AI

Review

A Comprehensive Review and a Taxonomy of Edge Machine Learning: Requirements, Paradigms, and Techniques

Wenbin Li *, Hakim Hacid, Ebtesam Almazrouei and Merouane Debbah

Technology Innovation Institute, Abu Dhabi P.O. Box 9639, United Arab Emirates; hakim.hacid@tii.ae (H.H.); ebtesam.almazrouei@tii.ae (E.A.); merouane.debbah@tii.ae (M.D.)
* Correspondence: wenbin.li@tii.ae

Abstract: The union of Edge Computing (EC) and Artificial Intelligence (AI) has brought forward the Edge AI concept to provide intelligent solutions close to the end-user environment, for privacy preservation, low latency to real-time performance, and resource optimization. Machine Learning (ML), as the most advanced branch of AI in the past few years, has shown encouraging results and applications in the edge environment. Nevertheless, edge-powered ML solutions are more complex to realize due to the joint constraints from both edge computing and AI domains, and the corresponding solutions are expected to be efficient and adapted in technologies such as data processing, model compression, distributed inference, and advanced learning paradigms for Edge ML requirements. Despite the fact that a great deal of the attention garnered by Edge ML is gained in both the academic and industrial communities, we noticed the lack of a complete survey on existing Edge ML technologies to provide a common understanding of this concept. To tackle this, this paper aims at providing a comprehensive taxonomy and a systematic review of Edge ML techniques, focusing on the soft computing aspects of existing paradigms and techniques. We start by identifying the Edge ML requirements driven by the joint constraints. We then extensively survey more than twenty paradigms and techniques along with their representative work, covering two main parts: edge inference, and edge learning. In particular, we analyze how each technique fits into Edge ML by meeting a subset of the identified requirements. We also summarize Edge ML frameworks and open issues to shed light on future directions for Edge ML.

Keywords: edge artificial intelligence; edge machine learning; distributed learning; distributed inference; federated learning; split learning; transfer learning; model approximation; model compression

check for updates

Citation: Li, W.; Hacid, H.; Almazrouei, E.; Debbah, M. A Comprehensive Review and a Taxonomy of Edge Machine Learning: Requirements, Paradigms, and Techniques. *AI* **2023**, *4*, 729–786. https://doi.org/10.3390/ai4030039

Academic Editor: Gianni D'Angelo

Received: 22 June 2023
Revised: 6 August 2023
Accepted: 15 August 2023
Published: 13 September 2023

1. Introduction

The tremendous success of Artificial Intelligence (AI) technologies [1] in the past few years has been driving both industrial and societal transformations through domains such as Computer Vision (CV), Natural Language Processing (NLP), Robotics, Industry 4.0, Smart Cities, etc. This success is mainly brought by deep learning, providing the conventional Machine Learning (ML) techniques with the capabilities of processing raw data and discovering intricate structures [2]. Daily human activities are now immersed with AI-enabled applications from content search and service recommendation to automatic identification and knowledge discovery.

The existing ML models, especially deep learning models, such as GPT-4 [3], Segment Anything [4], OVSeg [5], Make-A-Video [6], and Stable Diffusion [7], tend to rely on complex model structures and large model size to provide competitive performances. For instance, the largest WuDao 2.0 model [8] trained on 4.9TB of data has surpassed state-of-the-art levels on nine benchmark tasks with a striking 1.75 trillion parameters. As a matter of fact, large models have clear advantages on multi-modality, multi-task, and benchmark performance. However, such models require a relatively very large training datasets to be built as well as a large amount of computing resources during the training and inference phases. This dependency makes them usually closed to public access, and unsuitable to be directly deployed for end devices or even the small/medium enterprise level to provide real-time, offline, or privacy-oriented services.

In parallel with ML development, Edge Computing (EC) was firstly proposed in 1990 [9]. The main principle behind EC is to bring the computational resources at locations closer to end-users. This was intended to deliver cached content, such as images and videos, that are usually communication-expensive, and prevent heavy interactions with the main servers. This idea has later evolved to host applications on edge computing resources [10]. The recent and rapid proliferation of connected devices and intelligent systems has been further pushing EC from the traditional base station level or the gateway level to the end device level. This offers numerous technical advantages such as low latency, mobility, and location awareness support to delay-sensitive applications [11]. This serves as a critical enabler for emerging technologies like 6G, extended reality, and vehicle-to-vehicle communications, to mention only a few.

Edge ML [12], as the ML instantiation powered by EC and a union of ML and EC, has brought the processing in ML to the network edge and adapted ML technologies to the edge environment. In this work, the edge environment refers to the end-user side pervasive environment composed of devices from both the base station level and the end device level. In classical ML scenarios, users run ML applications on their resource-constrained devices (e.g., mobile phones, and Internet of Things (IoT) sensors and actuators), while the core service is performed on the cloud server. In Edge ML, either optimized models and services are deployed and executed in the end-user's device, or the ML models are directly built on the edge side. This computing paradigm provides ML applications with advantages such as real-time immediacy, low latency, offline capability, enhanced security and privacy, etc.

In order to illustrate the transformative potential and versatility of Edge ML across different sectors, we briefly explore several application sectors where Edge ML has already demonstrated substantial impact. The applications represent just a fraction of the potential of Edge ML, while the breadth and depth of Edge ML applications are expanding with the technique's evolution and its increased adoption.

- **Healthcare:** In healthcare, Edge ML enables real-time patient monitoring and personalized treatment strategies. Wearable sensors and smart implants equipped with Edge ML can process data locally, providing immediate health feedback [13]. This advancement permits the early detection of health irregularities and swift responses to potential emergencies, while also maintaining patient data privacy by avoiding the need for data transmission for analysis. Furthermore, in telemedicine, Edge ML could be used to interpret diagnostic imaging locally and provide immediate feedback to remote healthcare professionals, improving patient care efficiency and outcomes.
- **Autonomous Vehicles:** Edge ML is a key enabler for the advancements in the field of autonomous vehicles, which includes both Unmanned Aerial Vehicles (UAVs) and self-driving cars. These vehicles are packed with a myriad of sensors and cameras that generate enormous amounts of data per second [14]. Processing this data in real-time is crucial for safe and efficient operation, and sending all the data to a cloud server is impractical due to latency and bandwidth constraints. Edge ML, with its capability to process data at the edge, can help in reducing the latency, and enhancing real-time responses.
- **Smart City:** Edge ML plays a crucial role in the realization of smart cities [15], where real-time data processing is paramount. Applications such as intelligent traffic light control, waste management, and urban planning greatly benefit from ML models that can analyze sensor data on-site and respond promptly to changes in the urban environment. Moreover, Edge ML can power public safety applications such as real-time surveillance systems for crime detection and prevention. Here, edge devices like surveillance cameras equipped with ML algorithms can detect unusual activities or

behaviours and alert relevant authorities in real time, potentially preventing incidents and enhancing overall city safety.

- **Industrial IoT (IIoT):** In the realm of Industrial IoT [16], Edge ML is instrumental in predictive maintenance and resource management. With ML models running at the edge, real-time anomaly detection can be carried out to anticipate equipment failures and proactively schedule maintenance. Additionally, Edge ML can optimize operational efficiency by monitoring production line performance, tracking resource usage, and automating quality control processes.

However, the Edge ML's core research challenge remains how to adapt ML technologies to edge environmental constraints such as limited computation and communication resources, unreliable network connections, data sensitivity, etc., while keeping similar or acceptable performance. Research work was carried out in the past few years tackling different aspects of this meta-challenge, such as: model compression [17], transfer learning [18], and federated learning [19].

With the above-mentioned promising results in diverse areas, we noticed that very little work has been realized to deliver a systematic view of relevant Edge ML techniques, rather focusing on Edge ML in specific contexts. One example worth reporting is Wang et al. [20,21], who present a comprehensive survey on the convergence of edge computing and deep learning, which covers aspects of hardware, communication, models, as well as edge applications and edge optimization. The work is a good reference as an Edge ML technology stack. On the other hand, the analysis of edge ML paradigms are rather brief without a comprehensive analysis of diverse related problems and the matching solutions. Abbas et al. [22] review the role and impact of the relevant ML techniques in addressing the safety, security, and privacy challenges in the specific context of the IoT systems. Mustafa et al. [23] center around two significant themes in edge computing: Wireless Power Transfer (WPT) and Mobile Edge Computing (MEC). Their work surveys the methodologies of offloading tasks in MEC and WPT to end devices, and analyzes how the conjunction of WPT and MEC offloading can help overcome limitations in smart device battery lifetime and task execution delay. Murshed et al. [24] introduce a machine learning survey at the network edge, for which the training, inference, and deployment aspects are briefly summarized, and the technique coverage is limited and only focuses on representative techniques such as federated learning and quantization.

Compared to the existing works that briefly review the representative techniques, our paper aims to provide a panoramic view of Edge ML requirements, and offers a comprehensive technique review for edge machine learning on the soft computing aspects of model training and model inference. Throughout our review process, we strive for comprehensiveness, including more than twenty technique categories and over fifty techniques in this paper. Our work fills a significant gap in the literature by providing a single point of reference that offers extensive coverage of the Edge ML field. The paper delivers a more complete picture of the landscape of Edge ML, allowing readers to understand the full breadth and depth of the available techniques, their respective advantages and limitations, and their fit within different Edge ML contexts.

In contrast with [23], our paper concentrates on the integration of ML techniques in the edge environment, dealing with the aspects of data processing, model compression, distributed inference, and advanced learning paradigms to explore a broader range of techniques and paradigms. In comparison with [24], which covers representative works in topics of training, inference, applications, frameworks, software, and hardware, our paper focuses on model training and inference computing aspects, including a far richer list of techniques (e.g., in the edge learning section, we include thirteen technique categories benefiting Edge ML, compared to the three training techniques presented in [24]). We also analyze in detail the Edge ML requirements to provide a broad taxonomy and show how each technique can satisfy different Edge ML requirements as a systematic review.

Specifically, our paper answers the three following questions:

- What are the computational and environmental constraints and requirements for ML on the edge?
- What are the Edge ML techniques to train intelligent models or enable model inference while meeting Edge ML requirements?
- How can existing ML techniques fit into an edge environment regarding these requirements?

To answer the three above questions, this review is realized by firstly identifying the Edge ML requirements, and then individually reviewing existing ML techniques and analyzing if and how each technique can fit into edge by fulfilling a subset of the requirements. Following this methodology, our goal is to be as exhaustive as possible in the work coverage and provide a panoramic view of all relevant Edge ML techniques with a special focus on machine learning for model training and inference at the edge. Other topics, such as Edge ML hardware [25] and edge communication [26], are beyond the scope of this paper. As such, we do not discuss them in this review.

The remainder of the paper is organized as follows: Section 2 introduces the Edge ML motivation driven by the requirements. Section 3 provides an overview of all the surveyed edge ML techniques. In Sections 4 and 5, we describe each technique and analyze them, respectively, in relation to Edge ML requirements. Section 6 summarizes the technique review part, and Section 7 briefly introduces the frameworks supporting Edge ML implementation. Section 8 identifies the open issues and future directions in Edge ML. Section 9 concludes our work and sheds light on future perspectives.

2. Edge Machine Learning: Requirements

In the context of machine learning, be it supervised learning, unsupervised learning, or a reinforcement learning, an ML task could be either a training or an inference. As in every technology, it is critical to understand the underlying requirements that ensure proper expectations. By definition, the edge infrastructure is generally resource-constrained in terms of the following: computation power, i.e., processor and memory; storage capacity, i.e., auxiliary storage; and communication capability, i.e., network bandwidth. ML models, on the other hand, are commonly known to be hardware-demanding, with computationally expensive and memory-intensive features. Consequently, the union of EC and ML exhibits both constraints from edge environment and ML models. When designing edge-powered ML solutions, requirements from both the hosting environment and the ML solution itself need to be considered and fulfilled for suitable, effective, and efficient results.

We introduce in this section the Edge ML requirements, structured in three categories: (i) ML requirements, (ii) EC requirements, and (iii) overall requirements, which are composite indicators from ML and EC for Edge ML performance. The three categories of requirements are summarized in Figure 1.

2.1. ML Requirements

We foresee five main requirements an ML system should consider: (i) Low Task Latency, (ii) High Performance, (iii) Generalization and Adaptation, (iv) Labelled Data Independence, and (v) Enhanced Privacy and Security. We detail these in the following.

- **Low Task Latency:** Task latency refers to the end-to-end processing time for one ML task, in seconds (s), and is determined by both ML models and the supporting computation infrastructure. Low task latency is important to achieve fast or real-time ML capabilities, especially for time-critical use-cases such as autonomous driving. We use the term task latency instead of latency to differentiate this concept from communication latency, which describes the time for sending a request and receiving an answer.
- **High Performance:** The performance of an ML task is represented by its results and measured by general performance metrics such as top-n accuracy, and f1-score in

percentage points (pp), as well as use-case-dependent benchmarks such as General Language Understanding Evaluation (GLUE) benchmark for NLP [27] or Behavior Suite for reinforcement learning [28].

- **Generalization and Adaptation:** The models are expected to learn the generalized representation of data instead of the task labels, so as to be easily generalized to a domain instead of specific tasks. This brings the models' capability to solve new and unseen tasks and realize a general ML directly or with a brief adaptation process. Furthermore, facing the disparity between learning and prediction environments, ML models can be quickly adapted to specific environments to solve the environmental specific problems.
- **Enhanced Privacy and Security:** The data acquired from edge carry much private information, such as personal identity, health status, and messages, preventing these data from being shared in a large extent. In the meantime, frequent data transmission over a network threatens data security as well. The enhanced privacy and security requires the corresponding solution to process data locally and minimize the shared information.
- **Labelled Data Independence:** The widely applied supervised learning in modern machine learning paradigms requires large amounts of data to train models and generalize knowledge for later inference. However, in practical scenarios, we cannot assume that all data in the edge are correctly labeled. The independence of labelled data indicates the capability of an Edge ML solution to solve one ML task without labelled data or with few labelled data.

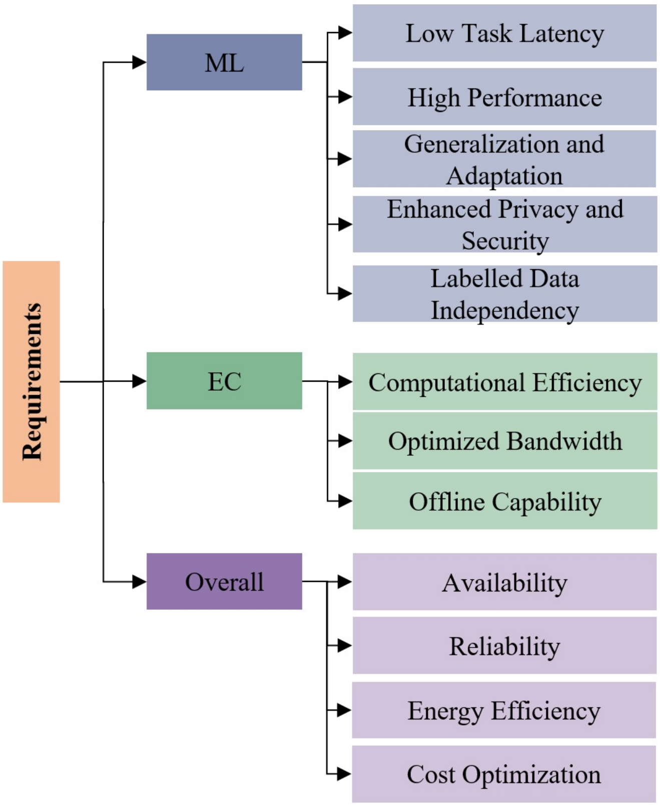


Figure 1. Edge ML Requirements.

2.2. EC Requirements

Three main edge environmental requirements of EC impact the overall Edge ML technology: (i) Computational Efficiency, (ii) Optimized Bandwidth, and (iii) Offline Capability, summarized below.

- **Computational Efficiency:** Refers to the efficient usage of computational resources to complete an ML task. This includes both processing resources measured by the number of arithmetic operations (OPs), and the required memory measured in MB.
- **Optimized Bandwidth:** Refers to the optimization of the amount of data transferred over network per task, measured by MB/Task. Frequent and large data exchanges over a network can raise communication and task latency. An optimized bandwidth usage expects Edge ML solutions to balance the data transfer over the network and local data processing.
- **Offline Capability:** The edge connectivity of edge devices is often weak and/or unstable, requiring operations to be performed on the edge directly. The offline capability refers to the ability to solve an ML task when network connections are lost or without a network connection.

2.3. Overall Requirements

The global requirements are composite indicators from ML and environmental requirements for Edge ML performance. We specify four overall requirements in this category: (i) Availability, (ii) Reliability, (iii) Energy Efficiency, and (iv) Cost Optimization.

- **Availability:** Refers to the percentage of time (in percentage points (pp)) that an Edge ML solution is operational and available for processing tasks without failure. For edge ML applications, availability is paramount because these applications often operate in real-time or near-real-time environments, and downtime can result in severe operational and productivity loss.
- **Reliability:** Refers to the ability of a system or component to perform its required functions under stated conditions for a specified period of time. Reliability can be measured using various metrics such as Mean Time Between Failures (MTBF) and Failure Rate.
- **Energy Efficiency:** Energy efficiency refers to the number of ML tasks obtained per power unit, in Task/J. The energy efficiency is determined by both the computation and communication design of Edge ML solutions and their supporting hardware.
- **Cost optimization:** Similar to energy consumption, edge devices are generally low-cost compared to cloud servers. The cost here refers to the total cost of realizing one ML task in an edge environment. This is again determined by both the Edge ML software implementation and its supporting infrastructure usage.

It should be noted that, depending on the nature of Edge ML applications, one Edge ML solution does not necessarily fulfill all the requirements above. The exact requirements for each specific Edge ML application vary according to each requirement's critical level to an application. For example, for autonomous driving, the task latency requirement is much more critical than the power consumption and cost optimization requirements.

3. Techniques Overview

Figure 2 shows a global view of edge Machine Learning techniques reviewed in this paper. We structure the related techniques into: (i) edge inference, and (ii) edge learning. The edge inference category introduces the technologies to accelerate the task latency of ML model inference. This is performed through, e.g., compressing existing models to consume less hardware resources or by dividing existing models into several parts for parallel inference collaboration. The edge learning category introduces solutions to directly build ML models on the edge side by learning locally from edge data. We detail the categories in the next sections.

Before introducing the details of each reviewed technique, we go through three basic machine learning paradigms, i.e., supervised learning, unsupervised learning, and reinforcement learning, to lay the theoretical foundation of ML. Briefly, supervised learning involves using an ML model to learn a mapping function between input data and the target variable from the labeled dataset. Unsupervised learning directly describes or extracts relationships in unlabeled data without any guidance from labelled data. Reinforcement learning is the process that an ML agent continuously interacts with its environment, performs actions to obtain awards, and learns to achieve a goal by the trial-and-error method.

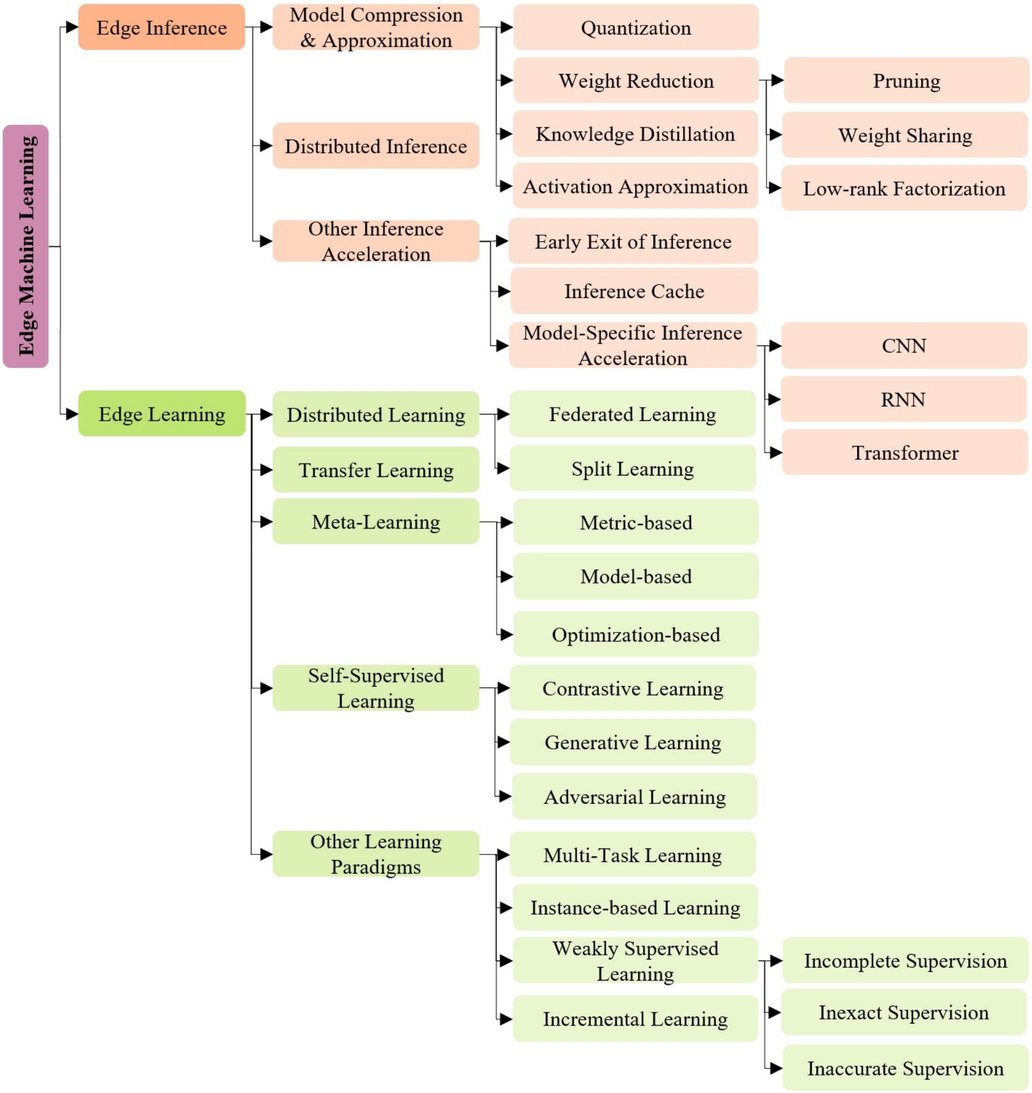


Figure 2. Edge ML Technique Overview.

Extending the work from [29], we give below the formal definition of the three basic learning paradigms. The objective of the definition is not merely to offer a conventional understanding of these methods, but more importantly, to create a conceptual bridge between these mainstream learning techniques and their specialized applications and adaptations for edge learning techniques, which helps to understand how these well-established AI techniques transform when applied to the context of Edge computing. Breakthroughs have been made in all three ML learning paradigms to derive meaningful data insights and bring intelligent capabilities, and the reviewed techniques in this paper all fit into the three general machine learning paradigms.

3.1. Supervised Learning

Supervised learning learns a function $f_\theta : X \rightarrow Y$ mapping inputs $x_i \in X$ to the corresponding outputs $y_i \in Y$ with the help of a labeled dataset D of m samples $D = \{(x_i, y_i)\}_{i=1}^{m}$, in which θ are ML model parameters (e.g., weights and biases in the case of neural network). The learning process aims at finding optimal or sub-optimal values for θ specific to the dataset D that minimizes an empirical loss function L_D through a training process (e.g., backward propagation in the case of neural network) as:

$$\theta_{SL} := \underset{\theta}{arg\,min}\; L_D(\theta), \tag{1}$$

where SL stands for "supervised learning". In practice, the labelled dataset D is often divided into training, validation, and testing datasets D^{tr}, D^{val}, D^{test} to train the model, guide the training process, and evaluate model performance after training, respectively [30].

Finding globally optimal values of θ_{SL} is computationally expensive, while in practice the training process is commonly an approximation to find sub-optimal θ_{SL} values guided by a predefined meta-knowledge ω including the initial model parameters θ, the training optimizer, and learning rate in the case of neural network, as:

$$\theta_{SL} \approx g_\omega(D, L_D), \tag{2}$$

where g_ω is an optimization procedure that uses predefined meta-knowledge ω, dataset D and loss function L_D to continuously update the model's parameters θ and output final θ_{SL}.

3.2. Unsupervised Learning

Training an ML model in the unsupervised manner is mostly similar to the supervised learning processing, except that the learned function $f_\theta : X \to X$ mapping input $x_i \in X$ to the same input x_i or other inputs. Unsupervised learning only uses unlabeled dataset $\bar{D}$ of n sample $\bar{D} = \{(x_i)\}_{i=1}^n$ to determine θ values specific to the dataset $\bar{D}$ that minimize an empirical loss function $L_{\bar{D}}$ through a training process, as:

$$\theta_{UL} := \underset{\theta}{arg\,min}\; L_{\bar{D}}(\theta), \tag{3}$$

where UL stands for "unsupervised learning". Furthermore, the same approximation is applied to unsupervised learning to efficiently fit the θ_{UL} to $\bar{D}$:

$$\theta_{UL} \approx g_\omega(\bar{D}, L_{\bar{D}}) \tag{4}$$

In addition to the above unsupervised learning paradigm which is used to train ML models, other unsupervised learning techniques such as clustering [31] apply predefined algorithms and computing steps to directly generate expected outputs (e.g., data clusters) from $\bar{D}$. In such context, the unsupervised learning approximates the values of specific algorithms' hyperparameters $\bar{\theta}_{UL}$ as:

$$\bar{\theta}_{UL} \approx g_\omega(\bar{D}, L_{\bar{D}}) \tag{5}$$

3.3. Reinforcement Learning

In the classic scenario of reinforcement learning where agents know the state at any given time step, the reinforcement learning paradigm can be formalized into a Markov Decision Process (MDP) as $M = (S, A, P, r, p_0, \gamma, T)$, where S is the set of states, A the set of actions, P the transition probability distribution defining $P(s_{t+1}|s_t, a_t)$ the transition probability from s_t to s_{t+1} via a_t, $r : S \times A \to \mathbb{R}$ the reward function, p_0 the probability distribution over initial states, $\gamma \in [0, 1]$ the discount factor prioritizing short- or long-term rewards by respectively decreasing or increasing it, T the maximum number of time steps. At a time step $t \in T$, a policy function π_θ, usually represented by a model in the case of deep reinforcement learning, is used to determine the action a_t that an agent performs at state $s_t : a_t = \pi_\theta(s_t)$, where θ is the parameters of the policy function; after the action a_t, the agent receives an award $r_t = r(s_t, \pi_\theta(s_t)), r_t \in \mathbb{R}$ and enters into a new state s_{t+1}. The interaction between agent and environment stops until a criterion is met, such as the rewards are maximized.

The objective of the reinforcement learning is to make agents learn to act and maximize the received rewards as follows:

$$\theta_{RL} := \underset{\theta}{arg\,min}\; \mathbb{E}_{traj} \sum_{t=1}^{T} \gamma^t r(s_t, \pi_\theta(s_t)), \tag{6}$$

where RL stands for "reinforcement learning", and $\mathbb{E}_{traj}$ is the expectation over possible trajectories $traj = (s_0, \pi_\theta(s_0), \ldots, s_T, \pi_\theta(s_T))$.

Similar to supervised and unsupervised learning, the sub-optimum of θ_{RL} are searched via an approximation process:

$$\theta_{RL} \approx g_\omega(M, L_M), \tag{7}$$

where g_ω is an optimization procedure that uses predefined meta-knowledge ω, the given MDP M and loss function L_M to produce final θ_{RL}.

4. Edge Inference

Edge inference techniques seek to enable large model inference on edge devices and accelerate the inference efficiency. The techniques can be categorized into three main groups: (i) model compression and approximation, (ii) distributed inference, and (iii) other inference acceleration techniques.

4.1. Model Compression and Approximation

A large amount of redundancy among the ML model parameters (e.g., neural network weights) has been observed [32], showing that a small subset of the weights is sufficient to reconstruct the entire neural network. Model compression and approximation are methods to transform ML models into smaller-size or approximate models with low-complexity computations. This is performed with the objective to reduce the memory use and the arithmetic operations during the inference, while keeping acceptable performances. Model compression and approximation can be broadly classified into three categories [33]: (i) Quantization, (ii) Weight Reduction, and (iii) Activation Function Approximation. We discuss these categories in the following:

4.1.1. Quantization

Quantization is the process of converting ML model parameters θ (i.e., weights and bias in neural networks) and activation outputs, represented in Floating Point (FP) format of high precision such as FP64 or FP32, into a low-precision format and then perform computing tasks such as training or inference. Different formats of quantization can be summarized as:

- **Low-Precision Floating-Point Representation**: A floating-point parameter describes binary numbers in the exponential form with an arbitrary binary point position such as 32-bit floating point (FP32), 16-bit floating point (FP16), 16-bit Brain Floating Point (BFP16) [34].
- **Fixed-Point Representation**: A fixed-point parameter [35] uses predetermined precision and binary point locations. Compared to a high-precision floating-point representation, the fixed-point parameter representation can offer faster, cheaper, and more power-efficient arithmetic operations.
- **Binarization and Terrorization**: Binarization [36] is the quantization of parameters into just two values, typically $-1, 1$ with a scaling factor. The terrorization [37] on the other hand adds the value 0 to the binary value set to express 0 in models.
- **Logarithmic Quantization**: In a logarithmic quantization [38], parameters are quantized into powers of two with a scaling factor. Work in [39] shows that a weight's representation range is more important than its precision in preserving network accuracy. Thus, logarithmic representations can cover wide ranges using fewer bits, compared to the other above-mentioned linear quantization formats.

In addition to the main exploited data types, AI-specific data formats and several quantization contributions exist in the literature and are introduced in Table 1.

Table 1. AI-specific data formats and model quantization works.

Work	**Contribution**	**Results and Insights**
Posit Representation [40]	The Posit is a data type designed to supersede IEEE Standard 754 floating-point numbers [41].	Provides larger dynamic range and higher accuracy than traditional floats, along with simpler hardware implementation and exception handling.
Fixed-Posit Representation [42]	Extended the Posit data type by maintaining a fixed count of the bits, unlike conventional posits that have a configurable number of regime and exponent bits.	Enhances the power and area efficiency compared to both Posit and 32-bit IEEE-754 floating-point representations. Improvements of up to 70% in power, 66% in area, and 26% in delay.
Tensor Cores [43]	NVIDIA's Tensor Cores is specialized hardware designed for performing the tensor/matrix computations with mixed-precision computations, enabling FP8 in the Transformer Engine, Tensor Float 32 (TF32) [44], and FP16.	Provide an order-of-magnitude higher performance with reduced precisions
8-bit Quantization [45]	An 8-bit quantization schema for MobileNet [46] on the ARM NEON-based implementation.	Model size reduction: 4×. Inference task latency reduction: up to 50%. Accuracy drop: 1.8% for COCO datasett [47].
SLQ [48]	A successive logarithmic quantization (SLQ) scheme to quantize the training error again when the quantization error is higher than a certain threshold.	Accuracy drop: less than 1.5% for AlexNet [49], SqueezeNet [50], and VGG-S [51] at 4 to 5-bit weight representation.
SLQ Training [52]	A specific training method for the Successive Logarithmic Quantization (SLQ) scheme.	Performance degradation of around 1% at 3-bit weight quantization.
Binary Network [53]	An accurate and efficient binary neural network for keyword-spotting applications, along with a binarization-aware training method for optimization.	Impressive 22.3 times speedup of task latency and 15.5 times storage-saving with only less than 3% accuracy drop on Google Speech Commands V1-12 task [54].
Binary Distilled Transformer [55]	A binarized multi-distilled transformer including a two-set binarization scheme, an elastic binary activation function with learned parameters, and a method for successively distilling models.	Accuracy of fully binarized transformer models approaches a full-precision BERT baseline on the GLUE language understanding benchmark within as little as 5.9%.
FMA [56]	A new class of Fused Multiply–Add (FMA) operators built on BFP16 arithmetic while maintaining accuracy comparable to that of the standard FP32.	Improved performance by 1.28–1.35× on ResNet compared to FP32.

To produce the corresponding quantized model, post-training quantization and quantization-aware training can be applied. Given an existing trained model, post-training quantization directly converts the trained model parameters and/or activation according to the conversion needs, to reduce model size and improve task latency during the inference phase. On the other hand, and instead of quantizing existing models, quantization-aware training is a method that trains an ML model by emulating inference-time quantization, which has proved to be better for model accuracy [57]. During the training of a neural network, quantization-aware training simulates low-precision behavior in the forward pass, while the backward pass based on backward propagation remains the same. The training process takes into account both errors from training data labels as well as quantization errors which accumulate in the total loss of the model, and hence the optimizer tries to reduce them by adjusting the parameters accordingly. Similarly, the work [58] analyzes the significant trade-off between energy efficiency and model accuracy, showing that the application of repair methods (e.g., ReAct-Net [59]), could largely offset the accuracy loss after quantization. Zhou et al. [60] analyzed various data precision combinations, concluding that accuracy deteriorates rapidly when weights are quantized to fewer than four bits.

The work [61] investigates the impact of data representation and bit-width reduction on CNN resilience, particularly in the context of safety-critical and resource-constrained systems. The results indicate that fixed-point data representation offers a superior trade-off between memory footprint reduction and resilience to hardware faults, especially for the LeNet-5 network, achieving a 4× memory footprint reduction at the expense of less than 0.45% critical faults without requiring network retraining.

Overall, moving from high-precision floating-point to lower-precision data representations is especially useful for ML models on edge devices with only low precision operation support such as Application-Specific Integrated Circuit (ASIC) and Field Programmable Gate Arrays (FPGA) to facilitate the trade-off between task accuracy and task latency. Quantization reduces the precision of parameters and/or activation, and thereby decreases the inference task latency by reducing the consumption of computing resources, while the workload reduction brought by cheaper arithmetic operations leads to energy and cost optimization as well. Moreover, quantization techniques make it feasible to run models on low-resource edge devices, increasing the availability of the application by allowing it to function with low connectivity. One the other hand, the techniques can also reduce the application robustness or resilience to hardware faults. Reduced precision leads to a model that is more susceptible to errors due to slight changes in the input or due to hardware faults, and thus decreases the reliability.

4.1.2. Weight Reduction

Weight reduction is a class of methods that removes redundant parameters from θ through pruning and parameter approximation. We reviewed the three following categories of methods in this paper:

- **Pruning**. The process of removing redundant or non-critical weights and/or nodes from models [17]: weight-based pruning removes connections between nodes (e.g., neurons in neural network) by setting relevant weights to zero to make the ML models sparse, while node-based pruning removes all target nodes from the ML model to make the model smaller.
- **Weight Sharing**. The process of grouping similar model parameters into buckets and reuse shared weights in different parts of the model to reduce model size or among models [62] to facilitate the model structure design.
- **Low-rank Factorization**. The process of decomposing the weight matrix into several low-rank matrices by uncovering explicit latent structures [63].

A node-based pruning method is introduced in [64] to remove redundant neurons in trained CNNs. In this work, similar neurons are grouped together following a similarity evaluation based on squared Euclidean distances and then pruned away. Experiments showed that the pruning method can remove up to 35% nodes in AlexNet with a 2.2% accuracy loss on the dataset of ImageNet [65]. A grow-and-prune paradigm is proposed in [66] to complement network pruning to learn both weights and compact Deep neural networks(DNNs) architectures during training. The method iteratively tunes the architecture with gradient-based growth and pruning of neurons and weight. Experimental results showed the compression ratio of 15.7× and 30.2× for AlexNet and VGG-16 network, respectively. This delivers a significant parameter and arithmetic operation reduction relative to pruning-only methods. In practice, pruning is often combined with a post-tuning or a retraining process to improve the model accuracy after pruning [67]. A Dense–Sparse–Dense training method is presented in [68], which introduces a post-training step to re-dense and recover the original model symmetric structure to increase the model capacity. This was shown to be efficient as it improves the classification accuracy by 1.1% to 4.3% on ResNet-50 [69], ResNet-18 [69], and VGG-16 [70]. The pruning method of SparseGPT is proposed in [71], showing that the large-scale generative pretrained transformer (GPT) family models can be pruned to at least 50% sparsity in one-shot, without any retraining, at minimal loss of accuracy. The driving idea behind this is an approximate sparse regression solver that runs entirely locally and relies solely on weight updates designed to preserve the input–output relationship for each layer. We also examine the layer-wise weight-pruning method presented in [72]. The method relies on a differential evolutionary layer-wise weight pruning operating in two distinct phases—a model-pruning phase, which analyzes each layer's pruning sensitivity and guides the pruning process, and a model-fine-tuning phase, where removed connections are considered for recovery to improve the model's capacity. Notably, the approach achieved impressive compression ratios of at least 10× across different models, with a standout 29× compression achieved for AlexNet. Another notable development in the field of structural pruning is the Dependency Graph (DepGraph) method [73]. This method represents a breakthrough in tackling the complex task of any structural pruning across a broad variety of neural architectures, from CNNs and RNNs to GNNs and Transformers. DepGraph introduces an automated system that models the dependency between layers to effectively group parameters that can be pruned together. The method demonstrates a promising performance across a multitude of tasks and architectures, including ResNet, DenseNet, MobileNet, and Vision transformer for images, GAT for graph, DGCNN for 3D point cloud, alongside LSTM for language.

The aforementioned pruning methods are static, as they permanently change the original network structure which may lead to a decrease in model capability. On the other hand, dynamic pruning [74] determines at run-time which layers, image channels (for CNN), or neurons would not participate in further model computing during a task. A dynamic channel pruning is proposed in [75]. This method dynamically selects which channel to skip or to process using feature boosting and suppression, which is achieved by the use of a side network trained together along the CNN to guide channel amplification and omission. This work achieved a 2× acceleration on ResNet-18 with 2.54% top-1 and 1.46% top-5 accuracy loss, respectively.

A multi-scale weight-sharing method is introduced in [76] to share weights among the convolution kernels of the same layer. To share kernel weights for multiple scales, the shared tuple of kernels is designed to have the same shape, and different kernels in the shared tuple are applied to different scales. With approximately 25% fewer parameters, the shared-weight ResNet model provides similar performance compared to the baseline ResNets [69]. Instead of looking up tables to locate the shared weight for each connection, HashedNets is proposed in [77] to randomly group connection weights into hash buckets via a low-cost hash function. These weights are tuned to adjust to the HashedNets weight sharing architecture with standard back-propagation during the training. Evaluations showed that HashedNets achieved a compression ratio of 64% with an around 0.7% accuracy improvement against a five-layer CNN baseline with the MNIST dataset [78]. Furthermore, the recent work [79] uses a gradient-based method to determine a threshold for attention scores at runtime, thereby effectively pruning inconsequential computations without significantly affecting model accuracy. Their proposed bit-serial architecture, known as LeOPArd, leverages this gradient-based thresholding to enable early termination, resulting in a significant boost in computational speed (1.9× on average) and energy efficiency (3.9× on average), with a minor trade-off in accuracy (<0.2% degradation). The related work of pruning is summarized in Table 2.

Structured matrices use repeated patterns within matrices to represent model weights to reduce the number of parameters. The circulant matrix, in which all row vectors are composed of the same elements and each row vector is shifted one element to the right relative to the preceding row vector, are often used as the structured matrix to provide a good compression and accuracy for RNN-type models [80,81]. The Efficient Neural Architecture Search (Efficient NAS) via parameter sharing is proposed in [82], in which only one shared set of model parameters is trained for several model architectures, also known as child models. The shared weights are used to compute the validation losses of different architectures. Sharing parameters among child models allows efficient NAS to deliver strong empirical performances for neural network design and use fewer GPU FLOP than automatic model design approaches. The NAS approach has been successfully applied to design model architectures for different domains [83] including CV and NLP.

Table 2. Summary of pruning works and main results.

Methods	Target Models	Main Results
Node-based Pruning [64]	CNNs	35% pruning, 2.2% accuracy loss
Grow-and-Prune [66]	DNNs	15.7×, 30.2× compression
Dense-Sparse-Dense [68]	ResNet, VGG-16	1.1% to 4.3% accuracy increase
SparseGPT [71]	GPT models	50% sparsity
Differential Layer-Wise Pruning [72]	LeNet, AlexNet, VGG16	10× to 29× compression
DepGraph Structural Pruning [73]	Any Model Structure	8–16× speedup, −6% to +0.3% accuracy change
Dynamic Channel Pruning [75]	ResNet-18	2× acceleration, 2.54% top-1, 1.46% top-5 accuracy loss
Multi-Scale Weight Sharing [76]	ResNets	25% fewer parameters
HashedNets [77]	5-layer CNN	64% compression, 0.7% accuracy improvement
Gradient-Based Pruning [79]	Transformers	1.9× speed, 3.9× energy efficiency, <0.2% accuracy loss

As for low-rank factorization, to find the optimal decomposed matrices to substitute the original weight matrix, Denton et al. [84] analyze three decomposition methods on pre-trained weight matrices: (i) singular-value decomposition, (ii) canonical polyadic decomposition, and (iii) blustering approximation. Experimental results on a 15-layer CNN demonstrate that singular-value decomposition achieved the best performance by a compression ratio of 2.4× to 13.4× on different layers along with a 0.84% point of top-one accuracy loss in the ImageNet dataset. A more recent work [85] proposes a data-aware low-rank compression method (DRONE) for weight matrices of fully-connected and self-attention layers in large-scale NLP models. As weight matrices in NLP models, such as BERT [86], do not show obvious low-rank structures, a low-rank computation could still exist when the input data distribution lies in a lower intrinsic dimension. The proposed method considers both the data distribution term and the weight matrices to provide a closed-form solution for the optimal rank-k decomposition. Experimental results show that DRONE can achieve 1.92× speedup on the Microsoft Research Paraphrase Corpus (MRPC) [87] task with only 1.5% loss in accuracy, and when DRONE is combined with distillation, it reaches 12.3× speedup on natural language inference tasks of MRPC, Corpus of Linguistic Acceptability (CoLA) [88], and Semantic Textual Similarity (STS) [89].

Overall, weight reduction directly reduces the ML model size by removing uncritical parameters. When performing tasks after weight reduction, ML models use less memory and require fewer arithmetic operations, which directly reduce the task latency with less workload and improve the computational resource efficiency. This is critical for time-sensitive applications to improve the perceived availability and responsiveness of the system. In addition, such improvements contribute to optimized energy consumption and cost. Similar to quantization, the weight-reduction techniques potentially make the model less resilient to certain types of hardware faults and such decrease the reliability. For example, a fault that affects a critical weight in a pruned network might have a bigger impact on the output than the same fault in an unpruned network, simply because there are fewer weights to 'absorb' the fault.

4.1.3. Knowledge Distillation

Knowledge Distillation is a procedure where a neural network is trained on the output of another network along with the original targets in order to transfer knowledge between the ML model architectures [90]. In this process, a large and complex network, or an ensemble model, is trained with a labelled dataset for a better task performance.

Afterwards, a smaller network is trained with the help of the cumbersome model via a loss function L, measuring the output difference of the two models. This small network should be able to produce comparable results, and in the case of over-fitting, it can even be made capable of replicating the results of the cumbersome network.

A knowledge-distillation framework for a fast objects detection task is proposed in [91]. To address the specific challenges of object detection in the form of regression, region proposals, and less-voluminous labels, two aspects are considered: (i) a weighted cross-entropy loss, to address the class imbalance, and (ii) a teacher-bounded loss, to handle the regression component and adaptation layers to better learn from intermediate teacher distributions. Evaluations with the datasets of Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL) [92], Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) [93], and COCO showed accuracy improvements by 3% to 5%. Wen et al. [94] argued that overly uncertain supervision of teachers can negatively influence the model's results. This is due to the fact that the knowledge from a teacher is useful but still not exactly right compared with a ground truth. Knowledge adjustment and dynamic temperature distillation are introduced in this work to penalize incorrect supervision and overly uncertain predictions from the teacher, making student models more discriminatory. Experiments on CIFAR-100 [95], CINIC-10 [96], and Tiny ImageNet [97] showed nearly state-of-the-art method accuracy.

MiniVit [98] proposes to compress vision transformers with weight sharing across layers and weight distillation. A linear transformation is added on each layer's shared weights to increase weight diversity. Three types of distillation for transformer blocks are considered in this work: (i) prediction-logit distillation, (ii) self-attention distillation, and (iii) hidden-state distillation. Experiments showed MiniViT can reduce the size of the pre-trained Swin-B transformer by 48% while achieving an increase of 1.0% in Top-1 accuracy on ImageNet.

Overall, knowledge distillation directly reduces the ML model size by simplifying model structures. Compared to the source model, the target model has a more compact and distilled structure with less parameters. Hence the workload of a task is reduced, leading to a better computational efficiency, higher availability, low task latency, and optimized energy consumption and cost. The distilled models potentially have higher reliability because they exert less stress on the hardware, reducing the likelihood of hardware faults or overheating. However, depending on the specific setup, the faults could have a greater impact on the model applications.

4.1.4. Activation Approximation

Besides the neural network's size complexity, i.e., in terms of the number of parameters, and architecture complexity, i.e., in the terms of layers, activation functions also impact the task latency of a neural network. Activation functions approximation replaces non-linear activation functions (e.g., *sigmoid* and *tanh*) in ML models with less computationally expensive functions (e.g., *ReLU*) to simplify the calculation or convert the computationally expensive calculation to series of lookup tables.

In an early work [99], the Piece-wise Linear Approximation of Non-linear Functions (PLAN) was studied. The *sigmoid* function was approximated by a combination of straight lines, and the gradient of the lines were chosen such that all the multiplications were replaced by simple shift operations. Compared to *sigmoid* and *tanh*, Hu et al. [100] show that *ReLU*, among other linear functions, is not only less computationally expensive but also proved to be more robust to handle the neural network vanishing gradient problem, in which the error dramatically decreases along with the back-propagation process in deep neural networks.

Activation approximation improves the computing resource usage by reducing the required number of arithmetic operations in ML models, and thus decreases the task latency with an acceptable increase in task error.

4.2. Distributed Inference

Distributed Inference divides ML models into different partitions and carries out a collaborative inference by allocating partitions to be distributed over edge resources and computing in a distributed manner [101].

The target edge resources to distribute the inference task can be broadly divided into three levels: (i) local processors in the same edge device [102], (ii) interconnected edge devices [101], and (iii) edge devices and cloud servers [103]. Among the three levels, an important research challenge is to identify the partition points of ML models by measuring data exchanges between layers to balance the usage of local computational resources and bandwidth among distributed resources.

To tackle the tightly coupled structure of CNN, a model parallelism optimization is proposed in [104], where the objective is to distribute the inference on edge devices via a decoupled CNN structure. The partitions are optimized based on channel group to partition the convolutional layers and then an input-based method to partition the fully connected layers, further exposing the high degree of parallelism. Experiments show that the decoupled structure can accelerate the inference of large-scale ResNet-50 by 3.21× and reduce 65.3% memory use with 1.29% accuracy improvement. Another distributed inference framework is also proposed in [105] to decompose a complex neural network into small neural networks and apply class-aware pruning on each small neural network on the edge device. The inference is performed in parallel while considering the available resources on each device. The evaluation shows that the framework achieves up to 17× speed-up when distributing a variant of VGG-16 over 20 edge devices, with around a 0.5% loss in accuracy.

Distributed inference can improve the end-to-end task latency by increasing the computing parallelism over a distributed architecture. At a price of bandwidth usage and network dependency, the overall energy efficiency and cost are optimized. By distributing the inference task, the load on individual devices is reduced, allowing more tasks to be processed concurrently, which can increase the availability of the ML application. In a distributed configuration, if one node fails, the task can be reassigned to another node, thereby increasing the overall system reliability.

4.3. Other Inference Acceleration Techniques

There exist other ways for accelerating inference in the literature. These have been categorized in a separate category as they are not as popular as the previously discussed techniques. These include: (i) Early Exit of Inference (EEoI), (ii) Inference Cache, and (iii) Model-Specific Inference Acceleration. We briefly review them in the following.

4.3.1. Early Exit of Inference (EEoI)

The Early Exit of Inference (EEoI) is powered by a deep network architecture augmented with additional side branch classifiers [106]. This allows prediction results for a large portion of test samples to exit the network early via these branches when samples can already be inferred with high confidence.

BranchyNet, proposed in [106], is based on the observation that features learned at an early layer of a network may often be sufficient for the classification of many data points. By adding branch structures and exit criteria to neural networks, BranchyNet is trained by solving a joint optimization problem on the weighted sum of the loss functions associated with the exit points. During the inference, BranchyNet uses the entropy of a classification result as a measure of confidence in the prediction at each exit point and allows the input sample to exit early if the model is confident in the prediction. Evaluations have been conducted with LeNet [78], AlexNet, and ResNet on MNIST, CIFAR-10 datasets, showing *BranchyNet* can improve accuracy and significantly reduce the inference time of the network by 2×–6×.

To improve the modularity of the *EEoI* methods, a plug-and-play technique named Patience-based Early Exit is proposed in [107] for single-branch models (e.g., ResNet, Transformer). The work couples an internal classifier with each layer of a pre-trained language model and dynamically stops inference when the intermediate predictions of the internal classifiers remain unchanged for a pre-defined number of steps. Experimental results with the ALBERT model [108] show that the technique can reduce the task latency by up to 2.42× and slightly improve the model accuracy by preventing it from overthinking and exploiting multiple classifiers for prediction.

EEoI can statistically improve the latency of inference tasks by reducing the inference workload at the price of a decrease in the accuracy. By increasing throughput, the technique leads to better availability. The side branch classifiers slightly increase the memory use during inference, while the task computational efficiency is higher as in most of cases where side branch classifiers can stop the inference earlier. In scenarios where a high level of certainty is needed, an early exit might introduce a higher probability of error, potentially compromising the reliability of the system. Generally, a correctly designed and trained EEoI technique is able to improve energy efficiency and optimize cost.

4.3.2. Inference Cache

Inference Cache saves models or models' inference results to facilitate future inferences of similar interest. This is motivated by the fact that ML tasks requested by nearby users within the coverage of an edge node may exhibit spatio-temporal locality [109]. For example, users within the same area might request recognition tasks for the same object of interest, which introduces redundant computation of deep learning inference.

Besides the *Cachier* [109], which caches ML models with edge server for recognition applications and shows 3× speedup in task latency, *DeepCache* [110] targets the cache challenge for a continuous vision task. Given input video streams, *DeepCache* firstly discovers the similarity between consecutive frames and identifies reusable image regions. During inference, *DeepCache* maps the matched reusable regions on feature maps and fills the reusable regions with cached feature map values instead of real Convolutional Neural Network (CNN) execution. Experiments show that *DeepCache* saves up to 47% inference execution time and reduces system energy consumption by 20% on average. A hybrid approach, semantic memory design (SMTM), is proposed in [111], combining inference cache with EEoI. In this work, low-dimensional caches are compressed with an encoder from high-dimensional feature maps of hot-spot classes. During the inference, SMTM extracts the intermediate features per layer and matches them with the cached features in fast memory: once matched, SMTM skips the rest of the layers and directly outputs the results. Experiments with AlexNet, GoogLeNet [112], ResNet50, and MobileNet V2 [113] show that SMTM can speed up the model inference over standard approaches with up to 2× and prior cache designs with up to 1.5× with only 1% to 3% accuracy loss.

Inference cache methods show their advantages of reducing task latency on continuous inference tasks or task batches. Since the prediction is usually made together with current input and previous caches, the accuracy can drop slightly. On the computational efficiency front, the cache lookup increases the computing workload and memory usage, while the global computational efficiency is improved across tasks, as the inference computation for each data sample does not start from scratch. Energy consumption and cost are reduced in the context of tasks sharing spatio-temporal similarity.

4.3.3. Model-Specific Inference Acceleration

Besides the above-mentioned edge inference techniques that can, in theory, be applied to most of ML model structures, other research efforts aim at accelerating the inference process for specific model structures. We briefly review the representative methods of inference acceleration for three mainstream neural network structures: (i) CNN, (ii) Recurrent Neural Network (RNN), and (iii) Transformers.

For CNN models, *MobileNets* [46] constructs small and low-latency models based on depth-wise separable convolution. This factorizes a standard convolution into a depth-wise convolution and a 1×1 convolution, as a trade off between latency and accuracy during inference. The latest version of *MobileNets* V3 [114] adds squeeze and excitation layers [115] to the expansion-filtering-compression block in *MobileNets* V2 [113]. As a result, it gives unequal weights to different channels from the input when creating the output feature maps. Combined with a later neural architecture search and NetAdapt [116], *MobileNets* V3-Large reaches 75.2% accuracy and 156ms inference latency on ImageNet classification with a single-threaded core on a Google Pixel 1 phone. *GhostNet* [117] also uses a depth-wise convolution to reduce the required high parameters and FLOPs induced by normal convolution: given an input image, instead of applying the filters on all the channels to generate one channel of the output, the input tensors are sliced into individual channels and the convolution is then applied only on one slice. During inference, x% of the input is processed by standard convolution and the output of this is then passed to the second depth-wise convolution to generate the final output. Experiments demonstrate that *GhostNet* can achieve higher recognition performance, i.e., 75.7% better top-1 accuracy than *MobileNets* V3 with similar computational cost on the ImageNet dataset. However, follow-up evaluations show that depth-wise convolution is more suitable for ARM/CPU and not friendly for GPU; thus, it does not provide a significant inference speedup in practice.

A real-time RNN acceleration framework is introduced in [118] to accelerate RNN inference for automatic speech recognition. The framework consists of a block-based structured pruning and several specific compiler optimization techniques including matrix reorder, load-redundant elimination, and a compact data format for pruned model storage. Experiments achieve real-time RNN inference with a Gated Recurrent Unit (GRU) model on an Adreno 640-embedded GPU and show no accuracy degradation when the compression rate is not higher than 10×.

Motivated by the way we pay visual attention to different regions of an image or correlate words in one sentence, a transformer is proposed in [119] showing encouraging results in various machine learning domains [120,121]. On the downside, transformer models are usually slower than competitive CNN models [122] in terms of task latency due to the massive number of parameters, quadratic-increasing computation complexity with respect to token length, non-foldable normalization layers, and lack of compiler-level optimizations. Current research efforts, such as [123,124], mainly focus on simplifying the transformer architecture to fundamentally improve inference latency, among which the recent EfficientFormer [125] achieves 79.2% top-1 accuracy on ImageNet-1K with only 1.6ms inference latency on an iPhone 12. In this work, a latency analysis is conducted to identify the inference bottleneck on different layers of vision transformer, and the EfficientFormer relies on a dimension-consistent structure design paradigm that leverages hardware-friendly 4D MetaBlocks and powerful 3D multi-scale hierarchical framework blocks along with a latency-driven slimming method to deliver real-time inference at MobileNet speed.

Generally, model-specific inference acceleration techniques lower the workload of an inference task and thus reduce the task latency within the same edge environment, resulting in higher availability. Though computational resources usage can vary among techniques, most work reports an acceptable accuracy loss in exchange for a considerable decrease in resources usage. In the case of model over-fitting, inference acceleration can improve the accelerated model accuracy. The total energy consumption and cost are therefore reduced. Under the assumption that the cached data are properly managed, the ML system provides consistent responses to the same input, which can enhance the reliability of the system.

5. Edge Learning

Edge learning techniques directly build *ML* models on native edge devices with local data. Distributed learning, transfer learning, meta-learning, self-supervised learning, and other learning paradigms fitting into Edge ML are reviewed in this section to tackle different aspects of Edge ML requirements.

5.1. Distributed Learning

Compared to cloud-based learning in which raw or pre-processed data are transmitted to cloud for model training, distributed learning (DL) in the edge divides the model training workload onto the edge nodes, i.e., edge servers and/or edge clients, to jointly train models with a cloud server by taking advantage of individual edge computational resources. Modern distributed learning approaches tend to only transmit locally updated model parameters or locally calculated outputs to the aggregation servers, i.e., cloud or edge, or the next edge node: in the server–client configuration, the aggregation server constructs the global model with all shared local updates [126]. On the other hand, in the peer-to-peer distributed learning setup, the model construction is achieved in an incremental manner along with the participating edge nodes together [127].

Distributed learning can be applied to all three basic ML paradigms, namely, supervised learning, unsupervised learning, and reinforcement learning. Instead of learning from one optimization procedure g_ω, distributed learning constructs a global model by aggregating the optimization results of all participant nodes, as formalized by Equation (8):

$$\theta \approx \bigsqcup_{i=1}^{n} g_{\omega^i}(\mathbb{D}^i, L_{\mathbb{D}^i}) \tag{8}$$

where g_{ω^i} is the optimization procedure driven by the meta-knowledge ω^i of the participant node $i, i \in n$, and n is the number of distributed learning nodes. $\mathbb{D}$ stands for the data used for learning, which can be for example the labelled dataset D for supervised learning, the unlabelled dataset $\bar{D}$ for unsupervised learning, or the MDP M for reinforcement learning. $L^i_{\mathbb{D}}$ is the corresponding loss on the given data $\mathbb{D}$ and $\bigsqcup$ is the aggregation algorithm (e.g., FedAvg [128] in the case of Federated Learning) to update the model by the use of all participants' optimization results (e.g., model parameters, gradients, outputs, etc.).

The edge distributed learning results into two major advantages:

- **Enhanced privacy and security:** Edge data often contain sensitive information related to personal or organizational matters that the data owners are reluctant to share. By transmitting only updated model parameters instead of the data, the distributed learning on the edge trains ML models in a privacy-preserving manner. Moreover, the reduced frequency of data transmission enhances the data security by restraining sensitive data only to the edge environment.
- **Communication and bandwidth optimization:** Uploading data to the cloud leads to a large transmission overhead and is the bottleneck of current learning paradigm [129]. A significant amount of communication is reduced by processing data in the edge nodes, and bandwidth usage optimized via edge distributed learning.

From the architectural perspective, there are three main organizational architectures [19,20] that exist to achieve distributed learning in the server–client configuration, as illustrated in Figure 3 and introduced as follows:

- **Cloud-enabled DL.** Given a number of distributed and interconnected edge nodes, cloud-enabled DL (see Figure 3a) constructs the global model by aggregating in the cloud the local models' parameters. These parameters are computed directly in each edge device. Periodically, the cloud server shares the global model parameters to all edge nodes so that the upcoming local model updates are made on the latest global model.
- **Edge-enabled DL.** In contrast to cloud-enabled DL, Edge-enabled DL (see Figure 3b) uses a local and edge server to aggregate model updates from its managed edge devices. Edge devices, with the management range of an edge server, contribute to the global model training on the edge aggregation server. Since the edge aggregation server is located near the edge devices, edge-enabled DL does not necessitate communications between the edge and the cloud, which thus reduces the communication latency and brings task offline capability. On the other hand, edge-enabled DL is often

resource-constrained and can only support a limited number of clients. This usually results in a degradation in task performance over time.

- **Hierarchical DL.** Hierarchical DL employs both cloud and edge aggregation servers to build the global model. Generally, edge devices within the range of a same-edge server transmit local data to the corresponding edge aggregation server to individually train local models, and then local models' parameters are shared with the cloud aggregation server to construct the global model. Periodically, the cloud server shares the global model parameters to all edge nodes (i.e., servers and devices), so that the upcoming local model updates are made on the latest global model. By this means, several challenges of distributed learning, such as Non-Identically Distributed Data (Non-IID) [130], imbalanced class [131], and the heterogeneity of edge devices [132] with diverse computation capabilities and network environments, can be targeted in the learning design. In fact, as each edge aggregation server is only responsible for training the local model with the collected data, the cloud aggregation server does not need to deal with data diversity and device heterogeneity across the edge nodes.

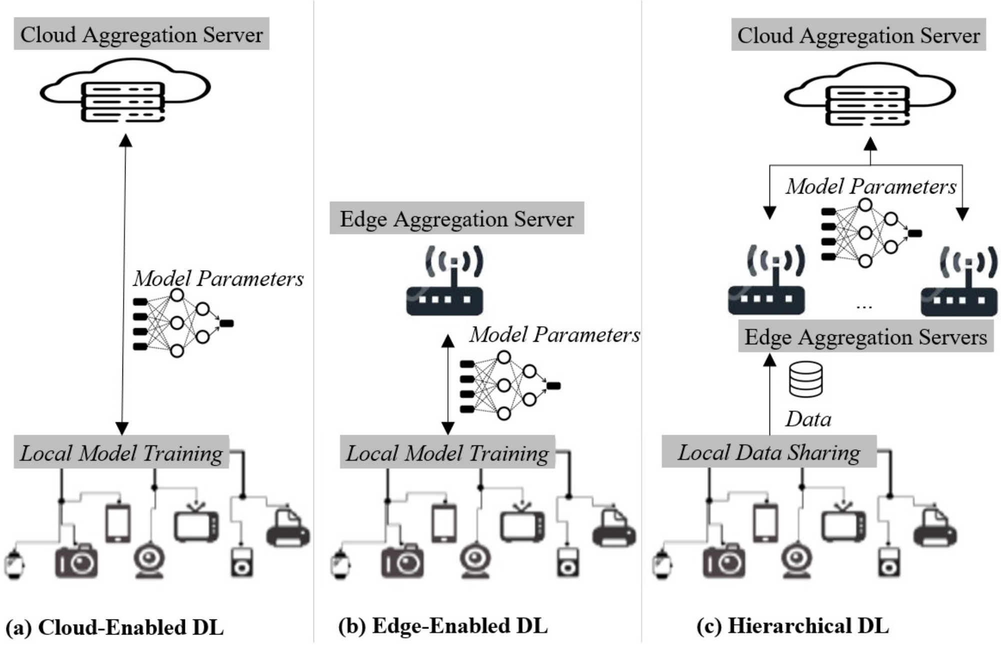


Figure 3. The distributed learning architectures available in the literature.

In the following, we review two distributed learning paradigms in the context of Edge ML: (i) federated learning, and (ii) split learning.

5.1.1. Federated Learning

Federated Learning (FL) [126] enables edge nodes to collaboratively learn a shared model while keeping all the training data on edge nodes, decoupling the ability to conduct machine learning from the need to store the data in the cloud. In each communication round, the aggregation server distributes the global model's parameters to edge training nodes, and each node trains its local model instance with newly received parameters and local data. The updated model parameters are then transmitted to the aggregation server to update the global model. The aggregation is commonly realized via federated average (FedAvg) [128] or Quantized Stochastic Gradient Descent (QSGD) [133] for neural networks, involving multiple local Stochastic Gradient Descent (SGD) updates and one aggregation by the server in each communication round.

FL is being widely studied in the literature. In particular, the survey in [19] summarizes and compares more than forty existing surveys on FL and edge computing regarding the covered topics. According to the distribution of training data and features among edge nodes, federated learning can be divided into three categories [134]: (i) Horizontal Federated Learning (HFL), (ii) Vertical Federated Learning (VFL), and (iii) Federated Transfer Learning (FTL). HFL refers to the federating learning paradigm where training data across edge nodes share the feature space but have different ones in samples. VFL federates models trained from data sharing the sample IDs but different feature space across edge nodes. Finally, FTL refers to the paradigm where data across edge nodes are correlated but differ in both samples and feature space.

HFL is widely used to handle homogeneous feature spaces across distributed data. In addition to the initial work of FL [126], showing considerable latency and throughput when performing the query suggestion task in mobile environments. HFL is highly popular in the healthcare domain [135] where it is, for instance, used to learn from different electronic health records across medical organizations without violating patients' privacy and improve the effectiveness of data-hungry analytical approaches. To tackle the limitation that HFL does not handle heterogeneous feature spaces, the continual horizontal federated learning (CHFL) approach [136] splits models into two columns corresponding to common features and unique features, respectively, and jointly trains the first column by using common features through HFL and locally trains the second column by using unique features. Evaluations demonstrate that CHFL can handle uncommon features across edge nodes and outperform the HFL models with are only based on common features.

As a more challenging subject than HFL, VFL is studied in [137] to answer the entity resolution question, which aims at finding the correspondence between samples of the datasets and learning from the union of all features. Since loss functions are normally not separable over features, a token-based greedy entity-resolution algorithm is proposed in [137] to integrate the constraint of carrying out entity resolution within classes on a logistic regression model. Furthermore, most studies of VFL only support two participants and focus on binary class logistic regression problems. A Multi-participant Multi-class Vertical Federated Learning (MMVFL) framework is proposed in [138]. MMVFL enables label sharing from its owner to other VFL participants in a privacy-preserving manner. Experiment results on two benchmark multi-view learning datasets, i.e., Handwritten and Caltech7 [139], show that MMVFL can effectively share label information among multiple VFL participants and match multi-class classification performance of existing approaches.

As an extension of the federated learning paradigm, FTL deals with the learning problem of correlated data from different sample spaces and feature spaces. FedHealth [140] is a framework for wearable healthcare targeting the FTL as a union of FL and transfer learning. The framework performs data aggregation through federated learning to preserve data privacy and builds relatively personalized models by transfer learning to provide adapted experiences in edge devices. To address the data scarcity in FL, an FTL framework for cross-domain prediction is presented in [141]. The idea of the framework is to share existing applications' knowledge via a central server as a base model, and new models can be constructed by converting a base model to their target-domain models with limited application-specific data using a transfer learning technique. Meanwhile, the federated learning is implemented within a group to further enhance the accuracy of the application-specific model. The simulation results on COCO and PETS2009 [142] datasets show that the proposed method outperforms two state-of-the-art machine learning approaches by achieving better training efficiency and prediction accuracy.

Besides the privacy-preserving nature of FL [143], and in addition to the research efforts on HFL, VFL, and FTL, challenges have been raised in federated learning oriented to security [144], communication [145], and limited computing resources [146]. This is important as edge devices usually have higher task and communication latency and are in vulnerable environments. In fact, low-cost IoT and Cyber-Physical System (CPS) devices are generally vulnerable to attacks due to the lack of fortified system security mechanisms. Recent advances in cyber-security for federated learning [147] reviewed several security attacks targeting FL systems and the distributed security models to protect locally residual data and shared model parameters. With respect to the parameter aggregation algorithm, the commonly used FedAvg employs the aggregation server to centralize model parameters; thus, attacking the central server breaks the FL's security and privacy. Decentralized FedAvg with momentum (DFedAvgM) [148] is presented on edge nodes that are connected by an undirected graph. In DFedAvgM, all clients perform stochastic gradient descent with momentum and communicate with their neighbors only. The convergence is proved under trivial assumptions, and evaluations with ResNet-20 on CIFAR-10 dataset demonstrate no significant accuracy loss when local epoch is set to 1.

From a communication perspective, although FL evades transmitting training data over a network, the communication latency and bandwidth usage for weights or gradients shared among edge nodes are inevitably introduced. The trade-off between communication optimization and the aggregation convergence rate is studied in [149]. A communication-efficient federated learning method with Periodic Averaging and Quantization (FedPAQ) is introduced. In FedPAQ, models are updated locally at edge devices and only periodically averaged at the aggregation server. In each communication round between edge training devices and the aggregation server, only a fraction of devices participate in the parameters aggregation. Finally, a quantization method is applied to quantize local model parameters before sharing with the server. Experiments demonstrate a communication–computation trade-off to improve communication bottleneck and FL scalability. Furthermore, knowledge distillation is used in communication-efficient federated learning technique FedKD [150]. In FedKD, a small mentee model and a large mentor model learn and distill knowledge from each other. It should be noted that only the mentee model is shared by different edge nodes and learns collaboratively to reduce the communication cost. In such a configuration, different training nodes have different local mentor models, which can better adapt to the characteristics of local datasets to achieve personalized model learning. Experiments with datasets on personalized news recommendations, text detection, and medical named entity recognition show that FedKD maximally can reduce 94.89% of communication cost and achieve competitive results with centralized model learning.

Federated learning on resource-constrained devices limit both communication and learning efficiency. The balance between convergence rate and allocated resources in FL is studied in [151], where an FL algorithm FEDL is introduced to treat the resource allocation as an optimization problem. In FEDL, each node solves its local training approximately till a local accuracy level is achieved. The optimization is based on the Pareto efficiency model [152] to capture the trade-off between the wall-clock training time and edge nodes energy consumption. Experimental results show that FEDL outperforms the vanilla FedAvg algorithm in terms of convergence rate and test accuracy. Moreover, computing resources can be not only limited but also heterogeneous at edge devices. A heterogeneity-aware federated learning method, Helios, is proposed in [153] to tackle the computational straggler issue. This implies that the edge devices with weak computational capacities among heterogeneous devices may significantly delay the synchronous parameter aggregation. Helios identifies each device's training capability and defines the corresponding neural network model training volumes. For straggling devices, a soft-training method is proposed to dynamically compress the original identical training model into the expected volume through a rotating neuron training approach. Thus, the stragglers can be accelerated while retaining the convergence for local training as well as federated collaboration. Experiments show that Helios can provide up to 2.5× training acceleration and maximum 4.64% convergence accuracy improvement in various collaboration settings.

Table 3 summarizes the reviewed works related to FL topics and challenges. Besides the efforts for security, communication, and resources, a personalized federated learning paradigm is proposed in [154], so that each client has their own personalized model as a result of federated learning. As the existence of a connected subspace containing diverse low-loss solutions between two or more independent deep networks has been discovered, the work combines this property with the model mixture-based personalized federated learning method for improved performance of personalization. Experiments on several benchmark datasets demonstrated that the method achieves consistent gains in both personalization performance and robustness to problematic scenarios possible in realistic services.

Table 3. FL related work.

Data and Features for FL		
HFL	VFL	FTL
[126,128,135,136]	[137,138]	[140,141]
Challenges		
Enhanced Security	Efficient Communication	Optimized Resources
[144,147,148]	[145,148–150]	[146,151,153]

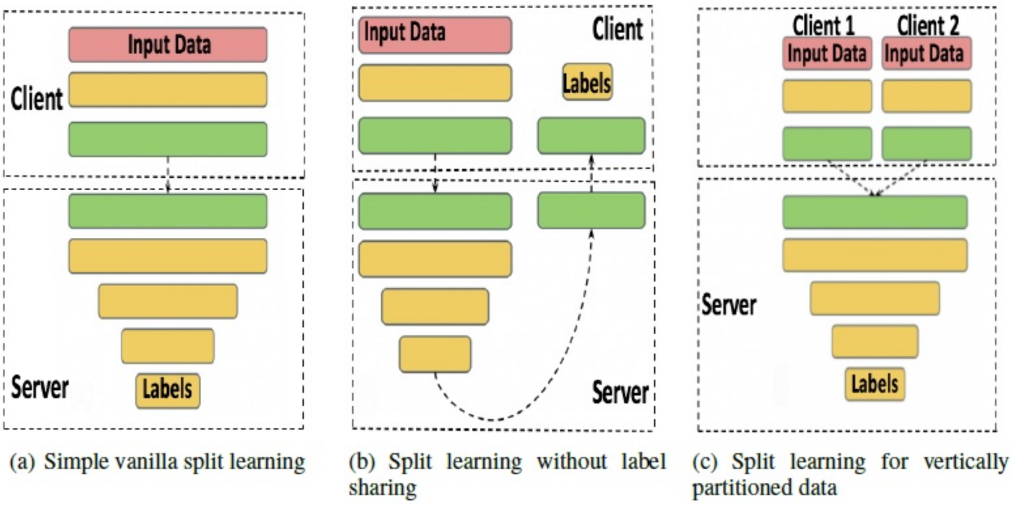
Overall, FL is designed primarily to protect data privacy during model training. Sharing models and performing distributed training increases the computation parallelism and reduces the communication cost, and thus reduces both the end-to-end training task latency and the communication latency. Moreover, specific FL design can provide enhanced security, optimized bandwidth usage and efficient computing resource usage. The edge-enabled FL as an instance of the edge-enabled DL can further bring offline capability to ML models. Generally, since local devices can continue training on the local data even if the network connection is down, which can improve the availability of the application during network failures or disruptions. Implementation requires the careful management and coordination of updates from multiple devices, handling devices with differing computational capabilities, and dealing with potential delays in communication. These factors can impact the reliability of applications of Federated Learning.

5.1.2. Split Learning

As another distributed collaborative training paradigm of ML models for data privacy, Split Learning (SpL) [155] divides neural networks into multiple sections. Each section is trained on a different node, either a server or a client. During the training phase, the forward process firstly computes the input data within each section and transmits the outputs of the last layer of each section to the next section. Once the forward process reaches the last layer of the last section, a loss is computed on the given input. The backward propagation shares the gradients reversely within each section and from the first layer of the last section to the previous sections. During the backward propagation, the model parameters are updated in the meantime. The data used during the training process are stored across servers or clients which take part in the collaborative training. However, none of the involved edge nodes can review data from other sections. The neural network split into sections and trained via SpL is called Split Neural Network (SNN).

The SpL method proposed in [155] splits the training between high-performance servers and edge clients, and orchestrates the training over sections into three steps: (i) training request, (ii) tensor transmission, and (iii) weights update. Evaluations with VGG and Resnet-50 models on MNIST, CIFAR-10, and ImageNet datasets show a significant reduction in the required computation operations and communication bandwidth by edge clients. This is because only the first few layers of SNN are computed on the client side, and only the gradients of few layers are transmitted during backward propagation. When a large number of clients are involved, the validation accuracy and convergence rate of SpL are higher than FL, as general non-convex optimization averaging models in a parameter space could produce an arbitrarily bad model [156].

The configuration choice to split a neural network across servers and clients are subject to design requirements and available computational resources. The work in [157] presents several configurations of SNN catering to different data modalities, of which Figure 4 illustrates three representative configurations: (i) in vanilla SpL, each client trains a partial deep network up to a specific layer known as the cut layer, and the outputs at the cut layer are sent to a server which completes the rest of the training. During the parameters update, the gradients are back-propagated at the server from its last layer until the cut layer. The rest of the back propagation is completed by the clients. (ii) In the configuration of SpL without label sharing, the SNN is wrapped around at the end layers of the servers. The outputs of the server layers are sent back to clients to obtain the gradients. During backward propagation, the gradients are sent from the clients to servers and then back again to clients to update the corresponding sections of the SNN. (iii) SpL for vertically partitioned data allows multiple clients holding different modalities of training data. In this configuration, each client holding one data modality trains a partial model up to the cut layer, and the cut layer from all the clients are then concatenated and sent to the server to train the rest of the model. This process is continued back and forth to complete the forward and backward propagation. Although the configurations show some versatile applications for SNN, other configurations remain to be explored.



(a) Simple vanilla split learning (b) Split learning without label sharing (c) Split learning for vertically partitioned data

Figure 4. Split Learning configurations [157].

Compared to FL, the SNN makes SpL a better option for resource-constrained environments. On the other hand, SpL performs slower than FL due to the relay-based training across multiple clients. To complement both learning paradigms, Split Federated Learning (SFL) [158] aims at bringing FL and SpL together for model privacy and robustness. SFL offers model privacy by network splitting and client-side model updates based on SpL, as well as shorter training latency by performing parallel processing across clients. Experiments demonstrate that SFL provides similar test accuracy and communication efficiency to SL, while significantly decreasing its computation time per global epoch than in SpL for multiple clients.

Overall, SpL largely improves training task latency by taking advantage of both server-side and edge-side computational resources. Compared to FL where all model gradients or weights are transmitted over a network, SpL only shares gradients of few layers of SNN and thus further optimizes the bandwidth usage. By reducing the amount of the transmitted data, split learning can help improve the availability of the application, especially in bandwidth-constrained environments. The SNN model performance is better compared to FL by avoiding FedAvg or QSGD during training. In addition to data privacy, which is enhanced by all distributed learning paradigms, SpL is excellent at preserving model privacy, as both the data and model structure are opaque across sections. Energy consumption and cost are thus reduced as a result of these SpL advantages. However, the implementation of Split Learning relies on proper partitioning handling, device heterogeneity management, and communication synchronisation, which can impact the reliability of the SpL applications.

5.2. Transfer Learning

Transfer Learning (TL) is inspired by humans' ability to transfer knowledge across domains. Instead of training models from scratch, TL aims at creating high-performance models on a target domain by transferring the knowledge from models of a different but correlated source domain [159]. The knowledge transfer in the context of transfer learning can be in the following three levels according to the discrepancy between domains:

- **Data Distribution.** The training data obtained in a specific spatial or temporal point can have different distribution to the testing data in an edge environment. The different data distribution, due to different facts such as co-variate shift [160], selection bias [161], and context feature bias [162], could lead to the degradation of model performance in a testing environment. The knowledge transfer between two different data distributions is a subtopic of transfer learning, known as Domain Adaptation (DA) [163].
- **Feature Space.** Contrary to the homogeneous transfer learning [18] which assumes that the source domain and the target domain consist of the same feature spaces, heterogeneous transfer learning tackles the (TL) case where the source and target domains have different feature spaces [164]. The heterogeneous transfer learning applies a feature space adaptation process to ease the difficulty to collect data within a target domain and expands the transfer learning to broader applications.
- **Learning Task Space.** Transfer learning also transfers knowledge between two specific learning tasks by use of the inductive biases of the source task to help perform the target task [165]. In this level, the data of the source and target task can have a same or different distribution and feature space. However, the specific source and target tasks are supposed to be similarly correlated either in a parallel manner, e.g., in the tasks of objects identification and person identification, or in a downstream manner, e.g., from a pretext learning task of image representation to a downstream task of an object detection task. It is worth mentioning that the knowledge generalization in an upstream manner from downstream tasks to out-of-distribution data is Domain Generalization (DG) [166].

As a learning paradigm focusing on the techniques to transfer knowledge between domains, transfer learning can be applied into all three basic learning categories, i.e., supervised learning, unsupervised learning, and reinforcement learning, for knowledge transfer between domains [165]. Based on the knowledge transfer process, two transfer learning techniques exist to build neural networks for the target domain: (i) Layer Freezing and (ii) Model Tuning. Layer Freezing is generally applied to transfer knowledge between domains that are correlated in a parallel manner and/or in situations where a target domain requests low training latency and has few training data. The process is summarized as follows.

1. Model Collection: An existing trained model on the source domain is acquired.
2. Layer Freezing: The first several layers from a source model are frozen to keep the previously learned representation, and the exact layers to freeze are determined by the source model layers which have learned the source data representation [167], i.e., usually the data-encoding part of a model.
3. Model Adjustment: The last few layers of the source model are deleted, and again the exact layers to delete are determined by the source model structure [168]. New trainable layers are added after the last layer of the modified source model to learn to turn the previous learned representation into outputs on the target domain.
4. Model Training: The updated model is trained with new data from the target domain.
5. Model Tuning: At last, an optional step is the tuning process usually based on model fine-tuning [169]. During this step, the entire newly trained model from the previous step is unfrozen and re-trained on the new data from the target domain with a low learning rate. The tuning process potentially further improves the model performance by adapting the newly trained representation to the new data.

On the other hand, Model Tuning is generally applied to transfer knowledge among domains that are correlated in a downstream manner and/or in situations where a target domain has sufficient training data. The process of tuning based transfer learning can be summarized as follows.

1. Model Pre-training: A model is pre-trained on the source domain to learn representations from source domain data.
2. Model Adjustment: As an optional step in tuning process, the last few layers of the source model are deleted, and new trainable layers are added after the last layer of the modified source model.
3. Model Tuning: The entire pre-trained model is trained on the new data from the target domain to map the learned representation to the target output.

During the two transfer learning processes, the parameters of the original model θ are updated to the new model parameters θ' with the dataset $\mathbb{D}'$ from the target domain through an optimization procedure g'_{ω}:

$$\theta' := g_{\omega'}(\mathbb{D}', L_{\mathbb{D}'}) \tag{9}$$

On the target domain, the meta-knowledge ω' and the optimization procedure $g_{\omega'}$ can be derived from the source domain during the transfer process; however, the focus of transfer learning is the knowledge transfer of model parameters from θ to θ'. Transfer learning building models based on previously learned knowledge in a correlated domain brings the following benefits.

- Training Efficiency. The speed of training new models is largely accelerated and uses much less computational resources compared to model training from scratch.
- Less Training Data. The model training or tuning process on the target model requires less training data, and this is especially useful in the case where there is a lot of data available from the source domain and relatively less data for target domain.
- Model Personalization. Transfer learning can quickly specialize pre-trained models to a specific environment and improve accuracy when the original pre-trained model cannot generalize well.

Transfer learning techniques are studied and compared in several surveys: an early study [164] associates the definition of transfer learning to the reasoning-based categories, and divides transfer learning into: (i) inductive transfer learning, (ii) transductive learning, and (iii) unsupervised learning, with respect to the source and target task spaces. To handle source and target feature space, homogeneous transfer learning is reviewed in [18,165], and heterogeneous transfer learning is analyzed in [164,165]. Regarding the domain adaptation for different data distributions, the state-of-the-art methods are summarized based on training loss in [170] for computer vision applications. In particular, recent research efforts tend to extend the scope of vanilla Domain Adaptation (DA) for different data distributions to different feature spaces or task spaces. The term "deep domain adaptation" is used in [170] to designate the methods that leverage deep neural networks and DA to solve both distribution shift and feature space differences. A Universal Domain Adaptation (UDA) method is described in [171] as a more general approach of transfer learning across a task space. UDA targets the supervised model transfer between domains where source and target have overlapped but have different label spaces. Without prior knowledge on the label sets from both domains, UDA is capable of classifying its samples correctly if it belongs to any class in the source label set or mark it as "unknown" otherwise. To address the unknown label classification, a Universal Adaptation Network (UAN) is introduced to quantify the transferability of each sample into a sample-level weighting mechanism based on both the domain similarity and the prediction uncertainty of each sample. Empirical results show that UAN works stably across different UDA settings and outperforms the state-of-the-art closed set, partial and open set domain adaptation methods.

The related works of transfer learning are highlighted in Table 4 and are introduced in the following. Regarding the layer freezing, one of the most popular application domains is healthcare, as the training data related to specific diseases can be difficult to obtain due to their rarity and the issue of privacy. Transfer Learning is applied in [172] to detect Parkinson's disease from speech symptom with layer freezing. In this work, the classification of patients with Parkinson's disease is realized with a CNN to analyze Mel-scale spectrograms in three different languages, i.e., Spanish, German, and Czech, via a transfer learning process. During the knowledge transfer, several consecutive layers are frozen to identify the layer topology characterizing the disease and others in the language. The results indicate that the fine-tuning of the neural network does not provide good performance in all languages, while the fine-tuning of individual layers improves the accuracy by up to 7%. Moreover, transfer learning among languages improves the accuracy up to 18% compared to a model training from scratch. Since fine-tuning and storing all the parameters is prohibitively costly and eventually becomes practically infeasible for pre-trained language models, a parameter-efficient fine-tuning method is proposed in [173] to optimize a small portion of the model parameters while keeping the rest fixed, drastically cutting down computation and storage costs.

Table 4. Highlighted related work of transfer learning.

Works	Methods	Insights
[172]	Layer Freezing	Frozen of consecutive layers to identify characterizing topology.
[173]	Parameter-Efficient Fine-tuning	Optimization of a small portion of parameters.
[174,175]	Large Model Tuning	Fine-tuning large pre-trained models.
[176]	Adapter Module Based Tuning	Few trainable parameters added per task.
[177]	Prompt Tuning	Competitive with scale, outperforms GPT-3's few-shots learning.
[178]	Prompt Tuning	Increased zero-shot accuracy with "Let's think step by step".
[179]	Instruction Fine-tuning	Large margin outperformance with FLan-PaLM 540B.
[180]	Hyper-parameter Tuning	Finding optimal model hyper-parameters.
[181]	Negative Transfer	Adversarial network to filter unrelated source data.

Concerning the model-tuning, fine-tuning large pre-trained models is an effective transfer mechanism in both CV [174] and NLP [175] domains. As the general fine-tuning creates an entirely new model for each downstream task, the method is not efficient when facing multiple downstream tasks. In fact, it results in the reproduction of the same-sized model multiple times. An adapter module-based tuning method is introduced in [176], where adapter modules extend the pre-trained models by only adding a few trainable parameters per task. The parameters of the original network remain fixed, yielding a high degree of parameter sharing. The experiment transferring BERT transformer to 26 diverse text classification tasks attain near state-of-the-art performance: on GLUE benchmark, the proposed method shows only 0.4% degradation compared to fine-tuned results, while adding only 3.6% parameters per task compared to the 100% parameter retraining of fine-tuning. Moreover, prompt tuning [177] is a simple yet effective method to learn prompts to perform specific downstream tasks without modifying models, which is especially useful when handling large language models and vision–language models. The study in [177] shows that prompt tuning becomes more competitive with scale: as models exceed billions of parameters, the proposed method matches the strong performance of model fine-tuning, and largely outperforms the few-shots learning of Generative Pre-trained Transformer 3 (GPT-3) [182]. As the prompt plays an important role in the model output, an interesting discovery is made in [178] to perform reasoning tasks with pre-trained Large Language Models (LLMs) by simply adding "Let's think step by step" before each output. The zero-shot accuracy is increased from 17.7% to 78.7% on MultiArith techmark [183] and from 10.4% to 40.7% on GSM8K benchmark [184] with an off-the-shelf 175B parameter model. As explored by the work, the versatility of this single prompt across very diverse reasoning tasks hints at untapped and understudied fundamental zero-shot capabilities of LLMs. This suggests high-level and multi-task broad cognitive capabilities may be extracted through simple prompting. Furthermore, an instruction fine-tuning method is described in [179] focusing on scaling the number of tasks, scaling the model size, and fine-tuning on chain-of-thought data. The resulting Flan-PaLM 540B instruction-fine tuned on 1.8K tasks outperforms PALM 540B by a large margin (+9.4% on average). At last, the tuning process is also applied to find optimal values for model hyper-parameters [180], which is however out of the scope of transfer learning.

Although transfer learning depends on the correlation between source and target domains to be effective, the similarities between domains are not always beneficial but can be misleading to the learning. Negative transfer [181] is the transfer process in which the target model is negatively affected by the transferred knowledge. It can be caused by several factors such as the domain relevance and the learner's capacity to find the transferable and beneficial part of the knowledge across domains. The work in [181] proposes a method relying on an adversarial network to circumvent negative transfer by filtering out unrelated source data. The harmful source data are filtered by a discriminator estimating both marginal and joint distributions to reduce the bias between source and target risks. The experiments involving four benchmarks demonstrate the effectiveness of filtering negative transfer and the improvement of model performance under negative transfer conditions.

Transfer Learning avoids building models from scratch and largely reduces the workload of training new models, which leads to the low training task latency, availability improvement, and efficient computation. In parallel, the required training data in the case of supervised learning is much less than required when training models from scratch. Thus, transfer learning can save expensive data-labeling efforts and drives conventional supervised learning more independent of labelled data. Regarding the edge requirements of model performance, transfer learning facilitates the construction of personalized models specific to individual edge environments and are expected to maintain a high model accuracy and reliability compared to generalized models. However, in practice, the model performance is determined by the quality of the source model, the training data in a target domain, and the correlation between the source and the target domains. Thus, the performance varies according to the specific configurations.

5.3. Meta-Learning

Taking the philosophy one step higher, and focusing on learning the learning process rather than specific tasks, meta-learning [185] is an advanced learning paradigm that observes and "remembers" previous learning experiences on multiple learning tasks, and then quickly learns new tasks from previous meta-data by analyzing the relation between tasks and solutions. The meta-learning solution for ML tasks is is realized in two levels [186]: (i) a base learner for each task, and (ii) a global meta-learner. The base learner solves task-specific problems and focuses on a single task, while the meta-learner integrates them using previous learned concepts to quickly learn the associated tasks. For a new task, meta-learning directly applies or updates the solution of the most similar task. In the case where no similar task is registered, meta-learning exploits the relation between tasks and solutions to propose an initial reference solution.

Meta-learning can also be applied to all three basic machine learning paradigms: supervised learning, unsupervised, and reinforcement learning. Regular supervised learning and unsupervised learning do not assume any given or predefined meta-knowledge. On the contrary, in supervised and unsupervised meta-learning, the goal is not only to realize a specific task but also to find the best meta-knowledge set, enabling the base learner to learn new tasks as quickly as possible. Regular reinforcement learning maximizes the expected reward on a single MDP, while meta reinforcement learning's intention is to maximize the expected reward over various MDPs by learning meta-knowledge. To summarize, instead of learning separately model parameters θ for all base learners, meta-learning actually focuses on learning the optimal or sub-optimal meta-knowledge ω^* for the global meta-learner, as formalized in Equation (10).

$$\omega^* := \underset{\omega}{arg\,min} \bigsqcup_{t=1}^{n} L_{\mathbb{D}^t}(g_{\omega^t}(\mathbb{D}^t, L_{\mathbb{D}^t})) \tag{10}$$

where g_{ω^t} is the optimization procedure driven by the meta-knowledge ω^t of the task $i, i \in n$, and n is the number of the considered base learner tasks. $\mathbb{D}^t$ is the data used for learning the base task t, $L_{\mathbb{D}^t}$ is the corresponding loss on the given data $\mathbb{D}^t$, $\bigsqcup$ is the aggregation algorithm (e.g., Model-Agnostic Meta-Learning (MAML) [187]) that finds the optimal meta-knowledge ω^* by minimizing the losses across different base learners.

Depending on the representation of the meta-knowledge, meta-learning techniques can be divided into three categories [29]: (i) metric-based meta-learning, (ii) model-based meta-learning, and (iii) optimization-based meta-learning.

1. Metric-based meta-learning learns the meta-knowledge in the form of feature space from previous tasks by associating the feature space with model parameters. New tasks are achieved by comparing new inputs, usually with unseen labels (also known as the query set), to example inputs (a.k.a. the support set) in the learned feature space. The new input will be associated to the label of the example input with which it shares the highest similarity. The idea behind metric-based meta-learning is similar to distance-based clustering algorithms, e.g., K-Nearest Neighbors (KNN) [188] or K-means [189], but with a learned model containing the meta-knowledge. Being simple in computation and fast at test-time with small tasks, metric-based meta-learning is inefficient when the tasks contain a large number of labels to compare, while the fact of relying on labelled examples makes the metric-based meta-learning both specialized at and limited by the supervised learning paradigm.
2. Model-based meta-learning relies on an internal or external memory component (i.e., a model) to save previous inputs and to empower the models to maintain a stateful representation of a task as the meta-knowledge. Specifically designed for fast training, the memory component can update its parameters in a few training steps with new data, either by the designed internal architecture or controlled by another meta-learner model [190]. When given new data on a specific task, the model-based meta-learning firstly processes the new data to train and alter the internal state of the model. Since the internal state captures relevant task-specific information, outputs can be generated for unseen labels of the same task or new tasks. Compared to the metric-based meta-learning, model-based meta-learning has a broader applicability to the three basic machine learning paradigms and brings flexibility and dynamics to the meta-learning technique via quick and dynamic model adjustment to new tasks and data.
3. Optimization-based meta-learning revises the gradient-based learning optimization algorithm so that the model is specialized at fast learning with a few examples, as the gradient-based optimization is considered to be slow to converge and inefficient with few learning data. Optimization-based meta-learning is generally achieved by a two-level optimization process [187]: base-learners are trained in a task-specific manner, while the meta-learner performs cross-task optimization in such a way that all base learners can quickly learn individual model parameters set for different tasks. Optimization-based meta-learning works better on wider task distributions and enables faster learning compared to the two previous meta-learning techniques. On the other hand, the global optimization procedure leads to an expensive computation workload, as each task's base-learner is considered [191].

In all the three meta-learning representations, one important characteristic of meta-learning is that during the testing phase, the resulting models are generalized and able to deal with the data labels, inputs, and the tasks on which models were not explicitly trained during the previous learning phase. Thus, data and task generalization as well as fast learning are the two main advantages of meta-learning.

Meta-learning widens the applicability of machine learning techniques and hence is applied into various domains such as few-shot learning in image classification [192], zero-shot learning for natural language processing [178], robot control [193], and reasoning [194]. Several surveys study the existing meta-learning techniques and works. In addition to [29], meta-learning in neural networks is studied in [191]. This work proposes a taxonomy and organizes the paper according to the representation of meta-knowledge, the meta-level optimizer, and the global objective of the meta-learning. Based on the type of the leveraging meta-data during the learning process, Vanschoren et al. [185] categorizes meta-learning techniques into: (i) learning from model evaluations, (ii) learning from task properties, and (iii) learning from prior models. Wang et al. [195] review the metric-based few-shot learning methods targeting the problem of data-intensive applications with little training data. Methods are grouped into three perspectives: data, model, and algorithm. The pros and cons of each perspective is analyzed in the work. The specific related work on Meta-Learning is summarized in Table 5 and described as follows.

The main challenge in meta-learning is to learn from prior experiences in a systematic and data-driven way [185]. For the metric-based meta-learning, a typical configuration of few-shot learning is N-way K-shot learning [196,197]. N-way refers to the number of classes N existing in the support set of the meta-testing phase. K-shot refers to the number of data samples K in each class in the support set. The few-shot learning tackles the supervised learning problem where models need to quickly generalize after training on few examples from each class. During the meta-training phase, the training dataset is divided into support set and query set, and the data embeddings are extracted from all training data, i.e., images. Each image from the query set is classified according to its embedding similarity with images from the support set. The model parameters are then updated by back-propagating the loss from the classification error of the query set. After training, the meta-testing phase classifies unseen labels from the meta-training phase (i.e., in Figure 5, images of unseen dog breeds are given during meta-testing) by use of the new support set.

Table 5. Summary of related work on Meta-Learning.

Works	Methods	Results and Insights
Metric-based: Few-shot learning [196,197]	N-way K-shot learning.	Efficient for small tasks, but inefficient for large label sets.
Metric-based: LSTM-based meta-learner [198]	Captures both short-term and long-term knowledge.	Rapidly converges a base learner to a locally optimal solution and in the meantime learns a task-common initialization as the base learner.
Metric-based: Zero-shot learning [199]	Two linear layers network modeling relationships among features, attributes, and classes.	Outperforms state-of-the-art approaches on several datasets by up to 17%.
Metric-based: Class distribution learning [200]	Represents each class as a probability distribution, defined as functions of the respective observed class attributes.	Leverages additional unlabeled data from unseen classes and improves estimates of their class-conditional distributions; Superior results in the same datasets compared to [199].
Metric-based: CLIP [201]	Pre-training of vision models from raw-text-describing images.	Benchmarks on over 30 CV datasets produce competitive results with fully supervised baselines
Model-based: MANN [190]	Model-based controller with an external memory component and Least Recently Used Access (LRUA) method.	Superior to LSTM in two meta-learning tasks.

Table 5. *Cont.*

Works	**Methods**	**Results and Insights**
Model-based: Drone adaptation [202]	Dynamics model with shared dynamics parameters and adaptation parameters.	Drone control with unknown payloads; autonomous determination of payload parameters and adjustment of flight control; performance improvement over non-adaptive methods on several suspended payload transportation tasks.
Optimization-based: MAML [187]	Gradient descent with model-specific updates.	General optimization tasks; simple and general, but higher-order derivatives potentially decrease performance.
Optimization-based: iMAML [203]	Approximation of higher-order derivatives.	General optimization tasks; more robust for larger optimization paths with same computational costs.
Optimization-based: online MAML [204]	Extension of MAML for online learning scenarios.	Continuous learning from newly generated data; strong in model specialization, but computation cost grows over time.

The work [198] proposes a Long Short-Term Memory (LSTM)-based meta-learner model in the few-shot regime. This is done to learn the exact optimization algorithm used to train another neural network classifier as the base learner: the meta-learner is trained to capture both short-term knowledge within a task and long-term common knowledge among all the tasks. This way, the meta-learner is able to rapidly converge a base learner to a locally optimal solution on each task and in the meantime learn a task-common initialization as the base learner. As a step further, zero-shot learning [205] does not require any example data as support set to perform new tasks or classify new classes which the model has not observed during the training phase. A simple zero-shot learning approach is introduced in [199] to model the relationships among features, attributes, and classes as a two linear layers network, where the weights of the second layer are not learned but are given by the environment. During the inference phase with new classes, the second layer is directly given so that the model can make predictions on the new labels. Despite being simple, the experiment results outperformed the state-of-the-art approaches on the datasets of Animals with Attributes (AwA) [206], SUN attributes (SUN) [207], and aPascal/aYahoo objects (aPY) [208] by up to 17% at the publication time.

Unlike [199] representing classes as fixed embeddings in a feature space, Verma et al. [200] represent each class as a probability distribution. The parameters of the distribution of each seen and unseen class are defined as functions of the respective observed class attributes. This allows the leveraging of additional unlabeled data from unseen classes and the improvement of the estimates of their class-conditional distributions via transductive or semi-supervised learning. Evaluations demonstrate superior results in the same datasets compared to [199]. In parallel to CV, the pre-trained large language models (LLMs) have proven to be excellent for few-shot learner [182] and zero-shot learner [178]. Furthermore, Contrastive Language-Image Pre-training (CLIP) [201] learns computer vision models directly from raw-text-describing images, which leverages a much boarder source of supervision instead of specific data labels. The pre-training of predicting "which caption goes with which image?" is realized on a dataset of 400 million image and text pairs from the Internet. After pre-training, natural language is used to reference learned visual concepts and describe new ones, enabling zero-shot transfer of the model to downstream tasks. The work matches the accuracy of the ResNet-50 model on ImageNet zero-shot without dataset-specific training, and benchmarks on over 30 CV datasets produce competitive results with fully supervised baselines.

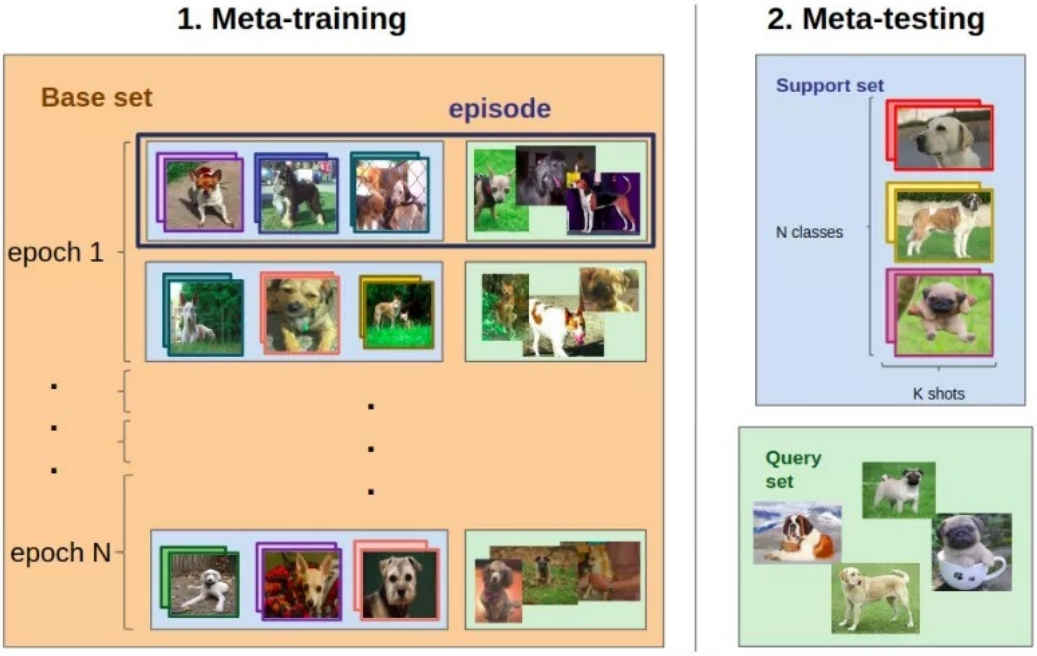


Figure 5. N-way K-shot learning setup [197].

As for model-based meta-learning, Memory-Augmented Neural Network (MANN) [190] contains a model-based controller, either feed-forward network or LSTM, to interact with an external memory component for memory retrieval and update. During training, the model learns to bind data representations to their labels regardless of the actual content of the data representation or label, and then the model maps these bound representations to appropriate classes for prediction. The memory writing and reading are powered by the proposed Least Recently Used Access (LRUA) method, and the MANN displays a performance superior to an LSTM in two meta-learning tasks on the Omniglot classification dataset [209] and sampled functions from a Gaussian process for regression.

A more concrete use case is illustrated in [202] to adapt drones to flight with unknown payloads, in which drones are expected to autonomously determine the payload parameters and adjust the flight control accordingly. During the training, a dynamics model with shared dynamics parameters and adaptation parameters are trained over K different payloads. During the testing, the robot infers the optimal latent variable representing the unknown payload by use of the learned dynamics parameters and the new sensed data. A model-predictive controller (MPC) then uses the trained dynamic model to plan and execute drone actions that follow the specified flight path. Experiments demonstrate the performance improvement of the proposed method compared to non-adaptive methods on several suspended payload transportation tasks.

With respect to optimization-based meta-learning, MAML [187] is a general optimization algorithm, compatible with any model that learns through gradient descent. In MAML, model-specific updates are made by one or more gradient descent steps. Instead of using second derivatives for meta-optimization of models, the meta-optimization proposes the First-Order MAML (FOMAML) to ignore the second derivative during MAML gradient computation to be less computationally expensive. MAML has obtained much attention due to its simplicity and general applicability. In the meantime, ignoring higher-order derivatives potentially decreases the model performance, and thus the iMAML [203] approximates these derivatives in a way that is less memory-consuming. While the iMAML is more robust for larger optimization paths, the computational costs roughly stay the same compared to MAML. Furthermore, online MAML [204] extends the MAML to online learning scenarios where models continuously learn in a potentially infinite time horizon from newly generated data and adapt to environmental changes. Being strong in model specialization, the computation cost, however, keeps growing over time.

Overall, meta-learning reduces supervised learning's dependency on labelled data by enabling models to learn new concepts quickly, which makes meta-learning particularly suitable for the edge side in the sense that it accelerates the training task. Another major advantage of meta-learning is the generalization capability that it brings to models to solve diverse tasks and the potential to realize general ML. Computational resource efficiency is higher for multiple model training, which leads to optimized energy consumption and cost optimization. Nevertheless, the global optimization procedure of optimization-based meta-learning may yet lead to expensive computation workload according to the number of base learners. Additional computation on the support dataset for metric-based meta-learning introduces extra workload during inference according to the dataset size (in such a case, the use of metric-based meta-learning is usually avoided). The generalization capability makes this application versatile and potentially more reliable, on the one hand; while on the other hand, its resource-intensive nature may impact its availability in resource-constrained environments.

5.4. Self-Supervised Learning

In contrast to supervised learning or reinforcement learning, human beings' learning paradigm is barely supervised and rarely reinforced. Self-Supervised Learning (SSL) is an unsupervised learning paradigm that uses self-supervision from original data and extracts higher-level generalizable features through unsupervised pre-training or optimization of contrastive loss objectives [186]. These learned feature representations are generalized and transferable, and thus can be tuned later to realize downstream tasks, and the pre-trained models are used as initial models to avoid training from scratch. During self-supervised learning, data augmentation techniques [210,211] are widely applied for contrast or generation purposes, and data labels are not required since pseudo labels can be estimated from trained models on similar tasks.

According to the loss objectives driving the training process, self-supervised learning can be summarized into three categories [212]: (i) generative learning, (ii) contrastive learning, and (iii) adversarial learning, as a combination of generative and contrastive learning. The architectures of the three categories are illustrated in Figure 6 to show the transformation from the traditional supervised learning process to a self-supervised learning approach, which can be particularly useful in Edge scenarios where label data are scarce.

- Generative Learning: Generative learning trains an encoder to encode the input into an explicit vector and a decoder to reconstruct the input from the explicit vector. The training simulates pseudo labels for unlabeled data and is guided by the reconstruction loss between the real input and the reconstructed input.
- Contrastive learning: Contrastive learning trains an encoder to respectively encode inputs into explicit vectors and measure similarity among inputs. The contrastive similarity metric is employed as the contrastive loss for model training. During the training, the contrastive learning calibrates label-free data against themselves to learn high-level generalizable representations.
- Adversarial Learning: Adversarial learning trains an encoder–decoder to generate fake samples and a discriminator to distinguish them from real samples in an adversarial manner. In other words, it learns to reconstruct the original data distribution rather than the samples themselves, and the distributional divergence between original and reconstructed divergence is the loss function to minimize during the training phase. The point-wise (e.g., word in texts) objective of the generative SSL is sensitive to rare examples and contrary to the high-level objective (e.g., texts) in classification tasks, which may result in inherent results without distribution data. Adversarial SSL abandons the point-wise objective and uses the distributional matching objectives for high-level abstraction learning. In the meantime, adversarial learning preserves the

decoder component abandoned by the contrastive SSL to stabilize the convergence with more expressiveness.

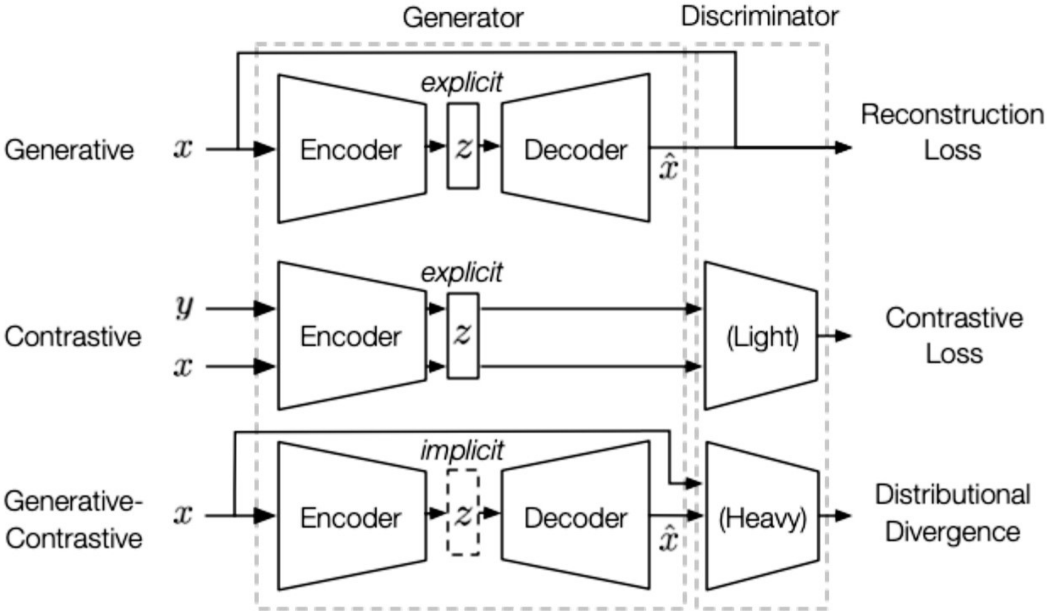


Figure 6. Self-Supervised Learning Architecture [212].

As an emerging field, self-supervised learning has received significant research attention. A comprehensive survey of the three above-mentioned SSL categories is presented in [212] including existing methods and representative works. Research works across several modalities of image, text, speech, and graphs are reviewed and compared in [213]. Digging in specific application domains, SSL works for visual feature learning and NLP representation learning are respectively analyzed in [214,215]; since graph-structured data are widely used and available over network, efforts on SSL of graph representation are compared in [216] to facilitate downstream tasks based on graph neural networks. The related work on SSL is summarized in Table 6 and introduced as follows.

Table 6. Summary of related work on Self-Supervised Learning.

Works	Methods	Results and Insights
Masked Prediction Method [182,217–219]	Generative SSL model that trains by filling in intentionally removed and missing data.	Effective to build pre-trained models in different areas like language, speech recognition, image regions, and graph edges.
Dat2Vec [217]	A general framework for SSL in speech, NLP and CV data. Predicts latent representations of the full input data based on a masked view of the input.	Performs competitively on major benchmarks in speech recognition, image classification, and natural language understanding.
MoCo [220]	Contrastive SSL with two encoders. Encodes two augmented versions of the same input images into queries and keys.	Outperforms its supervised pre-training counterpart in several CV tasks.
SimCLR [221]	A framework for contrastive learning of visual representations. Employs various image augmentation operations and contrastive cross-entropy loss.	Achieves a relative improvement of 7% over previous state-of-the-art, matching the performance of a supervised ResNet-50.
BiGANs [222]	Projects data back into the latent space to boost auxiliary supervised discrimination tasks.	Captures the difference at the semantic level without making any assumptions about the data.

Table 6. *Cont.*

Works	Methods	Results and Insights
BigBiGAN [223]	Extends the BiGAN model on representation learning by adding an encoder and correspondingly updating the discriminator.	Achieves the state-of-the-art in both unsupervised representation learning on ImageNet, and unconditional image generation.
Image Completion [224]	Uses adversarial SSL for image completion by training a global and local context discriminator networks.	Can complete images of arbitrary resolutions by filling in missing regions of any shape.
Image Inpainting [225]	Uses adversarial SSL for image completion in masked chest X-ray images.	Facilitates abnormality detection in the healthcare domain.
SSL with Federated Learning [226]	Empirical study of federated SSL for privacy preserving and representation learning with unlabeled data.	Tackles the non-IID data challenge of FL.
SSL with Meta Learning [186]	Reviews the intersection between SSL and meta-learning.	Shows how SSL can improve the generalization capability of models.
SSL with Transfer Learning [227]	The SSL applications within the transfer learning framework	Introduces methods for designing pre-training tasks across different domains.

Generative SSL often applies the masked prediction method [217] to train the model to fill in the intentionally removed and missing data. For instance, in the work [182], generative learning generates words in sentences in NLP by masking the words to generate in each step and updates the model parameters by minimizing the distance between the generated word and the masked word in the text. The same masking methods have proven to be effective to build pre-trained models by hiding speech time slices [228], image regions [218], and graph edges [219] in speech recognition.

In a multi-modal setting context, a more general framework is introduced in [217] as dat2vec for speech, NLP, and CV data. The idea is to predict latent representations of the full input data based on a masked view of the input in a self-distillation setup using a standard Transformer architecture. Instead of predicting modality-specific targets such as words, visual tokens, or units of human speech, data2vec predicts contextualized and multi-modal latent representations. Experiments on the major benchmarks of speech recognition Librispeech [229], image classification ImageNet-1K, and natural language understanding GLUE demonstrate a competitive performance to predominant approaches. Generative SSL is the mainstream method in NLP to train LLMs with texts from the Internet, while on the other hand SSL reveals less competitive results than contrastive SSL in CV domains of which the classification is the main objective.

Contrastive SSL creates multiple views of inputs [230] and compares them in the representation space to solve discrimination problems. During the learning, the distance between multi-views of the same data sample is minimized and the distance between different data samples is maximized. Negative sampling is a common for contrastive learning, but this process is often biased and time-consuming. Momentum Contrast (MoCo) [220] uses two encoders, an encoder and a momentum encoder, to encode two augmented versions of the same input images into queries and keys, respectively. During the training, positive pairs are constructed from queries of keys of current mini-batch, while negative pairs are constructed from queries of current mini-batch and keys from previous mini-batches to minimize the contrastive loss function InfoNCE [231]. In the experiments, MoCo outperforms its supervised pre-training counterpart in seven CV tasks on datasets including PASCAL and COCO.

To avoid explicitly using negative examples and prevent feature collapse, several data augmentation operations for images (e.g., original, crop, resize, color distort, Gaussian noise and blur, etc.) are introduced in [221] as a simple framework for contrastive learning (SimCLR) of visual representations. The learning with regularization and contrastive cross entropy loss benefits from a larger batch size and a longer training compared to the supervised counterpart: SimCLR achieves 76.5% top-1 accuracy, which is a 7% relative improvement over the previous state-of-the-art, matching the performance of a supervised

ResNet-50. Contrastive learning is found to be useful for almost all visual classification tasks due to the class-invariance modeling between different image instances but does not present a convincing result in the NLP benchmarks. The theory and applications of contrastive SSL to the domains such as NLP and graph learning where data are discrete and abstract is still challenging.

Inspired by the Generative Adversarial Networks (GAN) [232], adversarial SSL either focuses on generating with the learned complete representation of data or reconstructing the whole inputs with partial ones. Instead of learning from the latent distribution of task-related data distributions, Bidirectional Generative Adversarial Networks (BiGANs) [222] projects data back into the latent space to boost auxiliary supervised discrimination tasks. The learned distribution does not make any assumption about the data and thus captures the difference in the semantic level. BigBiGAN [223] discovers that a GAN with deeper and larger structures produces better results on downstream task and extends the BigGAN model on representation learning by adding an encoder and correspondingly updating the discriminator. Evaluations of the representation learning and generation capabilities of the BigBiGAN models achieve the state-of-the-art in both unsupervised representation learning on ImageNet, and unconditional image generation.

Adversarial SSL proves to be successful in image generation and processing, while still limited in NLP [233] and graph learning [234]. Alternatively, in-painting is a common use case for Adversarial SSL to reconstruct the entire inputs by filling in target regions with a relevant content, which allows the model to learn representations of different regions as well in order to process specific objects in images, detect anomalies in regions or reconstruct 3D images from 2D. A method of image completion is presented in [224] to complete images of arbitrary resolutions by filling in missing regions of any shape. A global discriminator and a local context discriminator are trained to distinguish real images from completed ones. The global discriminator assesses if the image is coherent as a whole, while the local discriminator ensures the local consistency of the generated patches at the completed region. The image completion network is then trained to fool both context discriminator networks. A similar work is reported in [225] to generate regions in masked chest X-ray images to facilitate the abnormality detection in the healthcare domain.

As the key method to alleviate the data labelling and annotation dependency, SSL demonstrates the boosting capability to power other learning paradigms, and the resulting solutions absorb merits from SLL and its incorporating learning paradigms. Federated SSL is empirically studied in [226] for both privacy preserving and representation learning with unlabeled data. A framework is also introduced to tackle the non-IID data challenge of FL. The intersection between SSL and meta-learning is reviewed in [186] showing models can best contribute to the improvement of model generalization capability. The models trained by SSL for pretext tasks with unlabeled data can be used by transfer learning to build state-of-the-art results. The self-supervised learning methods and their applications within the transfer learning framework are reviewed and summarized in [227].

Overall, the essential advantage of SSL is its capability to leverage the tremendous amount of unlabeled data to learn latent representations; thus, the labelled data dependency is largely alleviated during the learning process. The learned data representation via pretext task is in high-level generalization and can be easily used by downstream tasks to provide higher performance in various benchmarks, and thus SSL can improve reliability by learning a more comprehensive representation of the data. Although the arithmetic operations required by the training and task latency rises in certain learning setups with larger batch and more epochs, the testing performance is boosted as well. The final cost of a training task with SSL is much less compared to the same task requiring the manual labelling of data. While SSL can improve availability by reducing the need for labeled data, it often requires more computational resources for training as the model learns to understand the data structure and make predictions, which could cause an availability challenge on resource-constrained edge devices.

5.5. Other Learning Paradigms

Besides the four major learning techniques fitting to Edge ML, introduced in previously, in this section we briefly review relevant ML paradigms that potentially improve Edge ML solutions by satisfying a subset of its requirements.

5.5.1. Multi-Task Learning

Instead of building *n* models for *n* tasks, Multi-Task Learning (MTL) aims at using one ML model to realize multiple correlated tasks at the same time [235]. This is commonly achieved by training an entire model for all tasks, consisting of a commonly shared part among all tasks and a task-independent part. The commonly shared part of the model learns the common representation and task relations from all tasks' inputs, while the task-independent part computes and generates the final output for each task individually. During the multi-task learning, the model is trained in a way that data are mutualized among tasks to discover implicit task correlations. The learning process helps the model better find relevant features for each task and reduces the risk of over-fitting, so that all tasks' performance is improved via relevant features and tasks correlation [236]. Among the multiple tasks, each task can be a general learning task such as supervised tasks (e.g., classification or regression problems), unsupervised tasks (e.g., clustering problems), or reinforcement learning.

From the modelling perspective, MTL can be divided into: (i) hard parameter sharing and (ii) soft parameter sharing [237]. The hard parameter sharing generally shares the hidden layers among all tasks, while keeping several task-specific output layers. On the other hand, soft parameter sharing creates a set of parameters for each task of a similar structure, and the distance among the task parameters is then regularized during training [238] in order to encourage the parameters to be similar. The modelling structure is illustrated in Figure 7. The choice of the two modelling depends on the similarity among input data and task relation.

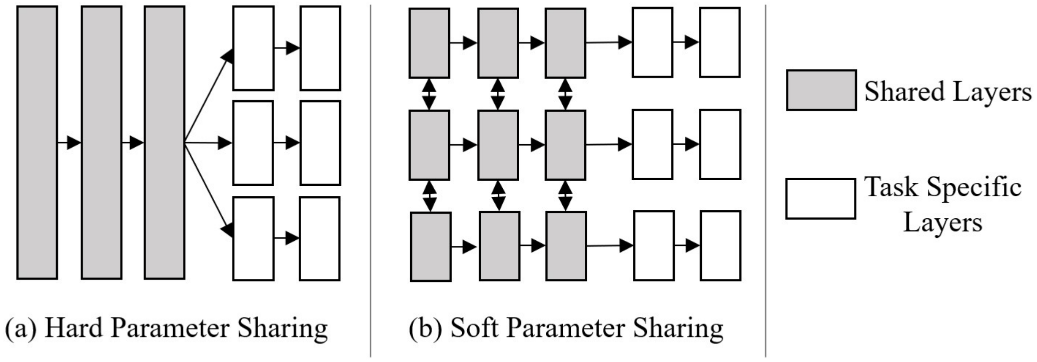


Figure 7. MTL modelling structure.

A number of works of MTL are surveyed and compared in [55,235,237], illustrating the overview of the literature and recent advances. One important research challenge of MTL lies in the multi-task modelling to take into account task and data relations for parameter structure sharing. An MTL model directly at the edge of the network is introduced in [239] for traffic classification and prediction. Based on autoencoders as the key building blocks for learning common features, the model anticipates information on the type of traffic to be served and the resource allocation pattern requested by each service during its execution. Simulation results produce higher accuracy and lower prediction loss compared to a single-task schema. The on-edge multi-task transfer learning is studied in [240], tackling data scarcity and resource constraints for task allocation. Instead of treating individual tasks equally, the work proposes to measure the impact of tasks on the overall decision performance improvement and quantify task importance with a Data-driven Cooperative Task Allocation (DCTA) approach. Experiments show that DCTA reduces task latency by 3.24×, and saves 48.4% energy consumption compared with the state-of-the-art when solving the task allocation with task importance for MTL.

Via common layers sharing among tasks, model parameters in MTL are largely decreased compared to multiple individual task models, and thus the computational workload is lower for the multiple task model. This leads to an improvement in task latency and computation efficiency. Via the learning of more relevant features and task correlations, the performance for correlated tasks is boosted, while the availability and the reliability are improved. Overall, in the context where multiple correlated tasks need to be performed, the MTL brings an efficient method for energy and cost optimization, making it suitable for the edge.

5.5.2. Instance-Based Learning

Instance-based Learning (IBL) [241], also called memory-based learning or lazy learning, compares new instances with already-seen instances to perform supervised learning tasks. Instead of learning an explicit representation mapping between features and instance labels, and predicting based on the learned representation, the key idea of IBL is to uniquely rely on seen instances to predict new instances. Commonly applied techniques of IBL are kNN, Radial Basis Function (RBF) [242], and Case-Based Reasoning (CBR) [243]. Among these techniques, kNN is widely used as a non-parametric model which simply retains all of the training instances and uses all of them to predict new instances based on a similarity distance between instances. In contrast to the metric-based meta-learning which generalizes the learned representation to unseen classes or tasks, IBL is suitable for rapidly realizing supervised learning tasks without generalization when the number of labels and retrained instances are small. Moreover, the technique can be easily extended to predict previously unseen instances by simply adding unseen instances in the prediction process. On the other hand, the computational complexity of IBL grows exponentially with the number of retained instances and the number of available labels, making the learning paradigm not suitable for performing large supervised tasks.

A Distributed storage and computation kNN algorithm (D-kNN) is introduced in [244]. It is based on cloud-edge computing for cyber–physical–social systems. The main contribution of the work lies in the optimization of distributed computation and storage of kNN and the efficient searching at distributed nodes to reduce the complexity of the algorithm. A CBR approach is described in [245] to optimize energy consumption in smart buildings. The approach is based on a multi-agent architecture deployed in a cloud environment with a wireless sensor network, where the agents learn human behaviors through CBR-enabled neural networks and manage device usage. Experiments in office buildings achieve an average energy saving of 41%.

IBL alleviates the labelled data dependency by reducing the amount of required labelled data to perform supervised learning tasks. Since the computational complexity of IBL scales with the problem complexity, the task latency, computation efficiency, availability and reliability, cost and energy consumption vary according to the specific task setup. The final performance of a model depends on the representativeness and the distribution of the instances as well.

5.5.3. Weakly Supervised Learning

Weakly Supervised Learning (WSL) comprises a family of learning techniques that train models to perform supervised tasks with noisy, limited, or imprecise labelled data from limited data labelling capacity [246]. Although the thorough labelling of edge data is not realistic to achieve by edge users in a continuous basis, the assumption can be made that users or edge applications can casually provide data labelling assistance under consensus. The casual data labelling in such a context may produce noisy, imprecise, or an insufficient amount of labelled data for supervised learning, and correspondingly requires specific learning paradigms to tackle the weak supervision problem.

According to the weakness of the labelled data quality, the problem of WSL can be divided into three categories [247]: (i) incomplete supervision, (ii) inexact supervision, and (iii) inaccurate supervision.

- Incomplete supervision refers to the problem that a predictive model needs to be trained from the ensemble of labelled and unlabeled data, where only a small amount of data is labelled, while other available data remain unlabeled.
- Inexact supervision refers to the problem that a predictive model needs to be trained from data with only coarse-grained label information. The multi-instance learning [248] is a typical learning problem of incomplete supervision where training data are arranged in sets, and a label is provided for the entire set instead of the data themselves.
- Inaccurate supervision concerns the problem that a predictive model needs to be trained from data that are not always labelled with ground-truth. A typical problem of inaccurate supervision is label noise [249], where mislabeled data are expected to be corrected or removed before model training.

Aiming the three problems of labelled data, weakly supervised learning brings techniques able to train models from data with low-quality labels and perform supervised tasks.

Existing work on WSL is introduced and summarized in [247] and then further developed in [250] by leveraging the data quantity and adaptability. In what relates to the incomplete supervision problems, active learning [251], inductive semi-supervised learning [252], and transductive learning [253] are three typical solutions for supplementing data labelling. The process of the three learning paradigms for incomplete supervision is illustrated in Figure 8 to provide a visual understanding of how these methods evolve from the conventional supervised learning paradigm and operate in scenarios of partial data labelling. Active learning is a technique where the learner interactively collects training data, typically by querying an oracle to request labels for new data in order to resolve ambiguity during the learning process [251]. Instead of querying all collected data points, the active learning goal is to only query the most representative data and use them for model training. The number of data used to train a model this way is often much smaller than the number required in conventional supervised learning, while the key idea behind it is that a learning paradigm can achieve higher accuracy with fewer training labels, if it is allowed to choose the data from which it learns [254].

Without queries, inductive semi-supervised learning labels the data with the help of the available labelled data and then trains the model [252]. The general process of semi-supervised learning is to firstly train a small model with the available labelled data to classify the unlabeled data, and then trains the final model with all data. Such an idea is driven by the assumption that similar data produce similar outputs in supervision tasks, and unlabeled data can be helpful to disclose which data are similar. Instead of training a small model to predict the unlabeled data, transductive learning [253] derives the values of the unknown data with unsupervised learning algorithms and labels the unlabeled data according to the clusters to which they belong. Then a model is trained by use of both the previously available and the newly labeled data. Compared to inductive semi-supervised learning, transductive learning considers all data when performing the data labeling that potentially improve the data labeling results. On the other hand, due to the fact no model is built for labelling, an update in the dataset will result in the repetition of the whole learning process. Active learning, inductive semi-supervised learning, and transductive learning are efficient in the situation where the acquisition of unlabeled data is relatively cheap while labeling is expensive.

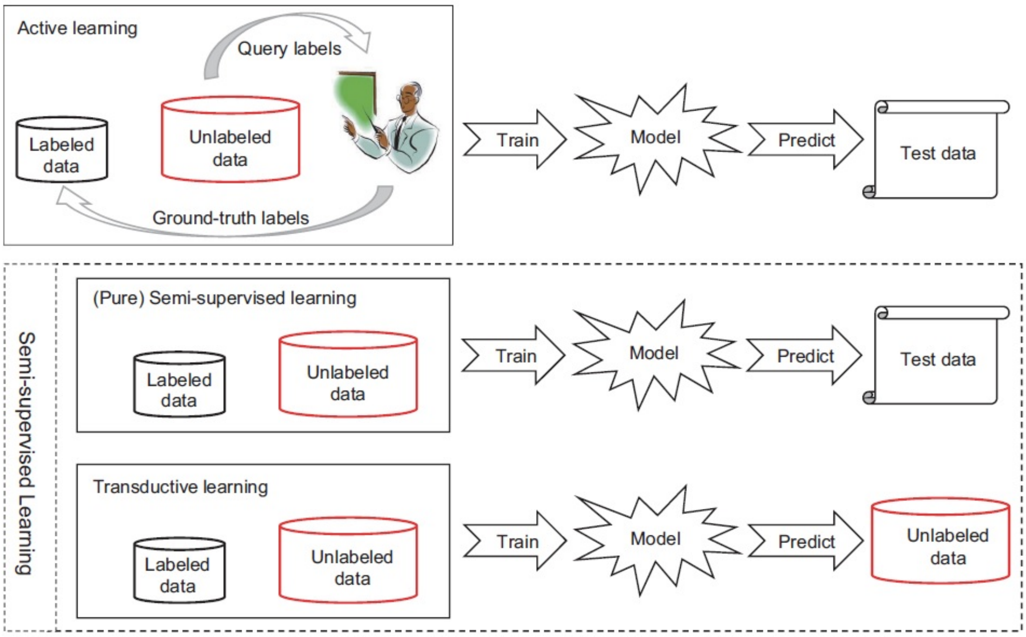


Figure 8. Incomplete Supervised Learning process [247].

Regarding the inexact supervision, multi-instance learning has been successfully applied to various tasks such as image classification [255], relation extraction [256], localization [257], and healthcare [258]. The main idea behind is to adapt single instance supervised learning algorithms for instance discrimination to the multi-instance representation for set discrimination. For the label noise problem, label smoothing [259] is a regularization technique that introduces noise for the labels and can improve both predictive performance and model calibration. The effect of label smoothing on model training with label noise is studied in [260,261], showing that the label-smoothing approach incorporating labeled instance centroid and its covariance reduces the influence of noisy labels during training [260]. Label smoothing is also competitive with loss-correction under label noise [261]. Moreover, loss correction is studied in [262] using a two-component mixture model as an unsupervised generative model of sample loss values during training to allow an online estimation of the probability that a sample is mislabelled, and the loss is corrected, relying on the network prediction.

Overall, targeting the learning problems where labelled data are scarce or imperfect, WSL mitigates the labelled data dependency. Focusing on the data labelling part, the task latency, cost, and energy consumption are optimized compared to manual labelling process. This enhance the availability of the model as fewer resources are required for data labeling. The quality of labels in WSL is often lower than in fully supervised learning, which may impact the reliability of the model's predictions.

5.5.4. Incremental Learning

Incremental learning [263], also called continual learning, is a machine learning paradigm that regularly processes periodically collected data and continuously integrates newly learned information to models in order to keep models up to date to the evolving data representation or task. Contrary to conventional offline learning, where all training data are available at the beginning of the learning process, and models are firstly built by learning all data batches or samples through epochs for prediction, incremental learning is suitable for learning problems where data are collected over time. In this case, the data distribution, the feature space, or even the task evolve over time. Thus, the trained model is expected to be periodically updated in order to capture and adapt to the new evaluations. Incremental learning takes advantage of a higher quality of data, closeness to the testing environment, and continuously personalizes the pre-trained model with new classes. This learning paradigm can maintain and improve task accuracy when an original pre-trained model cannot generalize well. Moreover, the incremental learning updates model locally and thus preserves the privacy in the case of local deployment.

With respect to the incremental learning setup, online learning, as an instantiation of incremental learning in an online scenario [264], continuously learns from data provided in sequence from a data stream and produces a series of versions of the same model for prediction. This is performed without having the complete training dataset available at the beginning. The model is deployed online to continuously realize intervened updates and predictions. In particular, as new data are usually generated very fast from the data stream such as in the case of Twitter data [265], online learning typically uses data samples for only one epoch training and then switches for newer samples. Furthermore, lifelong learning [266] is another incremental learning branch that is characterized by the time span of the learning process and refers to the incremental learning in an infinite time span, to accumulate the learned knowledge for future learning and problem solving.

One major challenge of incremental learning is the continuous model adaptation and efficient paradigm design of learning from new data. One typical cause is the concept drift [267] which occurs over time, leading to a change in the functional relationship between the model inputs and outputs. Furthermore, learning data of new classes, the model can forget previously learned knowledge. This refers to another cause: the catastrophic forgetting [268]. An early work [269] incorporates incremental learning with the partial instance memory of data samples from the boundaries of the induced concepts. The model updates are based on both previous and new samples. Online learning [264] employs a cross-distillation loss together with a two-step learning technique respectively for the new class data learning and the exemplar data learning to tackle catastrophic forgetting. Furthermore, it counts on the feature-based exemplary set update to mitigate the concept drift. This method outperforms the results of current state-of-the-art offline incremental learning methods on the CIFAR-100 and ImageNet-1000 datasets in online scenarios. To perform lifelong learning on edge devices with limited computation resources, a dynamically growing neural network architecture is introduced in [270] based on self-organization neural network (SONN) [271]. In the architecture, a CNN backbone is used as the encoder and the SONN is applied after as the classifier with the capability to grow the network when required to performance lifelong object detection on FPGA.

Incremental learning is excellent at autonomously adapting models to continuously changing environments of data, features, and task spaces, and thus improves the reliability. By learning from data closer to the prediction environment, the model performance on real environments is improved as well. In particular, the incremental learning fits well to the edge environment with limited computing resources, as data can be fetched for learning in a piecemeal manner and then discarded right after the training, which improves the availability. In an online setting, incremental learning consumes more network bandwidth and computation resources in exchange for a higher model performance and adaptation capability. The cost and energy consumption are increased.

6. Technique Review Summary

In this section, we summarize in Tables 7 and 8 all reviewed techniques with regard to the Edge ML requirements. The three left columns illustrate the individual techniques, or technique groups, while the top two rows list the Edge ML requirements. The following notations are used to facilitate the relationship descriptions between techniques and requirements.

- "+": The reviewed technique improves the corresponding Edge ML solution regarding the specific Edge ML requirement. For instance, quantization techniques reduce the inference task latency by simplifying the computation complexity.
- "−": The reviewed technique negatively impacts the corresponding Edge ML solution regarding the specific Edge ML requirement. For instance, quantization techniques lead to accuracy loss during inference due to the low precision representation of data.

- "*": The impact of the reviewed technique on the corresponding Edge ML solution varies according to the specific configurations or setup. For instance, transfer learning techniques improve the target model performance under the conditions that the source task and the target task are correlated, and the data quantity and quality on the target domain are sufficient. The weakness in data quantity or quality on the target domain can result in unsatisfactory model performance.
- "/": The reviewed technique does not directly impact the corresponding Edge ML solution regarding the specific Edge ML requirement. For instance, federated learning techniques do not directly improve or worsen the labelled data independence for a supervised learning process.

Moreover, the two following assumptions have been made to assure an objective evaluation of each Edge ML technique regarding the requirements:

- Appropriate modelling and learning: all models for ML tasks are designed and trained following the state-of-the-art solution. No serious over-fitting or under-fitting has occurred, so that the models' performance can be compared before and after applying the Edge ML techniques.
- Statistic scenario: When performing a task, statistic scenarios instead of the best or the worst scenario are considered for techniques evaluation, as certain techniques, e.g., Early Exit of Inference, can produce worse results compared to the corresponding conventional technique in extreme cases where all the side branch classifiers in a model do not produce high enough confidence and thus it fails to stop the inference earlier. However, statistically, the EEoI technique is able to improve energy efficiency and optimize cost when performing a number of running tasks.

From Tables 7 and 8, one can see that most of edge inference techniques focus on reducing inference workload to improve computational efficiency, task latency, and availability. Concerning the reliability, the compressed models put less stress on the hardware, potentially reducing the risk of hardware faults or failures, while in the meantime they are less resilient to hardware faults or slight changes in the input themselves. Distributed inference makes the inference execution of large models possible on the edge side by introducing a greater computational and communication workload for coordination and synchronization among edge clients. Regarding the distributed learning, split learning is able to offer a more competitive performance and privacy compared to federated learning, when the cloud server is available to cooperate on the training process. Distributed learning can potentially increase reliability by introducing additional nodes; however, on the other hand, the additional complexity to manage and synchronize different resources leads to more points of failure. The overall impact on reliability will depend on the detailed configurations of the distributed learning technique implementation. Transfer learning mainly focuses on accelerating the training task latency by facilitating knowledge sharing across domains, whilst meta-learning and self-supervised learning respectively provide an efficient and a consolidated way to learn the data representation instead of specific tasks from labeled and unlabeled data to facilitate the learning of new tasks. Moreover, other learning paradigms, i.e., instance-based learning and weakly supervised learning, provide alternative solutions to directly learn from instances or partially labelled data. Multi-task learning is efficient to reduce model size and discover task correlations for better performance when multiple correlated tasks need to be realized simultaneously. At last, incremental learning improves the model performance by continuously adapting models to the real environment by learning from ever-evolving data. The overall requirements of energy efficiency and cost optimization are met by most Edge ML techniques from different aspects of ML and EC.

Table 7. Edge ML: Techniques meet Requirements—Part I.

Edge ML Techniques			Edge ML Requirements							
			Machine Learning					Edge Computing		
			Low Task Latency	**High Performance**	**Generalization and Adaptation**	**Enhanced Privacy and Security**	**Labelled Data Independency**	**Computational Efficiency**	**Optimized Bandwidth**	**Offline Capability**
Edge Inference	Model Compression and Approximation	Quantization	+	*	/	/	/	+	/	/
		Weight Reduction	+	*	/	/	/	+	/	/
		Knowledge Distillation	+	–	/	/	/	+	/	/
		Activation Function Approximation	+	–	/	/	/	+	/	/
	Distributed Inference		+	/	/	/	/	–	–	–
	Other Inference Acceleration	Early Exit	+	-	/	/	/	+	/	/
		Inference Cache	+	–	/	/	/	+	/	/
		Model-Specific Inference Acceleration	+	–	/	/	/	+	/	/
Edge Learning	Distributed Learning	Federated Learning	+	–	/	+	/	–	+	–
		Split Learning	+	/	/	+	/	–	+	–
	Transfer Learning		+	*	/	/	+	+	/	/
	Meta-Learning		+	/	+	/	+	*	/	/
	Self-Supervised Learning		+	+	+	/	+	+	/	/
	Other Learning Paradigms	Multi-Task Learning	+	+	/	/	/	+	/	/
		Instance-based Learning	*	*	/	/	+	*	/	/
		Weakly Supervised Learning	+	/	/	/	+	/	/	/
		Incremental Learning	–	+	+	/	/	–	–	/

Table 8. Edge ML: Techniques meet Requirements—Part II.

Edge ML Techniques			**Edge ML Requirements**			
			Overall			
			Availability	**Reliability**	**Energy Efficiency**	**Cost Optimization**
Edge Inference	Model Compression and Approximation	Quantization	+	*	+	+
		Weight Reduction	+	*	+	+
		Knowledge Distillation	+	*	+	+
		Activation Function Approximation	+	*	+	+
	Distributed Inference		+	+	−	−
	Other Inference Acceleration	Early Exit	+	−	+	+
		Inference Cache	+	+	+	+
		Model-Specific Inference Acceleration	+	*	+	+
Edge Learning	Distributed Learning	Federated Learning	+	*	+	+
		Split Learning	+	*	+	+
	Transfer Learning		+	*	+	+
	Meta-Learning		*	+	*	*
	Self-Supervised Learning		*	+	+	+
	Other Learning Paradigms	Multi-Task Learning	+	+	+	+
		Instance-based Learning	*	*	*	*
		Weakly Supervised Learning	+	−	+	+
		Incremental Learning	+	+	+	+

7. Edge ML Frameworks

To facilitate the implementation of ML solutions on the Edge, specific frameworks have been developed targeting various devices and platforms. In this section, we briefly review the representative development frameworks supporting the Edge ML implementation. The future direction related to Edge ML frameworks is summarized together with other open issues in the next section.

For edge inference frameworks, the general process to deploy a model on the edge starts by choosing an existing trained model, and converting the model to a specific format supported by the framework. In parallel with the model conversion, model compression methods, such as quantization, are generally offered by frameworks and can be applied to reduce the model size. Finally, the model is deployed to the edge devices for inference tasks. Frameworks for edge inference can be grouped into two categories: (i) cross-platform frameworks and (ii) platform-specific frameworks.

Cross-Platform Inference Framework. Several popular native Edge ML frameworks exist to provide a general development solution across different devices and Operating Systems (OS). TensorFlow Lite [272] is a light version of TensorFlow [273] to deploy models on mobile and embedded devices, enabling quantization and on-device inference. The framework supports devices running with Android, iOS, or embedded Linux OS, as well as devices based on Micro-controllers (MCUs) and Digital Signal Processors (DSPs) without any OS. The main advantage of TensorFlow Lite is that it does not require OS support, any standard C or C++ libraries, or dynamic memory allocation. Alternatively, PyTorch Mobile [274], as the edge version of PyTorch [275], supports model inference on devices with an iOS, Android, or Linux system. It also integrates 8-bit kernels for quantization. The Embedded Learning Library (ELL) from Microsoft is a similar framework allowing the deployment of ML models onto resource-constrained platforms and small single-board computers such as Raspberry Pi, Arduino, and micro:bit running Windows, Linux, or macOS. To support both edge inference and edge training, CoreML [276] is a machine learning framework across all of Apple's OSs including macOS, iOS, tvOS, and watchOS.

Both training and inference are enabled, with pre-built features such as memory footprint reduction for performance optimization. However, since models are encapsulated inside an application in CoreML, additional computational complexity is added.

Platform-Specific Inference Framework. Besides the cross-platform frameworks, numerous platform-specific frameworks have been developed to support the inference task on specific processors including (i) General Purpose Processors (GPPs) such as CPUs and GPUs; (ii) Application-Specific Integrated Circuits (ASICs) such as Google TPU [277] and Intel Movidius VPU [278]; and (iii) MCUs such as STM32 [279]. Most of the existing mobile processors are based on Arm architecture. ARM Compute Library [280] offers a collection of low-level machine learning functions optimized for Cortex-A CPU and Mali GPUs architectures. Specifically, it provides the basic CNN building blocks for model inference in CV domains. Nvidia GPUs are widely used in PCs, and TensorRT [281], as the corresponding ML framework for edge inference across all Nvidia GPUs, supports distributed inference and numerous model structures extensively optimized for performance including optimizer and run-time that delivers low latency and high throughput for edge inference.

Besides GPPs, a large number of ASICs specific to Edge ML applications are manufactured [25]. To facilitate the direct use, ASICs are commonly integrated into System-on-a-Chip (SoC), System-on-Module (SoM) or Single-Board Computer (SBC), and the corresponding frameworks specific to ASIC vendors are developed to offer a complete ML computing solution. Aiming at the NVIDIA Jetson family of hardware, NVIDIA EGX Platform [282], a combination of Nvidia GPUs and VMware vSphere with NVIDIA Certified Systems, provides a full stack software infrastructure to deploy Edge ML applications for inference.

The Qualcomm Neural Processing SDK for ML [283] is an Edge ML framework to optimize and deploy trained neural networks on devices with Snapdragon SoC family from Qualcomm. The Qualcomm Neural Processing SDK supports convolutional neural networks and custom layers. Concerning MCUs, Neural Network on Micro-controller (NNoM) [284] is a high-level inference neural network library specifically for microcontrollers, which helps to convert a model trained with Keras a to native NNoM model for deployment. NNoM supports several convolutional structures including Inception [112], ResNet, and DenseNet [285] and recurrent layers such as simple RNN, GRU, and LSTM. X-CUBE-ML [286] is an ML expansion framework for STM32 MCU with the capability to convert and deploy pre-trained neural networks and classical machine learning models, and performance measurement functions for STM32 are directly integrated as well.

Again, in order to match the nature of the target ASICs and the embedding platforms, most of the frameworks must go through a process of conversion, modification, and, in some cases, complete retraining. Since most of native edge frameworks only support edge inference, we hereafter illustrate general distributed learning frameworks for learning purposes, which can be used in edge devices with relatively high computation capability such as PC and mobile phones. The frameworks are mainly developed to facilitate the parallel training (i.e., data-parallelism, model-parallelism, and 3D parallelism, as summarized in [287]) and federated learning.

Distributed Learning Framework. PyTorch [275] is one of the most popular deep learning frameworks, offering both data-parallelism and model-parallelism functions for model training. Built upon PyTorch, DeepSpeed [288] extends the PyTorch with additional 3D parallelism support as well as memory optimization for exascale model training. As an alternative ecosystem of PyTorch, Distributed API in Tensorflow provides data-parallel techniques for model training, while Mesh TensorFlow [289] extends the Distributed TensorFlow with model-parallelism and enables large model training on Google TPUs. Mindspore [290] is another general ML computing framework from Huawei with end-to-end ML capabilities for model development, execution, and deployment in various scenarios including distributed training and cloud-edge deployment. Regarding the federated learning frameworks, TensorFlow Federated [291] is the federated learning framework from the TensorFlow ecosystem on decentralized data. The framework enables developers to simulate the included federated learning algorithms on their models and data, as well as to experiment with novel algorithms. For deployment in real environments, Open Federated Learning [292] from Intel provides deployment scripts in bash and leverages certificates for securing communication. Finally, NVIDIA CLARA [293] includes full-stack GPU-accelerated libraries covering training schemes, communications, and hardware configurations for server-client federated learning with a central aggregation server.

8. Open Issues and Future Directions

Despite the diverse methods and paradigms of Edge ML and the initial success of their powered edge solutions, challenges and open issues are not rare in the Edge ML field, slowing down the technological progress. In this section, we summarize some open issues of Edge ML to shed light on its future directions.

Learning Generalization and Adaptation. Currently ML techniques are going through a transition from the learning of specific labels to the learning of data representations. Meta-learning and self-supervised learning provide intuitive manners to progress in this direction. Nevertheless, meta-learning usually relies on a support dataset to perform any task-specific adaptation, and self-supervised learning requires tuning as well for specific tasks. The generalization from representation learning brings general cognitive abilities to models, while automatic adaptation techniques to specific tasks such as zero-shot learning in NLP need to be further studied and explored so that specific tasks can be solved directly without performing any adaptation process. In the future, machine learning techniques will continue to advance from merely learning task-specific labels to more generalized data representations. This progression is poised to enhance machine learning models' cognitive abilities, pushing the achievement boundaries. This is particularly important to Edge ML as human intervention or guidance are not guaranteed compared to the cloud-based solutions.

Theoretical Foundation. With the rapid emergence of Edge ML techniques, the theoretical foundation related to the emerging techniques for optimal design and empirical validation are not up to date. For example, most model compression and approximation methods do not have mathematical proofs for the optimal compression ratio. Federated learning also may not converge in the training process, if the data distribution varies largely from clients. Finally, self-supervised learning continuously seeks optimal contrastive objective functions to optimize learning efficiency. Theoretical foundations are crucial to validate empirical conclusions from emerging fields and provide guidelines for the optimal design of Edge ML solutions. In the foreseeable future, significant advancements in the theoretical foundation of Edge ML techniques are expected. These advancements will primarily focus on the development of mathematical proofs and models for such Edge ML methods.

Architectures for Heterogeneity and Scalability. An Edge ML environment is known to be heterogeneous in distribution of entities such as data, device resources, network infrastructures, and even ML tasks and models. And with a large number of participant edge devices, bottlenecks have been identified affecting Edge ML performance. Such bottlenecks include the communication bottleneck in federated learning for gradient communications and the computational bottleneck in meta-learning when the support set is large. Furthermore, all edge devices are not often activated at the same time, and the temporal disparity feature makes it more challenging for the organizational architecture to manage the Edge ML solution. Adding local edge servers can alleviate the problem of the local perimeter, and to reach the global heterogeneity management with a large number of edge devices. Advanced distributed architectures for ML tasks are expected to synchronize and coordinate entities among all heterogeneity levels and deliver robust and scalable solutions for dynamic and adaptive aggregation in distributed setup. Given the inherent diversity in Edge ML environments, including variations in data, device resources, network infrastructures, and ML tasks and models, there is a pressing need for more flexible and adaptive architectures. As edge devices continue to proliferate, future work will need to address performance bottlenecks, such as communication constraints in federated learning and computational limitations in meta-learning.

Fortified Privacy. Privacy preservation is the primary objective in distributed learning and inference paradigms, as no data are shared outside of the local client. However, sensitive information can still be leaked via methods such as the reverse deduction of models. Although security- and privacy-oriented methods can improve the situation, a significant computation complexity is introduced in the edge devices in the meantime, increasing task latency and energy consumption. Novel and lightweight computing paradigms are expected to protect data and model leakage during information exchange and go from enhanced privacy to fortified privacy. The future direction is expected to focus on developing novel, lightweight computing paradigms that not only protect data from breaches but also prevent model leakage during information exchange. This future trend towards fortified privacy should bring forward the development of new methods and architectures that are efficient in terms of computational resources and energy usage. These advancements are expected to increase privacy without sacrificing performance, leading to a more secure and trustable Edge ML environment.

Hybrid Approach. With the reviewed techniques tackling different aspects of Edge ML requirements, hybrid strategies with more than one technique is now commonly adopted when designing Edge ML solution. Hybrid ML benefits from several techniques and can achieve better performance than the use of any single method. The integration of two or three techniques are popular in the reviewed literature, while with a given set of design requirements, complete hybrid approaches covering all Edge ML phases, including data preprocessing, learning, and inference, are missing. The hybrid approach with a thorough technical design for each phase can best contribute to the improvement of model capability, and thus is a direction worth exploring. We envision the future of Edge ML will entail designing holistic hybrid solutions that address every phase with careful technical consideration. This approach not only enhances model capability but also ensures robust performance in varying edge scenarios.

Data Quality Assurance. Nowadays, a huge amount of data is created on the edge devices at every second, but most of it cannot be directly used by ML without the labeling and preprocessing processes. As a step forward, self-supervised learning proves to be good at learning structured and unlabeled data. However, the data quality such as noisy data, non-IID data, imbalanced distribution, or data corruptions and errors, still impacts the learning results and tends to alter the model performance. Although a number of methods are introduced, the selection of suitable methods is determinant to the results and highly relies on expertise. Regular interaction with humans for labelling and selection of quality data are not realistic, especially for edge users, and thus embedded learning paradigms integrating native data selection for quality control and preprocessing of different input qualities is the future of Edge ML. In the future direction, assuring data quality will become increasingly important within the context of Edge ML. We envision the future of Edge ML to be geared towards the development of embedded learning paradigms, which integrate native data selection mechanisms for quality control and preprocessing. These embedded systems would automatically handle different input qualities, reducing the need for human intervention, and leading to more reliable, autonomous Edge ML systems.

Framework Extension. The number of frameworks keeps increasing for Edge ML. However, due to the resource-constrained nature of the edge environment, existing frameworks tend to be lightweight for resource efficiency and thus limited in their support of ML features and functions: most of the native Edge ML frameworks are only designed for edge inference, and involve additional steps and computation for model conversion. Device-specific frameworks often support a subset of neural network layers and activation functions, which requires model re-design and re-training before deployment as well. With the rapid development of computing capability on edge devices, the trade-off between resource efficiency and functionality can be further studied to extend the supporting edge features and functions. Looking towards the future, there is a pressing need for the extension and expansion of the available frameworks for Edge ML. With the rapid advancements in the computing capabilities of edge devices, there is an opportunity to reconsider the trade-off between resource efficiency and functionality. In the future, we anticipate a shift towards frameworks that strike a better balance between these two aspects, providing more comprehensive support for Edge ML features and functions without drastically increasing resource requirements. This progression towards more functionally rich and resource-efficient frameworks will significantly impact the design and implementation of Edge ML solutions and remains a promising avenue for exploration and development in the field.

Standardization. There are widespread standardization organizations (SDOs) on ML (e.g., ISO/IEC JTC 1/SC 42 Artificial Intelligence [294], ITU-T Focus Groups [295,296], IEEE SA Artificial Intelligence Systems, only to name a few) contributing to the community development and reference solutions. However, there is clearly very few ongoing activities within initiatives and SDOs (e.g., ETSI ISG EMC [297]) focused on defining native specifications for Edge ML solutions. Along with the uprising development of Edge ML technologies, Edge ML standards and specifications covering MLOps life cycle in the edge environment are expected to fill the gap in the Edge ML ecosystem and optimize ML at the edge for reference and guidance. Moving forward, it is expected that the standardization ecosystem will change, with a more dedicated focus on creating standards and specifications that cover the MLOps lifecycle in the edge environment. The development of such standards will likely address the current gaps in the Edge ML ecosystem, providing both guidance and reference solutions for implementing ML at the edge. Such standardization will not only streamline the development and deployment processes but also enhance system interoperability and reliability, making Edge ML more accessible and effective.

9. Conclusions

Due to the specific features of privacy preservation, low-latency experiences, and low energy consumption, edge-powered machine learning solutions have been rapidly emerging in end-user devices for services and applications in the domains of CV, NLP, healthcare, UAVs, etc. In this paper, we provide a comprehensive review of Edge ML techniques focusing on the two parts of ML solutions: (i) edge inference and (ii) edge learning. The review offers a panoramic view of the technique's perimeter through a thorough taxonomy. Recent and representative works are presented for each technique with its targeting Edge ML requirements. Edge ML frameworks are introduced, while open issues are identified for future research directions. To the best of our knowledge, this is the first review covering the entire and detailed technique perimeter of Edge ML learning and inference.

This paper can serve as a reference to select adaptive ML paradigms and build corresponding solutions in edge environments. Due to the large perimeter to cover, we adapt the review strategy to prioritize the technique width rather than the technique depth, and thus further work will focus on surveying more detailed research challenges and methods for targets and specific techniques' branches. In the meantime, we are also investigating scalable architectures for Edge ML solutions over heterogeneity infrastructural resources, data and tasks.

Author Contributions: Conceptualization, methodology, investigation, writing—original draft preparation and revision, W.L.; writing—review and editing, H.H.; supervision, funding acquisition, E.A.; supervision, funding acquisition, M.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, D.; Maslej, N.; Brynjolfsson, E.; Etchemendy, J.; Lyons, T.; Manyika, J.; Ngo, H.; Niebles, J.C.; Michael, J.; Sellitto, M.; et al. *The AI Index Report 2022*; AI Index Steering Committee, Stanford Institute for Human-Centered AI, Stanford University: Stanford, CA, USA, 2022; pp. 1–230. [CrossRef]
2. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
3. OpenAI. GPT-4 Technical Report. *arXiv* **2023**. [CrossRef]
4. Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C.; Gustafson, L.; Xiao, T.; Whitehead, S.; Berg, A.C.; Lo, W.Y.; et al. Segment Anything. *arXiv* **2023**. [CrossRef]
5. Liang, F.; Wu, B.; Dai, X.; Li, K.; Zhao, Y.; Zhang, H.; Zhang, P.; Vajda, P.; Marculescu, D. Open-Vocabulary Semantic Segmentation with Mask-adapted CLIP. *arXiv* **2022**. [CrossRef]
6. Singer, U.; Polyak, A.; Hayes, T.; Yin, X.; An, J.; Zhang, S.; Hu, Q.; Yang, H.; Ashual, O.; Gafni, O.; et al. Make-A-Video: Text-to-Video Generation without Text-Video Data. *arXiv* **2022**. [CrossRef]
7. GitHub-Stability-AI/Stablediffusion: High-Resolution Image Synthesis with Latent Diffusion Models, 2023. Available online: https://github.com/CompVis/latent-diffusion (accessed on 28 July 2023).
8. Romero, A. Wu Dao 2.0: A Monster of 1.75 Trillion Parameters | by Alberto Romero | Medium | Towards Data Science, 2021. Available online: https://towardsdatascience.com/gpt-3-scared-you-meet-wu-dao-2-0-a-monster-of-1-75-trillion-parameters-832cd83db484 (accessed on 28 July 2023).
9. Dilley, J.; Maggs, B.; Parikh, J.; Prokop, H.; Sitaraman, R.; Weihl, B. Globally distributed content delivery. *IEEE Internet Comput.* **2002**, *6*, 50–58. [CrossRef]
10. Davis, A.; Parikh, J.; Weihl, W.E. EdgeComputing: Extending enterprise applications to the edge of the internet. In Proceedings of the 13th International World Wide Web Conference on Alternate Track, Papers and Posters, WWW Alt 2004, New York, NY, USA, 19–21 May 2004; pp. 180–187. [CrossRef]
11. Khan, W.Z.; Ahmed, E.; Hakak, S.; Yaqoob, I.; Ahmed, A. Edge computing: A survey. *Future Gener. Comput. Syst.* **2019**, *97*, 219–235. [CrossRef]
12. Lee, Y.L.; Tsung, P.K.; Wu, M. Techology trend of edge AI. In Proceedings of the 2018 International Symposium on VLSI Design, Automation and Test, VLSI-DAT 2018, Hsinchu, Taiwan, 16–19 April 2018; pp. 1–2. [CrossRef]
13. Amin, S.U.; Hossain, M.S. Edge Intelligence and Internet of Things in Healthcare: A Survey. *IEEE Access* **2021**, *9*, 45–59. [CrossRef]
14. Yang, B.; Cao, X.; Xiong, K.; Yuen, C.; Guan, Y.L.; Leng, S.; Qian, L.; Han, Z. Edge Intelligence for Autonomous Driving in 6G Wireless System: Design Challenges and Solutions. *IEEE Wirel. Commun.* **2021**, *28*, 40–47. [CrossRef]
15. Lv, Z.; Chen, D.; Lou, R.; Wang, Q. Intelligent edge computing based on machine learning for smart city. *Future Gener. Comput. Syst.* **2021**, *115*, 90–99. [CrossRef]
16. Tang, S.; Chen, L.; He, K.; Xia, J.; Fan, L.; Nallanathan, A. Computational Intelligence and Deep Learning for Next-Generation Edge-Enabled Industrial IoT. *IEEE Trans. Netw. Sci. Eng.* 2022, *early access*. [CrossRef]
17. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A Survey of Model Compression and Acceleration for Deep Neural Networks. *arXiv* **2017**. [CrossRef]
18. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A Comprehensive Survey on Transfer Learning. *Proc. IEEE* **2021**, *109*, 43–76. [CrossRef]
19. Abreha, H.G.; Hayajneh, M.; Serhani, M.A. Federated Learning in Edge Computing: A Systematic Survey. *Sensors* **2022**, *22*, 450. [CrossRef] [PubMed]
20. Wang, X.; Han, Y.; Leung, V.C.; Niyato, D.; Yan, X.; Chen, X. Convergence of Edge Computing and Deep Learning: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* **2020**, *22*, 869–904. [CrossRef]
21. Wang, X.; Han, Y.; Leung, V.C.M.; Niyato, D.; Yan, X.; Chen, X. *Edge AI*; Springer: Singapore, 2020. [CrossRef]
22. Abbas, G.; Mehmood, A.; Carsten, M.; Epiphaniou, G.; Lloret, J. Safety, Security and Privacy in Machine Learning Based Internet of Things. *J. Sens. Actuator Netw.* **2022**, *11*, 38. [CrossRef]
23. Mustafa, E.; Shuja, J.; uz Zaman, S.K.; Jehangiri, A.I.; Din, S.; Rehman, F.; Mustafa, S.; Maqsood, T.; Khan, A.N. Joint wireless power transfer and task offloading in mobile edge computing: A survey. *Clust. Comput.* **2022**, *25*, 2429–2448. [CrossRef]
24. Sarwar Murshed, M.G.; Murphy, C.; Hou, D.; Khan, N.; Ananthanarayanan, G.; Hussain, F. Machine Learning at the Network Edge: A Survey. *ACM Comput. Surv.* **2021**, *54*, 1–37. [CrossRef]
25. Li, W.; Liewig, M. A survey of AI accelerators for edge environment. In *Advances in Intelligent Systems and Computing*; Rocha, A., Adeli, H., Reis, L.P., Costanzo, S., Orovic, I., Moreira, F., Eds.; Springer International Publishing: Cham, Switzerland, 2020; Volume 1160, pp. 35–44. [CrossRef]
26. Wang, S.; Zhang, X.; Zhang, Y.; Wang, L.; Yang, J.; Wang, W. A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications. *IEEE Access* **2017**, *5*, 6757–6779. [CrossRef]
27. Wang, A.; Singh, A.; Michael, J.; Hill, F.; Levy, O.; Bowman, S.R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *EMNLP 2018—2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP, Proceedings of the 1st Workshop, Brussels, Belgium, 1 November 2018*; Association for Computational Linguistics: Toronto, ON, Canada, 2018; pp. 353–355. [CrossRef]

28. Osband, I.; Doron, Y.; Hessel, M.; Aslanides, J.; Sezener, E.; Saraiva, A.; McKinney, K.; Lattimore, T.; Szepesvari, C.; Singh, S.; et al. Behaviour Suite for Reinforcement Learning. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. [CrossRef]
29. Huisman, M.; van Rijn, J.N.; Plaat, A. A survey of deep meta-learning. *Artif. Intell. Rev.* **2021**, *54*, 4483–4541. [CrossRef]
30. Xu, Y.; Goodacre, R. On Splitting Training and Validation Set: A Comparative Study of Cross-Validation, Bootstrap and Systematic Sampling for Estimating the Generalization Performance of Supervised Learning. *J. Anal. Test.* **2018**, *2*, 249–262. [CrossRef]
31. Golalipour, K.; Akbari, E.; Hamidi, S.S.; Lee, M.; Enayatifar, R. From clustering to clustering ensemble selection: A review. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104388. [CrossRef]
32. Denil, M.; Shakibi, B.; Dinh, L.; Ranzato, M.; De Freitas, N. Predicting parameters in deep learning. *Adv. Neural Inf. Process. Syst.* **2013**, *2*, 2148–2156. [CrossRef]
33. Wang, E.; Davis, J.J.; Zhao, R.; Ng, H.C.; Niu, X.; Luk, W.; Cheung, P.Y.; Constantinides, G.A. Deep neural network approximation for custom hardware: Where We've Been, Where We're going. *ACM Comput. Surv.* **2019**, *52*, 1–39. [CrossRef]
34. Wang, S.; Kanwar, P. BFloat16: The Secret to High Performance on Cloud TPUs—Google Cloud Blog. 2021. Available online: https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus (accessed on 28 July 2023).
35. Goyal, R.; Vanschoren, J.; van Acht, V.; Nijssen, S. Fixed-point Quantization of Convolutional Neural Networks for Quantized Inference on Embedded Platforms. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2021. [CrossRef]
36. Yuan, C.; Agaian, S.S. A comprehensive review of Binary Neural Network. *Artif. Intell. Rev.* **2023**. [CrossRef]
37. Liu, B.; Li, F.; Wang, X.; Zhang, B.; Yan, J. Ternary Weight Networks. In Proceedings of the ICASSP 2023—2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5. [CrossRef]
38. Lee, E.H.; Miyashita, D.; Chai, E.; Murmann, B.; Wong, S.S. LogNet: Energy-efficient neural networks using logarithmic computation. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, New Orleans, LA, USA, 5–9 March 2017; pp. 5900–5904. [CrossRef]
39. Lai, L.; Suda, N.; Chandra, V. Deep Convolutional Neural Network Inference with Floating-point Weights and Fixed-point Activations. *arXiv* **2017**. [CrossRef]
40. Gustafson, J.L.; Yonemoto, I.T. Beating Floating Point at its Own Game: Posit Arithmetic. *Supercomput. Front. Innov.* **2017**, *4*, 71–86. [CrossRef]
41. *IEEE Std 754-2008*; IEEE Standard for Floating-Point Arithmetic. IEEE: Piscataway, NJ, USA, 2008; pp. 1–70. [CrossRef]
42. Gohil, V.; Walia, S.; Mekie, J.; Awasthi, M. Fixed-Posit: A Floating-Point Representation for Error-Resilient Applications. *IEEE Trans. Circuits Syst. II Express Briefs* **2021**, *68*, 3341–3345. [CrossRef]
43. NVIDIA Corporation. Tensor Cores: Versatility for HPC & AI | NVIDIA. 2022. Available online: https://www.nvidia.com/en-us/data-center/tensor-cores/ (accessed on 28 July 2023).
44. What Is the TensorFloat-32 Precision Format? | NVIDIA Blog. Available online: https://blogs.nvidia.com/blog/2020/05/14/tensorfloat-32-precision-format/ (accessed on 28 July 2023).
45. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713. [CrossRef]
46. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**. [CrossRef]
47. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common objects in context. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2014; Volume 8693, pp. 740–755. [CrossRef]
48. Lee, S.; Sim, H.; Choi, J.; Lee, J. Successive log quantization for cost-efficient neural networks using stochastic computing. In Proceedings of the Design Automation Conference, Las Vegas, NV, USA, 2–6 June 2019; pp. 2–7. [CrossRef]
49. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
50. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. *arXiv* **2016**. [CrossRef]
51. Jin, X.; Du, X.; Sun, H. VGG-S: Improved Small Sample Image Recognition Model Based on VGG16. In Proceedings of the 2021 3rd International Conference on Artificial Intelligence and Advanced Manufacture, AIAM 2021, Manchester, UK, 23–25 October 2021; pp. 229–232. [CrossRef]
52. Oh, S.; Sim, H.; Lee, S.; Lee, J. Automated Log-Scale Quantization for Low-Cost Deep Neural Networks. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 742–751. [CrossRef]
53. Qin, H.; Ma, X.; Ding, Y.; Li, X.; Zhang, Y.; Tian, Y.; Ma, Z.; Luo, J.; Liu, X. BiFSMN: Binary Neural Network for Keyword Spotting. In Proceedings of the IJCAI International Joint Conference on Artificial Intelligence, Vienna, Austria, 23–29 July 2022; pp. 4346–4352. [CrossRef]
54. Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition. *arXiv* **2018**. [CrossRef]

55. Liu, Z.; Oguz, B.; Pappu, A.; Xiao, L.; Yih, S.; Li, M.; Krishnamoorthi, R.; Mehdad, Y. BiT: Robustly Binarized Multi-distilled Transformer. *arXiv* **2022**, arXiv:2205.13016.
56. Osorio, J.; Armejach, A.; Petit, E.; Henry, G.; Casas, M. A BF16 FMA is All You Need for DNN Training. *IEEE Trans. Emerg. Top. Comput.* **2022**, *10*, 1302–1314. [CrossRef]
57. Zhang, J.; Zhou, Y.; Saab, R. Post-training Quantization for Neural Networks with Provable Guarantees. *SIAM J. Math. Data Sci.* **2023**, *5*, 373–399. [CrossRef]
58. De Putter, F.; Corporaal, H. Quantization: How far should we go? In Proceedings of the 2022 25th Euromicro Conference on Digital System Design (DSD), Gran Canaria, Spain, 31 August–2 September 2022; pp. 373–382. [CrossRef]
59. Liu, Z.; Shen, Z.; Savvides, M.; Cheng, K.T. ReActNet: Towards Precise Binary Neural Network with Generalized Activation Functions. In Proceedings of the Computer Vision—ECCV 2020, Glasgow, UK, 23–28 August 2020; Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 143–159.
60. Zhou, S.; Wu, Y.; Ni, Z.; Zhou, X.; Wen, H.; Zou, Y. DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients. *arXiv* **2016**. [CrossRef]
61. Ruospo, A.; Sanchez, E.; Traiola, M.; O'Connor, I.; Bosio, A. Investigating data representation for efficient and reliable Convolutional Neural Networks. *Microprocess. Microsyst.* **2021**, *86*, 104318. [CrossRef]
62. Chu, X.; Zhang, B.; Xu, R. FairNAS: Rethinking Evaluation Fairness of Weight Sharing Neural Architecture Search. In Proceedings of the IEEE International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 12219–12228. [CrossRef]
63. Jaderberg, M.; Vedaldi, A.; Zisserman, A. Speeding up convolutional neural networks with low rank expansions. In Proceedings of the BMVC 2014—British Machine Vision Conference 2014, Nottingham, UK, 1–5 September 2014. [CrossRef]
64. Srinivas, S.; Babu, R.V. Data-free Parameter Pruning for Deep Neural Networks. *arXiv* **2015**. [CrossRef]
65. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* **2015**, *115*, 211–252. [CrossRef]
66. Dai, X.; Yin, H.; Jha, N.K. NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm. *IEEE Trans. Comput.* **2019**, *68*, 1487–1497. [CrossRef]
67. Yu, J.; Lukefahr, A.; Palframan, D.; Dasika, G.; Das, R.; Mahlke, S. Scalpel: Customizing DNN pruning to the underlying hardware parallelism. In Proceedings of the International Symposium on Computer Architecture, Toronto, ON, Canada, 24–28 June 2017; Volume F1286, pp. 548–560. [CrossRef]
68. Han, S.; Mao, H.; Gong, E.; Tang, S.; Dally, W.J.; Pool, J.; Tran, J.; Catanzaro, B.; Narang, S.; Elsen, E.; et al. DSD: Dense-sparse-dense training for deep neural networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017. [CrossRef]
69. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
70. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015. [CrossRef]
71. Frantar, E.; Alistarh, D. SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot. *arXiv* **2023**, arXiv:2301.00774.
72. Wu, T.; Li, X.; Zhou, D.; Li, N.; Shi, J. Differential Evolution Based Layer-Wise Weight Pruning for Compressing Deep Neural Networks. *Sensors* **2021**, *21*, 880. [CrossRef]
73. Fang, G.; Ma, X.; Song, M.; Mi, M.B.; Wang, X. DepGraph: Towards Any Structural Pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 16091–16101.
74. Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing* **2021**, *461*, 370–403. [CrossRef]
75. Gao, X.; Zhao, Y.; Dudziak, L.; Mullins, R.; Xu, C. Dynamic channel pruning: Feature boosting and suppression. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. [CrossRef]
76. Aich, S.; Yamazaki, M.; Taniguchi, Y.; Stavness, I. Multi-Scale Weight Sharing Network for Image Recognition. *Pattern Recognit. Lett.* **2020**, *131*, 348–354. [CrossRef]
77. Chen, W.; Wilson, J.T.; Tyree, S.; Weinberger, K.Q.; Chen, Y. Compressing neural networks with the hashing trick. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; Volume 3, pp. 2275–2284.
78. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2323. [CrossRef]
79. Li, Z.; Ghodrati, S.; Yazdanbakhsh, A.; Esmaeilzadeh, H.; Kang, M. Accelerating Atention through Gradient-Based Learned Runtime Pruning. In Proceedings of the 49th Annual International Symposium on Computer Architecture, New York, NY, USA, 18–22 June 2022; Volume 14, pp. 902–915. [CrossRef]
80. Wang, Z.; Lin, J.; Wang, Z. Accelerating Recurrent Neural Networks: A Memory-Efficient Approach. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2017**, *25*, 2763–2775. [CrossRef]
81. Wang, S.; Li, Z.; Ding, C.; Yuan, B.; Qiu, Q.; Wang, Y.; Liang, Y. C-LSTM: Enabling efficient LSTM using structured compression techniques on FPGAs. In Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2018, Monterey, CA, USA, 25–27 February 2018; pp. 11–20. [CrossRef]

82. Pham, H.; Guan, M.Y.; Zoph, B.; Le, Q.V.; Dean, J. Efficient Neural Architecture Search via parameter Sharing. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 10–15 July 2018; Volume 9, pp. 6522–6531.
83. Liu, Y.; Sun, Y.; Xue, B.; Zhang, M.; Yen, G.G.; Tan, K.C. A Survey on Evolutionary Neural Architecture Search. *IEEE Trans. Neural Networks Learn. Syst.* **2023**, *34*, 550–570. [CrossRef]
84. Denton, E.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. *Adv. Neural Inf. Process. Syst.* **2014**, *2*, 1269–1277. [CrossRef]
85. Chen, P.H.; Hsian-Fu, Y.; Dhillon, I.S.; Cho-Jui, H. DRONE: Data-aware Low-rank Compression for Large NLP Models. *Adv. Neural Inf. Process. Syst.* **2021**, *35*, 29321–29334.
86. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2019, Minneapolis, MN, USA, 2–7 June 2019; Volume 1, pp. 4171–4186. [CrossRef]
87. Dolan, W.B.; Brockett, C. Automatically Constructing a Corpus of Sentential Paraphrases. In Proceedings of the Third International Workshop on Paraphrasing (IWP2005), Jeju Island, Republic of Korea, 14 October 2005; pp. 9–16.
88. Warstadt, A.; Singh, A.; Bowman, S.R. Neural Network Acceptability Judgments. *Trans. Assoc. Comput. Linguist.* **2019**, *7*, 625–641. [CrossRef]
89. STSBenchmark. STSbenchmark-Stswiki. 2019. Available online: https://ixa2.si.ehu.eus/stswiki/index.php/STSbenchmark (accessed on 28 July 2023).
90. Borup, K.; Andersen, L.N. Even your Teacher Needs Guidance: Ground-Truth Targets Dampen Regularization Imposed by Self-Distillation. *Adv. Neural Inf. Process. Syst.* **2021**, *7*, 5316–5327. [CrossRef]
91. Chen, G.; Choi, W.; Yu, X.; Han, T.; Chandraker, M. Learning efficient object detection models with knowledge distillation. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 743–752.
92. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
93. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [CrossRef]
94. Wen, T.; Lai, S.; Qian, X. Preparing lessons: Improve knowledge distillation with better supervision. *Neurocomputing* **2021**, *454*, 25–33. [CrossRef]
95. Krizhevsky, A.; Hinton, G. Learning multiple layers of features from tiny images. *Cs.Toronto.Edu* **2009**, 1–58.
96. Darlow, L.N.; Crowley, E.J.; Antoniou, A.; Storkey, A.J. CINIC-10 is not ImageNet or CIFAR-10. *arXiv* **2018**. [CrossRef]
97. Le, Y.; Yang, X. *Tiny ImageNet Visual Recognition Challenge*; Stanford CS231N; Stanford University: Stanford, CA, USA, 2015.
98. Zhang, J.; Peng, H.; Wu, K.; Liu, M.; Xiao, B.; Fu, J.; Yuan, L. *MiniViT: Compressing Vision Transformers with Weight Multiplexing*; Technical Report; Microsoft: Redmond, DC, USA, 2022.
99. Amin, H.; Curtis, K.M.; Hayes-Gill, B.R. Piecewise linear approximation applied to nonlinear function of a neural network. *IEEE Proc. Circuits Devices Syst.* **1997**, *144*, 313–317. [CrossRef]
100. Hu, Z.; Zhang, J.; Ge, Y. Handling Vanishing Gradient Problem Using Artificial Derivative. *IEEE Access* **2021**, *9*, 22371–22377. [CrossRef]
101. Zhao, Z.; Barijough, K.M.; Gerstlauer, A. DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2348–2359. [CrossRef]
102. Lane, N.D.; Bhattacharya, S.; Georgiev, P.; Forlivesi, C.; Jiao, L.; Qendro, L.; Kawsar, F. DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices. In Proceedings of the 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN 2016, Vienna, Austria, 11–14 April 2016. [CrossRef]
103. Li, H.; Ota, K.; Dong, M. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
104. Du, J.; Zhu, X.; Shen, M.; Du, Y.; Lu, Y.; Xiao, N.; Liao, X. Model Parallelism Optimization for Distributed Inference Via Decoupled CNN Structure. *IEEE Trans. Parallel Distrib. Syst.* **2021**, *32*, 1665–1676. [CrossRef]
105. Hemmat, M.; Davoodi, A.; Hu, Y.H. EdgenAI: Distributed Inference with Local Edge Devices and Minimal Latency. In Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC, Taipei, Taiwan, 17–20 January 2022; pp. 544–549. [CrossRef]
106. Teerapittayanon, S.; McDanel, B.; Kung, H.T. BranchyNet: Fast inference via early exiting from deep neural networks. In Proceedings of the International Conference on Pattern Recognition, Cancun, Mexico, 4–8 December 2016; pp. 2464–2469. [CrossRef]
107. Zhou, W.; Xu, C.; Ge, T.; McAuley, J.; Xu, K.; Wei, F. BERT loses patience: Fast and robust inference with early exit. In Proceedings of the Advances in Neural Information Processing Systems, Virtual, 6–12 December 2020. [CrossRef]
108. Lan, Z.; Chen, M.; Goodman, S.; Gimpel, K.; Sharma, P.; Soricut, R. Albert: A Lite Bert for Self-Supervised Learning of Language Representations. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. [CrossRef]
109. Drolia, U.; Guo, K.; Tan, J.; Gandhi, R.; Narasimhan, P. Cachier: Edge-Caching for Recognition Applications. In Proceedings of the International Conference on Distributed Computing Systems, Atlanta, GA, USA, 5–8 June 2017; pp. 276–286. [CrossRef]

110. Xu, M.; Zhu, M.; Liu, Y.; Lin, F.X.; Liu, X. DeepCache: Principled cache for mobile deep vision. In Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM, New Delhi, India, 29 October–2 November 2018; pp. 129–144. [CrossRef]
111. Li, Y.; Zhang, C.; Han, S.; Zhang, L.L.; Yin, B.; Liu, Y.; Xu, M. Boosting Mobile CNN Inference through Semantic Memory. In Proceedings of the 29th ACM International Conference on Multimedia, MM 2021, Online, 20–24 October 2021; pp. 2362–2371. [CrossRef]
112. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1–9. [CrossRef]
113. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520. [CrossRef]
114. Howard, A.; Sandler, M.; Chen, B.; Wang, W.; Chen, L.C.; Tan, M.; Chu, G.; Vasudevan, V.; Zhu, Y.; Pang, R.; et al. Searching for mobileNetV3. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2021; pp. 1314–1324. [CrossRef]
115. Hu, J.; Shen, L.; Albanie, S.; Sun, G.; Wu, E. Squeeze-and-Excitation Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 2011–2023. [CrossRef]
116. Yang, T.J.; Howard, A.; Chen, B.; Zhang, X.; Go, A.; Sandler, M.; Sze, V.; Adam, H. NetAdapt: Platform-aware neural network adaptation for mobile applications. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2018; Volume 11214, pp. 289–304. [CrossRef]
117. Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; Xu, C. GhostNet: More features from cheap operations. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1577–1586. [CrossRef]
118. Dong, P.; Wang, S.; Niu, W.; Zhang, C.; Lin, S.; Li, Z.; Gong, Y.; Ren, B.; Lin, X.; Tao, D. RTMobile: Beyond real-time mobile acceleration of RNNs for speech recognition. In Proceedings of the Design Automation Conference, Online, 20–24 July 2020. [CrossRef]
119. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5999–6009. [CrossRef]
120. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16×16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.
121. Wang, T.; Roberts, A.; Hesslow, D.; Le Scao, T.; Chung, H.W.; Beltagy, I.; Launay, J.; Raffel, C. What Language Model Architecture and Pretraining Objective Work Best for Zero-Shot Generalization? *Proc. Mach. Learn. Res.* **2022**, *162*, 22964–22984.
122. Wang, X.; Zhang, L.L.; Wang, Y.; Yang, M. Towards Efficient Vision Transformer Inference: A First Study of Transformers on Mobile Devices. In Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications, HotMobile 2022, Orange County, CA, USA, 22–23 February 2022; pp. 1–7. [CrossRef]
123. Graham, B.; El-Nouby, A.; Touvron, H.; Stock, P.; Joulin, A.; Jégou, H.; Douze, M. LeViT: A Vision Transformer in ConvNet's Clothing for Faster Inference. In Proceedings of the IEEE International Conference on Computer Vision, Montreal, BC, Canada, 11–17 October 2021; pp. 12239–12249. [CrossRef]
124. Roh, B.; Shin, J.; Shin, W.; Kim, S. Sparse DETR: Efficient End-to-End Object Detection with Learnable Sparsity. *arXiv* **2021**. [CrossRef]
125. Li, Y.; Yuan, G.; Wen, Y.; Hu, E.; Evangelidis, G.; Tulyakov, S.; Wang, Y.; Ren, J. EfficientFormer: Vision Transformers at MobileNet Speed. *arXiv* **2022**. [CrossRef]
126. McMahan, B.; Daniel Ramage. Federated Learning: Collaborative Machine Learning without Centralized Training Data—Google Research Blog. Available online: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (accessed on 28 July 2023).
127. Wink, T.; Nochta, Z. An Approach for Peer-to-Peer Federated Learning. In Proceedings of the 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2021, Taipei, Taiwan, 21–24 June 2021; pp. 150–157. [CrossRef]
128. Brendan McMahan, H.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Ft. Lauderdale, FL, USA, 20–22 April 2017. [CrossRef]
129. Kang, Y.; Hauswald, J.; Gao, C.; Rovinski, A.; Mudge, T.; Mars, J.; Tang, L. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ACM SIGPLAN Not.* **2017**, *52*, 615–629. [CrossRef]
130. Zhu, H.; Xu, J.; Liu, S.; Jin, Y. Federated learning on non-IID data: A survey. *Neurocomputing* **2021**, *465*, 371–390. [CrossRef]
131. Wang, L.; Xu, S.; Wang, X.; Zhu, Q. Addressing Class Imbalance in Federated Learning. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021, Online, 2–9 February 2021; Volume 11B, pp. 10165–10173. [CrossRef]
132. Xu, C.; Qu, Y.; Xiang, Y.; Gao, L. Asynchronous Federated Learning on Heterogeneous Devices: A Survey. *ACM Comput. Surv* **2021**, *37*, 27. [CrossRef]

133. Alistarh, D.; Grubic, D.; Li, J.Z.; Tomioka, R.; Vojnovic, M. QSGD: Communication-efficient SGD via gradient quantization and encoding. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 1710–1721.
134. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–19. [CrossRef]
135. Xu, J.; Glicksberg, B.S.; Su, C.; Walker, P.; Bian, J.; Wang, F. Federated Learning for Healthcare Informatics. *J. Healthc. Inform. Res.* **2021**, *5*, 1–19. [CrossRef] [PubMed]
136. Mori, J.; Teranishi, I.; Furukawa, R. Continual Horizontal Federated Learning for Heterogeneous Data. In Proceedings of the International Joint Conference on Neural Networks, Padua, Italy, 18–23 July 2022. [CrossRef]
137. Nock, R.; Hardy, S.; Henecka, W.; Ivey-Law, H.; Patrini, G.; Smith, G.; Thorne, B. Entity Resolution and Federated Learning get a Federated Resolution. *arXiv* **2018**. [CrossRef]
138. Feng, S.; Yu, H. Multi-Participant Multi-Class Vertical Federated Learning. *arXiv* **2020**. [CrossRef]
139. Li, Y.; Nie, F.; Huang, H.; Huang, J. Large-scale multi-view spectral clustering via bipartite graph. *Proc. Natl. Conf. Artif. Intell.* **2015**, *4*, 2750–2756. [CrossRef]
140. Chen, Y.; Qin, X.; Wang, J.; Yu, C.; Gao, W. FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare. *IEEE Intell. Syst.* **2020**, *35*, 83–93. [CrossRef]
141. Wang, K.I.; Zhou, X.; Liang, W.; Yan, Z.; She, J. Federated Transfer Learning Based Cross-Domain Prediction for Smart Manufacturing. *IEEE Trans. Ind. Inform.* **2022**, *18*, 4088–4096. [CrossRef]
142. Ferryman, J.; Shahrokni, A. PETS2009: Dataset and challenge. In Proceedings of the 12th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance, PETS-Winter 2009, Snowbird, UT, USA, 7–12 December 2009. [CrossRef]
143. Li, Q.; Wen, Z.; Wu, Z.; Hu, S.; Wang, N.; Li, Y.; Liu, X.; He, B. A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 3347–3366. [CrossRef]
144. Lyu, L.; Yu, H.; Zhao, J.; Yang, Q. Threats to Federated Learning. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2020; Volume 12500, pp. 3–16. [CrossRef]
145. Yang, Z.; Xu, B.; Luo, W.; Chen, F. Autoencoder-based representation learning and its application in intelligent fault diagnosis: A review. *Meas. J. Int. Meas. Confed.* **2022**, *189*, 110460. [CrossRef]
146. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Vincent Poor, H. Federated Learning for Internet of Things: A Comprehensive Survey. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 1622–1658. [CrossRef]
147. Ghimire, B.; Rawat, D.B. Recent Advances on Federated Learning for Cybersecurity and Cybersecurity for Federated Learning for Internet of Things. *IEEE Internet Things J.* **2022**, *9*, 8229–8249. [CrossRef]
148. Sun, T.; Li, D.; Wang, B. Decentralized Federated Averaging. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 4289–4301. [CrossRef] [PubMed]
149. Reisizadeh, A.; Jadbabaie, A.; Mokhtari, A.; Hassani, H.; Pedarsani, R. FedPAQ: A Communication-Efficient Federated Learning Method with Periodic Averaging and Quantization. *Proc. Mach. Learn. Res.* **2020**, *108*, 2021–2031. [CrossRef]
150. Wu, C.; Wu, F.; Lyu, L.; Huang, Y.; Xie, X. Communication-efficient federated learning via knowledge distillation. *Nat. Commun.* **2022**, *13*, 2032. [CrossRef] [PubMed]
151. DInh, C.T.; Tran, N.H.; Nguyen, M.N.; Hong, C.S.; Bao, W.; Zomaya, A.Y.; Gramoli, V. Federated Learning over Wireless Networks: Convergence Analysis and Resource Allocation. *IEEE/ACM Trans. Netw.* **2021**, *29*, 398–409. [CrossRef]
152. Stiglitz, J.E. Self-selection and Pareto efficient taxation. *J. Public Econ.* **1982**, *17*, 213–240. [CrossRef]
153. Xu, Z.; Yu, F.; Xiong, J.; Chen, X. Helios: Heterogeneity-Aware Federated Learning with Dynamically Balanced Collaboration. In Proceedings of the Design Automation Conference, San Francisco, CA, USA, 5–9 December 2021; pp. 997–1002. [CrossRef]
154. Hahn, S.J.; Jeong, M.; Lee, J. Connecting Low-Loss Subspace for Personalized Federated Learning. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 14–18 August 2022; pp. 505–515. [CrossRef]
155. Gupta, O.; Raskar, R. Distributed learning of deep neural network over multiple agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [CrossRef]
156. Goodfellow, I.J.; Vinyals, O.; Saxe, A.M. Qualitatively characterizing neural network optimization problems. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015. [CrossRef]
157. Vepakomma, P.; Gupta, O.; Swedish, T.; Raskar, R. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv* **2018**. [CrossRef]
158. Thapa, C.; Arachchige, P.C.M.; Camtepe, S.; Sun, L. SplitFed: When Federated Learning Meets Split Learning. In Proceedings of the 36th AAAI Conference on Innovative Applications of Artificial Intelligence, AAAI 2022, Arlington, VA, USA, 17–19 November 2022; Volume 36, pp. 8485–8493. [CrossRef]
159. Panigrahi, S.; Nanda, A.; Swarnkar, T. A Survey on Transfer Learning. *Smart Innov. Syst. Technol.* **2021**, *194*, 781–789. [CrossRef]
160. Sugiyama, M.; Suzuki, T.; Nakajima, S.; Kashima, H.; Von Bünau, P.; Kawanabe, M. Direct importance estimation for covariate shift adaptation. *Ann. Inst. Stat. Math.* **2008**, *60*, 699–746. [CrossRef]

161. Huang, J.; Smola, A.J.; Gretton, A.; Borgwardt, K.M.; Schölkopf, B. Correcting Sample Selection Bias by Unlabeled Data. In Proceedings of the NIPS 2006: 19th International Conference on Neural Information Processing Systems, Whistler, BC, Canada, 8–9 December 2006; pp. 601–608. [CrossRef]
162. Singh, K.K.; Mahajan, D.; Grauman, K.; Lee, Y.J.; Feiszli, M.; Ghadiyaram, D. Don't Judge an Object by Its Context: Learning to Overcome Contextual Bias. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 11067–11075. [CrossRef]
163. Zhang, Y.; Liu, T.; Long, M.; Jordan, M.I. Bridging theory and algorithm for domain adaptation. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; pp. 12805–12823. [CrossRef]
164. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2010**, *22*, 1345–1359. [CrossRef]
165. Weiss, K.; Khoshgoftaar, T.M.; Wang, D.D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [CrossRef]
166. Zhou, K.; Liu, Z.; Qiao, Y.; Xiang, T.; Loy, C.C. Domain Generalization: A Survey. *arXiv* **2021**. [CrossRef] [PubMed]
167. Li, X.; Grandvalet, Y.; Davoine, F. Explicit inductive bias for transfer learning with convolutional networks. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, 10–15 July 2018; Volume 6, pp. 4408–4419. [CrossRef]
168. Zhi, W.; Chen, Z.; Yueng, H.W.F.; Lu, Z.; Zandavi, S.M.; Chung, Y.Y. Layer Removal for Transfer Learning with Deep Convolutional Neural Networks. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2017; Volume 10635, pp. 460–469. [CrossRef]
169. Chu, B.; Madhavan, V.; Beijbom, O.; Hoffman, J.; Darrell, T. Best practices for fine-tuning visual classifiers to new domains. In *Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2016. [CrossRef]
170. Wang, M.; Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, *312*, 135–153. [CrossRef]
171. You, K.; Long, M.; Cao, Z.; Wang, J.; Jordan, M.I. Universal domain adaptation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 2715–2724. [CrossRef]
172. Rios-Urrego, C.D.; Vásquez-Correa, J.C.; Orozco-Arroyave, J.R.; Nöth, E. Transfer learning to detect parkinson's disease from speech in different languages using convolutional neural networks with layer freezing. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer Science and Business Media Deutschland GmbH: Cham, Switzerland, 2020; Volume 12284, pp. 331–339. [CrossRef]
173. Ding, N.; Qin, Y.; Yang, G.; Wei, F.; Yang, Z.; Su, Y.; Hu, S.; Chen, Y.; Chan, C.M.; Chen, W.; et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nat. Mach. Intell.* **2023**, *5*, 220–235. [CrossRef]
174. Kara, O.; Sehanobish, A.; Corzo, H.H. Fine-tuning Vision Transformers for the Prediction of State Variables in Ising Models. *arXiv* **2021**. [CrossRef]
175. Howard, J.; Ruder, S. Universal language model fine-tuning for text classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, 15–20 July 2018; Volume 1, pp. 328–339. [CrossRef]
176. Houlsby, N.; Giurgiu, A.; Jastrzçbski, S.; Morrone, B.; de Laroussilhe, Q.; Gesmundo, A.; Attariyan, M.; Gelly, S. Parameter-efficient transfer learning for NLP. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; pp. 4944–4953. [CrossRef]
177. Lester, B.; Al-Rfou, R.; Constant, N. The Power of Scale for Parameter-Efficient Prompt Tuning. In Proceedings of the EMNLP 2021—2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 16–20 November 2021; pp. 3045–3059. [CrossRef]
178. Kojima, T.; Gu, S.S.; Reid, M.; Matsuo, Y.; Iwasawa, Y. Large Language Models are Zero-Shot Reasoners. *arXiv* **2022**. [CrossRef]
179. Chung, H.W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; et al. Scaling Instruction-Finetuned Language Models. *arXiv* **2022**. [CrossRef]
180. Schratz, P.; Muenchow, J.; Iturritxa, E.; Richter, J.; Brenning, A. Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data. *Ecol. Model.* **2019**, *406*, 109–120. [CrossRef]
181. Wang, Z.; Dai, Z.; Poczos, B.; Carbonell, J. Characterizing and avoiding negative transfer. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 11285–11294. [CrossRef]
182. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2020; Volume 33, pp. 1877–1901.
183. Roy, S.; Roth, D. Solving general arithmetic word problems. In Proceedings of the EMNLP 2015: Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1743–1752. [CrossRef]
184. Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. Training Verifiers to Solve Math Word Problems. *arXiv* **2021**. [CrossRef]
185. Vanschoren, J. Meta-Learning: A Survey. *arXiv* **2018**. [CrossRef]
186. Peng, H. A Brief Summary of Interactions Between Meta-Learning and Self-Supervised Learning. *arXiv* **2021**. [CrossRef]
187. Finn, C.; Abbeel, P.; Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Proceedings of the 35th International Conference on Machine Learning, ICML 2017, Sydney, Australia, 6–11 August 2017; Volume 3, pp. 1856–1868. [CrossRef]

188. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Cheng, D. Learning k for kNN Classification. *ACM Trans. Intell. Syst. Technol.* **2017**, *8*, 1–19. [CrossRef]
189. Hamerly, G.; Elkan, C. Learning the k in kmeans. In *Advances in Neural Information Processing Systems*; Thrun, S., Saul, L., Schölkopf, B., Eds.; MIT Press: Cambridge, MA, USA, 2004; Volume 17, pp. 1–8.
190. Santoro, A.; Bartunov, S.; Botvinick, M.; Wierstra, D.; Lillicrap, T. Meta-Learning with Memory-Augmented Neural Networks. In Proceedings of the 33rd International Conference on Machine Learning, ICML 2016, New York, NY, USA, 19–24 June 2016; Volume 4, pp. 2740–2751.
191. Hospedales, T.; Antoniou, A.; Micaelli, P.; Storkey, A. Meta-Learning in Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 5149–5169. [CrossRef] [PubMed]
192. Sun, X.; Wang, B.; Wang, Z.; Li, H.; Li, H.; Fu, K. Research Progress on Few-Shot Learning for Remote Sensing Image Interpretation. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2021**, *14*, 2387–2402. [CrossRef]
193. Gupta, A.; Mendonca, R.; Liu, Y.X.; Abbeel, P.; Levine, S. Meta-reinforcement learning of structured exploration strategies. In Proceedings of the Advances in Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018; pp. 5302–5311. [CrossRef]
194. Griffiths, T.L.; Callaway, F.; Chang, M.B.; Grant, E.; Krueger, P.M.; Lieder, F. Doing more with less: Meta-reasoning and meta-learning in humans and machines. *Curr. Opin. Behav. Sci.* **2019**, *29*, 24–30. [CrossRef]
195. Wang, Y.; Yao, Q.; Kwok, J.T.; Ni, L.M. Generalizing from a Few Examples: A Survey on Few-shot Learning. *ACM Comput. Surv.* **2020**, *53*, 1–34. [CrossRef]
196. Chen, W.Y.; Wang, Y.C.F.; Liu, Y.C.; Kira, Z.; Huang, J.B. A closer look at few-shot classification. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019. [CrossRef]
197. Bennequin, E. Meta-learning algorithms for Few-Shot Computer Vision. *arXiv* **2019**. [CrossRef]
198. Ravi, S.; Larochelle, H. Optimization as a model for few-shot learning. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017; pp. 1–11.
199. Romera-Paredes, B.; Torr, P.H. An embarrassingly simple approach to zero-shot learning. In Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6–11 July 2015; Volume 3, pp. 2142–2151. [CrossRef]
200. Verma, V.K.; Rai, P. A Simple Exponential Family Framework for Zero-Shot Learning. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2017; Volume 10535, pp. 792–808. [CrossRef]
201. Radford, A.; Kim, J.W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. Learning Transferable Visual Models From Natural Language Supervision. *Proc. Mach. Learn. Res.* **2021**, *139*, 8748–8763. [CrossRef]
202. Belkhale, S.; Li, R.; Kahn, G.; McAllister, R.; Calandra, R.; Levine, S. Model-Based Meta-Reinforcement Learning for Flight with Suspended Payloads. *IEEE Robot. Autom. Lett.* **2021**, *6*, 1471–1478. [CrossRef]
203. Rajeswaran, A.; Kakade, S.M.; Finn, C.; Levine, S. Meta-learning with implicit gradients. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 113–124. [CrossRef]
204. Finn, C.; Rajeswaran, A.; Kakade, S.; Levine, S. Online meta-learning. In Proceedings of the 36th International Conference on Machine Learning, ICML 2019, Long Beach, CA, USA, 9–15 June 2019; pp. 3398–3410. [CrossRef]
205. Wang, W.; Zheng, V.W.; Yu, H.; Miao, C. A survey of zero-shot learning: Settings, methods, and applications. *ACM Trans. Intell. Syst. Technol.* **2019**, *10*, 1–37. [CrossRef]
206. Lampert, C.H.; Nickisch, H.; Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 951–958. [CrossRef]
207. Patterson, G.; Hays, J. SUN attribute database: Discovering, annotating, and recognizing scene attributes. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 2751–2758. [CrossRef]
208. Farhadi, A.; Endres, I.; Hoiem, D.; Forsyth, D. Describing objects by their attributes. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 1778–1785. [CrossRef]
209. Lake, B.M.; Salakhutdinov, R.; Tenenbaum, J.B. The Omniglot challenge: A 3-year progress report. *Curr. Opin. Behav. Sci.* **2019**, *29*, 97–104. [CrossRef]
210. Shorten, C.; Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. *J. Big Data* **2019**, *6*, 1–48. [CrossRef]
211. Shorten, C.; Khoshgoftaar, T.M.; Furht, B. Text Data Augmentation for Deep Learning. *J. Big Data* **2021**, *8*, 1–34. [CrossRef] [PubMed]
212. Liu, X.; Zhang, F.; Hou, Z.; Mian, L.; Wang, Z.; Zhang, J.; Tang, J. Self-supervised Learning: Generative or Contrastive. *arXiv* **2021**. [CrossRef]
213. Ericsson, L.; Gouk, H.; Loy, C.C.; Hospedales, T.M. Self-Supervised Representation Learning: Introduction, advances, and challenges. *IEEE Signal Process. Mag.* **2022**, *39*, 42–62. [CrossRef]
214. Jing, L.; Tian, Y. Self-Supervised Visual Feature Learning with Deep Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 4037–4058. [CrossRef]
215. Kalyan, K.S.; Rajasekharan, A.; Sangeetha, S. AMMUS: A Survey of Transformer-based Pretrained Models in Natural Language Processing. *arXiv* **2021**. [CrossRef]

216. Xie, Y.; Xu, Z.; Zhang, J.; Wang, Z.; Ji, S. Self-Supervised Learning of Graph Neural Networks: A Unified Review. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 2412–2429. [CrossRef]
217. Baevski, A.; Hsu, W.N.; Xu, Q.; Babu, A.; Gu, J.; Auli, M. data2vec: A General Framework for Self-supervised Learning in Speech, Vision and Language. *arXiv* **2022**. [CrossRef]
218. Pathak, D.; Krahenbuhl, P.; Donahue, J.; Darrell, T.; Efros, A.A. Context Encoders: Feature Learning by Inpainting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 1–9.
219. Hu, Z.; Dong, Y.; Wang, K.; Chang, K.W.; Sun, Y. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Online, 6–10 July 2020; pp. 1857–1867. [CrossRef]
220. He, K.; Fan, H.; Wu, Y.; Xie, S.; Girshick, R. Momentum Contrast for Unsupervised Visual Representation Learning. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 9726–9735. [CrossRef]
221. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G. A simple framework for contrastive learning of visual representations. In Proceedings of the 37th International Conference on Machine Learning, ICML 2020, Vienna, Austria, 12–18 July 2020; pp. 1575–1585.
222. Donahue, J.; Darrell, T.; Krähenbühl, P. Adversarial feature learning. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017. [CrossRef]
223. Donahue, J.; Simonyan, K. Large scale adversarial representation learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–11. [CrossRef]
224. Iizuka, S.; Simo-Serra, E.; Ishikawa, H. Globally and locally consistent image completion. *ACM Trans. Graph.* **2017**, *36*. [CrossRef]
225. Tran, M.T.; Kim, S.H.; Yang, H.J.; Lee, G.S. Deep learning-based inpainting for chest X-ray image. In Proceedings of the 9th International Conference on Smart Media and Applications, Jeju, Republic of Korea, 17–19 September 2020; pp. 267–271. [CrossRef]
226. Zhuang, W.; Wen, Y.; Zhang, S. Divergence-Aware Federated Self-Supervised Learning. In Proceedings of the 10th International Conference on Learning Representations, ICLR 2022, Vienna, Austria, 7–11 May 2022. [CrossRef]
227. Mao, H.H. A Survey on Self-supervised Pre-training for Sequential Transfer Learning in Neural Networks. *arXiv* **2020**. [CrossRef]
228. Baevski, A.; Zhou, H.; Mohamed, A.; Auli, M. wav2vec 2.0: A framework for self-supervised learning of speech representations. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–12 December 2020. [CrossRef]
229. Panayotov, V.; Chen, G.; Povey, D.; Khudanpur, S. Librispeech: An ASR corpus based on public domain audio books. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, South Brisbane, Australia, 19–24 April 2015; pp. 5206–5210. [CrossRef]
230. Tian, Y.; Krishnan, D.; Isola, P. Contrastive Multiview Coding. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2020; Volume 12356, pp. 776–794. [CrossRef]
231. Oord, A.v.d.; Li, Y.; Vinyals, O. Representation Learning with Contrastive Predictive Coding. *arXiv* **2018**. [CrossRef]
232. Radford, A.; Metz, L.; Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. In Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, 2–4 May 2016. [CrossRef]
233. Clark, K.; Luong, M.T.; Le, Q.V.; Manning, C.D. Electra: Pre-Training Text Encoders As Discriminators Rather Than Generators. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, 26–30 April 2020. [CrossRef]
234. Dai, Q.; Li, Q.; Tang, J.; Wang, D. Adversarial network embedding. In Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, LA, USA, 2–7 February 2018; pp. 2167–2174. [CrossRef]
235. Zhang, Y.; Yang, Q. A Survey on Multi-Task Learning. *IEEE Trans. Knowl. Data Eng.* **2021**, *34*, 5586–5609. [CrossRef]
236. Li, W.; Zemi, D.H.; Redon, V.; Matthieu, L. Multi-Task Attention Network for Digital Context Classification from Internet Traffic. In Proceedings of the 2022 7th International Conference on Machine Learning Technologies (ICMLT), Rome, Italy, 11–13 March 2022; pp. 1–12. [CrossRef]
237. Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *arXiv* **2017**. [CrossRef]
238. Yang, Y.; Hospedales, T.M. Trace norm regularised deep multi-task learning. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017. [CrossRef]
239. Rago, A.; Piro, G.; Boggia, G.; Dini, P. Multi-Task Learning at the Mobile Edge: An Effective Way to Combine Traffic Classification and Prediction. *IEEE Trans. Veh. Technol.* **2020**, *69*, 10362–10374. [CrossRef]
240. Chen, Q.; Zheng, Z.; Hu, C.; Wang, D.; Liu, F. On-Edge Multi-Task Transfer Learning: Model and Practice with Data-Driven Task Allocation. *IEEE Trans. Parallel Distrib. Syst.* **2020**, *31*, 1357–1371. [CrossRef]
241. Aha, D.W.; Kibler, D.; Albert, M.K. Instance-based learning algorithms. *Mach. Learn.* **1991**, *6*, 37–66. [CrossRef]
242. Ghosh, J.; Nag, A. An Overview of Radial Basis Function Networks. *Stud. Fuzziness Soft Comput.* **2001**, *67*, 1–36. [CrossRef]
243. Watson, I.; Marir, F. Case-Based Reasoning: A Review. *Knowl. Eng. Rev.* **2009**, *9*, 327–354. [CrossRef]
244. Zhang, W.; Chen, X.; Liu, Y.; Xi, Q. A Distributed Storage and Computation k-Nearest Neighbor Algorithm Based Cloud-Edge Computing for Cyber-Physical-Social Systems. *IEEE Access* **2020**, *8*, 50118–50130. [CrossRef]

245. González-Briones, A.; Prieto, J.; De La Prieta, F.; Herrera-Viedma, E.; Corchado, J.M. Energy optimization using a case-based reasoning strategy. *Sensors* **2018**, *18*, 865. [CrossRef] [PubMed]
246. Ratner, A.; Varma, P.; Hancock, B.; Ré, C. Weak Supervision: A New Programming Paradigm for Machine Learning | SAIL Blog. 2019. Available online: http://ai.stanford.edu/blog/weak-supervision (accessed on 28 July 2023).
247. Zhou, Z.H. A brief introduction to weakly supervised learning. *Natl. Sci. Rev.* **2018**, *5*, 44–53. [CrossRef]
248. Wei, X.S.; Wu, J.; Zhou, Z.H. Scalable algorithms for multi-instance learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 975–987. [CrossRef] [PubMed]
249. Frénay, B.; Verleysen, M. Classification in the presence of label noise: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2014**, *25*, 845–869. [CrossRef]
250. Nodet, P.; Lemaire, V.; Bondu, A.; Cornuejols, A.; Ouorou, A. From Weakly Supervised Learning to Biquality Learning: An Introduction. In Proceedings of the International Joint Conference on Neural Networks, Shenzhen, China, 18–22 July 2021. [CrossRef]
251. Mohri, M.; Rostamizadeh, A.; Talwalkar, A. *Foundations of Machine Learning*, 2nd ed.; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2018.
252. van Engelen, J.E.; Hoos, H.H. A survey on semi-supervised learning. *Mach. Learn.* **2020**, *109*, 373–440. [CrossRef]
253. Rahman, S.; Khan, S.; Barnes, N. Transductive learning for zero-shot object detection. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 6081–6090. [CrossRef]
254. Settles, B. Active Learning Literature Survey. *Mach. Learn.* **2010**, *15*, 201–221. [CrossRef]
255. Sharma, Y.; Shrivastava, A.; Ehsan, L.; Moskaluk, C.A.; Syed, S.; Brown, D.E. Cluster-to-Conquer: A Framework for End-to-End Multi-Instance Learning for Whole Slide Image Classification. *Proc. Mach. Learn. Res.* **2021**, *143*, 682–698.
256. Eberts, M.; Ulges, A. An end-to-end model for entity-level relation extraction using multi-instance learning. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2021, Online, 19–23 April 2021; pp. 3650–3660. [CrossRef]
257. Luo, Z.; Guillory, D.; Shi, B.; Ke, W.; Wan, F.; Darrell, T.; Xu, H. Weakly-Supervised Action Localization with Expectation-Maximization Multi-Instance Learning. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2020; Volume 12374, pp. 729–745. [CrossRef]
258. Raju, A.; Yao, J.; Haq, M.M.H.; Jonnagaddala, J.; Huang, J. Graph Attention Multi-instance Learning for Accurate Colorectal Cancer Staging. In *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Cham, Switzerland, 2020; Volume 12265, pp. 529–539. [CrossRef]
259. Müller, R.; Kornblith, S.; Hinton, G. When does label smoothing help? In Proceedings of the NIPS'19: 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32, pp. 4694–4703.
260. Gao, W.; Wang, L.; Li, Y.F.; Zhou, Z.H. Risk minimization in the presence of label noise. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI 2016, Phoenix, AZ, USA, 12–17 February 2016; pp. 1575–1581.
261. Lukasik, M.; Bhojanapalli, S.; Menon, A.; Kumar, S. Does label smoothing mitigate label noise? In Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, 12–18 July 2020; Volume 119, pp. 6448–6458.
262. Arazo, E.; Ortego, D.; Albert, P.; O'Connor, N.; Mcguinness, K. Unsupervised Label Noise Modeling and Loss Correction. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; Volume 97, pp. 312–321.
263. Losing, V.; Hammer, B.; Wersing, H. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing* **2018**, *275*, 1261–1274. [CrossRef]
264. He, J.; Mao, R.; Shao, Z.; Zhu, F. Incremental learning in online scenario. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 13923–13932. [CrossRef]
265. Lin, J.; Kolcz, A. Large-scale machine learning at Twitter. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Scottsdale, AZ, USA, 20–24 May 2012; pp. 793–804. [CrossRef]
266. Ling, C.X.; Bohn, T. A Deep Learning Framework for Lifelong Machine Learning. *arXiv* **2021**. [CrossRef]
267. Lu, J.; Liu, A.; Dong, F.; Gu, F.; Gama, J.; Zhang, G. Learning under Concept Drift: A Review. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 2346–2363. [CrossRef]
268. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526. [CrossRef] [PubMed]
269. Maloof, M.A.; Michalski, R.S. Incremental learning with partial instance memory. *Artif. Intell.* **2004**, *154*, 95–126. [CrossRef]
270. Piyasena, D.; Thathsara, M.; Kanagarajah, S.; Lam, S.K.; Wu, M. Dynamically Growing Neural Network Architecture for Lifelong Deep Learning on the Edge. In Proceedings of the 30th International Conference on Field-Programmable Logic and Applications, FPL 2020, Gothenburg, Sweden, 31 August–4 September 2020; pp. 262–268. [CrossRef]
271. Marsland, S.; Shapiro, J.; Nehmzow, U. A self-organising network that grows when required. *Neural Netw.* **2002**, *15*, 1041–1058. [CrossRef]
272. Singh, A.; Bhadani, R. *Mobile Deep Learning with TensorFlow Lite, ML Kit and Flutter*; Packt Publisher: Birmingham, UK, 2020; 380p.
273. Pang, B.; Nijkamp, E.; Wu, Y.N. Deep Learning With TensorFlow: A Review. *J. Educ. Behav. Stat.* **2020**, *45*, 227–248. [CrossRef]

274. PyTorch. Home | PyTorch. 2022. Available online: https://pytorch.org/mobile/home/ (accessed on 28 July 2023).
275. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In Proceedings of the NIPS'19: 33rd International Conference on Neural Information Processing Systems, hlVancouver, BC, Canada, 8–14 December 2019; Wallach, H., Larochelle, H., Beygelzimer, A., Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8026–8037.
276. Marques, O. Machine Learning with Core ML. In *Springer Briefs in Computer Science*; Springer: Berlin/Heidelberg, Germany, **2020**; pp. 29–40. [CrossRef]
277. Cass, S. Taking AI to the edge: Google's TPU now comes in a maker-friendly package. *IEEE Spectr.* **2019**, *56*, 16–17. [CrossRef]
278. Ionice, M.H.; Gregg, D. The Movidius Myriad architecture's potential for scientific computing. *IEEE Micro* **2015**, *35*, 6–14. [CrossRef]
279. STMicroelectronics. STM32 32-bit ARM Cortex MCUs-STMicroelectronics. 2014. Available online: http://www.st.com/web/catalog/mmc/FM141/SC1169 (accessed on 28 July 2023).
280. Sun, D.; Liu, S.; Gaudiot, J.L. Enabling Embedded Inference Engine with ARM Compute Library: A Case Study. *arXiv* **2017**, arXiv:1704.03751.
281. Jeong, E.J.; Kim, J.; Tan, S.; Lee, J.; Ha, S. Deep Learning Inference Parallelization on Heterogeneous Processors with TensorRT. *IEEE Embed. Syst. Lett.* **2022**, *14*, 15–18. [CrossRef]
282. NVIDIA. EGX Platform for Accelerated Computing | NVIDIA. 2023. Available online: https://www.nvidia.com/en-us/data-center/products/egx/ (accessed on 28 July 2023).
283. Qualcomm Technologies, I. Qualcomm Neural Processing SDK for AI—Qualcomm Developer Network. 2021. Available online: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk (accessed on 28 July 2023).
284. GitHub—Majianjia/Nnom: A Higher-Level Neural Network Library for Microcontrollers. 2021. Available online: https://github.com/majianjia/nnom (accessed on 28 July 2023).
285. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 22–25 July 2017; pp. 2261–2269. [CrossRef]
286. STMicroelectronics. X-CUBE-AI—AI Expansion Pack for STM32CubeMX—STMicroelectronics. 2022. Available online: https://www.st.com/en/embedded-software/x-cube-ai.html (accessed on 28 July 2023).
287. Narayanan, D.; Shoeybi, M.; Casper, J.; LeGresley, P.; Patwary, M.; Korthikanti, V.; Vainbrand, D.; Kashinkunti, P.; Bernauer, J.; Catanzaro, B.; et al. Efficient Large-Scale Language Model Training on GPU Clusters Using Megatron-LM. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, St. Louis, MI, USA, 14–19 November 2021. [CrossRef]
288. Rasley, J.; Rajbhandari, S.; Ruwase, O.; He, Y. DeepSpeed: System Optimizations Enable Training Deep Learning Models with over 100 Billion Parameters. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Online, 6–10 July 2020; pp. 3505–3506. [CrossRef]
289. GitHub—Tensorflow/Mesh: Mesh TensorFlow: Model Parallelism Made Easier. Available online: https://github.com/tensorflow/mesh (accessed on 28 July 2023).
290. Chen, L. *Deep Learning and Practice with MindSpore*; Cognitive Intelligence and Robotics; Springer: Singapore, 2021; pp. 1–531. [CrossRef]
291. Google Inc. TensorFlow Federated. 2022. Available online: https://www.tensorflow.org/federated (accessed on 28 July 2023).
292. Intel. Intel/Openfl: An Open Framework for Federated Learning. Available online: https://github.com/intel/openfl (accessed on 28 July 2023).
293. Nvidia Clara. NVIDIA Clara | NVIDIA Developer. 2020. Available online: https://developer.nvidia.com/blog/federated-learning-clara/ (accessed on 28 July 2023).
294. Standards by ISO/IEC JTC. ISO—ISO/IEC JTC 1/SC 42—Artificial Intelligence. 2022. Available online: https://www.iso.org/committee/6794475/x/catalogue/ (accessed on 28 July 2023).
295. International Telecommunication Union (ITU). Focus Group on AI for Autonomous and Assisted Driving (FG-AI4AD). 2021. Available online: https://www.itu.int/en/ITU-T/focusgroups/ai4ad/Pages/default.aspx (accessed on 28 July 2023).
296. ITU-T FG-ML5G. Focus Group on Machine Learning for Future Networks Including 5G, 2018. Available online: https://www.itu.int/en/ITU-T/focusgroups/ml5g/Pages/default.aspx (accessed on 28 July 2023).
297. Dahmen-Lhuissier, S. ETSI—Multi-Access Edge Computing—Standards for MEC, 2020. Available online: https://www.etsi.org/technologies/multi-access-edge-computing (accessed on 28 July 2023).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.