and Mohak Shah. A survey of on-device machine learning: An algorithms and learning theory perspective. *ACM Trans. Internet Things*, 2(3), jul 2021. doi: 10.1145/3450494.

[13] A. Tak and S. Cherkaoui. Federated Edge Learning: Design Issues and Challenges. *IEEE Network*, 35(2):252–258, 2021. ISSN 0890-8044. doi: 10.1109/MNET.011.2000478.

[14] Jie Zhang, Zhihao Qu, Chenxi Chen, Haozhao Wang, Yufeng Zhan, Baoliu Ye, and Song Guo. Edge Learning: The Enabling Technology for Distributed Big Data Analytics in the Edge. *ACM Comput. Surv.*, 54(7):151:1–151:36, July 2021. ISSN 0360-0300. doi: 10.1145/3464419.

[15] M. G. Sarwar Murshed, Christopher Murphy, Daqing Hou, Nazar Khan, Ganesh Ananthanarayanan, and Faraz Hussain. Machine Learning at the Network Edge: A Survey. *ACM Computing Surveys*, 54(8):1–37, November 2022.

ISSN 0360-0300, 1557-7341. doi: 10.1145/3469029.

[16] Haftay Gebreslasie Abreha, Mohammad Hayajneh, and Mohamed Adel Serhani. Federated Learning in Edge Computing: A Systematic Survey. *Sensors*, 22:450, Jan 2022. ISSN 1424-8220. doi: 10.3390/s22020450.

[17] P. Boobalan, S.P. Ramu, Q.-V. Pham, K. Dev, S. Pandya, P.K.R. Maddikunta, T.R. Gadekallu, and T. Huynh-The. Fusion of Federated Learning and Industrial Internet of Things: A survey. *Computer Networks*, 212, 2022. doi: 10.1016/j.comnet.2022.109048.

[18] Praveen Joshi, Mohammed Hasanuzzaman, Chandra Thapa, Haithem Afli, and Ted Scully. Enabling all in-edge deep learning: A literature review. *IEEE Access*, 11:3431–3460, 2023. doi: 10.1109/ACCESS.2023.3234761.

[19] H. Cai, J. Lin, Y. Lin, Z. Liu, H. Tang, H. Wang, L. Zhu, and S. Han. Enable Deep Learning on Mobile Devices: Methods, Systems, and Applications. *ACM Transactions on Design Automation of Electronic Systems*, 27(3), 2022. ISSN 1084-4309. doi: 10.1145/3486618.

[20] Yiming Cui, Jiajia Guo, Xiangyi Li, Le Liang, and Shi Jin. Federated edge learning for the wireless physical layer: Opportunities and challenges. *China Communications*, 19(8):15–30, August 2022. ISSN 1673-5447. doi: 10.23919/JCC.2022.08.002.

[21] A. Imteaj, U. Thakker, S. Wang, J. Li, and M.H. Amini. A Survey on Federated Learning for Resource-Constrained IoT Devices. *IEEE Internet of Things Journal*, 9(1):1–24, 2022. ISSN 2327-4662. doi: 10.1109/JIOT.2021.3095077.

[22] Javier Mendez, Kay Bierzynski, M. P. Cuéllar, and Diego P. Morales. Edge Intelligence: Concepts, Architectures, Applications, and Future Directions. *ACM TRANSACTIONS ON EMBEDDED COMPUTING SYSTEMS*, 21(5):48:1–48:41, October 2022. ISSN 1539-9087. doi: 10.1145/3486674.

[23] Carlos Poncinelli Filho, Elias Marques, Victor Chang, Leonardo dos Santos, Flavia Bernardini, Paulo F. Pires, Luiz Ochi, and Flavia C. Delicato. A Systematic Literature Review on Distributed Machine Learning in Edge Computing. *Sensors*, 22(7):2665, January 2022. ISSN 1424-8220. doi: 10.3390/s22072665.

[24] Partha Pratim Ray. A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34(4):1595–1623, April 2022. ISSN 1319-1578. doi: 10.1016/j.jksuci.2021.11.019.

[25] Wenbin Li, Hakim Hacid, Ebtesam Almazrouei, and Merouane Debbah. A comprehensive review and a taxonomy of edge machine learning: Requirements, paradigms, and techniques. *AI*, 4(3):729–786, 2023. ISSN 2673-2688. doi: 10.3390/ai4030039.

[26] Haochen Hua, Yutong Li, Tonghe Wang, Nanqing Dong, Wei Li, and Junwei Cao. Edge Computing with Artificial Intelligence: A Machine Learning Perspective. *ACM Computing Surveys*, 55(9):184:1–184:35, Jan 2023. ISSN 0360-0300. doi: 10.1145/3555802.

[27] Shuai Zhu, Thiemo Voigt, JeongGil Ko, and Fatemeh Rahimian. On-device Training: A First Overview on Existing Systems, May 2023.

[28] Zhiyuan Wu, Sheng Sun, Yuwei Wang, Min Liu, Xuefeng Jiang, Runhan Li, and Bo Gao. Survey of Knowledge Distillation in Federated Edge Learning, Feb 2023.

[29] Kyle Hoffpauir, Jacob Simmons, Nikolas Schmidt, Rachitha Pittala, Isaac Briggs, Shanmukha Makani, and Yaser Jararweh. A Survey on Edge Intelligence and Lightweight Machine Learning Support for Future Applications and Services. *Journal of Data and Information Quality*, 15(2):20:1–20:30, June 2023. ISSN 1936-1955. doi: 10.1145/3581759.

[30] Vincenzo Barbuto, Claudio Savaglio, Min Chen, and Giancarlo Fortino. Disclosing edge intelligence: A systematic meta-survey. *Big Data and Cognitive Computing*, 7(1), 2023. ISSN 2504-2289.

[31] Silvana Trindade, Luiz F. Bittencourt, and Nelson L.S. da Fonseca. Resource management at the network edge for federated learning. *Digital Communications and Networks*, 10(3):765–782, 2024. ISSN 2352-8648. doi: https://doi.org/10.1016/j.dcan.2022.10.015.

[32] Piotr Grzesik and Dariusz Mrozek. Combining Machine Learning and Edge Computing: Opportunities, Challenges, Platforms, Frameworks, and Use Cases. *Electronics*, 13(3):640, January 2024. ISSN 2079-9292. doi: 10.3390/electronics13030640.

[33] Oumayma Jouini, Kaouthar Sethom, Abdallah Namoun, Nasser Aljohani, Meshari Huwaytim Alanazi, and Mohammad N. Alanazi. A survey of machine learning in edge computing: Techniques, frameworks, applications, issues, and research directions. *Technologies*, 12(6), 2024. ISSN 2227-7080. doi: 10.3390/technologies12060081.

[34] Jingke Tu, Lei Yang, and Jiannong Cao. Distributed Machine Learning in Edge Computing: Challenges, Solutions and Future Directions. *ACM Computing Surveys*, 57(5):132:1–132:37, January 2025. ISSN 0360-0300. doi: 10.1145/3708495.

[35] Francesco Piccialli, Diletta Chiaro, Pian Qi, Valerio Bellandi, and Ernesto Damiani. Federated and edge learning for large language models. *Information Fusion*, 117:102840, May 2025. ISSN 1566-2535. doi: 10.1016/j.inffus.2024.102840.

[36] Afonso Lourenço, João Rodrigo, João Gama, and Goreti Marreiros. On-device edge learning for IoT data streams: A survey, February 2025.

[37] Ninghui Jia, Zhihao Qu, Baoliu Ye, Yanyan Wang, Shihong Hu, and Song Guo. A Comprehensive Survey on Communication-Efficient Federated Learning in Mobile Edge Environments. *IEEE Communications Surveys & Tutorials*, 27(6):3710–3741, December 2025. ISSN 1553-877X. doi: 10.1109/COMST.2025.3535957.

[38] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. An overview on edge computing research. *IEEE Access*, 8: 85714–85728, 2020. doi: 10.1109/ACCESS.2020.2991734.
[39] Fang Liu, Guoming Tang, Youhuizi Li, Zhiping Cai, Xingzhou Zhang, and Tongqing Zhou. A Survey on Edge Computing Systems and Tools. *Proceedings of the IEEE*, 107(8):1537–1562, August 2019. ISSN 0018-9219, 1558-2256.
[40] S Ayyasamy. Edge computing research–a review. *Journal of Information Technology*, 5(1):62–74, 2023.
[41] Bingqian Lu, Jianyi Yang, and Shaolei Ren. Poster: Scaling up deep neural network optimization for edge inference. In *SEC*, pages 170–172, 2020. doi: 10.1109/SEC50012.2020.00025.
[42] Stephan Patrick Baller, Anshul Jindal, Mohak Chadha, and Michael Gerndt. Deepedgebench: Benchmarking deep neural networks on edge devices. In *IC2E*, pages 20–30, 2021.
[43] Carole-Jean Wu, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, Eldad Isaac, Yangqing Jia, Bill Jia, et al. Machine learning at facebook: Understanding inference at the edge. In *HPCA*, pages 331–344. IEEE, 2019. doi: 10.1109/HPCA.2019.00048.
[44] Colleen P. Bailey, Arthur C. Depoian, and Ethan R. Adams. Edge AI: Addressing the Efficiency Paradigm. In *2022 IEEE MetroCon*, pages 1–3, November 2022. doi: 10.1109/MetroCon56047.2022.9971140.
[45] Michael J. Kearns. *The Computational Complexity of Machine Learning*. MIT Press, 1990. ISBN 978-0-262-11152-2. Google-Books-ID: y5Txq1AkJoMC.
[46] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning. In *NeurIPS*, volume 33, pages 11285–11297, 2020.
[47] Ji Lin, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, and Song Han. On-device training under 256kb memory. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *NeurIPS*, volume 35, pages 22941–22954. Curran Associates, Inc., 2022.
[48] Arvin Tashakori, Wenwen Zhang, Z. Jane Wang, and Peyman Servati. Semipfl: Personalized semi-supervised federated learning framework for edge intelligence. *IEEE Internet of Things Journal*, 10(10):9161–9176, 2023.
[49] Kevin Afachao and Adnan M. Abu-Mahfouz. A review of intelligent iot devices at the edge. *ICAIoT*, pages 1–6, 2022.
[50] F. Bellotti, R. Berta, A. De Gloria, J. Doyle, and F. Sakr. Exploring Unsupervised Learning on STM32 F4 Microcontroller. In *Applications in Electronics Pervading Industry, Environment and Society*, volume 738, pages 39–46, 2021.
[51] W. Zhu and Z. Lu. Evaluation of time series clustering on embedded sensor platform. In *Euromicro DSD*, pages 187–191, 2021. doi: 10.1109/DSD53832.2021.00038.
[52] Mahmut Taha Yazici, Shadi Basurra, and Mohamed Medhat Gaber. Edge machine learning: Enabling smart internet of things applications. *Big Data and Cognitive Computing*, 2(3), 2018. ISSN 2504-2289. doi: 10.3390/bdcc2030026.
[53] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Aarti Singh and Jerry Zhu, editors, *AISTATS*, volume 54, pages 1273–1282, 20–22 Apr 2017.
[54] S. Abbas, A.A. Hejaili, G.A. Sampedro, M. Abisado, A.S. Almadhor, T. Shahzad, and K. Ouahada. A Novel Federated Edge Learning Approach for Detecting Cyberattacks in IoT Infrastructures. *IEEE Access*, 11:112189–112198, 2023.
[55] Z. Li, X. Cheng, J. Zhang, and B. Chen. Predicting Advanced Persistent Threats for IoT Systems Based on Federated Learning. In *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, volume 12382 LNCS, pages 76–89, 2021. doi: 10.1007/978-3-030-68851-6_5.
[56] J. Sidhpura, P. Shah, R. Veerkhare, and A. Godbole. FedSpam: Privacy Preserving SMS Spam Prediction. In *Neural Information Processing*, volume 1793 CCIS, pages 52–63. Springer Nature Singapore, 2023.
[57] S. Sriraman, S.S. Kannan, S. Ravishankar, and B. Bharathi. An On-device Federated Learning System for SMS Spam Classification. In *MIT URTC*, 2022. doi: 10.1109/URTC56832.2022.10002231.
[58] M. El Hanjri, H. Kabbaj, A. Kobbane, and A. Abouaomar. Federated Learning for Water Consumption Forecasting in Smart Cities. In *ICC*, pages 1798–1803, may 2023.
[59] Y. Shi, X. Li, and S. Chen. Towards Smart and Efficient Service Systems: Computational Layered Federated Learning Framework. *IEEE Network*, pages 1–8, 2023. doi: 10.1109/MNET.127.2200388.
[60] Y. Cheriguene, W. Jaafar, H. Yanikomeroglu, and C.A. Kerrache. Towards Reliable Participation in UAV-Enabled Federated Edge Learning on Non-IID Data. *IEEE Open Journal of Vehicular Technology*, 5:125–141, 2024.
[61] Zehua Sun, Yonghui Xu, Yong Liu, Wei He, Lanju Kong, Fangzhao Wu, Yali Jiang, and Lizhen Cui. A Survey on Federated Recommendation Systems. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2024. ISSN 2162-2388. doi: 10.1109/TNNLS.2024.3354924.
[62] Jian Li, Tongbao Chen, and Shaohua Teng. A comprehensive survey on client selection strategies in federated learning. *Computer Networks*, 251:110663, September 2024. ISSN 1389-1286. doi: 10.1016/j.comnet.2024.110663.
[63] Tianming Zang, Ce Zheng, Shiyao Ma, Chen Sun, and Wei Chen. A General Solution for Straggler Effect and Unreliable Communication in Federated Learning. In *ICC*, pages 1194–1199, May 2023. doi: 10.1109/ICC45041.2023.10279635.
[64] Bo Li, Mikkel N. Schmidt, Tommy S. Alstrøm, and Sebastian U. Stich. On the Effectiveness of Partial Variance Reduction in Federated Learning with Heterogeneous Data. In *CVPR*, pages 3964–3973, June 2023. doi: 10.1109/

CVPR52729.2023.00386.

[65] Pian Qi, Diletta Chiaro, Antonella Guzzo, Michele Ianni, Giancarlo Fortino, and Francesco Piccialli. Model aggregation techniques in federated learning: A comprehensive survey. *Future Generation Computer Systems*, 150:272–293, January 2024. ISSN 0167-739X. doi: 10.1016/j.future.2023.09.008.

[66] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach. In *NeurIPS*, volume 33, pages 3557–3568. Curran Associates, Inc., 2020.

[67] Shaoxiong Ji, Shirui Pan, Guodong Long, Xue Li, Jing Jiang, and Zi Huang. Learning Private Neural Language Modeling with Attentive Aggregation. In *IJCNN*, pages 1–8, July 2019. doi: 10.1109/IJCNN.2019.8852464.

[68] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates. In *ICML*, pages 5650–5659. PMLR, July 2018.

[69] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and Hugh Brendan McMahan. Adaptive Federated Optimization. In *ICLR*, October 2020.

[70] Y. Zeng, Y. Mu, J. Yuan, S. Teng, J. Zhang, J. Wan, Y. Ren, and Y. Zhang. Adaptive Federated Learning With Non-IID Data. *Computer Journal*, 66(11):2758–2772, 2023. doi: 10.1093/comjnl/bxac118.

[71] H. Gu, B. Guo, J. Wang, W. Sun, J. Liu, S. Liu, and Z. Yu. FedAux: An Efficient Framework for Hybrid Federated Learning. In *ICC*, pages 195–200, May 2022. doi: 10.1109/ICC45855.2022.9839129.

[72] B. Alhalabi, S. Basurra, and M.M. Gaber. FedNets: Federated Learning on Edge Devices Using Ensembles of Pruned Deep Neural Networks. *IEEE Access*, 11:30726–30738, 2023.

[73] H. Lv, Z. Zheng, T. Luo, F. Wu, S. Tang, L. Hua, R. Jia, and C. Lv. Data-Free Evaluation of User Contributions in Federated Learning. In *WiOpt*, 2021. doi: 10.23919/WiOpt52861.2021.9589136.

[74] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent. In *NeurIPS*, volume 30. Curran Associates, Inc., 2017.

[75] Yuncan Tang, Yongquan Liang, Yang Liu, Jinquan Zhang, Lina Ni, and Liang Qi. Reliable federated learning based on dual-reputation reverse auction mechanism in Internet of Things. *Future Generation Computer Systems*, 156:269–284, July 2024. ISSN 0167-739X. doi: 10.1016/j.future.2024.03.019.

[76] Linhao Meng, Yating Wei, Rusheng Pan, Shuyue Zhou, Jianwei Zhang, and Wei Chen. VADAF: Visualization for Abnormal Client Detection and Analysis in Federated Learning. *ACM Trans. Interact. Intell. Syst.*, 11(3-4):26:1–26:23, September 2021. ISSN 2160-6455. doi: 10.1145/3426866.

[77] Y. Sun, Y. Mao, and J. Zhang. MimiC: Combating Client Dropouts in Federated Learning by Mimicking Central Updates. *IEEE Transactions on Mobile Computing*, pages 1–13, 2023. doi: 10.1109/TMC.2023.3338021.

[78] Chen Dun, Mirian Hipolito, Chris Jermaine, Dimitrios Dimitriadis, and Anastasios Kyrillidis. Efficient and Light-Weight Federated Learning via Asynchronous Distributed Dropout. In *AISTATS*, pages 6630–6660. PMLR, April 2023.

[79] Hongbin Zhu, Yong Zhou, Hua Qian, Yuanming Shi, Xu Chen, and Yang Yang. Online Client Selection for Asynchronous Federated Learning With Fairness Consideration. *IEEE Transactions on Wireless Communications*, 22(4): 2493–2506, April 2023. ISSN 1558-2248. doi: 10.1109/TWC.2022.3211998.

[80] G.K. Gudur and S.K. Perepu. Zero-shot federated learning with new classes for audio classification. In *Proc. Interspeech 2021*, volume 2, pages 1041–1045, 2021. doi: 10.21437/Interspeech.2021-2264.

[81] Stefano Savazzi, Monica Nicoli, and Vittorio Rampa. Federated learning with cooperating devices: A consensus approach for massive iot networks. *IEEE Internet of Things Journal*, 7(5):4641–4654, 2020.

[82] Manupriya Gupta, Pavas Goyal, Rohit Verma, Rajeev Shorey, and Huzur Saran. Fedfm: Towards a robust federated learning approach for fault mitigation at the edge nodes. In *COMSNETS*, pages 362–370, 2022.

[83] A. Agrawal, D.D. Kulkarni, and S.B. Nair. On Decentralizing Federated Learning. In *SMC*, pages 1590–1595, oct 2020.

[84] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. SCAFFOLD: Stochastic Controlled Averaging for Federated Learning. In *ICML*, pages 5132–5143. PMLR, November 2020.

[85] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. Federated Learning with Non-IID Data. 2018. doi: 10.48550/arXiv.1806.00582.

[86] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated Optimization in Heterogeneous Networks. In *MLSys*, volume 2, pages 429–450, March 2020.

[87] L. Liu, J. Zhang, S.H. Song, and K.B. Letaief. Client-Edge-Cloud Hierarchical Federated Learning. In *ICC*, June 2020.

[88] M. Ma, L. Wu, W. Liu, N. Chen, Z. Shao, and Y. Yang. Data-aware Hierarchical Federated Learning via Task Offloading. In *GLOBECOM*, pages 3011–3016, 2022.

[89] Qinbin Li, Bingsheng He, and Dawn Song. Practical One-Shot Federated Learning for Cross-Silo Setting. In *IJCAI*, volume 2, pages 1484–1490, August 2021. doi: 10.24963/ijcai.2021/205.

[90] Yue Wang, Zhi Tian, Xin Fan, Yan Huo, Cameron Nowzari, and Kai Zeng. Distributed Swarm Learning for Internet of Things at the Edge: Where Artificial Intelligence Meets Biological Intelligence, oct 2022.

[91] Liangqi Yuan, Ziran Wang, Lichao Sun, Philip S. Yu, and Christopher G. Brinton. Decentralized Federated Learning: A Survey and Perspective. *IEEE Internet of Things Journal*, 11(21):34617–34638, November 2024. ISSN 2327-4662. doi: 10.1109/JIOT.2024.3407584.

[92] István Hegedűs, Gábor Danner, and Márk Jelasity. Gossip Learning as a Decentralized Alternative to Federated Learning. In José Pereira and Laura Ricci, editors, *Distributed Applications and Interoperable Systems*, Lecture Notes in Computer Science, pages 74–90, Cham, 2019. ISBN 978-3-030-22496-7.

[93] Dawei Xu, Chentao Lu, Chunhai Li, Chuan Zhang, Yongwei Tang, and Liehuang Zhu. Communication-Efficient Federated Swarm Learning for SAG-CM on Heterogeneous Data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 19:11940–11956, 2026. ISSN 2151-1535. doi: 10.1109/JSTARS.2026.3670968.

[94] K. Boubouh, R. Basmadjian, O. Ardakanian, A. Maurer, and R. Guerraoui. PePTM: An Efficient and Accurate Personalized P2P Learning Algorithm for Home Thermal Modeling. *Energies*, 16(18), 2023. doi: 10.3390/en16186594.

[95] J. Calo and B. Lo. Federated Blockchain Learning at the Edge. *Information*, 14(6), 2023. doi: 10.3390/info14060318.

[96] M.M. Amiri, T.M. Duman, D. Gunduz, S.R. Kulkarni, and H.V.P. Poor. Blind Federated Edge Learning. *IEEE Transactions on Wireless Communications*, 20(8):5129–5143, 2021.

[97] Bingnan Xiao, Xichen Yu, Wei Ni, Xin Wang, and H. Vincent Poor. Over-the-air federated learning: Status quo, open challenges, and future directions. *Fundamental Research*, 5(4):1710–1724, July 2025. ISSN 2667-3258. doi: 10.1016/j.fmre.2024.01.011.

[98] K.-Y. Liang, A. Srinivasan, and J.C. Andresen. Modular Federated Learning. In *IJCNN*, July 2022.

[99] Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8): 3710–3722, 2021. doi: 10.1109/TNNLS.2020.3015958.

[100] Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan, and Qiang Yang. Fate-llm: A industrial grade federated learning framework for large language models. *arXiv preprint arXiv:2310.10049*, 2023.

[101] Mengwei Xu, Dongqi Cai, Yaozong Wu, Xiang Li, and Shangguang Wang. Fwdllm: Efficient fedllm using forward gradient, 2024.

[102] A. Ranjan and B.C. Sahana. Efficient Federated Fine-Tuning via Zeroth-Order Optimization for Resource-Constrained Edge Devices. In *IEEE Int. Conf. Comput. Intell. Commun. Networks, CICN*, pages 1935–1940. Institute of Electrical and Electronics Engineers Inc., 2025. ISBN 979-833158733-8 (ISBN). doi: 10.1109/CICN67655.2025.11368000.

[103] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[104] Yuanyishu Tian, Yao Wan, Lingjuan Lyu, Dezhong Yao, Hai Jin, and Lichao Sun. FedBERT: When Federated Learning Meets Pre-training. *ACM Transactions on Intelligent Systems and Technology*, 13(4):66:1–66:26, August 2022. ISSN 2157-6904. doi: 10.1145/3510033.

[105] Zhengyang Lit, Shijing Sit, Jianzong Wang, and Jing Xiao. Federated split bert for heterogeneous text classification. In *IJCNN*, pages 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9892845.

[106] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[107] Jiang Tao, Zhen Gao, and Zhaohui Guo. Training Vision Transformers in Federated Learning with Limited Edge-Device Resources. *Electronics*, 11:2638, Jan 2022. ISSN 2079-9292. doi: 10.3390/electronics11172638.

[108] T.-M.H. Hsu, H. Qi, and M. Brown. Federated Visual Classification with Real-World Data Distribution. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12355 LNCS, pages 76–92, 2020. doi: 10.1007/978-3-030-58607-2_5.

[109] J. Jia, J. Mahadeokar, W. Zheng, Y. Shangguan, O. Kalinli, and F. Seide. Federated Domain Adaptation for ASR with Full Self-Supervision. In *INTERSPEECH*, pages 536–540, Seb 2022. doi: 10.21437/Interspeech.2022-803.

[110] D. Guliani, F. Beaufays, and G. Motta. Training speech recognition models with federated learning: A quality/cost framework. In *ICASSP*, pages 3080–3084, June 2021.

[111] Tuo Zhang, Tiantian Feng, Samiul Alam, Sunwoo Lee, Mi Zhang, Shrikanth S Narayanan, and Salman Avestimehr. Fedaudio: A federated learning benchmark for audio tasks. In *ICASSP*, pages 1–5. IEEE, 2023.

[112] Muhammad Ayat Hidayat, Yugo Nakamura, Billy Dawton, and Yutaka Arakawa. AGC-DP: Differential Privacy with Adaptive Gaussian Clipping for Federated Learning. In *MDM*, pages 199–208, July 2023. doi: 10.1109/MDM58254.2023.00042. ISSN: 2375-0324.

[113] Cynthia Dwork. Differential Privacy: A Survey of Results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer. ISBN 978-3-540-79228-4. doi: 10.1007/978-3-540-79228-4_1.

[114] Yi Zhang, Yunfan Lu, and Fengxia Liu. A Systematic Survey for Differential Privacy Techniques in Federated Learning. *Journal of Information Security*, 14(2):111–135, February 2023. doi: 10.4236/jis.2023.142008.

[115] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor. Federated Learning With Differential Privacy: Algorithms and Performance Analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469, 2020. ISSN 1556-6021. doi: 10.1109/TIFS.2020.2988575.

[116] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *SIGSAC*, CCS '17, pages 1175–1191, New York, NY, USA, October 2017. Association for Computing Machinery. ISBN 978-1-4503-4946-8. doi: 10.1145/3133956.3133982.

[117] Xing Zhang, Yuexiang Luo, and Tianning Li. A Review of Research on Secure Aggregation for Federated Learning. *Future Internet*, 17(7):308, July 2025. ISSN 1999-5903. doi: 10.3390/fi17070308.

[118] H. Zhou, H. Dai, G. Yang, and Y. Xiang. Robust Privacy-Preserving Federated Learning for Edge Computing With New Client Integration. *IEEE Transactions on Dependable and Secure Computing*, 2026. ISSN 15455971 (ISSN). doi: 10.1109/TDSC.2026.3651107.

[119] J. Jeniston Moses and S. Sophia. Federated Deep Learning for Privacy Preserving Diagnosis of Skin Diseases using Mobile Devices. In *Proc. Int. Conf. Intell. Cyber Phys. Syst. Internet Things, ICoICI*, pages 383–387. Institute of Electrical and Electronics Engineers Inc., 2025. ISBN 978-166545753-8 (ISBN). doi: 10.1109/ICoICI65217.2025.11253919.

[120] W. Li, Y. Xiang, J. Liu, and G. Chen. ESA-UFEL: Efficient and Secure Aggregation Framework for UAV-Assisted Federated Edge Learning. In *Proc. IEEE Int. Conf. Safe Prod. Informatiz., IICSPI*, pages 281–286. Institute of Electrical and Electronics Engineers Inc., 2025. ISBN 979-833153878-1 (ISBN). doi: 10.1109/IICSPI66775.2025.11437751.

[121] Y. Ding, W. Lu, K. Feng, Y. Gao, and B. Ren. Privacy-Aware Resource Collaboration for Secure UAV-Assisted Federated Edge Learning Systems. *IEEE Journal on Selected Areas in Communications*, 44:3678–3689, 2026. ISSN 07338716 (ISSN). doi: 10.1109/JSAC.2026.3669101.

[122] Z. Li, S. Zhang, and X. Zhou. FL-Transformer: A Privacy-Preserving Approach for Predictive Maintenance of Distributed Industrial Equipment Using Multi-Sensor Edge Intelligence. In *Int. Symp. Robot., Artif. Intell. Inf. Eng., RAIIE*, pages 354–359. Institute of Electrical and Electronics Engineers Inc., 2025. ISBN 979-833152452-4 (ISBN). doi: 10.1109/RAIIE65740.2025.11140579.

[123] Lingyun Wang, Hanlin Zhou, Yinwei Bao, Xiaoran Yan, Guojiang Shen, and Xiangjie Kong. Horizontal Federated Recommender System: A Survey. *ACM Comput. Surv.*, 56(9):240:1–240:42, May 2024. ISSN 0360-0300. doi: 10.1145/3656165.

[124] Chandra Thapa, Pathum Chamikara Mahawaga Arachchige, Seyit Camtepe, and Lichao Sun. SplitFed: When federated learning meets split learning. In *AAAI*, volume 36, pages 8485–8493, 2022. doi: 10.1609/aaai.v36i8.20825.

[125] D. Zou, X. Liu, L. Sun, J. Duan, R. Li, Y. Xu, W. Li, and S. Lu. FedMC: Federated Reinforcement Learning on the Edge with Meta-Critic Networks. In *IPCCC*, pages 344–351, 2022. doi: 10.1109/IPCCC55026.2022.9894336.

[126] S. Yue, J. Ren, J. Xin, D. Zhang, Y. Zhang, and W. Zhuang. Efficient Federated Meta-Learning over Multi-Access Wireless Networks. *IEEE Journal on Selected Areas in Communications*, 40(5):1556–1570, 2022. ISSN 0733-8716. doi: 10.1109/JSAC.2022.3143259.

[127] Ehsan Tanghatari, Mehdi Kamal, Ali Afzali-Kusha, and Massoud Pedram. Federated learning by employing knowledge distillation on edge devices with limited hardware resources. *Neurocomputing*, 531:87–99, April 2023. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.02.011.

[128] X. Qu, J. Wang, and J. Xiao. Quantization and Knowledge Distillation for Efficient Federated Learning on Edge Devices. In *HPCC-SmartCity-DSS*, pages 967–972, 2020. doi: 10.1109/HPCC-SmartCity-DSS50907.2020.00129.

[129] L. Liu, J. Zhang, S. Song, and K.B. Letaief. Hierarchical Federated Learning with Quantization: Convergence Analysis and System Design. *IEEE Transactions on Wireless Communications*, 22(1):2–18, 2023.

[130] Jihyeon Ryu, Dongho Won, and Youngsook Lee. A Study of Split Learning Model. In *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–4, January 2022. doi: 10.1109/IMCOM53663.2022.9721798.

[131] Bernardo Camajori Tedeschini, Mattia Brambilla, and Monica Nicoli. Split Consensus Federated Learning: An Approach for Distributed Training and Inference. *IEEE Access*, 12:119535–119549, 2024. ISSN 2169-3536. doi: 10.1109/ACCESS.2024.3446577.

[132] A. Ayad, M. Renner, and A. Schmeink. Improving the Communication and Computation Efficiency of Split Learning for IoT Applications. In *GLOBECOM*, 2021. ISBN 978-1-72818-104-2. doi: 10.1109/GLOBECOM46510.2021.9685493.

[133] Ngoc Duy Pham and Naveen Chilamkurti. Data Leakage Threats and Protection in Split Learning: A Survey. In *Proceedings of the 2023 International Conference on Intelligent Computing and Its Emerging Applications*, ICEA '23, pages 141–147, New York, NY, USA, December 2024. Association for Computing Machinery. ISBN 979-8-4007-0905-0. doi: 10.1145/3659154.3659189.

[134] Kamalesh Palanisamy, Vivek Khimani, Moin Hussain Moti, and Dimitris Chatzopoulos. SplitEasy: A Practical Approach for Training ML models on Mobile Devices. In *HotMobile*, pages 37–43, Feb 2021. doi: 10.1145/3446382.3448362.

[135] S. Liu, L. Xin, X. Lyu, and C. Ren. Masking-enabled Data Protection Approach for Accurate Split Learning. In *WCNC*, March 2023. ISBN 978-1-66549-122-8. ISSN: 1525-3511.

[136] S. Fu, F. Dong, D. Shen, and Q. He. Joint Quality Evaluation, Model Splitting and Resource Provisioning for Split Edge Learning. In *SECON*, pages 420–428, Seb 2023. ISBN 9798350300529. ISSN: 2155-5486.

[137] Zixuan Gu, Qiufeng Fan, Long Sun, Yang Liu, and Xiaojun Ye. VFLAIR-LLM: A Comprehensive Framework and Benchmark for Split Learning of LLMs. In *SIGKDD*, KDD '25, pages 5470–5481, New York, NY, USA, August 2025. Association for Computing Machinery. ISBN 979-8-4007-1454-2. doi: 10.1145/3711896.3737411.

[138] Ayush Chopra, Surya Kant Sahu, Abhishek Singh, Abhinav Java, Praneeth Vepakomma, Mohammad Mohammadi Amiri, and Ramesh Raskar. Adaptive Split Learning. In *Federated Learning Systems (FLSys) Workshop @ MLSys 2023*, July 2023.

[139] Ayush Chopra, Surya Kant Sahu, Abhishek Singh, Abhinav Java, Praneeth Vepakomma, Vivek Sharma, and Ramesh Raskar. AdaSplit: Adaptive Trade-offs for Resource-constrained Distributed Deep Learning, Dec 2021.

[140] Z. Cheng, X. Xia, M. Liwang, X. Fan, Y. Sun, X. Wang, and L. Huang. CHEESE: Distributed Clustering-Based Hybrid Federated Split Learning Over Edge Networks. *IEEE Transactions on Parallel and Distributed Systems*, 34(12):3174–3191, 2023. ISSN 1045-9219. doi: 10.1109/TPDS.2023.3322755.

[141] Jiaqi Xia, Meng Wu, and Pengyong Li. SHE-SFL: An efficient and privacy-preserving heterogeneous federated split learning architecture based on homomorphic encryption. *Future Generation Computer Systems*, 175:108101, February 2026. ISSN 0167-739X. doi: 10.1016/j.future.2025.108101.

[142] S. Fu, F. Dong, D. Shen, and T. Lu. Privacy-preserving model splitting and quality-aware device association for federated edge learning. *Software - Practice and Experience*, 2023. ISSN 0038-0644.

[143] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[144] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A Survey on Deep Transfer Learning. In Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, and Ilias Maglogiannis, editors, *ICANN*, pages 270–279, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01424-7.

[145] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, 109(1):43–76, January 2021. ISSN 1558-2256. doi: 10.1109/JPROC.2020.3004555.

[146] L. Yang, A.S. Rakin, and D. Fan. RepNet: Efficient On-Device Learning via Feature Reprogramming. In *CVPR*, pages 12267–12276, June 2022. doi: 10.1109/CVPR52688.2022.01196.

[147] H.-Y. Chiang, N. Frumkin, F. Liang, and D. Marculescu. MobileTL: On-Device Transfer Learning with Inverted Residual Blocks. In *AAAI*, volume 37, pages 7166–7174, 2023.

[148] B. Yang, O. Fagbohungbe, X. Cao, C. Yuen, L. Qian, D. Niyato, and Y. Zhang. A Joint Energy and Latency Framework for Transfer Learning over 5G Industrial Edge Networks. *IEEE Transactions on Industrial Informatics*, 18(1):531–541, 2022. doi: 10.1109/TII.2021.3075444.

[149] Seungkyu Choi, Jaekang Shin, and Lee-Sup Kim. Accelerating on-device dnn training workloads via runtime convergence monitor. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(5):1574–1587, 2022.

[150] K.M. Ahmed, A. Imteaj, and M.H. Amini. Federated Deep Learning for Heterogeneous Edge Computing. In *ICMLA*, pages 1146–1152, 2021. doi: 10.1109/ICMLA52953.2021.00187.

[151] J. Suzuki, S.F. Lameh, and Y. Amannejad. Using Transfer Learning in Building Federated Learning Models on Edge Devices. In *IDSTA*, pages 105–113, 2021. doi: 10.1109/IDSTA53674.2021.9660819.

[152] S. Liu, S. Xu, W. Yu, Z. Fu, Y. Zhang, and A. Marian. FedCT: Federated Collaborative Transfer for Recommendation. In *SIGIR*, pages 716–725, 2021. doi: 10.1145/3404835.3462825.

[153] D. Vucetic, M. Tayaranian, M. Ziaeefard, J.J. Clark, B.H. Meyer, and W.J. Gross. Efficient Fine-Tuning of BERT Models on the Edge. In *ISCAS*, pages 1838–1842, May 2022.

[154] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large Scale Incremental Learning. In *CVPR*, pages 374–382, 2019.

[155] Jingwei Zuo, George Arvanitakis, and Hakim Hacid. On Handling Catastrophic Forgetting for Incremental Learning of Human Physical Activity on the Edge, Feb 2023.

[156] Guangyuan SHI, JIAXIN CHEN, Wenlong Zhang, Li-Ming Zhan, and Xiao-Ming Wu. Overcoming Catastrophic Forgetting in Incremental Few-Shot Learning by Finding Flat Minima. In *NeurIPS*, volume 34, pages 6747–6761, 2021.

[157] Huong-Giang Doan, Hong-Quan Luong, Thi-Oanh Ha, and Thi Thanh Thuy Pham. An Efficient Strategy for Catastrophic Forgetting Reduction in Incremental Learning. *Electronics*, 12:2265, Jan 2023. ISSN 2079-9292. doi: 10.3390/electronics12102265.

[158] Qing Yang, Yudi Gu, and Dongsheng Wu. Survey of incremental learning. In *CCDC*, pages 399–404. IEEE, 2019.

[159] Muhammad Awais Hussain, Shih-An Huang, and Tsung-Han Tsai. Learning With Sharing: An Edge-Optimized Incremental Learning Method for Deep Neural Networks. *IEEE Transactions on Emerging Topics in Computing*, 11(2): 461–473, April 2023. ISSN 2168-6750. doi: 10.1109/TETC.2022.3210905.

[160] S. Disabato and M. Roveri. Incremental On-Device Tiny Machine Learning. In *AIChallengeIoT*, pages 7–13, 2020. doi: 10.1145/3417313.3429378.

[161] J. Liu, Z. Xie, D. Nikolopoulos, and D. Li. RIANN: Real-time incremental learning with approximate nearest neighbor on mobile devices. In *OpML*, 2020.

[162] Dawei Li, Serafettin Tasci, Shalini Ghosh, Jingwen Zhu, Junting Zhang, and Larry Heck. RILOD: near real-time incremental learning for object detection at the edge. In *SEC*, pages 113–126, NY, USA, November 2019. ACM. ISBN 978-1-4503-6733-2. doi: 10.1145/3318216.3363317.

[163] M. Rao, G. Chennupati, G. Tiwari, A. Kumar Sahu, A. Raju, A. Rastrow, and J. Droppo. Federated Self-Learning with Weak Supervision for Speech Recognition. In *ICASSP*, 2023.

[164] Sheng Yue, Ju Ren, Jiang Xin, Sen Lin, and Junshan Zhang. Inexact-ADMM Based Federated Meta-Learning for Fast and Continual Edge Learning. In *MobiHoc*, pages 91–100, July 2021. doi: 10.1145/3466772.3467038.

[165] Y. Luo, Z. Huang, Z. Zhang, Z. Wang, M. Baktashmotlagh, and Y. Yang. Learning from the past: Continual meta-learning with bayesian graph neural networks. In *AAAI*, pages 5021–5028, 2020. ISBN 978-1-57735-835-0.

[166] Ze-Han Wang, Zhenli He, Hui Fang, Yi-Xiong Huang, Ying Sun, Yu Yang, Zhi-Yuan Zhang, and Di Liu. Efficient On-Device Incremental Learning by Weight Freezing. In *ASP-DAC*, pages 538–543, Jan 2022. doi: 10.1109/ASP-DAC52403.2022.9712563.

[167] Antonio Carta, Andrea Cossu, Vincenzo Lomonaco, Davide Bacciu, and Joost van de Weijer. Projected Latent Distillation for Data-Agnostic Consolidation in Distributed Continual Learning, March 2023. arXiv:2303.15888.

[168] X. Zhang, H. Li, X. Chen, and X. Liu. Impact Patterns of Combining Model Pruning and Continual Learning on Model Performance. In *CogMI*, pages 27–33, 2021. doi: 10.1109/CogMI52975.2021.00013.

[169] Z. Wang, Z. Zhan, Y. Gong, G. Yuan, W. Niu, T. Jian, B. Ren, S. Ioannidis, Y. Wang, and J. Dy. SparCL: Sparse Continual Learning on the Edge. In *NeurIPS*, volume 35, 2022.

[170] Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms, Oct 2018.

[171] Mike Huisman, Jan N. van Rijn, and Aske Plaat. A survey of deep meta-learning. *Artificial Intelligence Review*, 54(6): 4483–4541, August 2021. ISSN 1573-7462. doi: 10.1007/s10462-021-10004-4.

[172] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-Learning in Neural Networks: A Survey, November 2020. arXiv:2004.05439.

[173] Zhongnan Qu, Zimu Zhou, Yongxin Tong, and Lothar Thiele. p-Meta: Towards On-device Deep Model Adaptation. In *SIGKDD*, pages 1441–1451, August 2022. doi: 10.1145/3534678.3539293. arXiv:2206.12705.

[174] B. Rosenfeld, B. Rajendran, and O. Simeone. Fast on-device adaptation for spiking neural networks via online-within-online meta-learning. In *2021 IEEE Data Science and Learning Workshop, DSLW 2021*, 2021. ISBN 978-1-66542-825-5. doi: 10.1109/DSLW51110.2021.9523405.

[175] D. Gao, X. He, Z. Zhou, Y. Tong, and L. Thiele. Pruning Meta-Trained Networks for On-Device Adaptation. In *CIKM*, pages 514–523, 2021. ISBN 978-1-4503-8446-9. doi: 10.1145/3459637.3482378.

[176] Feng Yu, Hui Lin, Xiaoding Wang, Sahil Garg, Georges Kaddoum, Satinder Singh, and Mohammad Mehedi Hassan. Communication-Efficient Personalized Federated Meta-Learning in Edge Networks. *IEEE Transactions on Network and Service Management*, 20(2):1558–1571, June 2023. ISSN 1932-4537. doi: 10.1109/TNSM.2023.3263831.

[177] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[178] H. Nam, J. Park, and S.-L. Kim. Active Wireless Split Learning via Online Cloud-Local Server Delta-Knowledge Distillation. In *ICC Workshops: Sustainable Communications for Renaissance*, pages 825–830, 2023. ISBN 9798350333077. doi: 10.1109/ICCWorkshops57953.2023.10283579.

[179] X. Xia, H. Yin, J. Yu, Q. Wang, G. Xu, and Q.V.H. Nguyen. On-Device Next-Item Recommendation with Self-Supervised Knowledge Distillation. In *SIGIR*, pages 546–555, 2022. doi: 10.1145/3477495.3531775.

[180] Y.-G. Qian, J. Ma, N.-N. He, B. Wang, Z.-Q. Gu, X. Ling, and S. Wassim. Two-stage Adversarial Knowledge Transfer for Edge Intelligence. *Ruan Jian Xue Bao/Journal of Software*, 33(12):4504–4516, 2022. ISSN 1000-9825. doi: 10.13328/j.cnki.jos.006352.

[181] Y. Zhou, X. Ma, D. Wu, and X. Li. Communication-Efficient and Attack-Resistant Federated Edge Learning with Dataset Distillation. *IEEE Transactions on Cloud Computing*, 11(3):2517–2528, 2023. doi: 10.1109/TCC.2022.3215520.

[182] C. Bai, X. Cui, and A. Li. Robust speech recognition model using multi-source federal learning after distillation and deep edge intelligence. In *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/2033/1/012158. Issue: 1.

[183] A. Hard, K. Partridge, N. Chen, S. Augenstein, A. Shah, H.J. Park, A. Park, S. Ng, J. Nguyen, I.L. Moreno, R. Mathews, and F. Beaufays. Production federated keyword spotting via distillation, filtering, and joint federated-centralized training. In *INTERSPEECH*, pages 76–80, Seb 2022. doi: 10.21437/Interspeech.2022-11050.

[184] S. Oh, J. Park, E. Jeong, H. Kim, M. Bennis, and S.-L. Kim. Mix2FLD: Downlink Federated Learning after Uplink Federated Distillation with Two-Way Mixup. *IEEE Communications Letters*, 24(10):2211–2215, 2020. doi: 10.1109/LCOMM.2020.3003693.

[185] J.-H. Ahn, O. Simeone, and J. Kang. Wireless Federated Distillation for Distributed Edge Learning with Heterogeneous Data. In *PIMRC*, Seb 2019. doi: 10.1109/PIMRC.2019.8904164.

[186] D.P. Nguyen, S. Yu, J.P. Muñoz, and A. Jannesari. Enhancing Heterogeneous Federated Learning with Knowledge Extraction and Multi-Model Fusion. In *ACM International Conference Proceeding Series*, pages 36–43, 2023. doi: 10.1145/3624062.3626325.

[187] Ilai Bistritz, Ariana Mann, and Nicholas Bambos. Distributed Distillation for On-Device Learning. In *NeurIPS*, volume 33, pages 22593–22604. Curran Associates, Inc., 2020.

[188] X. Xia, J. Yu, Q. Wang, C. Yang, N.Q.V. Hung, and H. Yin. Efficient On-Device Session-Based Recommendation. *ACM Transactions on Information Systems*, 41(4), 2023. doi: 10.1145/3580364.

[189] J. Yao, F. Wang, K. Jia, B. Han, J. Zhou, and H. Yang. Device-Cloud Collaborative Learning for Recommendation. In *SIGKDD*, pages 3865–3874, 2021. doi: 10.1145/3447548.3467097.

[190] Junhua Wong, Jeanne Nerbonne, and Qingxue Zhang. Ultra-efficient edge cardiac disease detection towards real-time precision health. *IEEE Access*, pages 1–1, 2023. doi: 10.1109/ACCESS.2023.3346893.

[191] I. Jang, H. Kim, D. Lee, Y.-S. Son, and S. Kim. Knowledge Transfer for On-Device Deep Reinforcement Learning in Resource Constrained Edge Computing Systems. *IEEE Access*, 8:146588–146597, 2020. doi: 10.1109/ACCESS.2020.3014922.

[192] Mohamed Riad Sebti, Andrea Accettola, Riccardo Carotenuto, and Massimo Merenda. Dataset Distillation Technique Enabling ML On-board Training: Preliminary Results. In Carmine Ciofi and Ernesto Limiti, editors, *SIE*, Lecture Notes in Electrical Engineering, pages 379–384, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-48711-8. doi: 10.1007/978-3-031-48711-8_46.

[193] Andrea Giovanni Accettola and Massimo Merenda. Dataset distillation as an enabling technique for on-device training in TinyML for IoT: an RFID use case. In *SpliTech*, pages 1–4, June 2023. doi: 10.23919/SpliTech58164.2023.10193138.

[194] H. Hu, S.M. Siniscalchi, C.-H.H. Yang, and C.-H. Lee. A VARIATIONAL Bayesian APPROACH TO LEARNING LATENT VARIABLES FOR ACOUSTIC KNOWLEDGE TRANSFER. In *ICASSP*, pages 1041–1045, May 2022. ISBN 978-1-66540-540-9. ISSN: 1520-6149.

[195] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[196] Alicja Kwasniewska, Maciej Szankin, Mateusz Ozga, Jason Wolfe, Arun Das, Adam Zajac, Jacek Ruminski, and Paul Rad. Deep Learning Optimization for Edge Devices: Analysis of Training Quantization Parameters. In *IECON*, volume 1, pages 96–101, Oct 2019. doi: 10.1109/IECON.2019.8927153.

[197] M.H. Ostertag, S. Al-Doweesh, and T. Rosing. Efficient Training on Edge Devices Using Online Quantization. In *DATE*, pages 1011–1014, 2020. ISBN 978-3-9819263-4-7. doi: 10.23919/DATE48585.2020.9116568.

[198] Yangchen Li, Ying Cui, and Vincent Lau. An Optimization Framework for Federated Edge Learning. *IEEE Transactions on Wireless Communications*, 22:934–949, Feb 2023. ISSN 1536-1276.

[199] S. Choi, J. Shin, Y. Choi, and L.-S. Kim. An optimized design technique of low-bit neural network training for personalization on IoT devices. In *DAC*, 2019. ISBN 978-1-4503-6725-7. ISSN: 0738-100X.

[200] Y. Chen, C. Hawkins, K. Zhang, Z. Zhang, and C. Hao. 3U-EdgeAI: Ultra-Low Memory Training, Ultra-Low Bitwidth Quantization, and Ultra-Low Latency Acceleration. In *GLSVLSI*, pages 157–162, 2021. ISBN 978-1-4503-8393-6.

[201] H. Li, R. Wang, W. Zhang, and J. Wu. One Bit Aggregation for Federated Edge Learning with Reconfigurable Intelligent Surface: Analysis and Optimization. *IEEE Transactions on Wireless Communications*, 22(2):872–888, 2023.

[202] G. Zhu, Y. Du, D. Gündüz, and K. Huang. One-Bit Over-the-Air Aggregation for Communication-Efficient Federated Edge Learning: Design and Convergence Analysis. *IEEE Transactions on Wireless Communications*, 20(3):2120–2135, 2021.

[203] Heju Li, Rui Wang, Jun Wu, and Wei Zhang. Federated edge learning via reconfigurable intelligent surface with one-bit quantization. In *GLOBECOM*, pages 1055–1060. IEEE, 2022.

[204] Y. Cui, J. Guo, C. Wen, and S. Jin. Communication-efficient Personalized Federated Edge Learning for Massive MIMO CSI Feedback. *IEEE Transactions on Wireless Communications*, pages 1–1, 2023.

[205] R. Chen, L. Li, K. Xue, C. Zhang, M. Pan, and Y. Fang. Energy Efficient Federated Learning Over Heterogeneous Mobile Devices via Joint Design of Weight Quantization and Wireless Transmission. *IEEE Transactions on Mobile Computing*, 22(12):7451–7465, 2023. doi: 10.1109/TMC.2022.3213766.

[206] Peixi Liu, Jiamo Jiang, Guangxu Zhu, Lei Cheng, Wei Jiang, Wu Luo, Ying Du, and Zhiqin Wang. Training time minimization for federated edge learning with optimized gradient quantization and bandwidth allocation. *Frontiers of Information Technology & Electronic Engineering*, 23(8):1247–1263, August 2022. ISSN 2095-9230.

[207] Jagmohan Chauhan, Young D. Kwon, and Cecilia Mascolo. Exploring On-Device Learning Using Few Shots for Audio Classification. In *EUSIPCO*, pages 424–428, August 2022. doi: 10.23919/EUSIPCO55093.2022.9909551. ISSN: 2076-1465.
[208] Y. Yamagishi, T. Kaneko, M. Akai-Kasaya, and T. Asai. Holmes: A Hardware-Oriented Optimizer Using Logarithms. *IEICE Transactions on Information and Systems*, E105D(12):2040–2047, 2022. ISSN 0916-8532.
[209] Xinlei Zhou and Dasen Yan. Model tree pruning. *International Journal of Machine Learning and Cybernetics*, 10: 3431–3444, 2019.
[210] Woosuk Kwon, Sehoon Kim, Michael W Mahoney, Joseph Hassoun, Kurt Keutzer, and Amir Gholami. A fast post-training pruning framework for transformers. *NeurIPS*, 35:24101–24116, 2022.
[211] Seungkyu Choi, Jaekang Shin, and Lee-Sup Kim. A convergence monitoring method for dnn training of on-device task adaptation. In *ICCAD*, pages 1–9. IEEE, 2021.
[212] Z. Ren, W. Fang, W. Xu, Z. Li, and Y. Hu. Research on Lightweight Model Training Technology of Federated Learning for Railway Defect Detection. *Tiedao Xuebao/Journal of the China Railway Society*, 45(4):77–83, 2023.
[213] S. Yu, P. Nguyen, A. Anwar, and A. Jannesari. Heterogeneous Federated Learning using Dynamic Model Pruning and Adaptive Gradient. In *CCGrid*, pages 322–330, 2023. doi: 10.1109/CCGrid57682.2023.00038.
[214] Y. Jiang, S. Wang, V. Valls, B.J. Ko, W.-H. Lee, K.K. Leung, and L. Tassiulas. Model Pruning Enables Efficient Federated Learning on Edge Devices. *IEEE Transactions on Neural Networks and Learning Systems*, 34(12):10374–10386, 2023. doi: 10.1109/TNNLS.2022.3166101.
[215] S. Liu, G. Yu, R. Yin, J. Yuan, L. Shen, and C. Liu. Joint Model Pruning and Device Selection for Communication-Efficient Federated Edge Learning. *IEEE Transactions on Communications*, 70(1):231–244, 2022.
[216] N. Mairittha, T. Mairittha, and S. Inoue. On-device deep personalization for robust activity data collection†. *Sensors (Switzerland)*, 21(1):1–22, 2021. ISSN 1424-8220. doi: 10.3390/s21010041.
[217] J. Han, Y. Ma, Q. Mei, and X. Liu. Deeprec: On-device deep learning for privacy-preserving sequential recommendation in mobile commerce. In *The Web Conference*, pages 900–911, 2021. doi: 10.1145/3442381.3449942.
[218] J. Lee, S. Kim, S. Kim, W. Jo, J.-H. Kim, D. Han, and H.-J. Yoo. OmniDRL: An Energy-Efficient Deep Reinforcement Learning Processor with Dual-Mode Weight Compression and Sparse Weight Transposer. *IEEE Journal of Solid-State Circuits*, 57(4):999–1012, 2022. doi: 10.1109/JSSC.2021.3138520.
[219] A. Hosny, M. Neseem, and S. Reda. Sparse Bitmap Compression for Memory-Efficient Training on the Edge. In *SEC*, pages 14–25, 2021. ISBN 978-1-4503-8390-5. doi: 10.1145/3453142.3491290.
[220] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. In *NeurIPS*, 2016.
[221] Erwei Wang, James J. Davis, Daniele Moro, Piotr Zielinski, Jia Jie Lim, Claudionor Coelho, Satrajit Chatterjee, Peter Y. K. Cheung, and George A. Constantinides. Enabling Binary Neural Network Training on the Edge. *ACM Transactions on Embedded Computing Systems*, 22(6):105:1–105:19, November 2023. ISSN 1539-9087. doi: 10.1145/3626100.
[222] Lorenzo Vorabbi, Davide Maltoni, and Stefano Santi. On-Device Learning with Binary Neural Networks, 2024.
[223] Yuya Fujiwara and Takayuki Kawahara. BNN Training Algorithm with Ternary Gradients and BNN based on MRAM Array. In *TENCON*, pages 311–316, oct 2023. doi: 10.1109/TENCON58879.2023.10322327. ISSN: 2159-3450.
[224] B. Penkovsky, M. Bocquet, T. Hirtzlin, J.-O. Klein, E. Nowak, E. Vianello, J.-M. Portal, and D. Querlioz. In-Memory Resistive RAM Implementation of Binarized Neural Networks for Medical Applications. In *DATE*, pages 690–695, 2020. ISBN 978-3-9819263-4-7. doi: 10.23919/DATE48585.2020.9116439.
[225] Ngoc Duy Pham, Hong Dien Nguyen, and Dinh Hoa Dang. Efficient binarizing split learning based deep models for mobile applications. *AIP Conference Proceedings*, 2406(1):020015, Seb 2021. ISSN 0094-243X. doi: 10.1063/5.0066470.
[226] Wulfram Gerstner and Werner M Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.
[227] Tianqi Tang, Rong Luo, Boxun Li, Hai Li, Yu Wang, and Huazhong Yang. Energy efficient spiking neural network design with rram devices. In *ISIC*, pages 268–271. IEEE, 2014.
[228] Edgar Lemaire, Loïc Cordone, Andrea Castagnetti, Pierre-Emmanuel Novac, Jonathan Courtois, and Benoît Miramond. An analytical estimation of spiking neural networks energy efficiency. In *ICONIP*, pages 574–587. Springer, 2022.
[229] Jianwei Xue, Lisheng Xie, Faquan Chen, Liangshun Wu, Qingyang Tian, Yifan Zhou, Rendong Ying, and Peilin Liu. Edgemap: An optimized mapping toolchain for spiking neural network in edge computing. *Sensors*, 23(14):6548, 2023.
[230] N. Skatchkovsky, H. Jang, and O. Simeone. Federated Neuromorphic Learning of Spiking Neural Networks for Low-Power Edge Intelligence. In *ICASSP*, pages 8524–8528, May 2020.
[231] Abdullah M. Zyarah, Nicholas Soures, and Dhireesha Kudithipudi. On-Device Learning in Memristor Spiking Neural Networks. In *ISCAS*, pages 1–5, May 2018. doi: 10.1109/ISCAS.2018.8351813. ISSN: 2379-447X.
[232] Nicholas Soures, Lydia Hays, Eric Bohannon, Abdullah M. Zyarah, and Dhireesha Kudithipudi. On-device STDP and synaptic normalization for neuromemristive spiking neural network. In *MWSCAS*, pages 1081–1084, August 2017. doi: 10.1109/MWSCAS.2017.8053115. ISSN: 1558-3899.

[233] Peter G. Stratton, Tara J. Hamilton, and Andrew Wabnitz. Unsupervised Feature Vector Clustering Using Temporally Coded Spiking Networks. In *IJCNN*, pages 1–7, June 2023. doi: 10.1109/IJCNN54540.2023.10191150. ISSN: 2161-4407.

[234] G. Tang, K. Vadivel, Y. Xu, R. Bilgic, K. Shidqi, P. Detterer, S. Traferro, M. Konijnenburg, M. Sifalakis, G.-J. van Schaik, and A. Yousefzadeh. SENECA: building a fully digital neuromorphic processor, design trade-offs and challenges. *Frontiers in Neuroscience*, 17, 2023. ISSN 1662-4548. doi: 10.3389/fnins.2023.1187252.

[235] Ali Safa, Jonah Van Assche, Mark Daniel Alea, Francky Catthoor, and Georges G.E. Gielen. Neuromorphic Near-Sensor Computing: From Event-Based Sensing to Edge Learning. *IEEE Micro*, 42(6):88–95, November 2022. ISSN 1937-4143.

[236] Geoffrey Hinton. The Forward-Forward Algorithm: Some Preliminary Investigations, Dec 2022. 10.48550/arXiv.2212.13345.

[237] Fabrizio De Vita, Rawan M. A. Nawaiseh, Dario Bruneo, Valeria Tomaselli, Marco Lattuada, and Mirko Falchetto. µ-FF: On-Device Forward-Forward Training Algorithm for Microcontrollers. In *SMARTCOMP*, pages 49–56, June 2023. doi: 10.1109/SMARTCOMP58114.2023.00024.

[238] Giorgia Dellaferrera and Gabriel Kreiman. Error-driven input modulation: Solving the credit assignment problem without a backward pass. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *ICML*, volume 162, pages 4937–4955, 17–23 Jul 2022.

[239] Fabrizio De Vita, Enrico Catalfamo, Rawan M. A. Nawaiseh, and Dario Bruneo. Sparse-FF: Enhancing Forward-Forward Efficiency by Promoting Sparsity. In *2025 IEEE 11th World Forum on Internet of Things (WF-IoT)*, pages 1–6, October 2025. doi: 10.1109/WF-IoT64238.2025.11270764.

[240] Jingxiao Ma, Priyadarshini Panda, and Sherief Reda. FF-INT8: Efficient Forward-Forward DNN Training on Edge Devices with INT8 Precision. In *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, pages 1–7, June 2025. doi: 10.1109/DAC63849.2025.11133061.

[241] Hamad AlHammadi, Meriam Mkadmi, Rohan Mitra, and Imran Zualkernan. Exploring the Forward-Forward Algorithm to Train Neural Networks for Camera Trap Images on the Edge. In *2024 IEEE 10th World Forum on Internet of Things (WF-IoT)*, pages 741–746, November 2024. doi: 10.1109/WF-IoT62078.2024.10811281.

[242] Albert Antony Sebastian, Rijul Muhammed, Sarthak Maloo, Rana Gharaibeh, Ali Reza Sajun, Imran Zualkernan, and Abdullah Shahid. Exploring the Forward-Forward algorithm in the Context of Crack Detection. In *Proceedings of the 2025 3rd International Conference on Machine Learning and Pattern Recognition*, MLPR '25, pages 28–35, New York, NY, USA, November 2025. Association for Computing Machinery. ISBN 979-8-4007-1333-0. doi: 10.1145/3760678.3760683.

[243] Mehdi Abbassi, Alberto Ancilotto, and Elsabetta Farella. Contextual Convolutions for Scalable Forward-Only Learning on Tiny Devices. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1718–1725, October 2025. doi: 10.1109/ICCVW69036.2025.00181.

[244] Ilenia Ficili, Enrico Catalfamo, Fabrizio De Vita, Dario Bruneo, and Antonio Puliafito. Enabling Efficient Federated Learning at the Edge through Sparse Forward-Forward Algorithm. In *2025 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pages 376–383, November 2025. doi: 10.1109/SmartIoT66867.2025.00061.

[245] C. Profentzas, M. Almgren, and O. Landsiedel. MiniLearn: On-Device Learning for Low-Power IoT Devices. In *EWSN*, 2022. ISSN: 2562-2331.

[246] D. Nadalini, M. Rusci, L. Benini, and F. Conti. Reduced precision floating-point optimization for Deep Neural Network On-Device Learning on microcontrollers. *Future Generation Computer Systems*, 149:212–226, 2023. ISSN 0167-739X.

[247] Shishir G. Patil, Paras Jain, Prabal Dutta, Ion Stoica, and Joseph Gonzalez. POET: Training neural networks on tiny devices with integrated rematerialization and paging. In *ICML*, volume 162, pages 17573–17583, 17–23 Jul 2022.

[248] T. Rahman, A. Wheeldon, R. Shafik, A. Yakovlev, J. Lei, O.-C. Granmo, and S. Das. Data Booleanization for Energy Efficient On-Chip Learning using Logic Driven AI. In *ISTM*, pages 29–36, 2022. ISBN 978-1-66547-116-9. doi: 10.1109/ISTM54910.2022.00014.

[249] A. Carta, G. Carfì, V. De Caro, and C. Gallicchio. Efficient Anomaly Detection on Temporal Data via Echo State Networks and Dynamic Thresholding. In *CEUR Workshop Proceedings*, volume 3350, pages 56–67, 2023.

[250] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCVW*, pages 3009–3018, 2019. doi: 10.1109/ICCVW.2019.00363.

[251] Hangyu Zhu, Jinjin Xu, Shiqing Liu, and Yaochu Jin. Federated learning on non-IID data: A survey. *Neurocomputing*, 465:371–390, November 2021. ISSN 0925-2312. doi: 10.1016/j.neucom.2021.07.098.

[252] Yanmeng Wang, Qingjiang Shi, and Tsung-Hui Chang. Why Batch Normalization Damage Federated Learning on Non-IID Data? *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2023. ISSN 2162-2388.

[253] Ahmad Chaddad, Yihang Wu, and Christian Desrosiers. Federated Learning for Healthcare Applications. *IEEE Internet of Things Journal*, 11(5):7339–7358, March 2024. ISSN 2327-4662. doi: 10.1109/JIOT.2023.3325822.

[254] Boubakr Nour, Soumaya Cherkaoui, and Zoubeir Mlika. Federated Learning and Proactive Computation Reuse at the Edge of Smart Homes. *IEEE Transactions on Network Science and Engineering*, 9(5):3045 – 3056, 2022.

[255] U.P. Chinchole and S. Raut. Federated Learning For Estimating Air Quality. In *ICCCNT*, 2021. doi: 10.1109/ICCCNT51525.2021.9580157.

[256] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. Overview and Importance of Data Quality for Machine Learning Tasks. In *SIGKDD*, pages 3561–3562, NY, USA, August 2020. ACM. ISBN 978-1-4503-7998-4.

[257] Steven Euijong Whang, Yuji Roh, Hwanjun Song, and Jae-Gil Lee. Data collection and quality challenges in deep learning: a data-centric AI perspective. *The VLDB Journal*, 32(4):791–813, July 2023. ISSN 0949-877X.

[258] Mohammed Djameleddine Belgoumri, Mohamed Reda Bouadjenek, Sunil Aryal, and Hakim Hacid. Data Quality in Edge Machine Learning: A State-of-the-Art Survey, June 2024. arXiv:2406.02600.

[259] J. Liu, J. Liu, Z. Xie, X. Ning, and D. Li. Flame: A Self-Adaptive Auto-Labeling System for Heterogeneous Mobile Processors. In *SEC*, pages 80–93, 2021. doi: 10.1145/3453142.3493611.

[260] Wenhai Lu, Jieren Cheng, Xiulai Li, and Ji He. A Review of Solving Non-IID Data in Federated Learning: Current Status and Future Directions. In Hai Jin, Yi Pan, and Jianfeng Lu, editors, *Artificial Intelligence and Machine Learning*, pages 58–72, Singapore, 2024. Springer Nature. ISBN 978-981-9712-77-9. doi: 10.1007/978-981-97-1277-9_5.

[261] Ching-Yi Lin and Radu Marculescu. Model Personalization for Human Activity Recognition. In *PerCom Workshops*, pages 1–7, March 2020. doi: 10.1109/PerComWorkshops48775.2020.9156229.

[262] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Guoliang Xing, and Jianwei Huang. Clusterfl: A clustering-based federated learning system for human activity recognition. *ACM Trans. Sen. Netw.*, 19(1), dec 2022. ISSN 1550-4859.

[263] M. Craighero, D. Quarantiello, B. Rossi, D. Carrera, P. Fragneto, and G. Boracchi. On-Device Personalization for Human Activity Recognition on STM32. *IEEE Embedded Systems Letters*, pages 1–1, 2023. ISSN 1943-0663. doi: 10.1109/LES.2023.3293458.

[264] Yongjie Du, Deyun Zhou, Yu Xie, Jiao Shi, and Maoguo Gong. Federated matrix factorization for privacy-preserving recommender systems. *Applied Soft Computing*, 111:107700, November 2021. ISSN 1568-4946. doi: 10.1016/j.asoc.2021.107700.

[265] Vasileios Perifanis and Pavlos S. Efraimidis. Federated Neural Collaborative Filtering. *Knowledge-Based Systems*, 242: 108441, April 2022. ISSN 0950-7051. doi: 10.1016/j.knosys.2022.108441.

[266] Saqib Hakak, Suprio Ray, Wazir Zada Khan, and Erik Scheme. A Framework for Edge-Assisted Healthcare Data Analytics using Federated Learning. In *BigData*, pages 3423–3427, December 2020.

[267] Marija Stojchevska, Mathias De Brouwer, Martijn Courteaux, Femke Ongenae, and Sofie Van Hoecke. From lab to real world: Assessing the effectiveness of human activity recognition and optimization through personalization. *Sensors*, 23(10), 2023. ISSN 1424-8220. doi: 10.3390/s23104606.

[268] J. Yang, Y. Sheng, Y. Zhang, W. Jiang, and L. Yang. On-Device Unsupervised Image Segmentation. In *DAC*, July 2023. doi: 10.1109/DAC56929.2023.10247959.

[269] A. Albaseer, M. Abdallah, A. Al-Fuqaha, and A. Erbad. Client Selection Approach in Support of Clustered Federated Learning over Wireless Edge Networks. In *GLOBECOM*, 2021. doi: 10.1109/GLOBECOM46510.2021.9685938.

[270] D. Wang, N. Zhang, and M. Tao. Clustered federated learning with weighted model aggregation for imbalanced data. *China Communications*, pages 41–56, 2022. doi: 10.23919/JCC.2022.08.004.

[271] Don Kurian Dennis, Tian Li, and Virginia Smith. Heterogeneity for the Win: One-Shot Federated Clustering. In *ICML*, pages 2611–2620, July 2021. ISSN: 2640-3498.

[272] Y.-Y. Hsieh, Y.-C. Lee, and C.-H. Yang. A cyclegan accelerator for unsupervised learning on mobile devices. In *ISCAS*, Oct 2020.

[273] S. Muthu, R. Tennakoon, R. Hoseinnezhad, and A. Bab-Hadiashar. Unsupervised video object segmentation: an affinity and edge learning approach. *International Journal of Machine Learning and Cybernetics*, 2022. doi: 10.1007/s13042-022-01615-6.

[274] D. Piyasena, M. Thathsara, S. Kanagarajah, S.K. Lam, and M. Wu. Dynamically Growing Neural Network Architecture for Lifelong Deep Learning on the Edge. In *FPL*, pages 262–268, 2020. doi: 10.1109/FPL50879.2020.00051.

[275] C. Li, H. Yang, Z. Sun, Q. Yao, J. Zhang, A. Yu, A.V. Vasilakos, S. Liu, and Y. Li. High-Precision Cluster Federated Learning for Smart Home: An Edge-Cloud Collaboration Approach. *IEEE Access*, 11:102157–102168, 2023.

[276] Z. Li, Z. Chen, X. Wei, S. Gao, C. Ren, and T.Q.S. Quek. HPFL-CN: Communication-Efficient Hierarchical Personalized Federated Edge Learning via Complex Network Feature Clustering. In *SECON workshops*, pages 325–333, Seb 2022. doi: 10.1109/SECON55815.2022.9918588.

[277] B. Gong, T. Xing, Z. Liu, W. Xi, and X. Chen. Towards Hierarchical Clustered Federated Learning with Model Stability on Mobile Devices. *IEEE Transactions on Mobile Computing*, pages 1–17, 2023. doi: 10.1109/TMC.2023.3332637.

[278] O. Aygün, M. Kazemi, D. Gündüz, and T.M. Duman. Hierarchical Over-the-Air Federated Edge Learning. In *ICC*, pages 3376–3381, May 2022.

[279] Jie Yan, Jing Liu, and Zhong-Yuan Zhang. CCFC: Bridging Federated Clustering and Contrastive Learning, January 2024. arXiv:2401.06634.

[280] Jie Yan, Jing Liu, Yi-Zi Ning, and Zhong-Yuan Zhang. CCFC++: Enhancing Federated Clustering through Feature Decorrelation, February 2024. arXiv:2402.12852.

[281] Jie Yan, Jing Liu, Ji Qi, and Zhong-Yuan Zhang. Privacy-Preserving Federated Deep Clustering based on GAN, October 2023. arXiv:2211.16965.

[282] Morris Stallmann and Anna Wilbik. Towards Federated Clustering: A Federated Fuzzy c-Means Algorithm (FFCM), January 2022. arXiv:2201.07316.

[283] W. Zhuang, Y. Wen, and S. Zhang. Joint Optimization in Edge-Cloud Continuum for Federated Unsupervised Person Re-identification. In *MM 2021*, pages 433–441, 2021. doi: 10.1145/3474085.3475182.

[284] Nan Lu, Zhao Wang, Xiaoxiao Li, Gang Niu, Qi Dou, and Masashi Sugiyama. Federated Learning from only Unlabeled Data with Class-Conditional-Sharing Clients. *arXiv preprint arXiv:2204.03304*, 2022.

[285] Keerthana Sivamayil, Elakkiya Rajasekar, Belqasem Aljafari, Srete Nikolovski, Subramaniyaswamy Vairavasundaram, and Indragandhi Vairavasundaram. A systematic study on reinforcement learning based applications. *Energies*, 16(3), 2023. ISSN 1996-1073. doi: 10.3390/en16031512.

[286] Muddasar Naeem, Syed Tahir Hussain Rizvi, and Antonio Coronato. A gentle introduction to reinforcement learning and its application in different fields. *IEEE Access*, 8:209320–209344, 2020. doi: 10.1109/ACCESS.2020.3038605.

[287] S.-C. Kao and T. Krishna. E3: A HW/SW Co-design Neuroevolution Platform for Autonomous Learning in Edge Device. In *ISPASS*, pages 288–298, 2021. doi: 10.1109/ISPASS51385.2021.00051.

[288] Hankz Hankui Zhuo, Wenfeng Feng, Yufeng Lin, Qian Xu, and Qiang Yang. Federated deep reinforcement learning, 2020. arxiv 1901.08277.

[289] Jiaju Qi, Qihao Zhou, Lei Lei, and Kan Zheng. Federated reinforcement learning: techniques, applications, and open challenges. *Intelligence & Robotics*, 2021. doi: 10.20517/ir.2021.02.

[290] Y. Xianjia, J.P. Queralta, J. Heikkonen, and T. Westerlund. Federated Learning in Robotic and Autonomous Systems. In *Procedia Computer Science*, volume 191, pages 135–142, 2021. doi: 10.1016/j.procs.2021.07.041.

[291] Chetan Nadiger, Anil Kumar, and Sherine Abdelhak. Federated reinforcement learning for fast personalization. In *AIKE*, page 123 – 127, 2019. doi: 10.1109/AIKE.2019.00031.

[292] Weicheng Xiong, Quan Liu, Fanzhang Li, Bangjun Wang, and Fei Zhu. Personalized federated reinforcement learning: Balancing personalization and experience sharing via distance constraint[formula presented]. *Expert Systems with Applications*, 238, 2024. doi: 10.1016/j.eswa.2023.122290.

[293] A. Jarwan and M. Ibnkahla. Edge-Based Federated Deep Reinforcement Learning for IoT Traffic Management. *IEEE Internet of Things Journal*, 10(5):3799–3813, 2023. doi: 10.1109/JIOT.2022.3174469.

[294] Xiaofei Wang, Chenyang Wang, Xiuhua Li, Victor C. M. Leung, and Tarik Taleb. Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet of Things Journal*, 7(10): 9441–9455, 2020. doi: 10.1109/JIOT.2020.2986803.

[295] T. Liu, T.K. Zhang, J. Loo, and Y.P. Wang. Deep Reinforcement Learning-Based Resource Allocation for UAV-Enabled Federated Edge Learning. *Journal of Communications and Information Networks*, 8(1), 2023.

[296] Y.G. Kim and C.-J. Wu. FedGPO: Heterogeneity-Aware Global Parameter optimization for Efficient Federated Learning. In *IISWC*, pages 117–129, 2022. doi: 10.1109/IISWC55918.2022.00020.

[297] G. Rjoub, O.A. Wahab, J. Bentahar, and A. Bataineh. Trust-driven reinforcement selection strategy for federated learning on IoT devices. *Computing*, 2022. doi: 10.1007/s00607-022-01078-1.

[298] H. Watanabe, M. Tsukada, and H. Matsutani. An FPGA-Based On-Device Reinforcement Learning Approach using Online Sequential Learning. In *IPDPSW*, pages 96–103, 2021. doi: 10.1109/IPDPSW52791.2021.00022.

[299] K. Rakesh, L.S. Kumar, R. Mittar, P. Chakraborty, P.A. Ankush, and S.K. Gairuboina. DNN based adaptive video streaming using combination of supervised learning and reinforcement learning. In *Communications in Computer and Information Science*, volume 1148 CCIS, pages 143–154, 2020. doi: 10.1007/978-981-15-4018-9_13.

[300] H. Zhang, A. Zhou, and H. Ma. Reinforcement learning-based real-time video streaming control and on-device training research. *Chinese Journal on Internet of Things*, 6(4):1–13, 2022. doi: 10.11959/j.issn.2096-3750.2022.00306.

[301] Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. *CoRR*, abs/1708.08611, 2017.

[302] T. Sen and H. Shen. Distributed Training for Deep Learning Models On An Edge Computing Network Using Shielded Reinforcement Learning. In *ICDCS*, pages 581–591, 2022. doi: 10.1109/ICDCS54860.2022.00062.

[303] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine learning*, 109(2):373–440, 2020.

[304] J. Park, J. Kwak, K. Kim, S.-S. Lee, and S.-J. Jang. Semi-Supervised Learning using Sequential Data for Mobile Applications. In *ICCE-Asia 2022*, 2022. doi: 10.1109/ICCE-Asia57006.2022.9954648.

[305] X. Pei, X. Deng, S. Tian, L. Zhang, and K. Xue. A Knowledge Transfer-Based Semi-Supervised Federated Learning for IoT Malware Detection. *IEEE Transactions on Dependable and Secure Computing*, 20(3):2127–2143, 2023. doi: 10.1109/TDSC.2022.3173664.

[306] A. Albaseer, B.S. Ciftler, M. Abdallah, and A. Al-Fuqaha. Exploiting Unlabeled Data in Smart Cities using Federated Edge Learning. In *IWCMC 2020*, pages 1666–1671, 2020. doi: 10.1109/IWCMC48107.2020.9148475.

[307] M. Tsukada, M. Kondo, and H. Matsutani. A neural network-based on-device learning anomaly detector for edge devices. *IEEE Transactions on Computers*, 69(7):1027–1044, 2020. doi: 10.1109/TC.2020.2973631.

[308] M. Wu, H. Matsutani, and M. Kondo. ONLAD-IDS: ONLAD-Based Intrusion Detection System Using SmartNIC. In *HPCC/DSS/SmartCity/DependSys 2022*, pages 546–553, 2022.

[309] V. Radu, P. Katsikouli, R. Sarkar, and M.K. Marina. A semi-supervised learning approach for robust indoor-outdoor detection with smartphones. In *SenSys 2014*, pages 280–294, 2014. doi: 10.1145/2668332.2668347.

[310] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, et al. A cookbook of self-supervised learning. *arXiv preprint arXiv:2304.12210*, 2023.

[311] Y. Gaol, J. Fernandez-Marques, T. Parcollet, P.P.B. De Gusmao, and N.D. Lane. Match to Win: Analysing Sequences Lengths for Efficient Self-Supervised Learning in Speech and Audio. In *2022 IEEE Spoken Language Technology Workshop, SLT 2022 - Proceedings*, pages 115–122, 2023. doi: 10.1109/SLT54892.2023.10023410.

[312] Z. Huo, D. Hwang, K.C. Sim, S. Garg, A. Misra, N. Siddhartha, T. Strohman, and F. Beaufays. Incremental Layer-Wise Self-Supervised Learning for Efficient Unsupervised Speech Domain Adaptation On Device. In *INTERSPEECH*, pages 4845–4849, Seb 2022. doi: 10.21437/Interspeech.2022-10904.

[313] J. Liu, X. Yu, and T. Rosing. Self-Train: Self-Supervised On-Device Training for Post-Deployment Adaptation. In *SmartIoT*, pages 161–168, 2022.

[314] J. Shi, Y. Wu, D. Zeng, J. Tao, J. Hu, and Y. Shi. Self-Supervised On-Device Federated Learning From Unlabeled Streams. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(12):4871–4882, 2023.

[315] Y. Wu, Z. Wang, D. Zeng, Y. Shi, and J. Hu. Enabling On-Device Self-Supervised Contrastive Learning with Selective Data Contrast. In *DAC*, pages 655–660, Dec 2021.

[316] Y. Wu, D. Zeng, Z. Wang, Y. Sheng, L. Yang, A.J. James, Y. Shi, and J. Hu. Federated Contrastive Learning for Dermatological Disease Diagnosis via On-device Learning. In *ICCAD*, Nov 2021.

[317] Fotis Kitsios, Maria Kamariotou, Aristomenis I. Syngelakis, and Michael A. Talias. Recent Advances of Artificial Intelligence in Healthcare: A Systematic Literature Review. *Applied Sciences*, 13(13):7479, Jan 2023. ISSN 2076-3417. doi: 10.3390/app13137479. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

[318] Adnan Qayyum, Junaid Qadir, Muhammad Bilal, and Ala I. Al-Fuqaha. Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering*, 14:156–180, 2020.

[319] Ahmad Zainuddin. Artificial Intelligence and Machine learning in the Healthcare Sector: A Review. *Malaysian Journal of Science and Advanced Technology*, July 2021.

[320] Eike Petersen, Yannik Potdevin, Esfandiar Mohammadi, Stephan Zidowitz, Sabrina Breyer, Dirk Nowotka, Sandra Henn, Ludwig Pechmann, Martin Leucker, Philipp Rostalski, and Christian Herzog. Responsible and regulatory conform machine learning for medicine: A survey of challenges and solutions. *IEEE Access*, 10:58375–58418, 2022. ISSN 2169-3536. doi: 10.1109/access.2022.3178382.

[321] Adnan Qayyum, Kashif Ahmad, Muhammad Ahtazaz Ahsan, Ala Al-Fuqaha, and Junaid Qadir. Collaborative Federated Learning for Healthcare: Multi-Modal COVID-19 Diagnosis at the Edge. *IEEE Open Journal of the Computer Society*, 3: 172–184, 2022. ISSN 2644-1268. doi: 10.1109/OJCS.2022.3206407.

[322] Zhuotao Lian, Qinglin Yang, Weizheng Wang, Qingkui Zeng, Mamoun Alazab, Hong Zhao, and Chunhua Su. DEEP-FEL: Decentralized, Efficient and Privacy-Enhanced Federated Edge Learning for Healthcare Cyber Physical Systems. *IEEE Transactions on Network Science and Engineering*, 9(5):3558–3569, Seb 2022. ISSN 2327-4697.

[323] Yeting Guo, Fang Liu, Zhiping Cai, Li Chen, and Nong Xiao. FEEL: A Federated Edge Learning System for Efficient and Privacy-Preserving Mobile Healthcare. In *ICPP*, pages 1–11, NY, USA, August 2020. ACM. ISBN 978-1-4503-8816-0.

[324] Juexiao Zhou, Longxi Zhou, Di Wang, Xiaopeng Xu, Haoyang Li, Yuetan Chu, Wenkai Han, and Xin Gao. Personalized and privacy-preserving federated heterogeneous medical image analysis with PPPML-HMI. *Computers in Biology and Medicine*, 169:107861, February 2024. ISSN 0010-4825. doi: 10.1016/j.compbiomed.2023.107861.

[325] Safa Bahri, Nesrine Zoghlami, Mourad Abed, and João Manuel R. S. Tavares. Big data for healthcare: A survey. *IEEE Access*, 7:7397–7408, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2018.2889180.

[326] Adnan Qayyum, Kashif Ahmad, Muhammad Ahtazaz Ahsan, Ala Al-Fuqaha, and Junaid Qadir. Collaborative Federated Learning for Healthcare: Multi-Modal COVID-19 Diagnosis at the Edge. *IEEE Open Journal of the Computer Society*, 3: 172–184, 2022. ISSN 2644-1268. doi: 10.1109/OJCS.2022.3206407.

[327] J. Chen, Y. Zheng, Y. Liang, Z. Zhan, M. Jiang, X. Zhang, D.S. Da Silva, W. Wu, and V.H.C. De Albuquerque. Edge2Analysis: A Novel AIoT Platform for Atrial Fibrillation Recognition and Detection. *IEEE Journal of Biomedical and Health Informatics*, 26(12):5772–5782, 2022. doi: 10.1109/JBHI.2022.3171918.

[328] V. Chandrika and S. Surendran. Incremental Machine Learning Model for Fetal Health Risk Prediction. In *SMART GENCON*, 2022. doi: 10.1109/SMARTGENCON56628.2022.10084232.

[329] D. Hou, R. Hou, and J. Hou. On-device Training for Breast Ultrasound Image Classification. In *CCWC*, pages 78–82, 2020. doi: 10.1109/CCWC47524.2020.9031146.

[330] T. Ravi Shanker Reddy and B.M. Beena. AI Integrated Blockchain Technology for Secure Health Care—Consent-Based Secured Federated Transfer Learning for Predicting COVID-19 on Wearable Devices. In *Lecture Notes in Networks and Systems*, volume 473, pages 345–356, 2023. doi: 10.1007/978-981-19-2821-5_30.

[331] Dr. Hari Om Sharan. Advancements and future directions in human activity recognition. *International Journal For Science Technology And Engineering*, 2023.

[332] M. M. Kamruzzaman. New Opportunities, Challenges, and Applications of Edge-AI for Connected Healthcare in Smart Cities. In *IEEE Globecom Workshops*, pages 1–6, Dec 2021. doi: 10.1109/GCWkshps52748.2021.9682055.

[333] Ali M. Hayajneh, Sami A. Aldalahmeh, Feras Alasali, Haitham Al-Obiedollah, Sayed Ali Zaidi, and Des McLernon. Tiny machine learning on the edge: A framework for transfer learning empowered unmanned aerial vehicle assisted smart farming. *IET Smart Cities*, 2023. doi: 10.1049/smc2.12072.

[334] Kavitha Rajamohan, Sangeetha Rangasamy, Alvis Abreo, Rohit Upadhyay, and Raison Sabu. Smart cities: Redefining urban life through iot. *Advances in systems analysis, software engineering, and high performance computing book series*, 2023. doi: 10.4018/978-1-6684-8785-3.ch016.

[335] J. Na, H. Zhang, X. Deng, B. Zhang, and Z. Ye. Accelerate personalized iot service provision by cloud-aided edge reinforcement learning: A case study on smart lighting. In *Lecture Notes in Computer Science*, volume 12571 LNCS, pages 69–84, 2020. doi: 10.1007/978-3-030-65310-1_6.

[336] Javier Manuel Aguiar-Perez and Maria Angeles Perez-Juarez. An insight of deep learning based demand forecasting in smart grids. *Sensors*, 23(3), 2023. ISSN 1424-8220. doi: 10.3390/s23031467.

[337] J. Jithish, Bithin Alangot, Nagarajan Mahalingam, and Kiat Seng Yeo. Distributed Anomaly Detection in Smart Grids: A Federated Learning-Based Approach. *IEEE Access*, 11:7157 – 7179, 2023. doi: 10.1109/ACCESS.2023.3237554.

[338] Afaf Taik, Boubakr Nour, and Soumaya Cherkaoui. Empowering Prosumer Communities in Smart Grid with Wireless Communications and Federated Edge Learning. *IEEE Wireless Communications*, 28(6):26 – 33, 2021.

[339] W. Lei, H. Wen, J. Wu, and W. Hou. MADDPG-based security situational awareness for smart grid with intelligent edge. *Applied Sciences (Switzerland)*, 11(7), 2021. doi: 10.3390/app11073101.

[340] Nhat Nam Tran Huu, Linh Mai, and Thanh Vo Minh. Detecting Abnormal and Dangerous Activities Using Artificial Intelligence on the Edge for Smart City Application. In *ACOMP*, pages 85 – 92, 2021. doi: 10.1109/ACOMP53746.2021.00018.

[341] Chunlei Bian, Yiming Xu, Li Wang, Haifeng Gu, and Fangjie Zhou. Abnormal behavior recognition based on edge feature and 3D convolutional neural network. In *YAC*, pages 661 – 666, 2020. doi: 10.1109/YAC51587.2020.9337685.

[342] Danni Yuan, Xiaoyan Zhu, Yaoru Mao, Binwen Zheng, and Tao Wu. Privacy-Preserving Pedestrian Detection for Smart City with Edge Computing. In *WCSP*, 2019. doi: 10.1109/WCSP.2019.8927923.

[343] S.V. Pandiyan and J. Rajasekharan. Federated Learning vs Edge Learning for Hot Water Demand Forecasting in Distributed Electric Water Heaters for Demand Side Flexibility Aggregation. In *IEEE PES Grid Edge Technologies Conference and Exposition*, 2023. doi: 10.1109/GridEdge54130.2023.10102739.

[344] A. Jaleel, M.A. Hassan, T. Mahmood, M.U. Ghani, and A. Ur Rehman. Reducing congestion in an intelligent traffic system with collaborative and adaptive signaling on the edge. *IEEE Access*, 8:205396–205410, 2020.

[345] Giorgos Constantinou, Gowri Sankar Ramachandran, Abdullah Alfarrarjeh, Seon Ho Kim, Bhaskar Krishnamachari, and Cyrus Shahabi. A crowd-based image learning framework using edge computing for smart city applications. In *BigMM*, pages 11 – 20, 2019.

[346] Harendra Singh Negi and Sushil Chandra Dimri. Machine learning enabled smart farming:the demand of the time. In *PDGC*, pages 490–495, 2022.

[347] Itika Sharma, Ayushe Sharma, and Sachin Kumar Gupta. Autonomous vehicles: Open-source technologies, considerations, and development. *Advances in Artificial Intelligence and Machine Learning*, pages 105–109, 2023.

[348] B. Yang, X. Cao, X. Li, C. Yuen, and L. Qian. Lessons Learned from Accident of Autonomous Vehicle Testing: An Edge Learning-Aided Offloading Framework. *IEEE Wireless Communications Letters*, 9(8):1182–1186, 2020.

[349] Itika Sharma, Ayushe Sharma, and Sachin Kumar Gupta. Asynchronous and Synchronous Federated Learning-based UAVs. In *ICA-SYMP*, pages 105–109, Jan 2023. doi: 10.1109/ICA-SYMP56348.2023.10044951.

[350] Jichao Chen, Omid Esrafilian, Harald Bayerlein, David Gesbert, and Marco Caccamo. Model-aided Federated Reinforcement Learning for Multi-UAV Trajectory Planning in IoT Networks. *arXiv.org*, abs/2306.02029, June 2023. ISSN 2331-8422. doi: 10.48550/arXiv.2306.02029.

[351] S. Chen and K. Mai. Towards Specialized Hardware for Learning-based Visual Odometry on the Edge. In *IROS*, pages 10603–10610, Oct 2022.

[352] D.C. Selvaraj, C. Vitale, T. Panayiotou, P. Kolios, C.F. Chiasserini, and G. Ellinas. Edge Learning of Vehicular Trajectories at Regulated Intersections. In *VTC*, Seb 2021. doi: 10.1109/VTC2021-Fall52928.2021.9625570.

[353] S. Zhang, S. Zhang, and L.K. Yeung. Energy-efficient Federated Edge Learning for Internet of Vehicles via Rate-Splitting Multiple Access. In *ISWCS*, Oct 2022. doi: 10.1109/ISWCS56560.2022.9940330.

[354] Hannes Werthner, Hans Robert Hansen, and Franceso Ricci. Recommender systems. In *HICSS*, volume 1, pages 167–167, 2007. doi: 10.1109/HICSS.2007.459.

[355] Xin Xin, Jiyuan Yang, Hanbing Wang, Jun Ma, Pengjie Ren, Hengliang Luo, Xinlei Shi, Zhumin Chen, and Zhaochun Ren. On the user behavior leakage from recommender system exposure. *ACM Trans. Inf. Syst.*, 41(3), feb 2023. ISSN 1046-8188.

[356] Peter Müllner. *User Privacy in Recommender Systems*. Springer Nature Switzerland, 2023. ISBN 978-3-031-28241-6.

[357] L. Yang, B. Tan, V.W. Zheng, K. Chen, and Q. Yang. Federated Recommendation Systems. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12500 LNCS: 225–239, 2020. ISSN 0302-9743.

[358] S. Wei, S. Meng, Q. Li, X. Zhou, L. Qi, and X. Xu. Edge-enabled federated sequential recommendation with knowledge-aware Transformer. *Future Generation Computer Systems*, 148:610–622, 2023.

[359] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. A federated graph neural network framework for privacy-preserving personalization. *Nature Communications*, 13(1):3091, June 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-30714-9.

[360] Z. Liu, L. Yang, Z. Fan, H. Peng, and P.S. Yu. Federated Social Recommendation with Graph Neural Network. *ACM Transactions on Intelligent Systems and Technology*, 13(4), 2022. ISSN 2157-6904. doi: 10.1145/3501815.

[361] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor. FedFast: Going beyond Average for Faster Training of Federated Recommender Systems. In *SIGKDD*, pages 1234–1242, 2020. ISBN 978-1-4503-7998-4. doi: 10.1145/3394486.3403176.

[362] Tianzi Zang, Yanmin Zhu, Haobing Liu, Ruohan Zhang, and Jiadi Yu. A Survey on Cross-domain Recommendation: Taxonomies, Methods, and Future Directions. *ACM Trans. Inf. Syst.*, 41(2):42:1–42:39, December 2022. ISSN 1046-8188. doi: 10.1145/3548455.

[363] Ziqi Yang, Zhaopeng Peng, Zihui Wang, Jianzhong Qi, Chaochao Chen, Weike Pan, Chenglu Wen, Cheng Wang, and Xiaoliang Fan. Federated Graph Learning for Cross-Domain Recommendation. In *NeurIPS*, volume 37, pages 64865–64888, December 2024.

[364] Wu Meihan, Li Li, Chang Tao, Eric Rigall, Wang Xiaodong, and Xu Cheng-Zhong. FedCDR: Federated Cross-Domain Recommendation for Privacy-Preserving Rating Prediction. In *CIKM*, CIKM '22, pages 2179–2188, Atlanta, GA, USA, 2022. Association for Computing Machinery. ISBN 978-1-4503-9236-5. doi: 10.1145/3511808.3557320.

[365] Liang Qu, Wei Yuan, Ruiqi Zheng, Lizhen Cui, Yuhui Shi, and Hongzhi Yin. Towards Personalized Privacy: User-Governed Data Contribution for Federated Recommendation. In *Web Conference*, WWW '24, pages 3910–3918, New York, NY, USA, May 2024. ACM. ISBN 979-8-4007-0171-9. doi: 10.1145/3589334.3645690.

[366] J. Qin, X. Zhang, B. Liu, and J. Qian. A split-federated learning and edge-cloud based efficient and privacy-preserving large-scale item recommendation model. *Journal of Cloud Computing*, 12(1), 2023. ISSN 2192-113X.

[367] Xiaolin Zheng, Zhongyu Wang, Chaochao Chen, Jiashu Qian, and Yao Yang. Decentralized Graph Neural Network for Privacy-Preserving Recommendation. In *CIKM*, CIKM '23, pages 3494–3504, New York, NY, USA, October 2023. Association for Computing Machinery. ISBN 979-8-4007-0124-5. doi: 10.1145/3583780.3614834.

[368] A. Hard, K. Partridge, C. Nguyen, N. Subrahmanya, A. Shah, P. Zhu, I.L. Moreno, and R. Mathews. Training keyword spotting models on Non-IID data with federated learning. In *INTERSPEECH*, pages 4343–4347, Oct 2020.

[369] I. Zualkernan, S. Dhou, J. Judas, A.R. Sajun, B.R. Gomez, and L.A. Hussain. An IoT System Using Deep Learning to Classify Camera Trap Images on the Edge. *Computers*, 11(1), 2022. doi: 10.3390/computers11010013.

[370] H. Wang, F. Li, W. Mo, P. Tao, H. Shen, Y. Wu, Y. Zhang, and F. Deng. Novel Cloud-Edge Collaborative Detection Technique for Detecting Defects in PV Components, Based on Transfer Learning. *Energies*, 15(21), 2022.

[371] Y. Chen, L. Chen, C. Hong, and X. Wang. Federated Multitask Learning with Manifold Regularization for Face Spoof Attack Detection. *Mathematical Problems in Engineering*, 2022. doi: 10.1155/2022/7759410.

[372] Nicholas Soures, Dhireesha Kudithipudi, Robin Bay Jacobs-Gedrim, Sapan Agarwal, and Matthew Marinella. Enabling On-Device Learning with Deep Spiking Neural Networks for Speech Recognition. *ECS Transactions*, 85(6):127, April 2018. ISSN 1938-5862.

[373] D. Guliani, L. Zhou, C. Ryu, T.-J. Yang, H. Zhang, Y. Xiao, F. Beaufays, and G. Motta. Enabling On-Device training of speech recognition models with federated dropout. In *ICASSP*, pages 8757–8761, May 2022.

[374] K.C. Sim, A. Chandorkar, F. Gao, M. Chua, T. Munkhdalai, and F. Beaufays. Robust continuous on-device personalization for automatic speech recognition. In *INTERSPEECH*, volume 6, pages 4451–4455, 2021.

[375] J. Park, S. Jin, J. Park, S. Kim, D. Sandhyana, C. Lee, M. Han, J. Lee, S. Jung, C. Han, and C. Kim. Conformer-Based on-Device Streaming Speech Recognition with KD Compression and Two-Pass Architecture. In *2022 IEEE Spoken Language Technology Workshop, SLT*, pages 92–99, 2023. ISBN 9798350396904. doi: 10.1109/SLT54892.2023.10023291.

[376] S. Pillay, A.J. MacDonald, R. Brito, H. Burd, G. O'Shea, A.J. Higginson, and M. Faragalli. Federated Learning on Edge Devices in a Lunar Analogue Environment. In *IGARSS*, pages 2006–2009, July 2023. ISBN 9798350320107. doi: 10.1109/IGARSS52108.2023.10282202.

[377] Qiyang Zhang, Xiangying Che, Yijie Chen, Xiao Ma, Mengwei Xu, Schahram Dustdar, Xuanzhe Liu, and Shangguang Wang. A Comprehensive Deep Learning Library Benchmark and Optimal Library Selection. *IEEE Transactions on Mobile Computing*, pages 1–14, 2023. ISSN 1558-0660. doi: 10.1109/TMC.2023.3301973.

[378] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Javier Fernandez-Marques, Yan Gao, Lorenzo Sani, Kwing Hei Li, Titouan Parcollet, Pedro Porto Buarque de Gusmão, and Nicholas D. Lane. Flower: A Friendly Federated Learning Research Framework, March 2022.

[379] Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. FedML: A Research Library and Benchmark for Federated Machine Learning, July 2020.

[380] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, March 2016. arXiv:1603.04467.

[381] Alexander Ziller, Andrew Trask, Antonio Lopardo, Benjamin Szymkow, Bobby Wagner, Emma Bluemke, Jean-Mickael Nounahon, Jonathan Passerat-Palmbach, Kritika Prakash, Nick Rose, Théo Ryffel, Zarreen Naowal Reza, and Georgios Kaissis. PySyft: A Library for Easy Federated Learning. In Muhammad Habib ur Rehman and Mohamed Medhat Gaber, editors, *Federated Learning Systems: Towards Next-Generation AI*, Studies in Computational Intelligence, pages 111–139. Springer, Cham, Cham, 2021. ISBN 978-3-030-70604-3.

[382] Anupam Borthakur, Asim Manna, Aditya Kasliwal, Dipayan Dewan, and Debdoot Sheet. FedERA: Framework for Federated Learning with Diversified Edge Resource Allocation. *Authorea Preprints*, Seb 2023. doi: 10.36227/techrxiv.24173898.v1.

[383] Dun Zeng, Siqi Liang, Xiangjing Hu, Hui Wang, and Zenglin Xu. FedLab: A Flexible Federated Learning Framework. *JMLR*, 24(100):1–7, 2023. ISSN 1533-7928.

[384] Jae Hun Ro, Ananda Theertha Suresh, and Ke Wu. Fedjax: Federated learning simulation with jax, 2021.

[385] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A Benchmark for Federated Settings, Dec 2019. arXiv:1812.01097.

[386] Mirian Hipolito Garcia, Andre Manoel, Daniel Madrigal Diaz, Fatemehsadat Mireshghallah, Robert Sim, and Dimitrios Dimitriadis. Flute: A scalable, extensible framework for high-performance federated learning simulations, 2022.

[387] Daniele Romanini, Adam James Hall, Pavlos Papadopoulos, Tom Titcombe, Abbas Ismail, Tudor Cebere, Robert Sandmann, Robin Roehm, and Michael A. Hoeh. PyVertical: A Vertical Federated Learning Framework for Multi-headed SplitNN, April 2021. arXiv:2104.00489.

[388] Patrick Foley, Micah J. Sheller, Brandon Edwards, Sarthak Pati, Walter Riviera, Mansi Sharma, Prakash Narayana Moorthy, Shih-han Wang, Jason Martin, Parsa Mirhaji, Prashant Shah, and Spyridon Bakas. OpenFL: the open federated learning library. *Physics in Medicine & Biology*, 67(21):214001, October 2022. ISSN 0031-9155.

[389] Weiming Zhuang, Xin Gan, Yonggang Wen, and Shuai Zhang. EasyFL: A Low-Code Federated Learning Platform for Dummies. *IEEE Internet of Things Journal*, 9(15):13740–13754, August 2022. ISSN 2327-4662.

[390] Konstantin Burlachenko, Samuel Horváth, and Peter Richtárik. FL_pytorch: optimization research simulator for federated learning. In *DistributedML*, pages 1–7, Dec 2021. doi: 10.1145/3488659.3493775.

[391] Sang-Soo Park, Dong-Hee Kim, Jun-Gu Kang, and Ki-Seok Chung. EdgeRL: A Light-Weight C/C++ Framework for On-Device Reinforcement Learning. In *ISOCC*, pages 235–236, oct 2021. ISSN: 2163-9612.

[392] Davide Nadalini, Manuele Rusci, Giuseppe Tagliavini, Leonardo Ravaglia, Luca Benini, and Francesco Conti. Pulp-trainlib: Enabling on-device training for risc-v multi-core mcus through performance-driven autotuning. In *SAMOS*, pages 200–216. Springer, 2022.

[393] Diarmuid O'Reilly-Morgan, Erika Duriakova, Elias Tragos, Neil Hurley, and Aonghus Lawlor. NodeRec+: A Light-weight Framework for Federated Recommender Systems. In *SIGIR*, SIGIR '25, pages 4076–4080, NY, USA, July 2025. Association for Computing Machinery. ISBN 979-8-4007-1592-1. doi: 10.1145/3726302.3730138.

[394] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, volume 32. Curran Associates, Inc., 2019.

[395] Atakan Aral and Vincenzo De Maio. Simulators and emulators for edge computing. In *Edge Computing: Models, Technologies and Applications*, pages 291–309. IET, June 2020. doi: 10.1049/PBPC033E_ch14.

[396] Qihua Zhou, Zhihao Qu, Song Guo, Boyuan Luo, Jingcai Guo, Zhenda Xu, and Rajendra Akerkar. On-Device Learning Systems for Edge Intelligence: A Software and Hardware Synergy Perspective. *IEEE Internet of Things Journal*, 8(15): 11916–11934, August 2021. ISSN 2327-4662. doi: 10.1109/JIOT.2021.3063147.

[397] Diana Marculescu, Dimitrios Stamoulis, and Ermao Cai. Hardware-Aware Machine Learning: Modeling and Optimization. In *ICCAD*, pages 1–8, November 2018. doi: 10.1145/3240765.3243479. ISSN: 1558-2434.

[398] Shail Dave, Riyadh Baghdadi, Tony Nowatzki, Sasikanth Avancha, Aviral Shrivastava, and Baoxin Li. Hardware Acceleration of Sparse and Irregular Tensor Computations of ML Models: A Survey and Insights. *Proceedings of the IEEE*, 109(10):1706–1752, October 2021. ISSN 1558-2256. doi: 10.1109/JPROC.2021.3098483.

[399] Kazim Ergun, Raid Ayoub, Pietro Mercati, and Tajana Simunic Rosing. Dynamic Reliability Management of Multi-gateway IoT Edge Computing Systems. *IEEE Internet of Things Journal*, 10(5):3864–3889, March 2023. ISSN 2327-4662, 2372-2541. doi: 10.1109/JIOT.2022.3185082.

[400] Atakan Aral and Ivona Brandic. Dependency Mining for Service Resilience at the Edge. In *SEC*, pages 228–242, October 2018. doi: 10.1109/SEC.2018.00024.

[401] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the Convergence of FedAvg on Non-IID Data, June 2020. arXiv:1907.02189.

[402] Viraaji Mothukuri, Reza M. Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, February 2021. ISSN 0167-739X.

[403] Xuefei Yin, Yanming Zhu, and Jiankun Hu. A Comprehensive Survey of Privacy-preserving Federated Learning: A Taxonomy, Review, and Future Directions. *ACM Comput. Surv.*, 54(6):131:1–131:36, July 2021. ISSN 0360-0300.

[404] Qipeng Xie, Siyang Jiang, Linshan Jiang, Yongzhi Huang, Zhihe Zhao, Salabat Khan, Wangchen Dai, Zhe Liu, and Kaishun Wu. Efficiency Optimization Techniques in Privacy-Preserving Federated Learning With Homomorphic Encryption: A Brief Survey. *IEEE Internet of Things Journal*, 11(14):24569–24580, July 2024. ISSN 2327-4662.

[405] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[406] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[407] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[408] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021.

[409] Haohe Liu, Qiao Tian, Yi Yuan, Xubo Liu, Xinhao Mei, Qiuqiang Kong, Yuping Wang, Wenwu Wang, Yuxuan Wang, and Mark D Plumbley. Audioldm 2: Learning holistic audio generation with self-supervised pretraining. *arXiv preprint arXiv:2308.05734*, 2023.

[410] Deepanway Ghosal, Navonil Majumder, Ambuj Mehrish, and Soujanya Poria. Text-to-audio generation using instruction-tuned llm and latent diffusion model. *arXiv preprint arXiv:2304.13731*, 2023.

[411] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *CVPR*, pages 22500–22510, 2023.

[412] Hannah Rose Kirk, Bertie Vidgen, Paul Röttger, and Scott A Hale. Personalisation within bounds: A risk taxonomy and policy framework for the alignment of large language models with personalised feedback. *arXiv preprint arXiv:2303.05453*, 2023.

[413] Nir Kshetri. Cybercrime and privacy threats of large language models. *IT Professional*, 25(3):9–13, 2023.

[414] Herbert Woisetschläger, Alexander Isenko, Shiqiang Wang, Ruben Mayer, and Hans-Arno Jacobsen. Federated fine-tuning of llms on the very edge: The good, the bad, the ugly. *arXiv preprint arXiv:2310.03150*, 2023.

[415] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*, October 2021.

[416] James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontanon. FNet: Mixing Tokens with Fourier Transforms. In *NAACL-HLT*, pages 4296–4313, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.319.

[417] Goutham Ramakrishnan, Aditya Nori, Hannah Murfet, and Pashmina Cameron. Towards compliant data management systems for healthcare ml. *arXiv preprint arXiv:2011.07555*, 2020.

[418] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[419] Joel Castaño, Silverio Martínez-Fernández, Xavier Franch, and Justus Bogner. Exploring the carbon footprint of hugging face's ml models: A repository mining study. In *ESEM*, pages 1–12, 2023. doi: 10.1109/ESEM56168.2023.10304801.

[420] Thibault Pirson and David Bol. Assessing the embodied carbon footprint of iot edge devices with a bottom-up life-cycle approach. *Journal of Cleaner Production*, 322:128966, 2021. ISSN 0959-6526.

[421] Stefano Savazzi, Sanaz Kianoush, Vittorio Rampa, and Mehdi Bennis. A framework for energy and carbon footprint analysis of distributed and federated edge learning. In *PIMRC*, pages 1564–1569. IEEE, 2021.

Received 28 September 2025; revised 12 March xxxx; accepted 5 June xxxx