Ad Hoc Networks 153 (2024) 103348

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Survey paper

A review of on-device machine learning for IoT: An energy perspective

Nazli Tekin[a,b,*], Ahmet Aris[a], Abbas Acar[a], Selcuk Uluagac[a], Vehbi Cagri Gungor[c]

[a] *Cyber-Physical Systems Security Lab, School of Computing and Information Science, Florida International University, Miami, FL 33174, USA*
[b] *Department of Software Engineering, Erciyes University, 38280 Kayseri, Turkey*
[c] *Department of Computer Engineering, Abdullah Gul University, 38080 Kayseri, Turkey*

A R T I C L E I N F O

Keywords:
Internet of Things
Machine Learning
Energy efficiency
Deep learning
Edge computing
Tiny machine learning

A B S T R A C T

Recently, there has been a substantial interest in on-device Machine Learning (ML) models to provide intelligence for the Internet of Things (IoT) applications such as image classification, human activity recognition, and anomaly detection. Traditionally, ML models are deployed in the cloud or centralized servers to take advantage of their abundant computational resources. However, sharing data with the cloud and third parties degrades privacy and may cause propagation delay in the network due to a large amount of transmitted data impacting the performance of real-time applications. To this end, deploying ML models on-device (i.e., on IoT devices), in which data does not need to be transmitted, becomes imperative. However, deploying and running ML models on already resource-constrained IoT devices is challenging and requires intense energy consumption. Numerous works have been proposed in the literature to address this issue. Although there are considerable works that discuss energy-aware ML approaches for on-device implementation, there remains a gap in the literature on a comprehensive review of this subject. In this paper, we provide a review of existing studies focusing on-device ML models for IoT applications in terms of energy consumption. One of the key contributions of this study is to introduce a taxonomy to define approaches for employing energy-aware on-device ML models on IoT devices in the literature. Based on our review in this paper, our key findings are provided and the open issues that can be investigated further by other researchers are discussed. We believe that this study will be a reference for practitioners and researchers who want to employ energy-aware on-device ML models for IoT applications.

1. Introduction

In recent years, the Internet of Things (IoT) has gained a significant attraction with the rapidly increasing number of connected smart devices such as smart thermostats, smart lights, and smart home appliances [1]. The sensing, processing, and decision-making capabilities of such smart devices offer effective solutions in many application fields. For instance, in the industry, autonomous monitoring and controlling the machinery systems reduce the operational and maintenance costs and enhance the service quality [2]. Moreover, IoT-enabled smart home systems provide remote access and control to home appliances and secure health care services [3].

Machine Learning (ML) brings intelligence to IoT applications. For example, a smart thermostat equipped with ML functionality can learn the routines of the users and automatically schedule them [4]. Moreover, a voice assistant with speech recognition [5], a smart lock with face recognition [6], and an ML-enabled router with intrusion detection [7] capabilities are just examples of ML-powered IoT applications.

Conventionally, ML algorithms are outsourced/deployed on the cloud or a centralized server to make decisions and future predictions. However, running the ML model on cloud servers can cause security, privacy, and latency issues. For example, large amounts of data collected from numerous IoT devices can cause congestion in the network that can result in unreliable communication and long delays [8], which is undesirable for real-time IoT applications. Geographically distant cloud environments may add up to this issue.

IoT edge computing is an emerging architecture that addresses such concerns [9]. In this architecture, edge devices can process the data collected from nearby IoT devices. Since the data does not need to be sent to the cloud, edge computing reduces the latency and congestion in the network. Nevertheless, data shared with the cloud or edge server may also pose a potential threat to the security and privacy of the user data. For instance, in the US, Florida Healthy Kids Corporation reported a healthcare data breach in 2021 impacting 3.5 million records [10]. Therefore, keeping the data locally and running ML algorithms on IoT devices is becoming more and more imperative for many applications.

With this dire need, various industry leaders started to tackle the problem of running ML models on IoT devices. Examples include but

* Corresponding author at: Department of Software Engineering, Erciyes University, 38280 Kayseri, Turkey.
E-mail addresses: ntekin@fiu.edu, nazlitekin@erciyes.edu.tr (N. Tekin), aaris@fiu.edu (A. Aris), aacar@fiu.edu (A. Acar), selcuk@cs.fiu.edu (S. Uluagac), cagri.gungor@agu.edu.tr (V.C. Gungor).

https://doi.org/10.1016/j.adhoc.2023.103348
Received 6 February 2023; Received in revised form 11 October 2023; Accepted 5 November 2023
Available online 10 November 2023

not limited to ARM's [11] artificial intelligence platform with a new Cortex-M processor that enables on-device ML processing on power-constraint IoT devices, Google's ML Kit [12] that supports on-device ML models, Apple's CoreML [13] framework that runs ML fully on-device, Android's [14] tools and methods for integrating ML into the apps, and Amazon's [15] tools for performing ML inference models locally on devices. Moreover, Apple [16] has introduced an on-device Deep Neural Network (on-device DNN) for speech recognition applications.

In addition to the efforts from the industry, a proliferation of studies in the literature also tried to pave the way for on-device ML for IoT applications. Tiny Machine Learning (TinyML) approach is introduced to integrate ML algorithms into tiny IoT devices and embedded systems. Moreover, Federated Learning (FL) approach that enables distributed training on IoT devices is proposed in the literature. However, despite the advantages in security, privacy, and latency as well as efforts of industry and academia, the energy consumption of ML remains a critical constraint for battery-powered IoT devices. In fact, computationally intensive ML tasks can drain the battery of IoT devices. Numerous studies were conducted in the literature that runs ML algorithms on IoT devices. These studies generally focus on the feasibility of processing ML models on-device for various IoT applications. Some studies provide a framework to enable the computation of ML models on IoT devices and evaluate them in terms of execution times [17,18]. In addition, some of these studies focus on designing hardware architecture to improve energy efficiency.

Despite a plethora of works, no prior study exists that systematizes the knowledge in energy-aware on-device ML models for IoT applications. In this work, we provide a systematic review of the on-device ML models for IoT applications in terms of energy consumption. Particularly, this work answers the following questions to provide a guide for researchers as well as practitioners who want to integrate ML into IoT devices for their applications:

- RQ1: What are the different approaches for deploying on-device ML models for IoT applications?
- RQ2: How do these approaches take the energy-awareness perspective into consideration?
- RQ3: Which tools/libraries are used for implementing energy-aware on-device ML for IoT applications? Are they publicly available?
- RQ4: What type of IoT devices are used as deployment targets for energy-aware on-device ML models?

Scope: In this study, we aim to review energy-aware on-device ML algorithms that are particularly proposed for IoT applications. We focus on works that perform either the training or inference part of the ML model on IoT devices and examine the energy-related metrics.

Survey Methodology: To find the relevant literature, we performed our search in the well-known digital libraries of Google Scholar [19], IEEE Xplore [20], ACM Digital Library [21], Elsevier Science Direct [22], and arXiv digital libraries/repositories [23]. Particularly, we searched for the following keywords in these libraries:

- ML `AND` (energy consumption `OR` energy efficiency) `AND` IoT,
- ML `AND` execution time `AND` IoT,
- On-device learning `AND` (energy consumption `OR` execution time) `AND` IoT.

We selected relevant works published in the last seven years (2016–2023). In total, we survey 37 papers for our work.

Contributions: The contributions of our work are summarized as follows:

- We provide a comprehensive review of the works that focus on deploying ML models on IoT devices. We identify targeted IoT applications and analyze available techniques and tools that are used for running ML models on IoT devices.
- We introduce a novel taxonomy for describing the energy-aware ML models performing on IoT devices.
- We also discuss the lessons learned and open research problems that can be further investigated.

Organization: The rest of the paper is organized as follows: Section 2 presents the previous surveys about ML in IoT applications and how our work differs from them. Section 3 provides the background information. Section 4 presents our novel taxonomy and reviews the energy-aware ML approaches performed on IoT devices. Section 5 discusses the lessons learned and open issues. Finally, Section 6 concludes the paper.

2. Related work

In the last decade, ML has become an active field of research for IoT systems in which a massive amount of data is produced. In this section, we give a brief overview of the related surveys on ML in IoT, edge computing, tinyML where ML models are optimized for resource-constrained devices, and federated learning. We also highlight the differences of our work from the relevant literature.

ML for IoT Applications: Several surveys in the literature provided an overview of studies utilizing ML models for IoT applications [24–26,30]. Particularly, Cui et al. [26] presented an overview of studies focusing on the application of ML models for IoT while Mahdavinejad et al. [24] gave an overview of the studies focusing on ML techniques that are used for IoT data analysis. Moreover, Qian [30] presented a survey of ML-based IoT applications and Messaoud et al. [25] introduced a novel taxonomy of ML models that are used to address the issues in WSN and IoT In addition, some recent studies provided surveys on ML-based solutions employed in different IoT application domains such as healthcare [27], industry [28], and security [29].

Edge Computing Surveys: Edge computing, in which the computation is performed near the edge of the network on customized computing devices, is a promising solution to address security, privacy, and latency concerns. Murshed et al. [31] conducted a survey on ML systems at edge devices that focuses on compression techniques. Moreover, Lim et al. [32] reviewed FLat edge devices in terms of techniques and frameworks. On the other hand, Chen et al. [42] and Wang et al. [33] provided surveys on Deep Learning (DL) algorithms on edge computing architecture.

TinyML Surveys: In addition to edge computing, another line of research to preserve privacy and latency without sacrificing security is to employ the ML model on resource-constraint devices by converting the model into a code that can run on these devices. To address energy and memory challenges of training and inference on-device, TinyML approach is proposed, which is an emerging concept for on-device learning. There are a few surveys focusing on the overview of the research in this direction in the literature. Dhar et al. [43] provided a review of algorithms and theoretical approaches that enable ML on resource-constrained devices. Dutta et al. [44] gave the definition, strengths and weaknesses of TinyML, and related techniques. Ray [35] presented use cases such as image classification, face detection and anomaly detection, in which TinyML is utilized. Doyu et al. [36] introduced the TinyML-as-a-Service concept and its design requirements.

Federated Learning (FL) Surveys: Another increasingly used approach for on-device learning is FL. In FL, the resource-constrained devices perform the training locally in a collaborative way without sharing the data with each other. The main objective of FL is privacy as the raw data does not leave the local device. Numerous surveys reviewed FL studies in the literature [37–41]. Zang et al. [37] and Nguyen et al. [38] analyzed the improvement of FL and summarized practical FL applications. Imteaj et al. [41] discussed the challenges of implementing FL in an IoT environment. Khan et al. [39] provided recent advances, open

Table 1
The overview of related surveys in the literature (see [24–41]).

Surveys	Scope						Summary
	Edge Computing	TinyML	Federated Learning	Energy Awareness	On-Device ML	IoT	
[24–30]	✗	✗	✗	✗	✗	✓	Presenting surveys of ML models used for IoT applications.
[31,32]	✓	✗	✓	✗	✓	✓	Focusing on ML models that can be deployed on edge devices.
[33,34]	✓	✗	✗	✗	✓	✓	Describing the challenges and optimization algorithms of ML models deployed on edge devices.
[35,36]	✗	✓	✗	✗	✓	✓	Providing surveys of TinyML models.
[37]	✗	✗	✓	✗	✓	✗	Providing characteristics and practical applications of federated learning models.
[38–40]	✗	✗	✓	✗	✓	✓	Presenting surveys of federated learning models for IoT applications.
[41]	✗	✗	✓	✗	✓	✓	Discussing on-device federated learning models for IoT applications.
This work	✓	✓	✓	✓	✓	✓	Presenting a survey of energy-aware on-device ML models for IoT applications.

research challenges, and taxonomy for FL in IoT. Kholod et al. [40] reviewed the open-source FL frameworks.

Differences from other works: We present the list of other related surveys focusing on the deployment of ML models for IoT applications in Table 1. The main difference of our work from prior studies is that we investigate the ML techniques on IoT in terms of energy perspective. To the best of our knowledge, this is the first study that reviews energy-aware on-device ML models for IoT applications. In total, we surveyed 28 works published in 2016–2022. In addition, we provide a novel taxonomy of energy-aware on-device ML approaches in IoT and provide the publicly available tools used for running ML models on IoT devices. The reasoning behind our categorization for deploying the ML models on resource-constrained IoT devices is that it would be helpful for practitioners and researchers to choose among the given approaches for the given task (i.e., ML deployment on IoT devices).

3. Background

In this section, we provide brief background information about ML and deep learning algorithms that can be adapted to IoT applications for tasks such as identification, classification, or detection. Additionally, we give very brief information on IoT devices, platforms, and energy consumption in IoT devices.

3.1. Machine learning algorithms

A typical ML process consists of two phases: training and inference. In *the training phase*, collected data instances are used to teach ML algorithms to perform convenient predictions. In *the inference phase*, a prediction is made by running data points onto a trained ML model. ML algorithms can be divided mainly into three categories: (1) Supervised learning, (2) Unsupervised learning, and (3) Semi-supervised learning. Fig. 1 illustrates an example of supervised learning, unsupervised learning, and semi-supervised learning.

3.1.1. Supervised learning

The supervised learning algorithms aim to find a mapping function to infer output from given input by using labeled data instances. In supervised learning approaches, the labeled data is used for classification or regression tasks. Popular supervised learning algorithms are k-Nearest Neighbor (k-NN), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and Logistic Regression (LR). A k-NN classifier is based on calculating the distance (e.g., Euclidean) to measure similarities between one point and its neighboring points. k-NN can be applied to IoT applications such as image classification [45] and human activity recognition [46]. On the other hand, the SVM classifier is based on identifying a hyper-plane that distinguishes types of classes. IoT applications using SVM include, but are not limited to disease detection such as cotton leaf disease [47] and crop disease [48], and speech and image classification [49]. In DT, the classification is employed by setting if-then conditions. Smart health care systems can provide disease detection by leveraging DT classifiers [50]. Moreover, RF is an ML algorithm that builds a large group of decision trees. In IoT, RF is used for several applications such as health care monitoring systems [51] and malware detection systems [52]. Finally, LR is a statistical method that uses the logistic function to predict binary variables. Examples of utilizing LR for IoT applications include security-focused applications such as attack and anomaly detection [53,54].

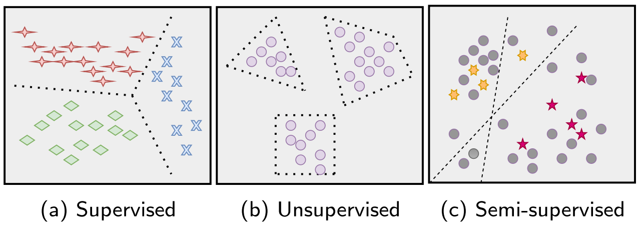

Fig. 1. A representation of (a) supervised learning; the red stars, green diamonds, and blue crosses indicate different classes in the labeled data and (b) unsupervised learning; the geometric shapes of purple circles indicate the different classes in the unlabeled data. In both cases, the dotted lines indicate the prediction (classification) result. (c) semi-supervised learning; gray circles indicate unlabeled data, orange six-point stars, and pink stars indicate differently labeled data. The dotted lines are decision boundaries.

3.1.2. Unsupervised learning

Unlike supervised learning, in unsupervised learning approaches, the unlabeled data are clustered based on their similarities and correlations. k-means clustering and Principal Component Analysis (PCA) are widely used unsupervised learning methods [55]. In k-means clustering, distance metrics are utilized to divide data points into predefined k clusters. The k-means clustering algorithm can be used in IoT applications such as smart home automation [56] and precision agriculture [57]. In addition, PCA is used for extracting important features to compress the dataset, hence reducing the feature dimension and computation complexity. PCA algorithm is used for IoT network traffic anomaly detection [58].

3.1.3. Semi-supervised learning

Semi-supervised learning approaches train the model by using both the labeled data and the unlabeled data. The aim of the model is to classify unlabeled data by utilizing a small amount of labeled data. The different types of semi-supervised learning algorithms are self-training, graph-based semi-supervised learning, and low-density separation. In the self-training method, the model is firstly trained with labeled data and then unlabeled data is classified with this model. After, the model is retrained with the new growing labeled data. In graph-based semi-supervised learning, unlabeled data are labeled by adopting a graph-based strategy. Later, the model is trained with the labeled data. Finally, in low-density separation, the unlabeled data are clustered by using a low-density region. Semi-supervised learning can be used in intrusion detection systems for IoT applications where there is a large amount of unlabeled data.

3.2. Deep learning algorithms

Deep Learning (DL) builds algorithms to create neural networks that can learn and make decisions. One common DL algorithm is Artificial Neural Network (ANN). ANN consists of an input layer, hidden layers, and an output layer as depicted in Fig. 2. These multiple layers are used to process data, compute complex tasks, and construct learning models. Moreover, ANN can be employed in several recognition and detection applications in IoT [59]. Another commonly used DL algorithm is the Convolutional Neural Network (CNN). CNN comprises a large number of convolution layers to extract and process features. In this case, the layers filter input to create a feature map that indicates significant feature information. The number of parameters can be significantly reduced in CNN compared to the neural network. Furthermore, CNN is widely used in IoT image classification and detection applications [60] and smart health systems [3]. Finally, Recurrent Neural Network (RNN) is another DL algorithm, which is commonly used for IoT applications. RNN enables to processing of long sequences of data and remembers previous values. RNN is mostly employed in several applications such as handwriting [61] or speech recognition [62]. Additionally, Long

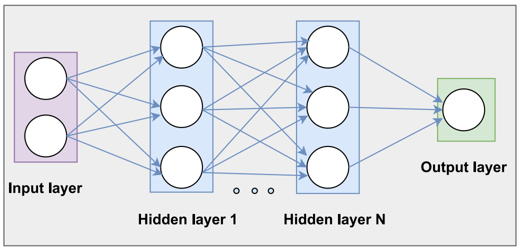

Fig. 2. The structure of an artificial neural network.

Short Term Memory Network (LSTM) is a gradient-based algorithm that can learn by remembering more previous values than RNN. Hence, LSTM provides long-term dependency learning.

3.3. IoT platforms

IoT devices are sensors, actuators, embedded and wearable devices that are able to collect and process data, and exchange data over the Internet or other networks [63–65]. There are various commercial off-the-shelf (COTS) IoT devices used for different IoT applications. For instance, in smart home applications, IoT devices such as smart thermostats, smart lights, and smart cameras are used. These devices are controlled via an app on a mobile device by the user and through the cloud. They can be controlled remotely and user interactions are logged. Moreover, there exist various commercial platforms to manage different IoT devices and allow interoperability between the devices. Some of the popular IoT platforms are Samsung SmartThings [66], Apple HomeKit [67], and AWS IoT [68]. In addition to these COTS IoT devices and platforms, there are several open platforms such as OpenHab [69] and Home Assistant [70]. Finally, the development boards are utilized by practitioners and researchers to develop IoT applications due to their similarity in terms of device resources. Some of the popular development boards are Raspberry Pi, Arduino, Intel Edison, BeagleBone Green Gateway, Tessel 2, and Teensy. Depending on the need, they can be equipped with varying hardware and software resources. For example, Aurdino has several models such as Uno, Mega 2560, or Nano 33 BLE with varying specifications.

3.4. Energy consumption in IoT

As mentioned before, IoT devices are generally designed to collect sensor data through the sensors placed far from each other, then gather them together through a hub, which also transmits the collected data into the cloud for further processing [71]. As IoT devices do not perform heavy energy-consuming tasks, they are mostly battery-powered. On the other hand, the hubs that connect multiple IoT devices or heavy-energy-consuming devices like CCTV cameras generally come with a wired connection. However, the overall market tendency is to have battery-powered IoT devices due to convenience and easy positioning. Therefore, energy consumption is very crucial, especially for battery-powered IoT devices as it can easily drain their batteries.

The tasks that consume energy for IoT devices include acquisition, processing, communication, and idle states. Thus, the total energy consumption of an IoT device can be defined using the following formula [72]:

$$\mathcal{E}_{total} = \mathcal{E}_{acq} + \mathcal{E}_{comm} + \mathcal{E}_{proc} + \mathcal{E}_{idle}. \tag{1}$$

The $\mathcal{E}_{acq}$, $\mathcal{E}_{comm}$, and $\mathcal{E}_{proc}$ are energy required for data acquisition from sensors, communications (i.e., transmitting and receiving), processing, respectively. The $\mathcal{E}_{idle}$ represents the energy consumed when the IoT device is in an idle state. The energy consumption of processing a task is given by [73]

$$\mathcal{E}_{proc} = \mathcal{P}_{proc} \times t, \tag{2}$$

where t indicates time spent and $\mathcal{P}_{proc}$ indicates the processing power and calculated as:

$$\mathcal{P}_{proc} = \mathcal{I} \times \mathcal{V}, \tag{3}$$

where $\mathcal{V}$ and $\mathcal{I}$ indicate the operating voltage and the current used during the operation, respectively. Therefore, time consumed for processing (i.e., execution time (s)) and communications (i.e., transmitting, receiving) are directly proportional to energy consumption and are considered as energy-related metrics in the literature. Likewise, power (W) and current (A) consumption are considered as energy-related metrics.

3.4.1. Energy consumption in ML algorithms

Running machine learning algorithms on IoT devices can substantially increase energy consumption. Many factors such as the machine learning algorithm's complexity, input data size, processing power of the device, and memory requirements have a significant impact on the processing energy consumption equation for a machine learning model.

ML Model Complexity: The complexity of the ML model plays a crucial role in energy consumption. More complex models typically involve a higher number of parameters, layers, or interactions, resulting in increased computational requirements. The device that requires more computations to train or make predictions consumes more processing energy. The training and inference computational complexity of commonly used ML algorithms implemented with scikit-learn is given in Table 2 [74]. Note that complexities can vary depending on specific implementations and optimizations.

k-NN: There is no explicit training process of k-NN, the algorithm only requires storing the entire training dataset. During inference, kNN involves comparing a new data point to all stored training data points to find the nearest neighbors. Therefore the computational complexity of inference is $O(mn)$ where m and n are the numbers of training samples and input features, respectively.

SVM: The training and inference computational complexity of SVM is $O(mn^2+n)$ and $O(mn)$, respectively. SVM employs kernel functions and matrix operations during training and inference, resulting in increased computational complexity, particularly with larger datasets. As the number of input features increases, the number of kernel computations increases, leading to significantly higher energy consumption.

LR: The training and inference computational complexity of LR is $O(mn^2 + m^3)$ and $O(n)$, respectively. The training phase involves operations such as matrix multiplications, inversions, and solving linear equations, increasing the complexity. Training on large datasets with a high number of features can pose challenges in terms of energy efficiency due to the potentially high time complexity involved in the process.

DT: DT uses comparison and recursive operators and information gain computation to form the basis of the decision-making process, allowing them to split the data based on feature values and make predictions. The training and inference computational complexity of DT is $O(mnlog(m))$ and $O(log(m))$, respectively. Therefore, it is more energy efficient in both training and inference compared to SVM and LR.

RF: RF combines the predictions from multiple decision trees through voting or averaging operations, depending on the task. The training and inference computational complexity of DT is $O(N_t mnlog(m))$ and $O(N_t log(m))$, respectively where N_t refers to the number of trees.

k-means: The computational complexity of the k-means algorithm is $O(mnc)$ which depends on not only the number of training samples and input features but also the number of clusters (*i.e., c*). The k-means algorithm primarily utilizes Euclidean distance and centroid calculations.

PCA: The PCA algorithm performs complex matrix operations such as eigen-decomposition with the computational complexity of $O(mn^2 + n^3c)$. The higher computational complexity requires higher energy consumption of IoT devices.

ANN: The computational complexity of ANN, $O(mnh^k i)$ is typically proportional to the number of hidden layers (*i.e., k*), the number of coefficients (*i.e., h*) in each layer and the number of iterations (*i.e., i*). The computational complexity thus energy consumption can increase significantly for larger and more complex networks with many layers and neurons. *Dataset and Input Features Size:* Larger dataset and input features size increase the computational complexity of all ML algorithms. Performing computations on a larger dataset and input features requires more processing power and execution time, resulting in increased processing energy consumption. A larger dataset and input feature size significantly increase the processing energy consumption for SVM and LR. In addition, as the dataset size increases, the execution time of DT and RF grows significantly. Consequently, the increased execution time leads to extended processing periods, impacting the processing energy consumption.

Processing Power of IoT Device: The processing power of an IoT device plays a crucial role in the energy consumption of machine learning algorithms. Higher processing power enables faster execution, leading to reduced processing time and potentially lower energy consumption. However, it is important to note that as the complexity of the machine learning model increases, there is a corresponding increase in processing energy consumption due to longer processing time. Table 3 demonstrates the most commonly used IoT device specifications including the type of processor, power, clock speed, and memory capacity.

Memory Requirements: ML algorithms with higher memory requirements consume more energy due to memory accesses and data transfers. When an algorithm operates on large amounts of data, it often necessitates frequent reading and writing operations in memory, leading to

Table 2
Computational and Space Complexity of ML algorithms.

ML Algorithm	Training	Inference	Model Size
k-NN	-	O(mn)	O(mn)
SVM	O($mn^2 + n$)	O(mn)	O(n)
LR	O($mn^2 + m^3$)	O(n)	O(n)
DT	O($mnlog(m)$)	O($log(m)$)	O(m)
RF	O($N_t mnlog(m)$)	O($N_t log(m)$)	O(N_tm)
k-means	O(mnc)	-	-
PCA	O($mn^2 + n^3c$)	-	-
ANN	O($mnh^k i$)	O(n)	-

m:number of training samples, **n:** number of input features, N_t: number of tree, c: number of clusters, **h:** number of coefficients, **k:** number of hidden layers,**i:** number of iterations.

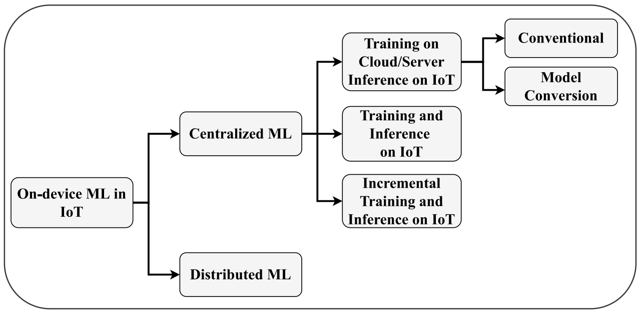
On-device ML in IoT
Centralized ML
Distributed ML
Training on Cloud/Server Inference on IoT
Training and Inference on IoT
Incremental Training and Inference on IoT
Conventional
Model Conversion

Fig. 3. Taxonomy of On-Device ML Models for IoT.

Table 3
IoT device specifications.

IoT Device	Processor	Power (V)	Clock Speed	RAM
Raspberry Pi 3B+ [75]	ARMv8-A64/32 Bit	5	1.4 GHz	1 GB
Raspberry Pi 4B [75]	ARMv8-A64/32 Bit	5	1.5 GHz	1,2,4,8 GB
Arduino Nano 33 BLE [76]	Cortex-M4F 32-bit	3.3	64 MHz	256 KB
Arduino Mega 2560 [76]	ATmega2560	16 MHz	16 MHz	8 KB
Arduino Uno [76]	ATmega328P	5	16 MHz	2 KB
Arduino Portenta H7 [76]	Cortex-M7	5	480 MHz	8 MB
ESP8266 [77]	Tensilica L106 32-bit	3.3	160 MHz	50 KB
ESP32 [77]	Xtensa LX6 32-bit	3.3	240 MHz	128 KB
nRF52840	Cortex-M4 32-bit	3	64 MHz	256 KB
Intel Galileo Gen 2 [78]	Intel Quark SoC*1000	7-15	400 MHz	256 MB
Intel EDISON Board [79]	Intel Quark SoC*Dual core	7-15	500 MHz	256 MB

higher energy consumption. Memory requirements of ML algorithms are shown in Table 2.

4. On-device ML in IoT

In this section, we present a novel taxonomy of on-device ML models that are utilized for IoT applications and discuss their energy consumption in detail. Fig. 3 demonstrates the taxonomy of these models. We first categorize the models according to the number of devices computing the ML model. In this regard, we classify on-device ML models in IoT mainly under two categories, which are as follows:

1. *Centralized ML:* In this category, the training phase is performed on a single device. This single point can be a computationally powerful resource (i.e., server or cloud) or an IoT device itself. However, in this approach, the inference is performed on the IoT device. Based on the training model computation architecture, the Centralized ML model can be implemented in three ways:
   (a) *Training on Cloud/Server - Inference on IoT:* The training phase is processed in the cloud or a centralized server (i.e., pre-training) while the inference phase is processed on IoT devices. This category is also divided into two sub-categories as follows:
      i. *Conventional:* The pre-trained model is directly deployed on IoT devices without any model conversion.
      ii. *Model Conversion:* The pre-trained model is converted into a C/C++ programming language code to be deployed on tiny IoT devices (i.e., microcontroller units (MCUs)).
   (b) *Training and Inference on IoT:* Both training and inference phases are performed on IoT devices.
   (c) *Incremental Training and Inference on IoT:* The training is performed in the cloud or centralized server, but the model is continuously updated by training with the local data on IoT devices.
2. *Distributed ML:* In this category, the training is performed locally on each IoT device in the network and each IoT device sends the model weights (i.e., parameters) to the cloud. After that, the global model is periodically aggregated in the cloud. Finally, the cloud distributes the global model to each IoT device. With this, the training phase is performed collaboratively on IoT devices without requiring the local data to be sent to the cloud.

In the rest of this section, we review the relevant studies in the literature in terms of their energy consumption.

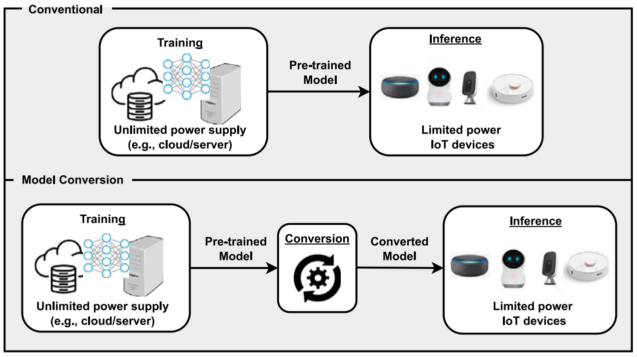
Conventional
Training
Unlimited power supply (e.g., cloud/server)
Pre-trained Model
Inference
Limited power IoT devices
Model Conversion
Training
Unlimited power supply (e.g., cloud/server)
Pre-trained Model
Conversion
Converted Model
Inference
Limited power IoT devices

Fig. 4. An illustration of Training on Cloud/Server and Inference on IoT Devices model with Conventional and Model Conversion approaches.

4.1. Centralized ML models

In the Centralized ML model, training is performed on a single point (i.e., server, cloud, or IoT device) whereas the inference is on the IoT device. The Centralized ML model can be implemented in three ways according to the location of the training phase: *Training on Cloud/Server - Inference on IoT*, *Training and Inference on IoT*, and *Incremental Training and Inference on IoT*. In this subsection, we explain each of these categories and briefly explain the studies that fall under these categories.

4.1.1. Training on cloud/server and inference on IoT device

ML algorithms require a huge amount of memory and computation power that makes them challenging to run on resource-constrained IoT devices. As inference requires less computation compared to training, some researchers suggested the approach, where the training phase is outsourced to the cloud or server [80]. The Training on Cloud/Server and Inference on IoT Device model is divided into two categories, namely, *Conventional* and *Model Conversion*. An illustration of Training on Cloud/Server and Inference on IoT Device model with Conventional and Model Conversion approaches is depicted in Fig. 4.

• *Conventional:* In the *Conventional* method, the pre-trained ML model is directly deployed on IoT devices without a model conversion. Various studies employed the Conventional method of training on cloud/server and inference on IoT devices and tried to minimize energy consumption. For instance, researchers directly deployed ML models to IoT devices and evaluated the energy consumption [81], employed optimization methods (e.g., compression, quantization, algorithmic complexity reduction) to reduce the computational overhead [82,83], and utilized various feature extraction methods [84,85].

Curtin and Matthews [81] constructed a CNN model with two layers on Raspberry Pi utilizing Tensor Flow [86] and Keras [87] to classify images for a wildlife monitoring application. They expressed that the lifetime of Raspberry Pi with four Alkaline AA batteries while processing image classification systems is 10 h. Zualkernan et al. [88] provided an automated animal classification application using DL on edge devices and they concluded that on average, each on-device inference takes only 0.2 ms time per image on a Raspberry Pi.

In addition to the studies using thresholds to reduce the model layers, some researchers proposed the use of optimization techniques to reduce the computational overhead. The authors in [82] employed a compression method based on linear quantization and confirmed that both the computational overhead and the model size of the RNNs model are reduced for the BreathPrint authentication system. Thus, compared to CNNs, using RNNs in this way provided them with more lightweight processing on IoT devices such as smartphones, smartwatches, and Raspberry Pi. Furthermore, in [83], researchers developed SparseSep, a new optimization technique based on sparse coding and separation of deep learning model layers, to enable them to work on wearable devices. This new technique exploits three software components: Layer Compression Compiler (LCC) is for compressing the DNN or CNN model while preserving high accuracy. Sparse Inference Runtime (SIR) leverages the sparse weight matrices to reduce computational overhead. Finally, Convolution Separation Runtime (CSR) is for compressing the convolution layers to further diminish the model for IoT wearable devices. The evaluations demonstrate that the running time and energy consumption of optimized deep learning models on wearable are reduced significantly. In another study [89], researchers used Stochastic Gradient Descent with Momentum (SGDM) optimization technique to compress the DNN model and deploy the pre-trained model in IoT device for fall detection in smart home applications.

In terms of the studies employing various feature selection schemes, Azariadi et al. [84] proposed a real-time Electrocardiogram (ECG) classification system that utilizes an SVM classifier on IoT device (i.e., Intel's Galileo board). They used Discrete Wavelet Transform (DWT) to extract features for ECG samples since it provides higher accuracy. Additionally, they performed design space exploration to select the best feature vector in terms of computation cost efficiency. Similarly, in [85], the authors presented a multi-stage feature selection approach to reduce the computational complexity for an activity recognition application deployed on the wearable.

• *Model Conversion:* In the Centralized ML with Model Conversion, the pre-trained ML model is converted into a C/C++ code to be deployed on tiny IoT devices. The model Conversion approach benefits from tinyML, which is an approach that aims to run optimized ML algorithms on resource-constrained devices with lower energy consumption. One of the key advantages of tinyML is that it provides data privacy. Additionally, processing on-device locally enables real-time decisions. Moreover, transmitting large amounts of raw data to the cloud is an energy-intensive process. Hence, tinyML does not require sending user data to the cloud increases energy efficiency, and thereby, a hot topic of research and practice where researchers from the industry and academia are actively working. While the industry side has generally focused on developing practical tools for tinyML, researchers utilized these tools or proposed new tools that use alternative number representations [90].

In terms of the industry investment in tinyML, several tinyML tools have become available such as TensorFlow Lite by Google, Embedded Learning Library (ELL) by Microsoft and ARM-NN, Common Microcontroller Software Interface Standard NN (CMSIS-NN) by ARM, STM32Cube-AI by ST Electronics to take advantage of ML for IoT applications. A few papers exist in the literature that utilize these tools. In [91], the authors evaluated the performance of STM32Cube-AI and TensorFlow Lite for ANN optimization. They also analyzed the performance of a quantization method in terms of energy and memory consumption. Similarly, in [92], the authors provided a comparison of TensorFlow and TensorFlow Lite using face recognition applications on IoT devices in terms of memory usage and inference time. Likewise, Giardano et al. [93] designed a battery-less face recognition application using TensorFlow Lite that runs on ARM Cortex-M4F microcontrollers and scavenges energy from thermal and solar harvesters. An energy-efficient ML inference method using XCube-AI is provided by Daghero et al. [80]. The idea behind this method is that there is no need for a complex ML model for easy inputs. To this end, they applied a score margin threshold (i.e., the difference between the largest and second-largest score at the last model layer) as a stopping criterion. The proposed method uses class-dependent thresholds instead of a single threshold for inference and achieves a better accuracy with a 60% reduction in energy consumption for on-device image classification. Furthermore, researchers proved that even DL models such as CNNs that require higher computation power can be deployed on IoT edge devices.

Other than the industry, tinyML frameworks were proposed by several research groups and open-source projects. To execute the supervised learning algorithms such as SVM, DT, LR, and MLP (Multilayer Perceptron) on resource-constrained IoT devices, the study in [90] introduced a software tool called EmbML [102] that converts the classifier into a C++ code. EmbML enables the fixed-point representation where data is stored in constant variables. Therefore, fixed-point representation saves memory and reduces the computation cost of operations. In [106], the researchers utilized another framework, namely emlearn [99], to apply MLP models for user recognition on wearable devices. They indicated that energy consumption of the recognition on a wearable is directly related to the complexity level of NN. Thus, the higher the complexity of a NN, the higher the energy consumption since the processing time increases. In addition to the tools developed by different research groups, open-source tinyML frameworks also exist. One such open-source tinyML framework is MicroMLGen which converts ML models into C code for microcontroller platforms. In [106], the authors compared the performance of ML models generated by various tinyML tools. Specifically, they compared SVM generated by MicroMLGen, MLP generated by emlearn, and DT generated by emlearn. As a result, the performance of DT generated by emlearn accomplished better accuracy with lower computation time, thereby lowering energy consumption. In addition, the researchers in [107] leveraged the microMLGen framework to deploy the SVM model with PCA for locomotion activities detection. In that case, the model with the GY-801 Motion Unit accelerator obtains two times better performance in terms of the consumed time for classification on-device compared to the SVM model without PCA. Another framework is MCUNet [103] which jointly designs the neural architecture search (TinyNAS) and the lightweight inference engine (TinyEngine) to enable running DL on MCUs. In another study, Sudharsan et al. [18] provided a method that converts DT models to C. The C-converted version of the DT model generated by the proposed method does not utilize SRAMs during execution and benefits from the larger capacity of flash memories. The authors showed that the proposed model provides four times faster inference time compared to the model converted by sklearn-porter [108], m2cgen [109], and emlearn [99] libraries. On the other hand, [110] provided the Edge Impulse online platform for developing tinyML models that can be deployed in various resource-constrained devices. In [111] proposed an energy-efficient model for detecting fruits in the smart farming environment by using the Edge Impulse framework.

Table 4 summarizes the TinyML frameworks and provides information about the provider, output language, compatible libraries, supported IoT platforms, source code, and whether a specific tool allows on-device training (i.e., incremental training on the device). Here, on-device training means updating the pre-trained model deployed on tiny IoT devices with new local data. As outlined in the table, a few of the tools support the on-device training. Detail explanations of the incremental training are given in Section 4.1.3.

Table 4
TinyML frameworks.

Framework	Provider	Output	Compatible Libraries	Supported IoT Platforms	Source Code	On-device Training
TensorFlow Lite [94]	Google	C++	TensorFlow	Arduino, ESP32, Sparkfun Edge,Texas Instruments Dev Boards	Open source	✓
ELL [95]	Microsoft	C/C++	CNTK, Darknet, ONNX	Raspberry Pi, Arduino	Open source	✗
ARM-NN [96]	ARM	C	TensorFlow, Caffe, ONNX	ARM Cortex-A, ARM Mali, ARM Ethos	Open source	✗
CMSIS-NN [97]	ARM	C	TensorFlow, Caffe, PyTorch	ARM Cortex M	Open source	✗
STM 32Cube.AI [98]	STMicroelectronics	C	Keras, TensorFlow, Caffe	STM32	Closed source	✗
emlearn [99]	Open source project	C	Keras, Python/Scikit-learn	Atmega, ESP8266, ESP32, ARM Cortex M (STM32)	Open source	✗
MicroMLGen [100]	Open source project	C	Python/Scikit-learn	Arduino, ESP32, ESP8266	Open source	✗
TinyOL [17]	Academic research project	C	TensorFlow	Arduino Nano 33 BLE	Closed source	✓
ML-MCU [101]	Open source project	C++	N/A	Arduino	Open source	✓
EmbML [102]	Open source project	C++	WEKA, Python/Scikit-learn	Arduino, Teensy	Open source	✗
MCUNet [103]	Open source project	C/C++	Pythorch, TensorFlow	STM32F746	Open source	✗
MicroTL [104]	Open source project	C	Pythorch	nRF-52840-DK	Open source	✓
Edge Impulse [105]	Open source project	C++	TensorFlow, Keras	Arduino Nano 33 BLE, nRF-52840-DK, Syntiant TinyML Board	Open source	✗

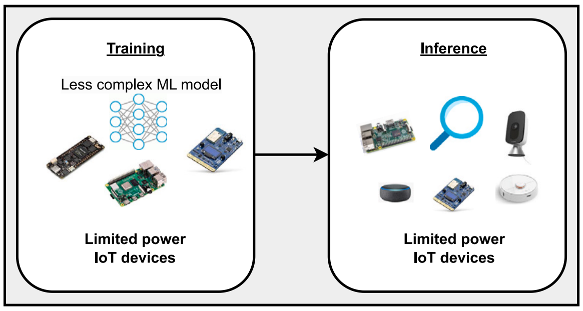

Fig. 5. An illustration of training and inference on IoT devices model.

4.1.2. *Training and inference on IoT device*

Although training the model on cloud/server solves the low computational power concern of IoT devices, the transmission of personal and sensitive user data for training may cause privacy and security issues. Hence, implementing both training and inference parts of ML methods on-device becomes critical. An illustration of the Training and Inference on IoT Devices model is shown in Fig. 5.

A few studies in the literature have provided solutions to address the problem of training and inference on IoT devices. In [112], local low-bit quantization and batch normalization freezing methods are introduced to enable CNN training on IoT devices. Moreover, they designed a bit-flexible multiply and accumulate (MAC) method to accelerate the computation of the learning system hence, maximizing the energy efficiency. In [46], authors introduced an optimization strategy to perform training and inference phases on resource-constrained devices for human activity recognition systems. They used the Chi2 filtering method for feature selection to reduce input data. They evaluated the supervised learning algorithms models such as LR, RF, k-NN, NB–Gaussian, Linear SVM, MLP, and DT in terms of accuracy and execution time. The results demonstrated that the execution time of ML methods is reduced by 80% while maintaining 90% accuracy. In [113], the researchers investigated the feasibility of running ML algorithms such as RF, SVM, and MLP on edge devices. They tested these algorithms in terms of power consumption, execution time, and accuracy. As a result, SVM performs faster inference with less power consumption, whereas RF performs with greater accuracy. In our previous work [74], we evaluated the energy consumption of running supervised ML algorithms on IoT devices. Experimental results showed that DT performs better in both training and inference.

Training and inference on IoT devices enable real-time applications while preserving data privacy and allowing offline intelligence. In [114], a feed-forward CNN is utilized to reduce computation complexity while training on-device for image classification applications. Traditionally, back-propagation is used to update the weights of each neuron in the NN. However, back-propagation is computation-intensive for IoT devices. For this reason, they built a cascade of multi-stage linear squared regressors without utilizing back-propagation to provide

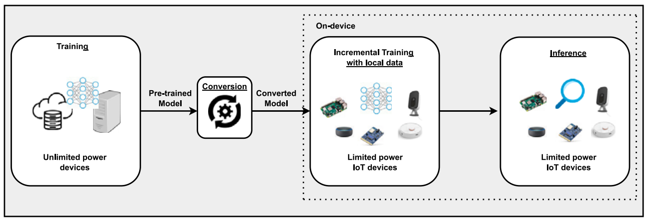

Fig. 6. An illustration of incremental training and inference on IoT devices.

a faster model. As a result, the faster model reduces the computation energy cost. Moreover, to decrease the computation cost of feature extraction before training, Saab transforms [114] in which the image is converted from spatial-only space to spectral-only space was utilized. Another study [115] introduced an anomaly detection model based on sparsity and random neural networks. The model benefits from Locality Sensitive Hashing (LSH) which requires less computation and memory. In addition, [116] presented the feasibility of NN training in resource-constraint IoT devices such as Arduino Portenta H7 and Arduino Nano 33 BLE.

4.1.3. *Incremental training and inference on IoT device*

Traditional ML utilizes historical data for training purposes. Therefore, examining new similar data achieves reliable accuracy. However, the accuracy may decline if the pre-trained model encounters different data. And, retraining a model on the cloud requires a large amount of data transmission causing significant energy consumption. Further, the previously collected data that needs to be stored for training leads to a huge amount of memory consumption. To address these concerns, incremental training, in which the pre-trained model is continuously updated with the new local data on IoT devices, was proposed [17, 117,118]. An illustration of incremental training and inference on IoT devices is depicted in Fig. 6. A few studies present frameworks that enable retraining the model on IoT devices with local data. In [17], Tiny ML with online learning (tinyOL) was proposed. The tinyOL framework provides an additional layer that can be executed on the fly for post-training. They evaluated the model by implementing it on the Arduino Nano 33 BLE board for anomaly classification problems. PCA was used for the data dimension reduction. The average time consumption of inference and online learning were reported as 1748 μs and 1921 μs, respectively. Further, researchers have improved the tinyOL framework by utilizing Complex Event Processing (CEP) to perform learning on industrial IoT devices [117]. In [118], the feasibility of model retraining on a Raspberry Pi was shown. They evaluated the model retraining for IoT device classification and category classification applications. To reduce the energy consumption of computation while retraining, the number of layers of NN was fixed. As a result, freezing three layers of NN was able to decrease the training time by 13.3%. In [45], the incremental on-device learning system based on DL and k-NN was proposed. Task dropping was utilized to approximate the pre-trained CNN model. The performance of the proposed solution was evaluated in two scenarios, speech command recognition and image classification applications on two platforms (i.e., Raspberry Pi 3B+ and STM32F76ZI). Furthermore, the ML-MCU framework was presented by [119] to enable incremental training and inference on IoT devices. The framework supports local training and on-the-fly methods without using the memory of resource-constrained devices. Additionally, Optimised-Stochastic Gradient Descent (Opt-SGD) and Optimized One-VersusOne (Opt-OVO) algorithms were provided to perform binary and multi-class classifier training on MCUs. In [120], researchers designed MicroTL, an open-source transfer learning framework, to enable re-training the DNN model on IoT devices. MicroTL utilizes the frozen pre-trained hidden layers and only re-trains the Fully-Connected (FC) layer(s). The framework consumed three times less energy by re-training on IoT devices rather than transmitting all local data to the cloud. In [121], researchers designed an algorithm to enable on-device learning with 256 KB memory. They proposed Quantization Aware Scaling (QAS) to allow fully quantized training for both forward and backward. In addition, they proposed a sparse update that prunes the gradient during backpropagation to reduce memory usage and computation costs. Table 5 summarizes the works on centralized ML models.

Some studies in the literature focused on hardware architecture design such as an accelerator to support training on resource-constrained devices [112,122,123]. In [112], the authors suggested low-bit training of deep learning on IoT devices. A new quantization method was proposed to reduce storage requirements and computational complexity. In addition, to support training on IoT devices, they designed an accelerator that obtains 12.5% higher energy efficiency. In [122], the authors proposed a deep learning weight update architecture, namely TrainWare (TW), which decreases memory usage. In that case, they achieved a 30.2% energy consumption reduction. Another architecture called DeepTrain based on homogeneous computing for training on embedded platforms was proposed by [123]. As a result, these hardware architecture designs provided energy-efficient training.

4.1.4. *Summary of centralized ML studies*

In this subsection, we provide discussions on the reviewed centralized ML works.

- *IoT Applications:* Among all reviewed studies, we found that energy-aware centralized ML-based solutions were used for applications such as image classification [80,81,88], health care systems such as human activity recognition [46,90,126], cancer and disease detection [113], facial recognition [92,93,106], and security systems such as anomaly detection [17,117] and intrusion detection [74]. For healthcare applications, ML model requires sensitive user data to perform training that results in potential privacy leaks. Similarly, security systems require privacy preservation and timely detection. Therefore, on-device ML design is needed to meet these demands.
- *ML Algorithms:* In centralized ML model, some of the studies performed supervised learning algorithms such as SVM [84,85,90,107, 124], DT [85,90,124] and NB [85] whereas others performed DL algorithms such as ANN [91], NN [112,117,118], CNNs [80,93,114], RNN [82], and MLP [46,90,106,113,124].
- *IoT Device:* Most of the studies considered Raspberry Pi [45,46,81, 88,113,114] as an IoT device, while some considered smartphones and smartwatches [82], Arduino [17,90,117,124], and ARM processors [93,126].
- *Tools/Frameworks:* Python/Sckitlearn [46,90,114] is a very common library to implement both ML and DL algorithms. Another commonly used library is TensorFlow [81,112,127] developed by Google. In addition, TensorFlow Lite DL framework was used by studies that perform inference on IoT devices [91,92,118] . There are other frameworks developed by particular developers such as MicroMLGen [124], EmbML [90,107], emlearn [106], tinyOL [117], ML-MCU [119], MCUNet [103], MicroTL [104] and Edge Impulse [105].
- *Metrics:* Most of the studies considered execution times (e.g., processing time spent for training or inference) as a metric to measure the energy consumption [45,46,82,83,93,113,118]. Likewise, [85] provided response time (ms) that includes the time needed for feature extraction and classification. Different from these studies, a few studies evaluated the energy consumption in Joule [74,80,83,91] and power consumption [74,116,120].

4.2. *Distributed ML models*

Distributed ML is an approach where multiple IoT devices collaboratively train the models. In particular, distributed learning comprises three phases, local training, model aggregation, and model distribution.

Table 5
Summary of studies utilizing Centralized ML.

		Work	IoT Application	ML Algorithm	IoT Device	Tool/Framework	Metrics (Unit)
Train on Cloud/Server and Inference on IoT Device	Conventional	[81]	Image classification	DL	Raspberry Pi 3	Python, Tensor Flow, Keras	Current (A), lifetime (h)
		[88]	Image classification	DL	Raspberry Pi 4B	Python	Pre-processing time (s), inference time (s)
		[82]	Breathing-based authentication	RNN	Nexus5, Pixel, Smartwatch,Rasspberry Pi	N/A	Inference time (s)
		[84]	Arrhythmia detection	SVM	Intel Galileo board	Matlab/LIBSVM	Execution time (s)
		[83]	Audio speaker verification	DNN, CNN	Qualcomm Snapdragon 400	N/A	Energy consumption (mJ), inference time (ms)
		[89]	Fall Detection Smart Home	DNN	Raspberry Pi 3	Python	Classification time (s)
		[85]	Human activity recognition	DT, NB, SVM	NRF51822	N/A	Response time (ms)
	Model Conversion	[91]	Li-Ion batteries parameter estimation	ANN, CNN, Tiny NN	STM32	Tensorflow Lite, X-CUBE-AI	Inference time (ms), inference energy consumption (nJ)
		[92]	Facial recognition	DL	Raspberry Pi	Tensoflow Lite	Detection time (s)
		[124]	N/A	SVM, MLP, DT, and RF	Arduino Uno 8 bit processor	MicroMLGen	Processing time (ms)
		[80]	Image classification and speech recognition	CNN	STM32H743 MCU	XCUBE-AI	Energy (mJ)
		[90]	Human activity recognition	DT, SVM,MLP, LR	Arduino Mega 2560, MK20DX256VLH7 (Teensy 3.2), MK66FX1M0VMD18 (Teensy 3.6)	WEKA, EmbML, Python/Scitkilearn	Classification time (μs)
		[106]	On-device user recognition	DL, MLP	ATmega328P	emlearn	Processing time (ms), current consumption (μA)
		[107]	Locomotion activities detection	SVM, PCA	ESP32	MicroMLGen	Classification time (ms)
		[117]	Anomaly detection	NN	Arduino Nano 33 BLE	tinyOL	Inference speed (ms)
		[125]	Keyword spotting, visual wake words, image classification	NN	Arduino Nano 33 BLE ,ESP 32	Edge Impulse	Inference time (ms)
		[111]	Smart agriculture	NN	Arduino Portenta H7	Edge Impulse	Current consumption (mA)
		[93]	Face detection	CNN	ARM Cortex-M	CMSIS-ARM	Inference time (s)
Training and Inference on IoT Device		[112]	Handwriting personalization	NN	N/A	Tensorflow	Energy eff.
		[114]	Breast ultrasound image classification	CNN	Raspberry Pi 3	Python/Scitklearn	Training energy eff.
		[46]	Human activity recognition	LR, LR, RF, k-NN, NB, SVM, MLP, DT	Raspberry Pi, Raspberry Zero	Python/Scitklearn	Inference time (s), training time (s)
		[113]	Breast cancer detection, glass identification	MLP, SVM, RF, LR	Raspberry Pi 3	Python	Training time (s), inference time (s), energy consumption
		[74]	IoT intrusion detection	k-NN, NB, DT, MLP, RF, LR	Raspberry Pi, Azure IoT Kit	Python	Training time (ms), inference time (ms), energy consumption (J), power consumption(W)
		[115]	Anomaly detection	NN	Raspberry Pi	Python	Inference time (ms)
		[116]	Keyword Spotting	NN	Arduino Portenta H7, Arduino Nano 33 BLE	N/A	Training time (ms), inference time (ms), current consumption (mA)
Incremental Training on IoT Device		[45]	Image classification, speech command identification	SVM, k-NN	Raspberry Pi 3B+, Embedded System (STM32F7)	N/A	Training time (s), inference time (s)
		[17]	Anomaly detection, anomaly classification	NN	Arduino Nano 33 BLE	tinyOL	Inference time (μs),online learning time (μs)
		[118]	IoT device classification category classification	NN (FC, LSTM, Conv1D)	Raspberry Pi	Tensorflow Lite	Training time (s) , inference time (s)
		[120]	Image Recognition	NN	nRF-52840-DK board 3	MicroTL	Training time (s), inference time (s), energy consumption (mJ), current consumption (mA)
		[121]	Image Recognition	NN	STM32F746 3	MCUNet	training time (ms)
		[119]	Breast cancer detection, Heart disease detection	N/A	nRF52840 (Adafruit Feather),STM32f103c8 (Blue Pill), Generic ESP32, ATSAMD21G18 (Adafruit METRO)	ML-MCU (Opt-SGD, Opt-OVO)	Inference time (ms), training time (ms)

ANN: Artificial Neural Network, CNN: Convolutional Neural Network, DL: Deep Learning, DNN: Deep Neural Network, DT: Decision Tree, **k-NN:** k-Nearest Neighbour, LR: Logistic Regression, LSTM: Long Short Term Memory, MLP: Multilayer Perception, NB: Naive Bayes, NN: Neural Network, PCA: Principal Component Analysis, RF: Random Forest, RNN: Recurrent Neural Network, SVM: Support Vector Machines.

Each IoT device performs local training with its own data and uploads the weights (i.e., parameters) of the local model to the cloud/server. Finally, the cloud/server aggregates the local model weights to create a global ML model and distributes the global ML model to each IoT device [37]. As the local data never leaves the IoT device, it addresses the latency, privacy, and security concerns with minimal computation and communication channel usage on the IoT device. An illustration of a distributed ML model is shown in Fig. 7.

FL has tremendous potential in applications where privacy is a critical concern and data is generated on devices. However, running FL on battery-powered IoT devices is still challenging in terms of energy consumption. [128] evaluated an FL implementation on IoT devices (e.g., Raspberry Pi) that leads to imbalanced data generation in terms of memory usage, training time, communication cost, and power consumption. In addition, [130] demonstrated the feasibility of executing FL on IoT devices when a simple CNN model is used. On the other hand, in [129], researchers provided an energy-efficient deep anomaly detection system based on FL that utilizes LSTM and SVM ML methods. To extract fine-grained features, they proposed attention mechanism-based CNN. Further, to reduce the number of gradients (i.e., weight

Table 6
Summary of studies utilizing Distributed ML.

Work	IoT Application	ML Algorithm	IoT Device	Tool/Framework	Metrics
[128]	Speech command recognition	CNN	Raspberry Pi 3	PyTorch	Training time (s), energy consumption (Wh)
[126]	Passive body detection	CNN, 2NN	ARM Cortex-A72 processor	Python, TensorFlow	Training time (ms)
[129]	Anomaly detection	LSTM	N/A	Python	Running time (min)
[127]	Anomaly detection	LSTM	Raspberry Pi 4B	Python, Tensorflow	Power consumption of training (mW)
[130]	Image Recognition	CNN, LSTM, MLP	Raspberry Pi 4B	PyThorch	Training time (s)
[131]	Malware detection	CNN, k-NN, RF, LR	Broadcom BCM2711	N/A	Training energy consumption (mJ)

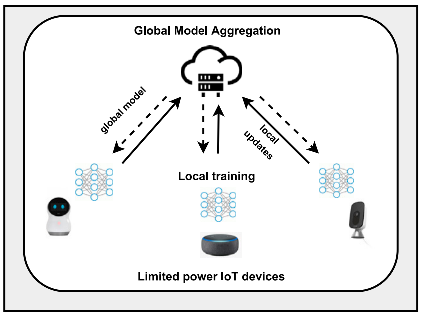

Fig. 7. An illustration of distributed ML model.

matrix) of the local models, a gradient compression mechanism was presented.

In [127], researchers proposed an FL-based anomaly detection system where training is locally performed using the data on IoT devices which reduces the communication energy cost. They demonstrated that FL can be a promising solution that provides better detection accuracy, faster detection, and energy efficiency. Similarly, to reduce the energy consumption of IoT devices and provide real-time detection, [131] proposed an FL-based malware detection system integrated with robust and active protection with an intelligent defense strategy at the IoT device. The proposed system performed 25% less energy consumption compared to classical FL.

Furthermore, an increasing number of devices in the network may cause aggregation failure on the server. In [126], authors presented a fully distributed FL approach that updates global models locally on devices. They tested the presented method on industrial IoT applications where the NN model was distributively trained and the model was aggregated on IoT devices. Table 6 summarizes the works on distributed ML models.

4.2.1. *Summary of distributed ML studies*

In this section, we provide discussions on the reviewed distributed ML works.

- *IoT Applications:* Distributed ML models were applied to speech command recognition [126], image recognition [130],passive body detection [128], anomaly detection [127,129] and malware [131] detection. The distributed ML model is effective particularly when the network bandwidth is limited or allocated to other critical tasks. Besides, it is efficient when the application requires a real-time response.
- *ML Algorithms:* In the distributed ML model, all of the studies employed DL algorithms such as CNN [126,128] and LSTM [127,129, 131].
- *IoT Devices:* Three studies [118,127,128] preferred Raspberry Pi while [126] preferred device with ARM Cortex-A72 processor and [131] preferred Broadcom BCM2711 board as the IoT device.
- *Tools/Frameworks:* Most of the studies [126,128,129,131] utilized Python-based tools to implement DL algorithms. In addition, [128, 129] used TensorFlow library in their study.
- *Metrics:* [126,128] provided training time as the energy performance metric whereas [129] provided running time. On the other hand, [127] provided power consumption during training while [131] provided energy consumption.

4.3. *Summary of energy efficient techniques*

In this section, we provide discussions on energy-efficient techniques such as quantization, pruning, and compression to enable deploying neural network (NN) on resource-constrained IoT devices.

4.3.1. *Quantization*

In NN, the weights, activations, and other intermediate values are usually represented using higher precision floating-point numbers (*e.g.,* 32-bit or 16-bit) which leads to increased memory storage requirements and computational complexity. Quantization is a technique used to reduce the memory and computational requirements of NN by decreasing the higher precision of numerical values to lower bit-width representations, such as 8-bit or even binary values. This reduction in precision enables more efficient storage, faster computations, and improved energy efficiency in the network.

There are several quantization techniques commonly used for NN. For instance, fixed-point quantization [132] is a quantization technique widely used in NN where all computations performed using integer arithmetic are significantly more cost-effective than those using floating-point arithmetic. The other one is ternary quantization [133] in which multiplications are performed using binary operations or limited to a maximum of two multiplications per activation when asymmetric scaling factors are utilized. This approach allows for a compression rate of up to 16 times with moderate accuracy loss. Another one is binary quantization [134] where all arithmetics are done via binary operations resulting in a compression rate of 32 times. However, it often results in high accuracy loss. Lastly, mixed precision quantization [135] is employed in NN where the precision sensitivity of each layer or even row of weights is individually tailored. This approach allows for optimization by utilizing the advantages of binary, ternary, and integer quantization techniques based on the specific requirements of each layer or row of weights.

4.3.2. Pruning

Pruning is a commonly used technique in NN to reduce the number of parameters and meet specific computation and storage requirements. It involves removing redundant or less important parameters or filters from the model. By eliminating these redundant components, the computation required during the inference process is reduced. Therefore, pruning parameters or filters from the convolutional layers enables minimizing computational cost. In addition, pruning can be performed simultaneously with the training process rather than applying it solely after training has been completed [121].

The two major techniques for pruning models include filter pruning and weight pruning [136]. Filter pruning is a technique that involves evaluating the importance of individual filters in an NN and subsequently eliminating those filters that are considered unnecessary or less significant. On the other hand, weight pruning is a technique used to reduce the number of parameters in a neural network by identifying and removing unnecessary or less important individual weights. By loading the pruned model into IoT, the frequency of accessing external memory is greatly reduced, leading to improved power consumption efficiency. Accordingly, the computational resources can be utilized more efficiently, resulting in faster and more energy-efficient training and inference process.

4.3.3. Knowledge distillation

Knowledge distillation is a technique used to transfer knowledge from a larger, more complex model (teacher) to a simplified model (student). The goal is to enable the student model to mimic the behavior and performance of the complex model while being more computationally efficient and having a smaller memory footprint. Knowledge distillation can be especially beneficial when the large model achieves high accuracy but requires significant computational resources. By distilling the knowledge into a student model, it enables the deployment of NN on low-cost IoT devices [137,138].

4.4. Summary of on-device ML design

In this section, we provide an overview of end-to-end on-device ML system design. There are several steps can be followed to achieve energy-efficient and optimized training and inference tasks.

Model Training: There are three distinct approaches to training the model on-device ML design. The first approach leverages high-performance hardware or cloud computing resources, enabling the achievement of high accuracy and performance. This approach is well-suited for tasks demanding substantial computational resources and extensive datasets. Alternatively, the second approach is the training of ML models on relatively powerful IoT devices, such as Raspberry Pi. This approach demonstrates how ML techniques can effectively adapt to devices with limited resources. The last approach is to deploy a pre-trained ML model and re-train (*i.e.,* incremental training) it on low-power IoT devices such as MCUs.

Determining when and how to employ each of these training methodologies according to application requirements, and applicable conditions is essential for building effective on-device ML solutions. The conditions include but not limited to computational resource requirements, dataset size, energy efficiency, real-time inference, security, and privacy concerns. ML models, especially DL models, often require substantial computational power, including GPUs or TPUs. If an IoT device lacks the necessary hardware resources to handle the complexity of the model it becomes necessary to offload the training to more powerful cloud or server infrastructure. In addition, if the dataset is too large to fit into the memory of an IoT device, cloud or server-based training becomes a practical choice for handling and processing big data. Moreover, training ML models can be computationally intensive and energy-consuming, which may not align with the power constraints of IoT devices. However, it is important to note that transmitting a high volume of data to the cloud can also be energy-consuming. Therefore the balance needs to be established when making a decision.

On the other hand, incremental training and inference on-device enhance the real-time applications that require low latency. Furthermore, for applications where data privacy and security are paramount, training ML models directly on IoT devices can keep sensitive data localized and reduce the risk of data breaches during transmission to the cloud. Besides, some applications require offline operations so that cloud access is unavailable. In summary, evaluating these factors in the context of IoT application requirements is crucial for selecting training approaches.

Model Compression: The compression techniques of quantization, pruning, and knowledge distillation can be applied to reduce the complexity and size of machine learning models, enabling their deployment and execution on IoT devices. Furthermore, it is possible to use multiple compression techniques simultaneously to further compress the model. For instance, one can initiate the compression process by applying pruning [132] to eliminate redundant parameters, connections, or filters from the model. This initial step effectively reduces the overall model size. Subsequently, quantization [135] can be utilized to further decrease the precision or bit-width of the remaining weights and activations, resulting in additional compression and memory savings. In addition to pruning and quantization, knowledge distillation [137] can be simultaneously applied to achieve even higher levels of compression. However, it is essential to acknowledge that employing multiple compression techniques concurrently may impact the accuracy of the compressed model. Thus, it becomes crucial to carefully manage the trade-off between compression and accuracy.

Type of IoT Device: The selection of appropriate IoT devices for specific applications holds considerable significance. IoT devices can indeed be broadly categorized into two categories: those with higher processing power and memory capacity (*e.g.,* Raspberry Pi) and those with lower processing power and memory capacity (*e.g.,* MCUs). For IoT devices with higher processing power and memory capacity such as Raspberry Pi, implementing the DL models like CNNs or RNNs is more efficient. It should be noted that not all IoT devices possess the capability to perform on-device training, especially for DL models, except for notable examples like Raspberry Pi. In addition, it is more efficient for applications involving image classification, or voice recognition, which often require higher computational capabilities and memory.

On the other hand, the other category of IoT devices such as MCUs are predominantly constrained by their significantly lower power and limited memory capacities, rendering them unsuitable for training tasks. However, MCUs such as ESP32, Arduino Uno, Arduino Nano 33 BLE, and STM 32, can still be leveraged effectively for on-device inference tasks even pre-trained with DL models. It is noteworthy that certain MCUs such as Arduino Nano 33 BLE, STM32F7463, and nRF-52840-DK board alongside Raspberry Pi, support incremental training capabilities, enabling the updating of models directly on the device itself. Moreover, they are better suited for applications involving anomaly detection, keyword spotting, and activity recognition.

Model Deployment: The compressed ML model can be directly deployed to IoT devices such as Raspberry Pi, while model conversion is necessary for IoT devices such as MCUs to ensure compatibility with the target on-device deployment environment. Several frameworks and tools have been developed to support specific IoT platforms, enabling the optimized performance and compatibility of ML model deployment on these platforms. A summary of these frameworks and tools can be found in Table 4.

5. Lessons learned and open issues

In the prior section, we reviewed and analyzed the energy-aware ML models in IoT applications. In this section, we discuss lessons learned from the reviewed works and provide open issues.

Device Support: A researcher or practitioner who wants to test his/her novel ML approach for IoT needs to choose the type of IoT device to conduct experiments on. The type of IoT device can be either COTS IoT devices or development boards. While COTS IoT devices are closer to the final product, they are either not programmable at all or allow only limited programming such as writing your own app to control the device. On the other hand, the effort required to test your approach on development boards is much less than the COTS IoT devices because of its wide range of supported programming languages and software. In literature, researchers considered mostly Raspberry Pi and MCUs to represent their IoT devices, as evident in Table 4, Table 5, and Table 6. Probably due to the limitations with programming and libraries/APIs for supporting ML, the researchers did not perform any energy-aware ML studies on COTS IoT devices. As such devices are widely used in smart environments (smart homes, offices, buildings, etc.), this is an important gap in both academia and industry. Therefore, the evaluation of on-device ML algorithms on COTS IoT devices is needed.

Energy Perspectives: Energy consumption of ML algorithms is a serious concern for IoT applications, especially for battery-powered IoT devices. Thus, in this work, we investigated the studies that work on on-device ML models in terms of energy consumption. We found that a few studies have examined energy consumption metrics in Jolue or power consumption in Watt. Instead, most of the studies consider computation costs since they can be attributed to energy consumption. From these studies, we note execution times (i.e., training time and inference time) as significant metrics as it is directly proportional to energy consumption. When comparing various on-device ML algorithms, it would not be fair to only consider the execution time because of the trade-off between accuracy and execution time. For example, since DL algorithms require high computational power, many optimization techniques (i.e., quantization, pruning, and compression) were proposed by researchers to enable DL in IoT. However, the optimized models usually lead to lower accuracy. In addition to academia, some companies from the industry such as Google, Microsoft, and ARM also developed frameworks to run ML on tiny IoT devices as summarized in Table 4. Therefore, developing energy-efficient algorithms for on-device ML for IoT applications is still a hot research area, in which the researchers investigate to find the optimum trade-off between accuracy and energy efficiency for real-life applications.

Application Domains: ML has been adapted to several application domains in IoT and energy consumption of on-device ML models for IoT has been analyzed for various applications such as image classification, speech and facial recognition, body and activity recognition, health care applications (e.g., cancer and disease detection, breast ultrasound image), device fingerprinting and security applications (e.g., anomaly and malware detection, authentication), and battery parameter estimation. Although the list of applications seems diverse, there exist many more application domains in IoT where ML is actively utilized and energy consumption is vital. Examples of such application domains that require energy-aware ML models include but not limited to agriculture, smoke and gas detection, industrial IoT applications in smart factories, and smart city applications.

Privacy: Privacy is another important factor when designing on-device ML algorithms for IoT applications. Among centralized training models, Train-on-Cloud and Incremental Learning approaches may not provide privacy as the data is shared with a third party (i.e., centralized cloud). On the other hand, the Train-on-IoT learning model provides privacy since the data never leaves the local IoT device (i.e., both training and inference is performed on the IoT device.) On-device ML with distributed training is considered to be an alternative approach to preserve privacy. However, membership attacks (i.e., whether the data is used during the training), partial reconstruction attacks (i.e., partial recovery of the user data) as well as perfect reconstruction attacks challenge this assumption. Therefore, in this regard, the research of designing a privacy-preserving on-device ML with a distributed training approach remains an open research problem.

Security: The ML models employed by various devices may be exposed to adversarial attacks that causes the model failure. Addressing adversarial attacks to on-device ML models for IoT applications is very critical since these attacks may cause serious consequences that may affect human health. Therefore, providing robustness to on-device ML models against adversarial attacks is of great importance as future research direction.

Efficient and Scalable ML in IoT: IoT deployments can have a large number of devices that produce a high volume of data with various diversities. Preprocessing, training, and inference of such big data for energy-efficient ML in IoT requires innovative approaches to process the data. Novel parallel computing approaches deployed at the IoT edge can be a remedy for this problem with scalable techniques to collect, store, process the big data coming from numerous IoT devices and perform scalable ML. In addition, as the memory of IoT and edge devices are significantly lower compared to cloud resources, the memory efficiency of ML algorithms plays a crucial role in efficient ML in IoT. As reviewed in this paper, researchers proposed several approaches to fit and run ML models in IoT devices such as model conversions. However, more efforts are needed in this regard to run ML in resource-constrained IoT devices.

Real-time Performance: The efforts of the industry and the academic community in bringing ML closer to IoT help to achieve not only better privacy but also better latency which is imperative for the real-time performance of IoT applications. As the collected data does not need to be transferred to a far cloud data center, ML-based IoT applications can process the data, perform inference, and return the result in a much shorter amount of time. Although not every IoT application needs real-time guarantees, energy-efficiency of real-time IoT applications is gaining popularity and already a few studies analyzed the energy efficiency of ML in IoT image classification [81] and arrhythmia detection [84] applications. More research efforts are needed in this regard to pushing energy efficiency in ML-based real-time IoT applications.

6. Conclusion

On-device ML models are swiftly gaining popularity due to their usage in IoT applications that need privacy-preserving and real-time response. However, performing energy-intensive ML tasks on battery-constrained IoT devices is a huge challenge. Numerous studies have proposed solutions to address this challenge. In this work, we comprehensively reviewed these studies that focus on on-device ML models deployed on IoT devices and their energy consumption. We provided a taxonomy of on-device ML approaches for IoT applications. Furthermore, we examined the algorithms and techniques such as compression and quantization and tools to run ML on IoT devices. Finally, we provided the lessons learned and the open issues.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was partially supported by the U.S. National Science Foundation (Awards: NSF-CAREER CNS-1453647, 2039606, 2219920), Cyber Florida, and Microsoft. Dr. Tekin was supported by the Scientific and Technological Research Council of Turkey (TUBITAK-BIDEB) 2219—International Postdoctoral Research Scholarship Program. The views expressed are those of the authors only, not of the funding agencies.

References

[1] J. Franco, A. Aris, B. Canberk, A.S. Uluagac, A survey of honeypots and honeynets for Internet of Things, industrial Internet of Things, and cyber-physical systems, IEEE Commun. Surv. Tutor. 23 (4) (2021) 2351–2383.
[2] L. Da Xu, W. He, S. Li, Internet of Things in industries: A survey, in: IEEE Transactions on Industrial Informatics, vol. 10, IEEE, 2014, pp. 2233–2243.
[3] A.I. Newaz, A.K. Sikder, M.A. Rahman, A.S. Uluagac, A Survey on Security and Privacy Issues in Modern Healthcare Systems: Attacks and Defenses, vol. 2, ACM New York, NY, USA, 2021, pp. 1–44.
[4] Google Nest Help, How Nest thermostats learn, 2022, [Online] https://support.google.com/googlenest/answer/9247510?hl=en (Accessed 3 March 2022).
[5] Amazon, What is conversational AI?, 2022, [Online] https://developer.amazon.com/en-US/alexa/alexa-skills-kit/conversational-ai (Accessed 3 March 2022).
[6] K. Bruton, Smart door locks with facial recognition, 2022, [Online] https://brinkshome.com/smartcenter/smart-door-locks-with-facial-recognition (Accessed 3 March 2022).
[7] S.K.M. Ajay Kattepur, How AI-enabled router configuration improves QoS in 5G networks, 2022, [Online] https://www.ericsson.com/en/blog/2021/7/router-configuration (Accessed 3 March 2022).
[8] F. Samie, L. Bauer, J. Henkel, From cloud down to things: An overview of machine learning in Internet of Things, in: IEEE Internet of Things Journal, vol. 6, IEEE, 2019.
[9] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, M. Imran, The role of edge computing in Internet of Things, in: IEEE Communications Magazine, vol. 56, IEEE, 2018, pp. 110–115.
[10] J. McKeon, This year's largest healthcare data breaches, 2021, [Online] https://healthitsecurity.com/features/this-years-largest-healthcare-data-breaches (Accessed 8 March 2022).
[11] A. Harrod, New AI technology from arm delivers unprecedented on-device intelligence for IoT, 2020, [Online] https://www.businesswire.com/news/home/20200210005346/en/New-AI-Technology-From-Arm-Delivers-Unprecedented-On-Device-Intelligence-for-IoT (Accessed 12 March 2022).
[12] S. Ravi, Custom on-device ML models with Learn2Compress, 2018, [Online] https://ai.googleblog.com/2018/05/custom-on-device-ml-models.html (Accessed 12 March 2022).
[13] Apple Developer, Core ML overview, 2022, [Online] https://developer.apple.com/machine-learning/core-ml/ (Accessed 3 March 2022).
[14] Amazon, Build smarter apps with machine learning, 2022, [Online] https://developer.android.com/ml (Accessed 3 March 2022).
[15] Amazon, AWS IoT greengrass ML inference, 2022, [Online] https://aws.amazon.com/greengrass/ml/ (Accessed 3 March 2022).
[16] Apple, Hey siri: An on-device DNN-powered voice trigger for Apple's personal assistant, 2017, [Online] https://machinelearning.apple.com/research/hey-siri (Accessed 3 March 2022).
[17] H. Ren, D. Anicic, T.A. Runkler, Tinyol: Tinyml with online-learning on microcontrollers, in: 2021 International Joint Conference on Neural Networks, IJCNN, IEEE, 2021, pp. 1–8.
[18] B. Sudharsan, P. Patel, J.G. Breslin, M.I. Ali, Ultra-fast machine learning classifier execution on IoT devices without sram consumption, in: Int. Conf. on Pervasive Computing and Communications Workshops and Other Affiliated Events, IEEE, 2021.
[19] Google scholar, 2022, [Online] https://scholar.google.com/ (Accessed 13 March 2022).
[20] IEEE XPlore Digital Library, 2022, [Online] https://ieeexplore.ieee.org/ (Accessed 3 March 2022).
[21] ACM digital library, 2022, [Online] https://dl.acm.org/ (Accessed 3 March 2022).
[22] ScienceDirect, 2022, [Online] https://www.sciencedirect.com/ (Accessed 13 March 2022).
[23] Arxiv, 2022, [Online] https://arxiv.org/ (Accessed 13 March 2022).
[24] M.S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, A.P. Sheth, Machine learning for Internet of Things data analysis: A survey, Digit. Commun. Netw. 4 (2018) 161–175.
[25] S. Messaoud, A. Bradai, S.H.R. Bukhari, P.T.A. Qung, O.B. Ahmed, M. Atri, A survey on machine learning in Internet of Things: Algorithms, strategies, and applications, Internet of Things 12 (2020) 100314.
[26] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, J. Qin, A survey on application of machine learning for Internet of Things, Int. J. Mach. Learn. Cybern. 9 (2018).
[27] S. Durga, R. Nag, E. Daniel, Survey on machine learning and deep learning algorithms used in Internet of Things (IoT) healthcare, in: 3rd International Conference on Computing Methodologies and Communication, IEEE, 2019, pp. 1018–1022.
[28] Z. Ullah, F. Al-Turjman, L. Mostarda, R. Gagliardi, Applications of artificial intelligence and machine learning in smart cities, Comput. Commun. 154 (2020) 313–323.
[29] S.M. Tahsien, H. Karimipour, P. Spachos, Machine learning based solutions for security of Internet of Things (IoT): A survey, J. Netw. Comput. Appl. 161 (2020) 102630.
[30] B. Qian, J. Su, Z. Wen, D.N. Jha, Y. Li, Y. Guan, D. Puthal, P. James, R. Yang, A.Y. Zomaya, et al., Orchestrating the development lifecycle of machine learning-based IoT applications: A taxonomy and survey, in: ACM Computing Surveys, vol. 53, ACM New York, NY, USA, 2020, pp. 1–47.
[31] M.S. Murshed, C. Murphy, D. Hou, N. Khan, G. Ananthanarayanan, F. Hussain, Machine learning at the network edge: A survey, in: ACM Computing Surveys, vol. 54, ACM New York, NY, 2021.
[32] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 2031–2063.
[33] X. Wang, Y. Han, V.C. Leung, D. Niyato, X. Yan, X. Chen, Convergence of edge computing and deep learning: A comprehensive survey, IEEE Comm. Surv. Tutor. 22 (2020).
[34] Q. Zhou, Z. Qu, S. Guo, B. Luo, J. Guo, Z. Xu, R. Akerkar, On-device learning systems for edge intelligence: A software and hardware synergy perspective, IEEE Internet Things J. 8 (15) (2021) 11916–11934.
[35] P.P. Ray, A review on TinyML: State-of-the-art and prospects, J. King Saud Univ.-Comp. Inform. Sci. (2021).
[36] H. Doyu, R. Morabito, M. Brachmann, A TinyMLaaS ecosystem for machine learning in IoT: Overview and research challenges, in: International Symposium on VLSI Design, Automation and Test, VLSI-DAT, IEEE, 2021, pp. 1–5.
[37] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, Y. Gao, A survey on federated learning, in: Knowledge-Based Systems, vol. 216, Elsevier, 2021.
[38] D.C. Nguyen, M. Ding, P.N. Pathirana, A. Seneviratne, J. Li, H.V. Poor, Federated learning for Internet of Things: A comprehensive survey, IEEE Comm. Surv. Tutor. (2021).
[39] L.U. Khan, W. Saad, Z. Han, E. Hossain, C.S. Hong, Federated learning for Internet of Things: Recent advances, taxonomy, and open challenges, IEEE Comm. Surv. Tutor. (2021).
[40] I. Kholod, E. Yanaki, D. Fomichev, E. Shalugin, E. Novikova, E. Filippov, M. Nordlund, Open-source federated learning frameworks for IoT: A comparative review and analysis, Sensors 21 (1) (2021) 167.
[41] A. Imteaj, U. Thakker, S. Wang, J. Li, M.H. Amini, A survey on federated learning for resource-constrained IoT devices, IEEE Internet Things J. 9 (1) (2021).
[42] J. Chen, X. Ran, Deep learning with edge computing: A review, in: Proceedings of the IEEE, vol. 107, 2019, pp. 1655–1674.
[43] S. Dhar, J. Guo, J. Liu, S. Tripathi, U. Kurup, M. Shah, A survey of on-device machine learning: An algorithms and learning theory perspective, ACM Trans. Internet Things 2 (3) (2021) 1–49.
[44] L. Dutta, S. Bharali, TinyML meets IoT: A comprehensive survey, in: Internet of Things, vol. 16, Elsevier, 2021, 100461.
[45] S. Disabato, M. Roveri, Incremental on-device tiny machine learning, in: Proceedings of the 2nd Int. Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things, 2020, pp. 7–13.
[46] O. Gómez-Carmona, D. Casado-Mansilla, F.A. Kraemer, D. López-de Ipiña, J. García-Zubia, Exploring the computational cost of machine learning at the edge for human-centric Internet of Things, in: Future Generation Computer Systems, Elsevier, 2020.
[47] A.A. Sarangdhar, V. Pawar, Machine learning regression technique for cotton leaf disease detection and controlling using IoT, in: 2017 International Conference of Electronics, Communication and Aerospace Technology, vol. 2, IEEE, 2017.
[48] H. Orchi, M. Sadik, M. Khaldoun, On using artificial intelligence and the Internet of Things for crop disease detection: A contemporary survey, in: Agriculture, vol. 12, Multidisciplinary Digital Publishing Institute, 2022, p. 9.
[49] N.N. An, N.Q. Thanh, L. Yanbing, F. Wu, Combining deep neural network with SVM to identify used in IoT, in: 15th International Wireless Communications & Mobile Computing Conference, IWCMC, IEEE, 2019, pp. 1145–1149.
[50] A.K. Prasad, SMART asthma alert using IoT and predicting threshold values using decision tree classifier, in: Applications of Internet of Things, Springer, 2021, pp. 141–150.
[51] P. Kaur, R. Kumar, M. Kumar, A healthcare monitoring system using random forest and Internet of Things (IoT), Multimedia Tools Appl. 78 (14) (2019) 19905–19916.
[52] H. Oz, A. Aris, A. Levi, A.S. Uluagac, A survey on ransomware: Evolution, taxonomy, and defense solutions, ACM Comput. Surv. 54 (11s) (2022) 1–37.
[53] M. Hasan, M.M. Islam, M.I.I. Zarif, M. Hashem, Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches, in: Internet of Things, vol. 7, Elsevier, 2019, 100059.
[54] F. Abbasi, M. Naderan, S.E. Alavi, Anomaly detection in Internet of Things using feature selection and classification based on Logistic Regression and Artificial Neural Network on N-BaIoT dataset, in: 5th International Conference on Internet of Things and Applications, IoT, IEEE, 2021, pp. 1–7.

[55] S. Bandyopadhyay, S. Thakur, J. Mandal, Product recommendation for e-commerce business by applying principal component analysis (PCA) and K-means clustering: benefit for the society, Innov. Syst. Softw. Eng. 17 (1) (2021) 45–52.
[56] P.C. Siswipraptini, R.N. Aziza, M. Asura, R.R.A. Siregar, M.A. Jabar, K-means clustering algorithm for smart home automation, in: 8th International Conference on Control, Mechatronics and Automation, ICCMA, IEEE, 2020, pp. 207–211.
[57] J. Stewart, R. Stewart, S. Kennedy, Dynamic IoT management system using k-means machine learning for precision agriculture applications, in: Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing, 2017, pp. 1–8.
[58] D.H. Hoang, H.D. Nguyen, A PCA-based method for IoT network traffic anomaly detection, in: 20th International Conference on Advanced Communication Technology, IEEE, 2018.
[59] A. Acar, H. Fereidooni, T. Abera, A.K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi, S. Uluagac, Peek-a-boo: I see your smart home activities, even encrypted!, in: Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2020, pp. 207–218.
[60] X. Zeng, K. Cao, M. Zhang, MobileDeepPill: A small-footprint mobile deep learning system for recognizing unconstrained pill images, in: Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, 2017, pp. 56–67.
[61] N.T. Nam, P.D. Hung, Padding methods in convolutional sequence model: an application in Japanese handwriting recognition, in: Proceedings of the 3rd International Conference on Machine Learning and Soft Computing, 2019, pp. 138–142.
[62] H. Sak, A. Senior, F. Beaufays, Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014, Preprint ArXiv:1402.1128.
[63] L. Babun, K. Denney, Z.B. Celik, P. McDaniel, A.S. Uluagac, A survey on IoT platforms: Communication, security, and privacy perspectives, Comput. Netw. 192 (2021) 108040.
[64] A.K. Sikder, G. Petracca, H. Aksu, T. Jaeger, A.S. Uluagac, A survey on sensor-based threats and attacks to smart devices and applications, IEEE Communications Surveys & Tutorials 23 (2) (2021) 1125–1159.
[65] L.P. Rondon, L. Babun, A. Aris, K. Akkaya, A.S. Uluagac, Survey on enterprise Internet of Things systems (E-IoT): A security perspective, Ad Hoc Netw. 125 (2022) 102728.
[66] Samsung, 2022, [Online] https://www.smartthings.com/ (Accessed 3 March 2022).
[67] Apple, 2022, [Online] https://www.apple.com/shop/accessories/all/homekit (Accessed 3 March 2022).
[68] Amazon, 2022, [Online] https://aws.amazon.com/iot/ (Accessed 3 March 2022).
[69] openH.A.B. Community, Openhab, 2022, [Online] https://www.openhab.org/ (Accessed 3 March 2022).
[70] Home Assistant, https://www.home-assistant.io/.
[71] N. Tekin, H.U. Yildiz, V.C. Gungor, Node-level error control strategies for prolonging the lifetime of wireless sensor networks, IEEE Sens. J. 21 (13) (2021) 15386–15397.
[72] N. Tekin, V.C. Gungor, Analysis of compressive sensing and energy harvesting for wireless multimedia sensor networks, Ad Hoc Netw. 103 (2020) 102164.
[73] N. Tekin, V.C. Gungor, The impact of error control schemes on lifetime of energy harvesting wireless sensor networks in industrial environments, Comput. Stand. Interfaces 70 (2020) 103417.
[74] N. Tekin, A. Acar, A. Aris, A.S. Uluagac, V.C. Gungor, Energy consumption of on-device machine learning models for IoT intrusion detection, in: Internet of Things, Elsevier, 2022, 100670.
[75] Raspberry Pi, Raspberry pi, 2023, [Online] https://datasheets.raspberrypi.com/?_gl=1*1byd9qf*_ga*NTc5NDI4OTUuMTY4ODU1MjcyOA..*_ga_22FD70LWDS*MTY4ODU1MjcyNy4xLjEuMTY4ODU1MjgwMC4wLjAuMA (Accessed 5 July 2023).
[76] Arduino, Arduino, 2023, [Online] https://docs.arduino.cc/ (Accessed 5 July 2023).
[77] Espressif, Espressif, 2023, [Online] https://www.espressif.com/en/support/download/documents/development-board (Accessed 5 July 2023).
[78] Intel, Intel, 2023, [Online] https://www.intel.com/content/dam/www/public/us/en/documents/datasheets/galileo-g2-datasheet.pdf (Accessed 5 July 2023).
[79] Intel, Intel, 2023, [Online] https://cdn-shop.adafruit.com/datasheets/EdisonDatasheet.pdf(Accessed 5 July 2023).
[80] F. Daghero, A. Burrello, D.J. Pagliari, L. Benini, E. Macii, M. Poncino, Energy-efficient adaptive machine learning on IoT end-nodes with class-dependent confidence, in: 27th International Conference on Electronics, Circuits and Systems, IEEE, 2020, pp. 1–4.
[81] B.H. Curtin, S.J. Matthews, Deep learning for inexpensive image classification of wildlife on the Raspberry Pi, in: 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference, IEEE, 2019.
[82] J. Chauhan, S. Seneviratne, Y. Hu, A. Misra, A. Seneviratne, Y. Lee, Breathing-based authentication on resource-constrained IoT devices using recurrent neural networks, in: Computer, vol. 51, IEEE, 2018, pp. 60–67.
[83] S. Bhattacharya, N.D. Lane, Sparsification and separation of deep learning layers for constrained resource inference on wearables, in: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM, 2016, pp. 176–189.
[84] D. Azariadi, V. Tsoutsouras, S. Xydis, D. Soudris, ECG signal analysis and arrhythmia detection on IoT wearable medical devices, in: 5th Int. Conf. on Modern Circuits and Systems Technologies, IEEE, 2016, pp. 1–4.
[85] D. Sprute, M. König, On-chip activity recognition in a smart home, in: 2016 12th International Conference on Intelligent Environments, IE, IEEE, 2016, pp. 95–102.
[86] TensorFlow, TensorFlow: Large-scale machine learning on heterogeneous systems, 2022, [Online] https://www.tensorflow.org/ (Accessed 2 February 2022).
[87] Keras, 2022, [Online] https://keras.io/ (Accessed 2 February 2022).
[88] I.A. Zualkernan, S. Dhou, J. Judas, A.R. Sajun, B.R. Gomez, L.A. Hussain, D. Sakhnini, Towards an IoT-based deep learning architecture for camera trap image classification, in: Global Conference on Artificial Intelligence and Internet of Things, IEEE, 2020.
[89] M. Nasir, K. Muhammad, A. Ullah, J. Ahmad, S.W. Baik, M. Sajjad, Enabling automation and edge intelligence over resource constraint IoT devices for smart home, in: Neurocomputing, vol. 491, Elsevier, 2022, pp. 494–506.
[90] L.T. da Silva, V.M. Souza, G.E. Batista, EmbML Tool: Supporting the use of supervised learning algorithms in low-cost embedded systems, in: 31st Int. Conf. on Tools with Artificial Intelligence, IEEE, 2019, pp. 1633–1637.
[91] G. Crocioni, D. Pau, J.-M. Delorme, G. Gruosso, Li-Ion batteries parameter estimation with tiny neural networks embedded on intelligent IoT microcontrollers, IEEE Access 8 (2020).
[92] Y.H. Wu, H. Luo, Apply edge intelligence to IoT based home automation, in: 17th Int. Conf. on Applied Computing, 2020.
[93] M. Giordano, P. Mayer, M. Magno, A battery-free long-range wireless smart camera for face detection, in: Proceedings of the 8th International Workshop on Energy Harvesting and Energy-Neutral Sensing Systems, 2020, pp. 29–35.
[94] TensorFlow, Tflite-micro, 2022, [Online] https://github.com/tensorflow/tflite-micro (Accessed 2 February 2022).
[95] Microsoft, ELL, 2020, [Online] https://github.com/microsoft/ELL (Accessed 2 February 2022).
[96] MikeJKelly, ARM-software, 2022, [Online] https://github.com/ARM-software/armnn (Accessed 2 February 2022).
[97] GuentherMartin, ARM-software, 2021, [Online] https://github.com/ARM-software/CMSIS_5 (Accessed 2 February 2022).
[98] STMicroelectronics, Stmicroelectronics, 2022, [Online] https://www.st.com/content/st_com/en.html (Accessed 3 March 2022).
[99] Jonnor, Emlearn, 2021, [Online] https://github.com/emlearn/emlearn (Accessed 2 February 2022).
[100] eloquentarduino, MicroMLGen, 2020, [Online] https://github.com/eloquentarduino/micromlgen (Accessed 2 February 2022).
[101] bharathsudharsan, ML-MCU, 2021, [Online] https://github.com/bharathsudharsan/ML-MCU (Accessed 2 February 2022).
[102] lucastsutsui, EmbML, 2021, [Online] https://github.com/lucastsutsui/EmbML (Accessed 2 February 2022).
[103] J. Lin, MCUNet, 2020, [Online] https://github.com/mit-han-lab/mcunet (Accessed 2 February 2022).
[104] C. Profentzas, MicroTL, 2021, [Online] https://github.com/chrpro/MicroTL (Accessed 4 January 2023).
[105] Edge impulse, 2023, [Online] https://docs.edgeimpulse.com/docs/ (Accessed 30 January 2023).
[106] R. Sanchez-Iborra, A. Skarmeta, Who is wearing me? TinyDL-based user recognition in constrained personal devices, in: IET Computers & Digital Techniques, Wiley Online Library, 2021.
[107] R. Yauri, R. Acosta, M. Jurado, M. Rios, Evaluation of principal component analysis algorithm for locomotion activities detection in a tiny machine learning device, in: Engineering International Research Conference, EIRCON, IEEE, 2021, pp. 1–4.
[108] D. Morawiec, Sklearn-porter, 2017, [Online] https://pypi.org/project/sklearn-porter/0.4.0/ (Accessed 2 February 2022).
[109] dependabot, M2cgen, 2020, [Online] https://github.com/BayesWitnesses/m2cgen (Accessed 2 February 2022).
[110] V. Janapa Reddi, A. Elium, S. Hymel, D. Tischler, D. Situnayake, C. Ward, L. Moreau, J. Plunkett, M. Kelcey, M. Baaijens, et al., Edge Impulse: An MLOps platform for tiny machine learning, Proceedings of Machine Learning and Systems 5 (2023).
[111] C. Nicolas, B. Naila, R.-C. Amar, Tinyml smart sensor for energy saving in Internet of Things precision agriculture platform, in: Thirteenth International Conference on Ubiquitous and Future Networks, ICUFN, IEEE, 2022, pp. 256–259.
[112] S. Choi, J. Shin, Y. Choi, L.-S. Kim, An optimized design technique of low-bit neural network training for personalization on IoT devices, in: Proceedings of the 56th Annual Design Automation Conference, 2019, pp. 1–6.
[113] M.T. Yazici, S. Basurra, M.M. Gaber, Edge machine learning: Enabling smart Internet of Things applications, in: Big Data and Cognitive Computing, vol. 2, Multidisciplinary Digital Publishing Institute, 2018, p. 26.

[114] D. Hou, R. Hou, J. Hou, On-device training for breast ultrasound image classification, in: 10th Annual Computing and Communication Workshop and Conference, IEEE, 2020.
[115] S. Leroux, P. Simoens, Sparse random neural networks for online anomaly detection on sensor nodes, Elsevier, 2022,
[116] N.L. Giménez, F. Freitag, J. Lee, H. Vandierendonck, Comparison of two microcontroller boards for on-device model training in a keyword spotting task, in: 2022 11th Mediterranean Conference on Embedded Computing, MECO, IEEE, 2022, pp. 1–4.
[117] H. Ren, D. Anicic, T.A. Runkler, The synergy of complex event processing and tiny machine learning in industrial IoT, in: Proceedings of the 15th ACM International Conference on Distributed and Event-Based Systems, 2021, pp. 126–135.
[118] R. Kolcun, D.A. Popescu, V. Safronov, P. Yadav, A.M. Mandalari, Y. Xie, R. Mortier, H. Haddadi, The case for retraining of ML models for IoT device identification at the edge, 2020, Preprint ArXiv:2011.08605.
[119] B. Sudharsan, J.G. Breslin, M.I. Ali, ML-MCU: A framework to train ML classifiers on MCU-based IoT edge devices, IEEE Internet Things J. (2021).
[120] C. Profentzas, M. Almgren, O. Landsiedel, Microtl: Transfer learning on low-power IoT devices, in: 2022 IEEE 47th Conference on Local Computer Networks, LCN, IEEE, 2022, pp. 1–8.
[121] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, S. Han, On-device training under 256kb memory, Adv. Neural Inf. Process. Syst. 35 (2022) 22941–22954.
[122] S. Choi, J. Sim, M. Kang, L.-S. Kim, TrainWare: A memory optimized weight update architecture for on-device convolutional neural network training, in: Proceedings of the International Symposium on Low Power Electronics and Design, 2018, pp. 1–6.
[123] D. Kim, T. Na, S. Yalamanchili, S. Mukhopadhyay, DeepTrain: A programmable embedded platform for training deep neural networks, in: IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 37 (11) (2018) 2360–2370.
[124] R. Sanchez-Iborra, A.F. Skarmeta, Tinyml-enabled frugal smart objects: Challenges and opportunities, IEEE Circuits Syst. Mag. 20 (3) (2020) 4–18.
[125] S. Hymel, C. Banbury, D. Situnayake, A. Elium, C. Ward, M. Kelcey, M. Baaijens, M. Majchrzycki, J. Plunkett, D. Tischler, et al., Edge Impulse: An MLOps platform for tiny machine learning, arXiv preprint arXiv:2212.03332 (2022).
[126] S. Savazzi, M. Nicoli, V. Rampa, Federated learning with cooperating devices: A consensus approach for massive IoT networks, IEEE Internet Things J. 7 (5) (2020).
[127] T.T. Huong, T.P. Bac, D.M. Long, T.D. Luong, N.M. Dan, B.D. Thang, K.P. Tran, et al., Detecting cyberattacks using anomaly detection in industrial control systems: A federated learning approach, Comput. Industry 132 (2021) 103509.
[128] Y. Gao, M. Kim, S. Abuadbba, Y. Kim, C. Thapa, K. Kim, S.A. Camtepe, H. Kim, S. Nepal, End-to-end evaluation of federated learning and split learning for Internet of Things, 2020, Preprint ArXiv:2003.13376.
[129] Y. Liu, S. Garg, J. Nie, Y. Zhang, Z. Xiong, J. Kang, M.S. Hossain, Deep anomaly detection for time-series data in industrial IoT: a communication-efficient on-device federated learning approach, IEEE Internet Things J. 8 (8) (2020).
[130] A. Das, T. Brunschwiler, Privacy is what we care about: Experimental investigation of federated learning on edge devices, in: Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things, 2019, pp. 39–42.
[131] S. Shukla, P.S. Manoj, G. Kolhe, S. Rafatirad, On-device malware detection using performance-aware and robust collaborative learning, in: 58th ACM/IEEE Design Automation Conference, DAC, IEEE, 2021, pp. 967–972.
[132] P. Prakash, J. Ding, R. Chen, X. Qin, M. Shu, Q. Cui, Y. Guo, M. Pan, Iot device friendly and communication-efficient federated learning via joint model pruning and quantization, IEEE Internet Things J. 9 (15) (2022) 13638–13650.
[133] Y. Li, Y. Bao, W. Chen, Fixed-sign binary neural network: An efficient design of neural network for Internet-of-Things devices, IEEE Access 8 (2020) 164858–164863.
[134] N. Tonellotto, A. Gotta, F.M. Nardini, D. Gadler, F. Silvestri, Neural network quantization in federated learning at the edge, Inform. Sci. 575 (2021) 417–436.
[135] N. Bruschi, A. Garofalo, F. Conti, G. Tagliavini, D. Rossi, Enabling mixed-precision quantized neural networks in extreme-edge devices, in: Proceedings of the 17th ACM International Conference on Computing Frontiers, 2020, pp. 217–220.
[136] C.-H. Wang, K.-Y. Huang, Y. Yao, J.-C. Chen, H.-H. Shuai, W.-H. Cheng, Lightweight deep learning: An overview, IEEE Consum. Electron. Mag. (2022).
[137] K. Su, C.M. Intisar, Q. Zhao, Y. Tomioka, Knowledge distillation for real-time on-road risk detection, in: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress, DASC/PiCom/CBDCom/CyberSciTech, IEEE, 2020, pp. 110–117.
[138] Z. Tao, Q. Xia, S. Cheng, Q. Li, An efficient and robust cloud-based deep learning with knowledge distillation, IEEE Trans. Cloud Comput. 11 (2) (2023) 1733–1745, http://dx.doi.org/10.1109/TCC.2022.3160129.

Nazli Tekin received a B.S. degree in computer engineering from Koc University, Istanbul, Turkey, in 2011, and a Ph.D. degree in electrical and computer engineering from Abdullah Gül University, Kayseri, Turkey, in 2020. She is an Assistant Professor at the Department of Software Engineering, Erciyes University, Kayseri. Currently, she is a post-doctoral associate at the Cyber-Physical Systems Security Lab, Department of Electrical and Computer Engineering, Florida International University, Miami, FL, USA. Her research interests include wireless sensor networks, the Industrial Internet of Things, and security.

Ahmet Aris is a Research Assistant Professor in the Department of Electrical and Computer Engineering at Florida International University. He is conducting research in Cyber-Physical Systems Security Lab (CSL) at Florida International University under the supervision of Dr. A. Selcuk Uluagac. He earned both PhD and MSc. in Computer Engineering from the Graduate School of Science, Engineering and Technology at Istanbul Technical University, Turkey. He also worked at Medianova CDN R&D Center as an R&D Analyst. In addition, he conducted research in the Networked Embedded Systems (NES) Group at Swedish Institute of Computer Science (SICS) as a visiting researcher. His research interests include IoT Security, Network Security, Web Security, and Malware.

Abbas Acar received his MSc and Ph.D. degrees in the Department of Electrical and Computer Engineering at Florida International University in 2019 and 2020, respectively. Before that, he received his B.S. degree in Electrical and Electronics Engineering from Middle East Technical University in 2015. His research interests include continuous authentication, IoT security/privacy, and homomorphic encryption. More information can be obtained from https://web.eng.fiu.edu/aacar/

Selcuk Uluagac is currently an Eminent Scholar Chaired Professor in the School of Computing and Information Science at Florida International University (FIU), where he leads the Cyber-Physical Systems Security Lab, with an additional courtesy appointment in the Department of Electrical and Computer Engineering. Before FIU, he was a Senior Research Engineer at Georgia Tech and Symantec. He holds a PhD from Georgia Tech and MS from Carnegie Mellon University. He received US National Science Foundation (NSF) CAREER Award (2015), Air Force Office of Sponsored Research's Summer Faculty Fellowship (2015), and University of Padova (Italy)'s Faculty Fellowship (2016), and Google's ASPIRE Research award in security and privacy (2021). He is an expert in the areas of cybersecurity and privacy with an emphasis on their practical and applied aspects and teaches classes in these areas. He has hundreds of research papers/studies/publications in the most reputable venues. His research in cybersecurity and privacy has been funded by numerous government agencies and industries, including the US NSF, the US Dept. of Energy, the US Air Force Research Lab, US Dept. of Labor, Cyber Florida, Google, Microsoft, Trend Micro, and Cisco, inter alia. He is very entrepreneurial and visionary in his research. Many of his research ideas have resulted in patents with one licensed to a company recently. He has served on the program committees of top-tier security conferences such as IEEE Security & Privacy ("Oakland"), NDSS, Usenix Security, inter alia. He was the TPC Chair of ACM CCS 2023 Security & ML Track, TPC Co-Chair of 2022 IEEE CNS. In 2018, he co-chaired the NIST's National Initiative for Cybersecurity Education Annual Expo and Conference. Currently, he serves on the editorial boards of IEEE Transactions of Information Forensics and Security (Deputy Editor-in-Chief), IEEE Transactions on Mobile Computing, and Elsevier Computer Networks. And, he is very active in the local and national community; his research has been covered by different media outlets (TV, online, published) numerous times. More information can be obtained from https://users.cs.fiu.edu/~suluagac/.

Vehbi Cagri Gungor received his Ph.D. degree in electrical and computer engineering from the Broadband and Wireless Networking Laboratory, Georgia Institute of Technology, Atlanta, GA, USA, in 2007. Currently, he is a Full Professor and Dean of the Faculty of Computer Science, Abdullah Gul University in Kayseri, Turkey. His current research interests are in machine learning, machine-to-machine communications, next-generation wireless networks, wireless ad hoc and sensor networks. Dr. Gungor has authored more than 100 papers in refereed journals and international conference proceedings, and has been serving as an editor, reviewer and program committee member to numerous journals and conferences in these areas. He is also the recipient of TUBITAK Young Scientist Award in 2017, Science Academy Young Scientist Award (BAGEP) in 2017, Turkish Academy of Sciences Distinguished Young Scientist Award (TUBA-GEBIP) in 2014, the IEEE Trans. on Industrial Informatics Best Paper Award in 2012, the European Union FP7 Marie Curie RG Award in 2009.