technologies

MDPI

Review

A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions

Oumayma Jouini [1,2], Kaouthar Sethom [1], Abdallah Namoun [3], Nasser Aljohani [3], Meshari Huwaytim Alanazi [4,*] and Mohammad N. Alanazi [5]

1 Innov'COM Laboratory, Higher School of Communication of Tunis (SUPCOM), Technopark Elghazala, Ariana 2083, Tunisia; oumayma.jouini@supcom.tn (O.J.)
2 National Engineering School of Tunis, University of Tunis El Manar, Tunis 1002, Tunisia
3 Faculty of Computer and Information Systems, Islamic University of Madinah, Madinah 42351, Saudi Arabia; a.namoun@iu.edu.sa (A.N.); naljohani@iu.edu.sa (N.A.)
4 Computer Science Department, College of Sciences, Northern Border University, Arar 91431, Saudi Arabia
5 College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 13318, Saudi Arabia; alanazi@imamu.edu.sa
* Correspondence: meshari.alanazi@nbu.edu.sa

Abstract: Internet of Things (IoT) devices often operate with limited resources while interacting with users and their environment, generating a wealth of data. Machine learning models interpret such sensor data, enabling accurate predictions and informed decisions. However, the sheer volume of data from billions of devices can overwhelm networks, making traditional cloud data processing inefficient for IoT applications. This paper presents a comprehensive survey of recent advances in models, architectures, hardware, and design requirements for deploying machine learning on low-resource devices at the edge and in cloud networks. Prominent IoT devices tailored to integrate edge intelligence include Raspberry Pi, NVIDIA's Jetson, Arduino Nano 33 BLE Sense, STM32 Microcontrollers, SparkFun Edge, Google Coral Dev Board, and Beaglebone AI. These devices are boosted with custom AI frameworks, such as TensorFlow Lite, OpenEI, Core ML, Caffe2, and MXNet, to empower ML and DL tasks (e.g., object detection and gesture recognition). Both traditional machine learning (e.g., random forest, logistic regression) and deep learning methods (e.g., ResNet-50, YOLOv4, LSTM) are deployed on devices, distributed edge, and distributed cloud computing. Moreover, we analyzed 1000 recent publications on "ML in IoT" from IEEE Xplore using support vector machine, random forest, and decision tree classifiers to identify emerging topics and application domains. Hot topics included big data, cloud, edge, multimedia, security, privacy, QoS, and activity recognition, while critical domains included industry, healthcare, agriculture, transportation, smart homes and cities, and assisted living. The major challenges hindering the implementation of edge machine learning include encrypting sensitive user data for security and privacy on edge devices, efficiently managing resources of edge nodes through distributed learning architectures, and balancing the energy limitations of edge devices and the energy demands of machine learning.

Keywords: machine learning; Internet of Things; IoT devices; edge intelligence; edge learning; artificial intelligence; deep learning; review

Citation: Jouini, O.; Sethom, K.; Namoun, A.; Aljohani, N.; Alanazi, M.H.; Alanazi, M.N. A Survey of Machine Learning in Edge Computing: Techniques, Frameworks, Applications, Issues, and Research Directions. *Technologies* **2024**, *12*, 81. https://doi.org/10.3390/technologies12060081

Academic Editors: Xavier Fernando and Mario Munoz-Organero

Received: 5 April 2024
Revised: 20 May 2024
Accepted: 29 May 2024
Published: 3 June 2024

Correction Statement: This article has been republished with a minor change. The change does not affect the scientific content of the article and further details are available within the backmatter of the website version of this article.

1. Introduction

During the past ten years, there has been a growing interest in wireless communication technology, and the number of IoT devices has risen significantly. It is forecasted to reach more than 26 billion devices connected to the Internet by 2030 [1], as depicted in Figure 1. These devices generate a large volume of data per second, have limited computing power, small memory capacities, and lack the self-sufficient intelligence to process raw data locally and make independent decisions. Resource-constrained IoT devices include sensors, smart fridges, and smart lights.

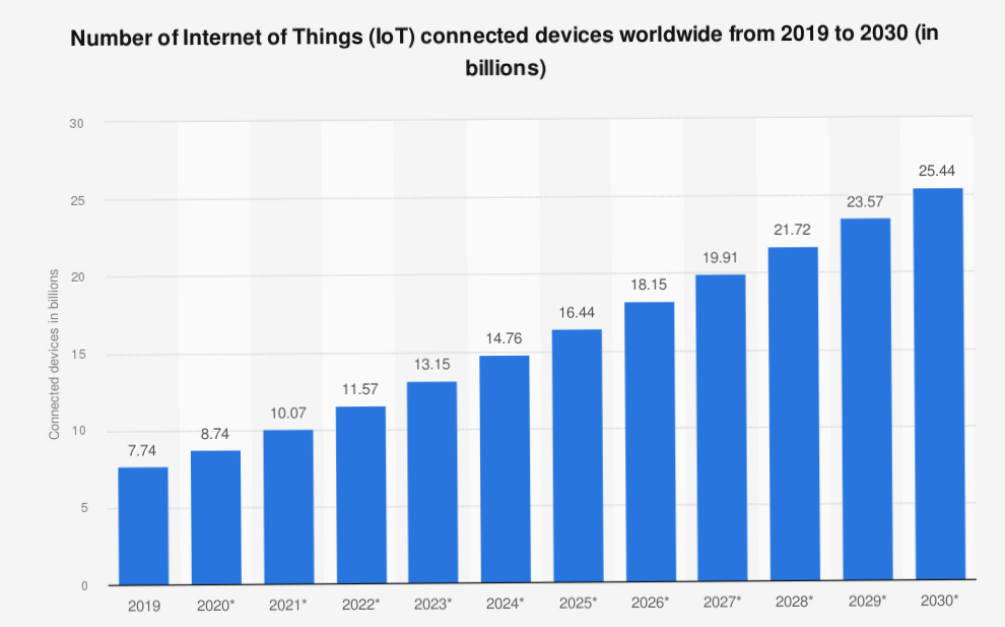

Figure 1. The Growth of IoT Devices between 2019 and 2030.

Feeding data into a machine learning (ML) system is among the most effective methods for extracting information and making decisions from IoT devices. Unfortunately, the processing capacity of these devices hinders the deployment of ML algorithms. Traditionally, the solution has been to offload all data to a centralized cloud system for further processing. However, this approach often leads to high latency, bandwidth saturation, and user data privacy concerns, as the data must be transferred and stored in the cloud indefinitely. Moreover, with the constant development of the IoT industry and the massive amount of data collected by each device, offloading all data to the cloud is becoming increasingly challenging (Figure 1). To address these issues, various solutions have been proposed. One effective solution is to process data as close as possible to its source and only transmit essential data to remote servers for further processing. This approach, known as edge computing or edge intelligence, offers several advantages.

In fact, edge intelligence (EI) proposes relocating data processing at the edge of the IoT network, which includes gateways, embedded devices, and edge servers near where the data is collected rather than a distant location. This is achieved by placing computing devices near the data-generating devices, spreading computation across the network's layers. For example, the IoT gateway may preprocess the data before transmitting the temporary results to the cloud, which will complete the rest of the processing. Integrating ML capabilities into edge computing could reduce data dimensionality. The preprocessing would be executed at the edge node to extract important information transmitted to the cloud for advanced analysis, decreasing bandwidth requirements and workload in the cloud systems. By processing all data locally, the system not only saves on bandwidth but also preserves privacy. A similar term is fog computing, which refers to an infrastructure where the cloud is brought closer to IoT devices. This system reduces delays and improves security by handling computations closer to the network's edge.

The primary distinction between fog and edge computing comes down to where data processing occurs. Edge computing handles data directly on sensor-equipped devices or on nearby gateway devices. Meanwhile, in the fog model, data processing occurs a bit farther from the edge, usually on devices linked through a local area network (LAN). Bringing the intelligence closer to the source can be implemented in edge or fog nodes. We delve deeper into this concept in Sections 3 and 4. Another solution discussed is running ML algorithms on end devices (embedded intelligence), which is mostly addressed as on-device learning. This concept can introduce substantial improvements to the IoT infrastructure. Applying ML directly on the device enables real-time decision-making through data analysis, tailored

behavior personalized specifically for each consumer, congestion prevention and, above all, to preserve the confidentiality of the user's data. Several research efforts have been made in order to successfully deploy ML in the IoT device (closer to the data source). These smart devices will be able to process sensed data, analyze the problem, and operate directly. In order to achieve this, many devices have been developed with improved GPUs and CPUs capable of handling the complexity of machine learning and deep learning (DL).

Additionally, lighter versions of frameworks and platforms that can support on-device machine learning have been developed. Section 4 of this paper delves deeper into this topic. The final solution examined in this paper involves integrating computation across the layers of the IoT network. The deployment of complex ML algorithms on the device itself is still limited due to the computing and memory constraints. Therefore, it is recommended to distribute the data processing among IoT devices, gateways, fog, and cloud computing. For instance, the IoT device can preprocess the data using lightweight embedded models and only offload the necessary data to the cloud for additional processing. This concept focuses on deploying ML models across all layers of the network, leveraging the hierarchical structure of the IoT system. Each layer, from embedded devices to gateways to edge servers to cloud servers, is capable of performing a certain amount of computation, with the capabilities increasing as we move up the hierarchy. This distributed approach enables the efficient processing of data, from the edge to the cloud, by allocating tasks to the most suitable computational entity at each level. Model partitioning is another solution in deep learning, in which certain layers are processed on the device, while others are processed on an edge server or within the cloud. Due to the computational cycles of other edge devices, this approach may be able to reduce latency. Evidently, after computing the first layers of the deep neural network (DNN) model, the intermediate results are comparably smaller, facilitating faster network transmission to an edge server. We have discussed this aspect further in the paper.

Although there is still much progress to be made in standardizing IoT architecture and technologies, Figure 2 can precisely identify the major components of the architecture as commonly utilized across a variety of applications. The architecture key components are divided into blocks in Figure 2. Each block illustrates a representation of the explained element, and arrows connect the blocks to show how each element interacts with the others. Text blocks are also included, with bulleted lists of the essential parts of each key factor. The IoT architecture includes :

1. **Perception or Sensing Layer:** the perception layer includes the physical components, such as IoT devices, sensors, and actuators. It is in charge of identifying objects and gathering information from them. we can find different types of sensors depending on the application [2]. ML can be implemented in this layer, which we identify as embedded intelligence. It is further explained in the next section.
2. **Network Layer:** network or transmission layer is in charge of routing and transmitting the data collected from the physical objects to the upper layers. The communication can be wired or wireless. The communication protocols commonly utilized in IoT include Wi-Fi, Bluetooth, IEEE 802.15.4, Z-wave, LTE-Advanced, RFID, Near Field Communication (NFC), and ultra-wide bandwidth (UWB).
3. **Processing Layer:** the middleware or processing layer collects and processes large amounts of data from the network layer. It possesses the capability to administer and deliver a range of services to the underlying layers. Among the technologies used in this layer are databases, cloud computing, and big data processing modules. The computation load could be divided between the fog/edge and cloud servers.
4. **Application Layer:** the application layer is in charge of providing users with services designed for specific applications. It defines a variety of IoT applications, like Smart Home, Smart City, and Smart Health.

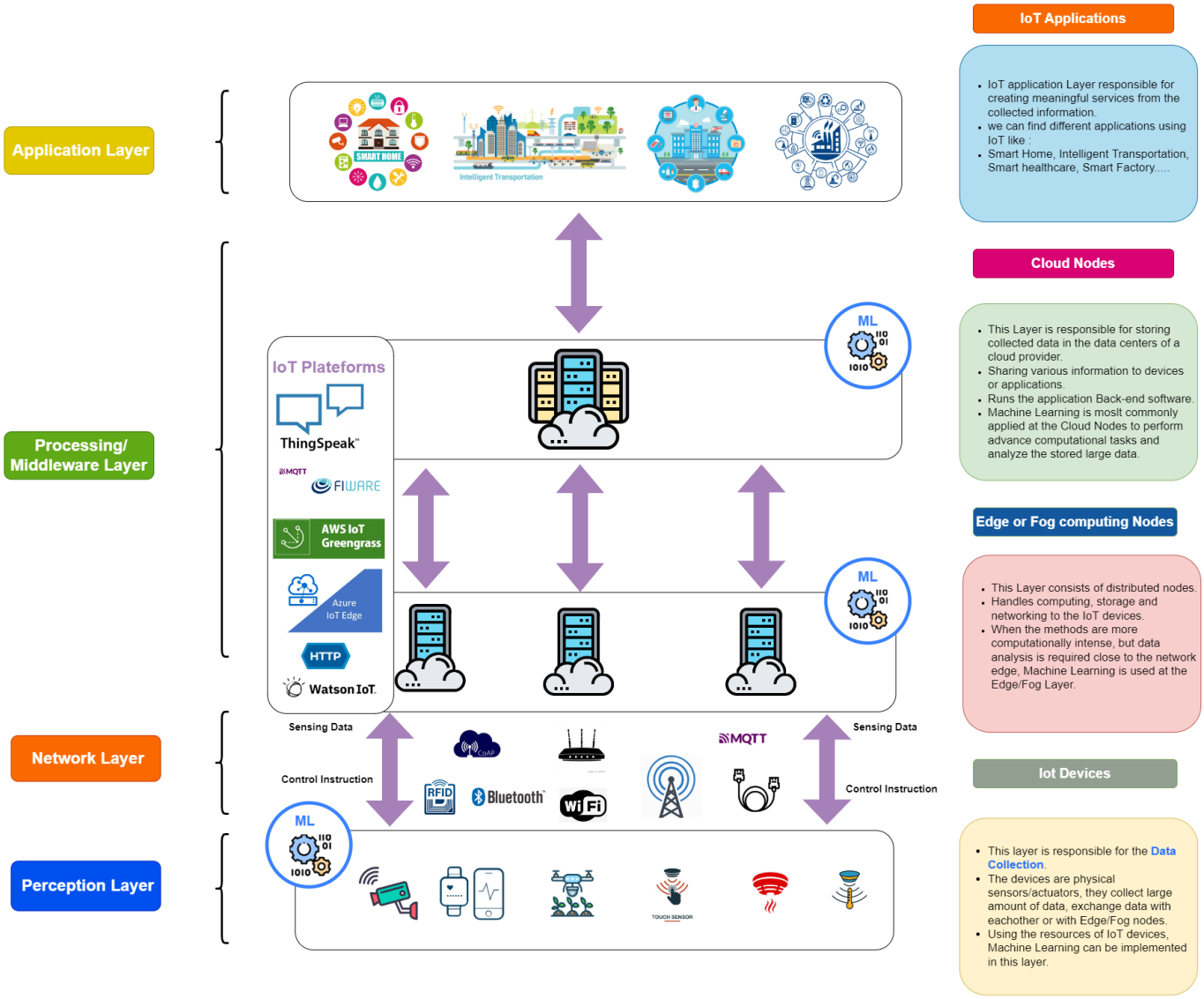

Figure 2. Essential Components of an IoT Architecture.

Furthermore, Figure 2 illustrates the significance of machine learning within the IoT infrastructure. ML techniques are applicable at various points, including IoT nodes, edge or fog servers, and cloud servers, adapted to suit the requirements of the specific application.

Combining artificial intelligence into edge computing applications has been explored by numerous review papers. In Grzesik et al. [3] provide a comprehensive survey on the integration of machine learning and edge computing within edge devices. The authors examine the benefits and challenges of combining AI in these devices. However, they have not explored the opportunities of applying AI to different architectures of the IoT. In their study, Chang et al. [4] investigate the combination of IoT and AI through the utilization of both edge computing and cloud resources. They delve into seven distinct IoT application scenarios, concentrating on techniques that facilitate the optimal deployment of AI models. Nonetheless, they overlooked the exploration of AI techniques and the potential benefits and applications of AI in edge-based scenarios and did not explore the opportunities of distributing the intelligence in all the IoT networks. Opposed to these reviews, our study examines the use of machine learning in various IoT fields, discusses the integration of intelligence into multiple layers of the IoT network, explores recent developments in this field and future work, and addresses related issues. Additionally, we employ a structured

method based on ML classification techniques to analyze all publications on ML in IoT that are available via an API. The primary contributions of this paper are outlined as follows:

- We conduct a thorough investigation into the current state of the art regarding ML in the IoT. It involves categorizing ML approaches based on their deployment within the IoT architecture, their application domains, and the developed frameworks and hardware to facilitate ML integration.
- We deliver an in-depth analysis of diverse ML techniques, their characteristics, suitability for IoT, and innovative solutions to overcome associated challenges.
- We explore the opportunities and challenges of seamlessly integrating IoT devices with embedded intelligence while addressing the combination of the computational overload across the cloud, fog, and edge layers.
- We employ a machine learning approach to analyze publications concerning ML in the IoT in various applications, training precise classifiers to categorize publications based on key phrases found in their titles and abstracts. The insights provided enable other researchers to replicate the analysis with updated publications in the future.
- Based on a comprehensive review of IoT and machine learning literature, we highlight key challenges and promising research directions for optimizing machine learning on IoT systems, with a focus on edge computing as the primary paradigm.

The rest of this paper is structured as follows: Section 2 presents ML techniques, focusing on the requirements of ML algorithms in IoT. Section 3 reviews the fundamentals of edge computing. In Section 4, we explain the requirements of integrating ML in the IoT network, focusing on different possible architectures, including available frameworks and hardware to facilitate the deployment of ML models. We also review cutting-edge research on ML applications at the edge for various use cases, positioning them on the cloud-to-edge spectrum. In Section 5, we investigate the open challenges encountered in this field. Then, in Section 6, we analyze ML and IoT articles, classify, and label them using an ML approach. Finally, Section 7 outlines the future directions of machine learning at the edge, while Section 8 concludes this review.

2. Machine Learning Techniques

Machine learning (ML) is a subset of artificial intelligence (AI) focused on improving performance through automatic learning from experience. ML relies on training machines with large datasets to make better decisions. As defined by Mitchell, ML involves learning from experience E to improve performance measure P on tasks T [5]. ML algorithms can be classified into four major categories: supervised, unsupervised, semi-supervised, and reinforcement learning. Figure 3 demonstrates a detailed taxonomy of ML algorithms.

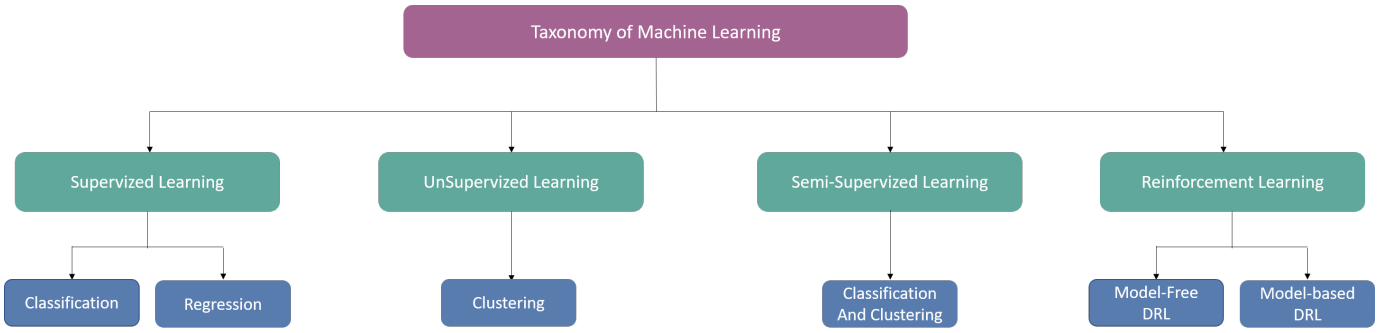

Figure 3. Taxonomy of Machine Learning Algorithms.

2.1. Supervised Learning

Supervised learning is one of the most used types of learning, the majority of ML algorithms are supervised. Supervised machine learning uses labeled datasets to train algorithms where the input and output are known in advance. The goal is to train the data to create a predictive model that can accurately anticipate responses when exposed to

unseen data [6–8]. Supervised machine learning implements classification and regression techniques. Classification is utilized for estimating discrete responses, while regression is employed when dealing with continuous labels. Some of the most popular supervised machine learning algorithms are support vector machine (SVM) and random forest (RF), which are described in the following paragraphs. Their popularity rises because they can be implemented and executed in resource-constrained environments and because of their ability to be distributed in the cloud-to-things network.

SVM is a powerful but simple ML algorithm used in classification and regression [9]. It utilizes a hyperplane to separate data points into distinct classes, aiming to maximize the margin from the closest data points, referred to as support vectors. The margin is crucial, as a larger margin indicates better classification accuracy. When a new data point is introduced, SVM predicts its label based on its position relative to the hyperplane. While SVM inherently works with linearly separable data, it can handle nonlinear datasets by transforming them into higher-dimensional spaces. This transformation is achieved through the kernel trick, which aids in finding the optimal hyperplane without the need for explicit mapping to higher dimensions, thereby reducing computational complexity. Several solutions have been developed to deploy SVM in resource-constrained embedded environments. This paper proposed a real-time tomato classification system using a model that combines convolutional neural networks (CNN) and SVM [10]. The suggested model includes two parts: the EfficientNetB0 CNN model for feature extraction, while SVM handles classification tasks. Notably, to optimize model inference on the embedded platform, they employed TensorRT and performed quantization to lower the model's computational complexity while maintaining accuracy. The hybrid model was deployed on the embedded single-board NVIDIA Jetson TX1, achieving an average accuracy of 93.54% for classifying static tomato images. During real-time implementation, it achieved an average inference speed of 15.6 frames per second, showcasing its capability for rapid decision-making. While the paper demonstrates significant advancements in real-time tomato classification using a hybrid CNN-SVM model deployed on an embedded system, issues such as ensuring data privacy and edge security continue to raise significant concerns. Additionally, optimizing algorithms and hardware to minimize energy consumption is essential for prolonged operation in resource-constrained environments.

RF is a simple yet effective algorithm. It tackles both regression and classification tasks by constructing a "forest" of decision trees through randomization, often employing the bagging algorithm [11,12]. In regression, RF averages outputs from multiple trees, while in classification, it selects the most voted outcome. RF stands out for its accuracy, robustness, and capacity to handle large datasets without overfitting. However, only applying current ML algorithms may not ensure optimal performance. Several studies have introduced solutions to address this challenge, such as FS-GAN, a federated self-supervised learning architecture proposed to facilitate automatic traffic analysis and synthesis across diverse datasets [13]. By leveraging federated learning, FS-GAN coordinates local model training processes across different datasets, enabling decentralized systems like edge computing networks to train models on multiple devices without transferring data to a central server. This approach addresses privacy concerns, minimizes communication overhead, and facilitates efficient model training in distributed environments.

2.2. Unsupervised Learning

Unsupervised learning analyses datasets without predefined outputs, focusing only on the provided input. Its objective is to identify patterns and structures within unlabeled data. Unlike supervised learning, there are no provided outcomes, the system observes only input features and aims to cluster data accordingly [14,15]. Clustering is among the most widely used unsupervised learning methods. It splits the data into groups according to similarity. Data with similar features are placed in the same groups, while dissimilar points are placed in other groups. Clustering is mostly used in video-on-demand, marketing, smart grid, etc. Figure 4 illustrates the concepts of supervised, unsupervised, and reinforcement learning.

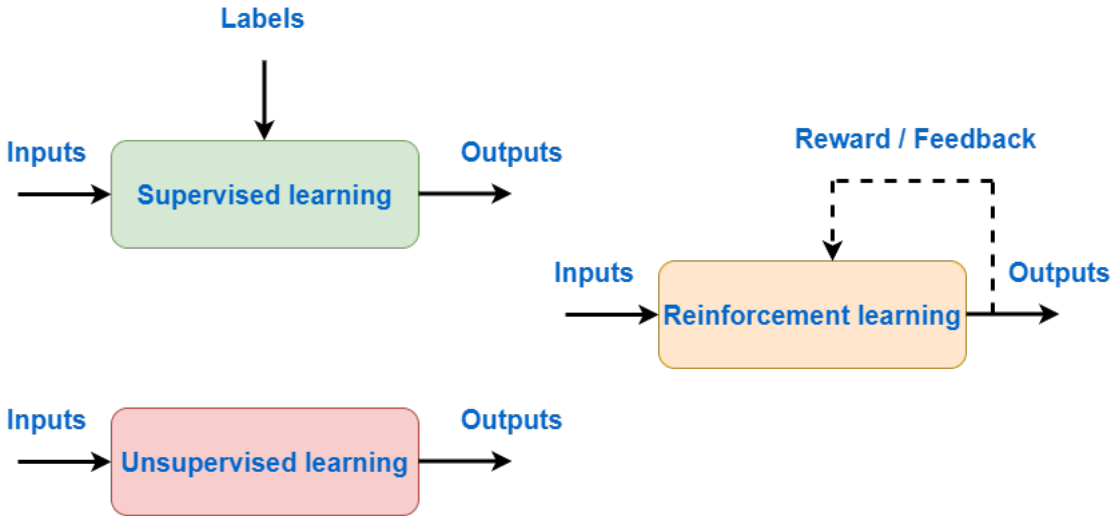

Figure 4. Supervised/Unsupervised/Reinforcement Learning.

2.3. Semi-Supervised Learning

Semi-supervised learning uses both labeled and unlabeled data, optimizing efficiency and accuracy. Labeled data is typically scarce and costly to obtain, requiring skilled human annotators. In contrast, unlabeled data is easier to gather. Leveraging both types, semi-supervised learning offers a compelling solution for enhancing algorithms with less human effort while achieving higher accuracy [16,17]. These models enhance unsupervised learning by leveraging small amounts of labeled data, offering simplicity and cost-efficiency compared to supervised learning. Generative models, Graph-based models [18], mixture models And EM, and semi-supervised support vector machines [16] are some of the algorithms used in semi-supervised learning.

2.4. Reinforcement Learning

Reinforcement learning relies on interaction with the environment, where machines learn via experiments in order to improve their actions relying on rewards or penalties [7]. Notably, Q-learning stands out as a prominent algorithm in this field [19,20]. Unlike other methods, reinforcement learning does not rely on data but rather on past experiences. It's widely applied in autonomous systems like race cars, robotic navigation, and AI gaming, forming a reward-based learning system. Various researchers have devised innovative solutions to deploy reinforcement-based models on the edge [21]. In this study, the author introduces a distributed edge intelligence sharing scheme aimed at improving learning efficiency and service quality in edge intelligence systems. This scheme allows edge nodes to enhance learning performance by exchanging their intelligence, tackling challenges like repetitive model training and overfitting caused by limited data samples. The approach is structured as a multi-agent Markov decision process, the scheme employs an ensemble deep reinforcement learning algorithm to optimize intelligence exchange among edge nodes [22].

2.5. Deep Learning

Deep learning approaches are well-suited for handling vast quantities of data and computationally intensive tasks such as image identification, speech recognition, synthesis, and others. As the demand for CPU power continues to rise, robust GPUs are increasingly being utilized to execute deep learning tasks. Deep neural networks, like artificial neural networks (ANNs), are created using deep learning. The term "deep" refers to the architecture of the network, which is characterized by numerous hidden layers [23]. Each deep learning model is made up of several layers. The input data is sequentially processed through the layers, with matrix multiplications performed at each layer. A layer's output

becomes the following layer's input. The last output from the final layer can be a feature representation or a classification result. When a model has multiple layers, it is called a deep neural network [24].

In deep learning, the number of layers is proportional to the number of features extracted. Unlike traditional methods, deep learning techniques automatically determine the features, so no feature calculation or extraction is required before using the method. With the advancement of DL, multiple network structures have been developed. CNN is a special case of the DNN family. CNN is performed when the matrix multiplications include convolutional filter operations [25]. CNN classifiers are commonly used for image classification, computer vision, and video analysis. Recurrent neural networks (RNN) are DNNs designed specifically for time series prediction, including loop connections within their layers to retain state information and facilitate predictions on sequential inputs, solving problems related to machine translation, audio data in speech recognition [26]. LSTM (long short-term memory) is a variant of RNN [27]. An LSTM unit comprises a cell, an input gate, an output gate, and a forget gate. These gates control the flow of information into and out of the cell, enabling the cell to maintain relevant information over time. The forget gate, in particular, can identify which information is retained and which is discarded. Designing a good DNN model for a specific application is difficult due to the large number of hyperparameters involved. During the design process, decisions often involve compromises on system metrics. For instance, a DNN model with high accuracy typically needs greater memory capacity to store all the model parameters compared to a model with lower accuracy. The choice of success metric depends on the application domain where DL is implemented. Islam et al. developed a custom CNN model for detecting tomato diseases, achieving an impressive 99% accuracy. Their model outperformed other CNNs based on the training time and computational cost. Additionally, they demonstrated the feasibility of running their model on drone devices. However, further analysis and experiments are required to effectively deploy their model on edge devices [28].

3. Edge Computing for IoT

Current computing and storage paradigms include cloud, fog, and edge computing. They are another important part of the internet of thing's architecture, the storage layer of a computing system for handling large amounts of data. A reliable data gathering, storage, computation, and analytic framework unfolds from the integration of these paradigms with the IoT.

3.1. Cloud Computing

Cloud computing is a centralized framework that provides on-demand computing resources such as servers, networks, storage, etc. [29]. The data must be transmitted to the data centers for additional processing and analysis in order to be usable, as illustrated in Figure 5. The cloud architecture is considered to have high latency and can congest the network because of its high load balancing. Since IoT's data are high, only cloud computing is no longer an option. Cloud computing can be divided into numerous types:

- **Software as a Service (SaaS):** Applications run on a service provider in the cloud, they are hosted, managed in a distant computer and connect to users via Internet.
- **Platform as a Service (PaaS):** Offers a cloud environment with all necessary resources to develop and build ready-to-use applications but without expense and hassle of purchasing and managing hardware or softwares.
- **Infrastructure as a Service (IaaS):** Offers computing resources to business, from servers to storage and networks. it is a pay-as-you-go, internet-based service model.

In the last few decades, the IoT has become increasingly prevalent, and the number of connected devices has grown exponentially. This has resulted in unprecedented volumes of data being generated by these devices.

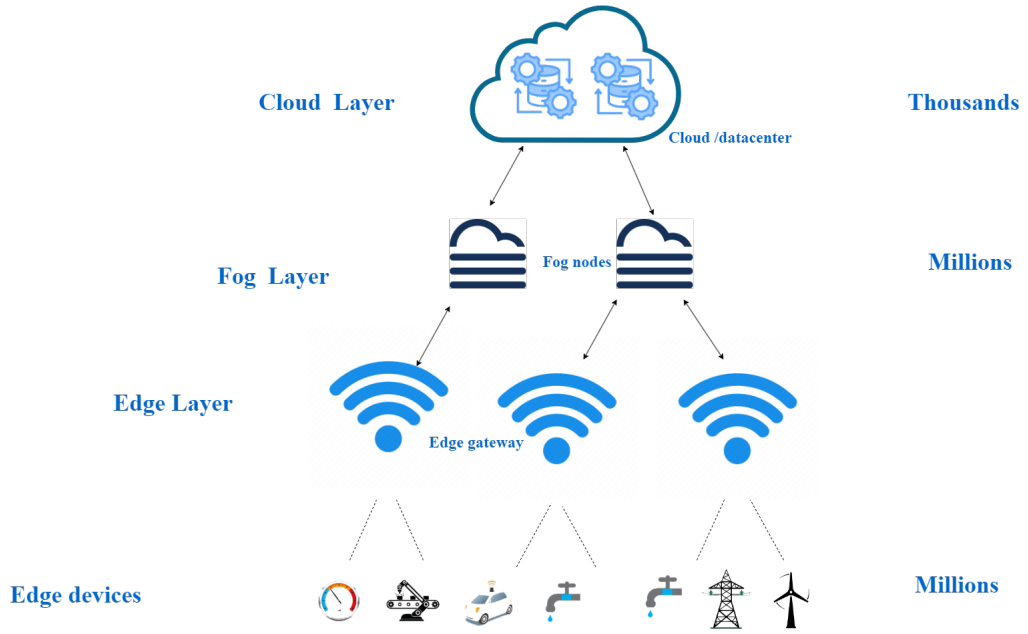

Figure 5. Fog and Edge Computing Layers.

3.2. Fog Computing

Fog computing is considered a shift from a centralized paradigm (CC) to a more decentralized system. It was created by CISCO back in 2012 [30] and refers to a virtualized platform that offers computation, storage, and communications services across edge devices and standard cloud computing data centers. These centers are often located at the network's edge, though not exclusively. Fog architecture consists of fog nodes (FNs) that can be placed anywhere along end devices and in the cloud. An FN can be constituted by various devices including switches, routers, servers, access points, IoT gateways, video surveillance cameras, or any other device equipped with processing capabilities, storage capacity, and network connectivity. This enables communication among the fog layer and end devices through a range of protocol layers and non-IP-based access technologies. The abstraction within the fog layer hides the complexity of various FNs offering a suite of services such as data processing, storage, resource allocation, and security. These functionalities are leveraged by a Service Orchestration Layer to efficiently distribute resources tailored to meet the distinct requirements of the end devices [31]. Integrating a fog layer into IoT-based systems to lower latency, conserve energy, and enhance real-time responsiveness has been the focus of various research studies. In the study by Singh et al. [32], a framework was introduced to track student stress and generate real-time alerts for stress forecasting. Data processing and cleaning were performed locally in the fog layer to minimize the workload in the cloud, followed by loading the data to the cloud layer for additional processing. The authors utilized multinomial Naive Bayes techniques to estimate emotion scores and categorize stress events. Additionally, they employed bidirectional Long Short-Term Memory (BiLSTM) for speech texture analysis and Visual Geometry Group (VGG16) for facial expression analysis.

3.3. Edge Computing

Edge computing, which was introduced prior to cloud computing, has gained substantial momentum, especially with the rise of the IoT. It enables IoT data computation and analysis at the network's edge, near the data source. Within IoT networks, the primary goal is to address critical needs such as security, privacy, low latency, and high bandwidth. Edge computing lowers the amount of data transmitted between nodes, resulting in cost savings and reduced network bandwidth consumption. Furthermore, edge computing serves as a complement to existing cloud computing platforms, enabling efficient data

processing. Depending on the IoT application, direct data processing on IoT devices may be necessary, supplementing centralized cloud infrastructure common in traditional setups. The concept of edge computing arises as processing capabilities are brought closer to end network elements. Edge computing and fog computing both involve processing data closer to where it's generated. While edge computing processes data directly on devices or within the local network, aiming to reduce latency and bandwidth usage by handling tasks locally before sending relevant information to centralized servers. Fog computing, on the other hand, extends this concept by deploying the FN at various points within the network, including both edge devices and traditional cloud data centers. This allows for a more distributed architecture, enabling efficient data processing and analysis across multiple network tiers. So, while edge computing is more localized, fog computing takes a broader approach, encompassing a wider range of network locations and resources. Edge intelligence extends the foundation of edge computing by directly integrating intelligent algorithms and models into edge devices. At the edge, it enables local data analysis, decision-making, and autonomous behavior. Edge intelligence enables real-time insights, reduced reliance on cloud resources, and improved autonomy in IoT environments by embedding machine learning and artificial intelligence capabilities on edge devices. It uses edge computing infrastructure to perform intelligent tasks and data processing, enhancing the overall capabilities of IoT networks. By bringing computational power and intelligence closer to the data source, combining edge computing and AI enables efficient and effective IoT deployments. It meets critical requirements like security, privacy, low latency, and high bandwidth while decreasing reliance on centralized cloud resources. This convergence enables smarter and more autonomous IoT systems, allowing devices at the network's edge to perform advanced analytics, decision-making, and real-time responses.

4. Implementing Machine Learning at the Edge

ML the edge requires several fundamental factors, including hardware, frameworks, model compression, and model selection. These elements have an impact on both the effectiveness and efficiency of edge-based AI applications. Hardware selection must strike a balance between computational power and energy efficiency while also taking into account specific use cases and resource constraints. In addition, model optimization techniques such as compression and selection are critical in adapting machine learning models for edge deployment.

4.1. IoT Hardware and Frameworks Employed in Edge Intelligence

IoT is a term that was presented in early 1999 by Kevin Ashton. It is an interconnection between physical world objects, embedded intelligent devices, and the internet. This is how the expression Internet of Things came to be in the first place. The key concept of IoT is connecting our day-to-day devices (cellphones, cars, lamps, etc.) via the internet to facilitate our work [2]. The connected devices have multiple types of sensors attached to them, such as temperature, speed, etc. to monitor and capture information from the physical world. Then data are sent to the cloud in order to be viewed, analyzed, and saved. IoT plays a crucial role in measuring and exchanging data between the physical world and virtual world. IoT devices must communicate with one another (the network or transmission layer in the 3-layer IoT architecture), the communication could be short or long range. Wireless technologies like Bluetooth, ZigBee, and Wi-Fi are suitable for short-range communication, while mobile networks like GSM, GPRS, 3G, 4G, LTEWiMAX, LoRa, Sigfox, CAT M1, NB-IoT and 5G are employed for long-range communication. Utilizing an IoT platform such as Mainflux, Thingspeak, Thingsboard, DeviceHive or Kaa to handle M2M communication and using protocols including MQTT, AMQP, STOMP, CoAP, XMPP, and HTTP has become common practice in IoT systems. However, with the rapid development of this new paradigm, our world is crowded with billions of smart IoT devices that generate a massive amount of data every day, hence the possibility to congest the network. In following section,

we will discuss the advancement in the IoT devices and frameworks that made it possible to embed complex ML models in resource-constrained devices.

4.1.1. IoT Hardware in Edge Intelligence: Design and Selection

One of the current challenges of integrating intelligence into IoT devices was the hardware's limited resources and its ability to handle on-device machine learning inference or training. That is why several companies have worked hard to develop an optimal hardware selection, considering factors like accuracy, energy consumption, throughput, and cost. The following section explains the hardware that has been specifically invented to deploy, build, and support ML at the network edge. To achieve high performance using a machine learning model, you will need rich GPUs that can handle big data with low latency. This is no longer the case since there are many efforts that have been made in the resource-constrained devices. Lately, we have noticed types of small devices that can support ML, even deep learning models. In the list below, we will introduce the most commonly used devices for deploying ML models at the edge.

The Raspberry Pi

Among the most commonly used devices for edge computing is the Raspberry Pi, a single-board computer designed by the Raspberry Pi Foundation [33]. It has been used to run machine learning inference without the need for additional hardware. The Raspberry Pi 3 Model B features a Quad Cortex A53 @ 1.2 GHz CPU, a 400 MHz VideoCore IV GPU, and 1 GB SDRAM. In comparison, the Raspberry Pi 4 Model 4 is an enhanced version, offering increased speed and power. It includes Gigabit Ethernet, built-in wireless networking, and Bluetooth capabilities. The Raspberry Pi 4 Model 4 boasts a Quad Cortex A72 @ 1.5 GHz CPU and supports up to 8 GB SDRAM.

NVIDIA's Jetson

NVIDIA Jeston is a new product developed by the Company NVIDIA fully dedicated for edge computing. the platform high-performance, low-power and energy efficiency makes it suitable for embedded applications and deploying Intelligence on the device. One of the popular Jetson products is the NVIDIA Jeston Nano Developer Kit developed by NVIDIA Corporation in Santa Clara, CA, USA [34], Jetson Nano is the entry-level board of the NVIDIA Jetson family. It is suitable for running Neural Networks calculations due to it's 128 Maxwell's GPU cores. it has 4 USB ports, a micro USB, an Ethernet port, two ribbon connectors, and a jack socket to provide additional power if needed. The Jetson Nano is compatible with all major DL frameworks, such as TensorFlow, pyTorch, Caffe, MXNet... Another Module from the Jetson family is Nvidia Jetson TX2 Series Jetson TX2, the Nvidia Jetson modules's medium-sized board, is larger than Jetson Nano. It proves useful when it comes to computer vision and deep learning. it has the Dual-Core NVIDIA Denver 2 64-Bit CPU and Quad-Core Arm® Cortex®-A57 MPCore processor. It is very suitable for running ML algorithms with its 256-core NVIDIA Pascal™ GPU. Jetson Tx2 proved be very efficient in real-time vehicle detection with high accuracy [35].

Nvidia Jetson AGX Xavier can function as a GPU workstation for edge AI applications with as little as 30 W power supply and delivering to 32 TOPs AI performance. Jetson AGX Orin Series are considered the most powerful in the Jetson modules, offering up to 275 TOPS and 8 times the performance compared to the previous generation for simultaneous AI inference. It is well-suited for a variety of applications, including manufacturing, logistics, retail, and healthcare, due to its high-speed interface that accommodates multiple sensors.

Arduino Nano 33 BLE Sense

Arduino Nano 33 BLE Sense is another AI enabled development board [36]. It can be considered one of the cheapest boards available in the market. The Nano 33 BlE Sense is simple and easy to use. It has an extensive library and resources which support TensorFlow lite, that offer the possibility to create your Ml model and upload it using

the Arduino IDE. Although, it is limited to executing a single program at a time, It has a built-in microphone, acceleromter and a 9axis inertial sensor that makes it excellent for wearable devices. Arduino Nano Board 33 is used various applications such as speech recognition [37] and Smart Health [38].

STM32 Microcontrollers

STM32 [39] is a family of 32-bit microcontroller integrated circuits produced by STMicroelectronics. This family includes various models based on different ARM Cortex processor cores, such as the Cortex-M33F, Cortex-M7F, Cortex-M4F, Cortex-M3, Cortex-M0+, and Cortex-M0. Despite the differences in the specific Cortex cores used, all STM32 microcontrollers are built around the same 32-bit ARM processor architecture. Each microcontroller has a processor core, static RAM, flash memory, a debugging interface, and various peripherals internally. In order to implement AI at the edge for the STM32, STMicroelectronics has developed specialized libraries for its devices, specifically the STM32Cube. The AI Toolkit allows pre-trained NNs to be integrated into STM32 ARM CortexM-based microcontrollers. It generates from Tensorflow and Keras STM32-compatible C code-provided NN models, as well as models in the standard ONNX format. STM32Cube has an intriguing feature to execute large NNs; it stores weights and activation buffers either in external flash memory or RAM.

SparkFun Edge

SparkFun Edge is the result of a collaboration between Google and Ambiq to develop a real-time audio analysis Dev-Board [40]. It is utilized for voice and gesture recognition purposes. A distinguishing feature of this board is the inclusion of the Apollo3 Blue microcontroller from Ambiq Micro, which features a 32-bit ARM Cortex-M4F CPU with direct memory access. Operating at 48 MHz CPU clock speed (96 MHz with TurboSPOTTM), and extremely low power consumption of 6 uA/MHz. Despite its power efficiency, it can easily run TensorFlow Lite, offering 384 KB of SRAM and 1 MB of Flash memory. This development board also features two microphones, a 3-axis accelerometer from STMicroelectronics, and a camera connector.

Google Coral Dev Board

Coral Dev [41] is designed for conducting fast on-device ML inference due to its integrated Google Edge TPU, which is an Application-Specific Integrated Circuit. Coral Development Board is a single-board computer with wireless capabilities for high-speed machine learning inference. It has a removable system-on-Module. The operating system on this board is Mendel, a Debian Linux variant. Supports TensorFlow Lite for model inference, Python and C++ as programming languages This development board is primarily used for image classification and object detection, but it can be used for a variety of other tasks. Working with this Dev Board is made easier by good documentation and support available online.

Beaglebone AI

The BeagleBone AI is a brand-new, high-end single-board computer. It's designed to help developers build ML and computer vision applications. This board is equipped by a Texas Instruments AM5729 system-on-chip. Powered by a Texas Instruments AM5729 system-on-chip, it features a dual-core C66x digital signal processor (DSP) and four embedded vision engine (EVE) cores. This configuration simplifies the exploration of machine learning applications in everyday life due to an optimized Texas Instruments Deep Learning (TIDL) framework and a machine learning OpenCL API with pre-installed toolkits. The BeagleBone AI is compatible with all popular machine learning frameworks, such as TensorFlow, Caffe, MXNet, and others. It is capable of executing tasks like image classification and object detection.

The selection of hardware platforms is crucial in determining the performance and capabilities of deployed solutions in edge computing and AI-enabled embedded systems. From widely accessible options like the Raspberry Pi, offering flexibility at a low cost, to specialized platforms like NVIDIA's Jetson series, providing high-performance computing tailored for deep learning tasks, each hardware option presents distinct benefits and limitations. For instance, while platforms such as Arduino Nano 33 BLE Sense offer simplicity and affordability, they may lack the computational power necessary for complex AI tasks. However, solutions like Google Coral Dev Board and BeagleBone AI excel in high-speed on-device ML computing but at a higher cost. Understanding these trade-offs is essential for selecting the most suitable hardware platform based on the specific needs of the application at hand, whether it be real-time audio analysis, computer vision, or any IoT deployments.

4.1.2. Edge Intelligence Frameworks and Libraries

Since the rapid explosion of ML-based applications and the unfeasibility of using only cloud-based processing, edge computing has gained momentum, leading to the deployment of most deep learning applications at the edge. As a result, new techniques to support DL models at the edge are required. This rising demand has prompted the creation of customized ML frameworks designed for edge computing architecture. These frameworks are critical in allowing the creation and deployment of lightweight models capable of operating effectively with limited resources. They also ensure efficient inference and learning processes in resource-constrained environments by optimizing model size, complexity, and computational requirements. This section classifies and explains the popular AI frameworks and libraries utilized in the reviewed research papers:

TensorFlow

TensorFlow stands out as one of the most commonly used open source frameworks for creating, training, evaluating, and deploying machine learning models [42,43]. It was developed by the Google brain team in 2011 and made open-source by 2015. TensorFlow flexible architecture enables developers to use a single API for distributing computation across multiple CPUs and GPUs, whether it be on desktops, servers, or mobile devices. This adaptability makes it particularly well-suited for distributed machine learning algorithms. TensorFlow supports a wide range of applications [44], with a focus on deep neural network [45]. It runs on Windows, Linux, Mac and even Android. its main programming languages are Python, Java and C.

TensorFlow Lite is a lightweight version of TensorFlow for resource-constrained devices. It enables on-device learning and makes ML run everywhere with low latency. TensorFlow light is designed for on-device inference rather than training purposes. It achieves reduced latency by compressing the pre-trained DNN model.

OpenEI

OpenEI is an open lightweight framework for edge intelligence [46]. OpenEI allows performing AI computations, data sharing capabilities on low-power devices suitable for deployment at the edge. OpenEI is made up of three parts, a package manager for running real-time DL tasks and training models locally, a model selector for choosing the best models for various edge hardware, and a library with a RESTful API for data sharing. The objective of OpenEI is to ensure that once deployed, any edge hardware will have intelligent capabilities.

Core ML

Core ML is Apple's ML framework. It enhances on-device performance by integrating the CPU, GPU, and Neural Engine while minimizing memory usage and power consumption [47]. Executing the full model on the user's device avoids the necessity for a network connection, safeguarding the privacy of the user's data and ensuring responsiveness of your

app. Core ML supports a large selection of ML algorithms like ensemble learning, SVM, ANN, and linear models. However, Core ML does not support unsupervised models [48].

Caffe

Caffe is a fast and flexible DL framework created by the Berkeley Vision and Learning Center (BVLC), with contributions from a community of developers on GitHub. Caffe enables research projects, big manufacturing applications, and startup prototypes in vision, speech recognition, robotics, neuroscience, astronomy, and multimedia [49]. Caffe is a BSD-licensed C++ library that supports the training and deployment of CNNs and other DL models. It also integrates with Python/Numpy and MATLAB. Caffe2 a new lightweight version of Caffe, was designed to facilitate data processing with DL directly on end devices. Caffe2 incorporates numerous new computation patterns, such as distributed computation, mobile, and reduced precision computation, expanding upon the capabilities of the original Caffe framework. With cross-platform libraries, Caffe2 facilitates deployment across multiple platforms, enabling developers to harness the computing power of GPUs in both cloud and edge environments [50]. It's worth noting that Caffe2 has been integrated into PyTorch.

Pytorch

PyTorch is a popular open-source library for building DL models. It was developed by Facebook. This package uses dynamic computation and uses Python concepts [51]. PyTorch carries PyTorch Mobile, which supports the execution of ML models on edge devices, mainly iOS and Android [52]. Both PyTorch and Caffe2 have their own set of benefits. Caffe2 is more towards industrial usage with a huge emphasis on mobile, whereas PyTorch is oriented toward research. In 2018, the Caffe2 and PyTorch projects came together to form PyTorch 1.0. This new framework combines the user experience of the PyTorch frontend with the scaling, deployment, and embedding capabilities of the Caffe2 backend. For the aforementioned reasons, simplicity and ease of usage are what make PyTorch popular among deep learning researchers and data scientists.

Apache MXNet

MXNet [53] is a deep-learning library that is both effective and versatile. It was created by the University of Washington and Carnegie Mellon University to support CNN and LSTM networks. It works by combining symbolic expression with tensor computation to maximize efficiency and flexibility. It is built to run on a variety of platforms (cloud or edge) and can perform training and inference tasks. It also supports the Ubuntu Arch64 and Raspbian ARM-based operating systems, in addition to the Windows, Linux, and OSX operating systems. MXNet is a lightweight library that can be integrated into various host languages and operate within a distributed setup. Several other projects have been published by some of the industry's top-leading tech giants to move ML functions from the cloud to the edge, like Azure IoT Edge from Microsoft and AWS IoT Greengrass from Amazon.

The need for implementing machine learning models at the edge continues to rise, which has encouraged the development of customized frameworks specifically designed for edge computing architectures. These frameworks are pivotal in facilitating the creation and deployment of lightweight models capable of operating effectively within resource-constrained environments. One of the most often used AI frameworks and libraries in modern literature is TensorFlow; it stands out for its flexibility and cross-platform compatibility. TensorFlow Lite, a lightweight version, specifically targets resource-constrained devices, ensuring low-latency on-device computing. OpenEI, an open lightweight framework, supports AI computations and data sharing capabilities on low-power devices deployed at the edge, aiming to integrate any edge hardware with intelligent capabilities post-deployment. Core ML, Apple's proprietary framework, optimizes on-device performance while maintaining user data privacy and application responsiveness. Furthermore, Caffe and its

lightweight successor, Caffe2, serve as fast and flexible deep learning frameworks suitable for various applications, from research projects to industrial prototypes. PyTorch, favored by deep learning researchers and data scientists, boasts dynamic computation and native support for edge deployment via PyTorch Mobile. Apache MXNet, with effectiveness and adaptability, is capable of supporting a wide range of platforms and tasks, from CNNs to LSTM networks, positioning itself as a lightweight library amenable to deployment in distributed environments. These frameworks, alongside cloud-to-edge migration tools like Azure IoT Edge and AWS IoT Greengrass, show how researchers and companies are working together to make edge devices smarter with AI, thereby revolutionizing the landscape of edge machine learning. A comparison of multiple frameworks and libraries for edge intelligence is presented in Table 1.

Table 1. A Comparison of Frameworks and Libraries for Edge Intelligence.

Framework Library	Development Language	Edge Device Requirements	Open Source	Task	Applications
TensorFlow Lite	C++, Java	Mobile Embedded Device	Yes	Inference	Computer Vision [54], Object Detection [55].
Caffe2	C++	Multiple Platforms	Yes	Training and Inference	Image Analysis [56], Video Analysis [57].
Core ML	Python	Apple Devices	No	Inference	Image analysis [58]
MXNet	Python, C++	Multiple Platforms	Yes	Training and Inference	Image Recognition [59] Text Recognition [60]
PyTorch	Python	Multiple Platforms	Yes	Training and Inference	Image Recognition [61] Text Recognition [62]
AWS IoT Greengrass	Python, Node.JS, Java, C and C++	Multiple Platforms	Yes	Inference	Precision Agriculture [63], Autonomous [64] Vehicles [65]
Edge2Train	C++	MCUs supported by Arduino IDE	Yes	Training and Inference	Video Analysis [66]
OpenEI	–	Multiple Platforms	Yes	Training and Inference	Various Applications
TensorRT	C++	NVIDIA GPU	No	Inference	Image Classification [67]
DeepThings	C/C++	Multiple Platforms	Yes	Training and Inference	Object Detection

4.2. Model Compression

Model compression involves lowering the size and computational demands of ML models without compromising their performance. It enables us to deploy the model on tiny devices. Model compression techniques allow the implementation of resource-intensive DL models in resource-limited edge devices by minimizing the model's parameter count or training DL models scaled down from their original size. Because edge devices have limited storage and processing power, compressing ML models is critical to ensuring efficient deployment. There are several techniques used in edge intelligence for model compression, like pruning, quantization, knowledge distillation, and low-rank factorization [68]. The two main ways for lowering the model size while maintaining a high accuracy are lower precision (fewer bits per weight), which is known as quantization and fewer weights, which is known as pruning. Post-training quantization reduces memory needs and execution speed while maintaining accuracy. Using ML frameworks like TensorFlow and Keras, this technique can be used both after and before training. Furthermore, pruning algorithms in neural networks remove redundant connections, reducing computing demands and program memory. Combining quantization and pruning approaches provides

a comprehensive model compression solution that optimizes both size and performance for deployment on smaller devices [69]. Huang et al. introduced DeepAdapter, a cloud-based-edge-device framework aimed at optimizing DL models for mobile web applications. Within DeepAdapter, a context-aware pruning algorithm is employed to reduce model size and computational demands while preserving performance. This adaptive approach replaces fixed network pruning methods, ensuring that pruned models are finely tuned to current resource constraints. Through the pruning process, redundant parameters or connections are selectively eliminated from the deep neural network models. Resulting in a more streamlined model that is better suited for deployment on resource-constrained devices, ultimately improving efficiency and performance in mobile web environment [70]. These authors propose an iterative algorithm that optimizes the partitioning and compression of a base DNN model. By dynamically adjusting strategies based on rewards, the approach efficiently maximizes performance while meeting resource constraints [71]. This study also validates that pruned models demonstrate accelerated inference and reduced memory usage by introducing GNN-RL, a pruning method that combines graph neural networks (GNNs) and reinforcement learning for topology-aware compression [72].

4.3. Architecture at the Edge (Where to Implement ML in IoT Architecture?)

Deploying ML algorithms in IoT devices was nearly impossible due to their limited computation power and small memories. The typical solution suggests offloading data processing to the cloud, but this leads to greater latency, privacy issues, and decreasing bandwidth. To resolve these issues, many research studies have been made about distributing machine learning algorithms on four important architectures as we can see in the Figure 6. Deploying machine learning algorithms in IoT applications is used in a variety of fields; we can organize the important applications as follows:

- **Smart Health:** To enhance patient well-being, innovative devices have emerged. For instance, adhesive plasters equipped with wireless sensors can observe wound status and transmit data to a doctor remotely, eliminating the necessity for the doctor's physical presence. Additionally, wearable devices and tiny implants can track and relay various health metrics such as heart rate, blood oxygen levels, blood sugar levels, and body temperature. Notably, there are sensors designed to forecast health events, like seizures. For instance, a wearable device mentioned in a study by Samie et al. [73] predicts epileptic seizures, alerting the patient beforehand.
- **Smart Transportation:** By leveraging in-vehicle sensors, mobile devices, and city-installed appliances, we can provide improved route recommendations, streamline parking space reservations, conserve street lighting, implement telematics for public transportation, prevent accidents [74], and enable autonomous driving.
- **Smart Agriculture:** Using sensors and embedded devices for soil scanning and water, light, humidity, and temperature control. Introducing intelligence and IoT devices to farmers to enhance crop quality and yield while optimizing human labor [75,76].
- **Surveillance Systems:** Smart cameras can collect video from multiple locations on the street. Smart security systems can identify suspects or prevent dangerous situations with real-time visual object recognition.
- **Smart Home:** Conventional household appliances, such as refrigerators, washing machines, and light bulbs have evolved by integrating internet connectivity, enabling communication between devices and authorized users. This connectivity enhances device management and monitoring while optimizing energy consumption rates. Moreover, the availability of smart home sensors introduces features such as smart locks and home assistants, further enhancing the functionality and convenience of modern homes.
- **Smart Environment:** Wireless sensors dispersed all over the city offer the ideal infrastructure for monitoring a wide range of environmental conditions. Enhanced weather stations can leverage barometers, humidity sensors, and ultrasonic wind

sensors. Moreover, intelligent sensors can oversee the city's air quality and water pollution levels [77].

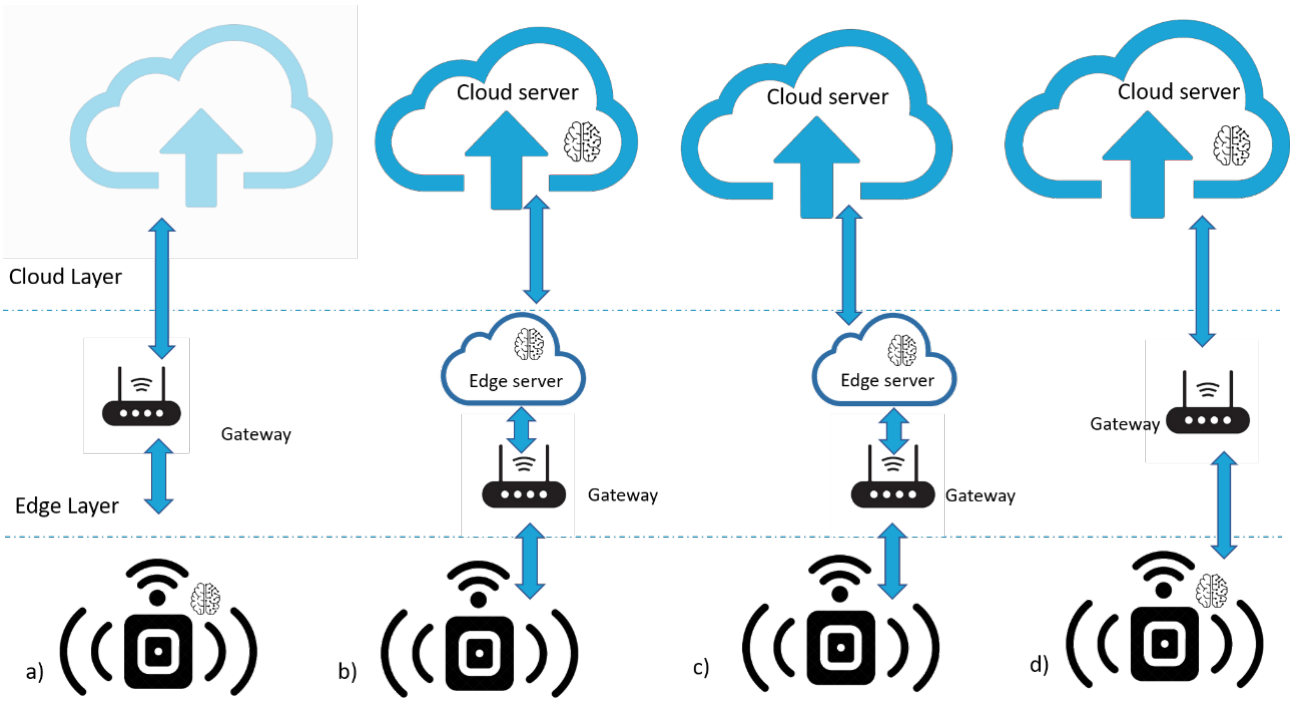

Figure 6. (**a**) On-Device Computation, (**b**) Distributed Edge Server-based Architecture, (**c**) Edge Computing, and (**d**) Joint Computation between the Cloud and the Device.

There are numerous other machine learning based IoT applications and various examples of how ML could be applied in these fields. It is explained in detail in many different other reviews. In this paper, we have decided to review and classify the integration of machine learning algorithms in IoT devices based on the location of the ML model in the architecture previously discussed, which is a new concept as far as we know. While writing this review, only English papers were taken into account. Furthermore, the reviewed papers were published between 2017 and 2024 to analyze the recent advancements in this field. The most advanced machine learning applications in IoT are categorized by application domain, input data type, machine learning techniques used, and where they fall on the cloud-to-things range.

4.3.1. On-Device Intelligence

ML models were primarily deployed on robust devices such as computers, servers, and specialized hardware. However, recent advancements have enabled the implementation of ML training and inference on low-power chips. These advancements involve both innovative hardware designs and software frameworks to extend the intelligence to the edge. On-device intelligence, also known as embedded intelligence, empowers devices to perform data processing, machine learning, and artificial intelligence tasks directly without relying on cloud or external server resources. This method provides numerous advantages, including lower latency, enhanced privacy, security, and improved performance in scenarios where network connectivity is limited or unreliable. In this study, Faleh et al. demonstrate the feasibility of deploying CNN models on embedded devices [78]. They used the Jetson Nano board to enable real-time mask recognition directly at the edge device. Utilizing a lightweight CNN classifier model, MobileNetV2, trained on a central server, they deployed it on the embedded Jetson Nano device linked to an alarm system to detect missing masks. The authors achieved an accuracy rate over 99% according to their experimental results. In this work [79], the authors presented a cost-effective intelligent irrigation

system designed to forecast environmental factors using embedded intelligence. They conducted a comparative analysis across various frameworks, including the deployment of an LSTM/GRU-based model on a Raspberry Pi board. Their findings highlight the accuracy of LSTM and GRU in predicting environmental factors. Bansal et al. [74] introduced DeepBus, a pothole detection system leveraging IoT to identify surface irregularities on roads in real-time. Road data is collected using a Raspberry Pi 3B+ equipped with an accelerometer and gyroscope. Then, the data undergoes labelling, preprocessing to remove missing data, and outlier detection. The authors evaluated and trained various ML models, finding that the RF classifier reached an accuracy of 86.8% for pothole detection on the collected dataset. After these initial phases, the model is implemented on smartphones, where it continuously collects data from the phone's sensors and detects potholes and bumps in real-time. This live data on potholes is made accessible to all users via a real-time map to enable smart transportation. Drivers can receive warnings based on this information, and their locations can be shared with civic authorities for prompt repair of road damages. Collecting and sharing real-time road data with users and civic authorities raises privacy and security concerns, posing significant challenges in protecting sensitive information and ensuring compliance with data protection regulations in IoT-based systems. Samie et al. [73] offered a new efficient algorithm that predicts epileptic seizure on IoT devices with limited resources. In fact, all the related work on epileptic seizure prediction applies SVM, Random Forest and CNN classifiers. However, based on the authors, these algorithms are not suitable for implementation on small, low-power portable IoT devices due to their complex features and higher computation processing requirements. Therefore, the authors proposed a new seizure prediction model. The EEG data which records electrical activity in the brain and can be measured with wearable or implantable sensors, are pre-processed (filtered and segmented) on the MSP432 IoT device. Then, features are extracted and selected to be passed to the logistic regression classifier, which is commonly used in seizure prediction (due to its light computation and low memory). After that, the remaining features, along with the logistic regression model output, are sent to the gateway (smartphone device also used by the patient) to eXtreme Gradient Boosting (XGBoost) for classification and post-processing. While the proposed approach achieves high accuracy and efficiency on constrained devices, it simplifies features and reduces the size of data segments, which may limit the model's effectiveness in real-world epileptic seizure detection scenarios. Balancing the need to keep models lightweight for deployment on resource-constrained IoT devices while ensuring sufficient complexity and robustness to accurately detect seizures in diverse real-world scenarios is equally important. Various solutions have been developed to handle this challenge. For instance, this study utilized quantization to develop a lightweight CNN model capable of capturing complex EEG signal features while maintaining high performance in environments with limited computational resources [80]. Transfer learning techniques have emerged as a solution to reduce the need for intensive feature engineering, thus resulting in a lighter model. In this study, the authors developed a system for detecting wheat rust [81]. Utilizing a CNN model based on transfer learning on an NVIDIA Jetson Nano, they achieved on-device inference. Their smart edge device comprises a camera for capturing crop images, a trained ResNet-50 for disease classification, and a touchscreen interface to display results to farmers. This system aids farmers in monitoring crop health, facilitating swift and accurate disease identification in real-time. On-device intelligence holds the potential to significantly transform numerous fields by allowing real-time, resource-efficient data analysis and decision-making on IoT devices. By processing data locally, on-device intelligence reduces latency, enhances privacy and security, improves reliability in low-connectivity environments, optimizes resource utilization, and enables scalability and flexibility in IoT ecosystems.

While numerous hardware and software solutions exist for edge ML, designing and deploying ML models on end devices remains a complex task. To deploy an AI model effectively on embedded devices, researchers and developers must carefully consider factors such as hardware selection. ML and DNN models demand significant computational

resources due to intensive linear algebra operations and large-scale matrix/vector multiplications, rendering traditional processors inefficient. As a result, various novel specialized processing architectures and methodologies have emerged to address these challenges and achieve goals such as compact size, cost-effectiveness, reduced power consumption, low latency, and enhanced computational efficiency for edge devices. The optimal hardware for integration into edge devices encompasses a range of options tailored to specific applications. These include Application-Specific Integrated Circuits (ASICs), Field-Programmable Gate Arrays (FPGAs), processors based on Reduced Instruction Set Computing (RISC), and embedded Graphics Processing Units (GPUs) [82]. A notable advancement in IoT hardware is the emergence of ultra-low-power AI accelerators, executing ML and DL tasks directly on the chip or device through parallel computation. Strategies like model design and compression, as well as techniques to reduce inference time on end devices, are crucial for successful AI model deployment on embedded devices.

4.3.2. Edge Intelligence

After discussing the implementation of ML techniques on device computing, we move forward where the processing and the algorithm's execution would be on the edge layer. Edge intelligence broadens device intelligence to a wider concept. While device intelligence relates to the device's ability to process and analyze data locally, such as on a smartphone, tablet, or IoT device, edge AI refers to the ability to process and analyze data near its source, typically at the edge, instead of relying on centralized cloud-based services. Edge AI includes not only individual devices, but also computing nodes or servers strategically located near data sources. In this study, the authors explore the use of edge computing in an intelligent aquaculture system [83]. They implemented a heterogeneous architecture consisting of various devices such as NVIDIA Jetson Nx and Jetson Nano to collect water sensor data and to perform real-time video-based fish detection using DL. Wardana et al. [77] introduced a new DL model that integrates one- dimensional CNN and LSTM network to anticipate a short-term hourly PM2.5 concentration at 12 distinct nodes according to a dataset by Zhang et al. [84]. Then, the CNN-LSTM was optimized with TensorFlow Lite to generate a lightweight model suitable for edge devices, such as the Raspberry Pi 3 Model B+ and Raspberry Pi 4 Model B boards. After comparing their proposed model to approximately 20 different deep learning methods, the authors concluded that their hybrid CNN-LSTM model surpasses other models. Proietti et al. [75] developed an edge intelligence approach to manage a Greenhouse. In this work, the authors presented an LSTM Encoder-Decoder-based system in a greenhouse to detect anomalies in plants and manage their growth and control equipment. The system layout in the greenhouse included an Arduino MKR that housed all the sensors responsible for environmental data collection (pressure, humidity, air temperature...), a Full-HD camera, and an Arduino Uno equipped with the actuators and an LED lamp to provide light for taking pictures. A Raspberry PI 4 model B and NVIDIA Jetson Nano as edge servers, which are connected to both the Arduino MKR and Arduino Uno to receive sensor data (MKR) and send control instructions (Uno). The images are taken and processed using ELA (Easy Leaf Area) in order to calculate the Leaf Area. The authors feeds the LSTM model the temperature, relative Humidity, pressure, light Intensity, UVA and Leaf Area data after pre-processing and reshaping into a suitable format for LSTM. After tuning and training the model in both NVIDIA and Raspberry Pi, it was obvious that with one LSTM layer and 64 LSTM units, they obtained the best results. Jetson Nano may be faster than Raspberry Pi. However, Nano was not capable of completing the analysis of different hyperparameter choices. Rumy et al. [76] presented an automatic rice leaf disease detection system that applies a lightweight ML model based only on edge computing. In order to make the image classification model, the authors experimented with several machine learning algorithms to pre-process images of healthy and infected leaves, but they concluded that the supervised learning random forest is the most efficient. After training, they exported and transformed the image classification model to Raspberry Pi edge device, which was used to classify and detect the healthy

leaves from the infected ones. They achieved an accuracy rate of 97.50%, and using this method can reduce data transmission costs by about 86.66%, which in turn lowers latency. Using edge computing, the authors in this study in [85] presented a novel approach to online water quality monitoring and early warning. They presented an advanced version of the backpropagation neural network (BPNN), enhanced by a new hybrid optimization approach that combines the cuckoo search algorithm and the Nelder-Mead simplex method. This novel method successfully adjusted the BPNN's weight and deviation parameters in an effort to increase the system's precision and dependability when assessing water quality. However, more improvements are needed to reach a better level of accuracy.

The research papers presented in this section demonstrate the significance of combining machine learning models with edge computing to achieve efficient and real-time data processing in a variety of IoT applications. The studies show that hybrid deep learning models can be successfully implemented and optimized on the edge, demonstrating the advantage they offer over other methods. Researchers have achieved accurate and resource-efficient solutions for anticipating environmental factors, managing greenhouse conditions and detecting plant diseases, as well as unmanned aerial vehicle (UAV) applications, by harnessing edge-device intelligence. For instance, in a recent study by Liu et al. [86], they highlighted the importance of integrating edge computing into vehicular networks to enhance traffic safety and elevate travel comfort. By incorporating Mobile Edge Computing (MEC) into vehicular networks, the study addressed the challenges and opportunities of shifting cloud resources closer to the edge of the network. The research provided an extensive review of advancements in Vehicular Edge Computing (VEC), covering areas such as task offloading, caching mechanisms, data sharing protocols, flexible network management strategies, and security and privacy considerations. Despite the significant progress made in VEC, several issues remain unresolved. These include enhancing Quality of Service (QoS), improving scalability, and strengthening security and privacy measures. Addressing these challenges opens up avenues for further research and innovation in the field of VEC.

4.3.3. Edge-Cloud Joint Computation

In a joint computation between the cloud and the edge layers of a network, certain tasks or computations are carried out locally at the edge, while others are offloaded to centralized cloud servers for more intensive processing or analysis. This strategy maximizes resource utilization by using the advantages of both edge and cloud computing. For example, data preprocessing or initial analysis may occur at the edge for rapid response, while complex analytics or long-term storage take place in the cloud. Joint computation enhances system efficiency, scalability, and flexibility, making it useful for applications that require a balance between local processing and centralized resources, such as smart cities, intelligent transportation systems, or distributed surveillance networks. In this study, they introduced an innovative architectural framework to connect smart monitoring robotic devices in healthcare facilities [87]. Their framework consists of three layers: the IoT node layer, the edge layer, and the cloud layer. Then, data is collected and routed directly to the edge node, and from there, it is offloaded to the cloud server for primary data processing. However, the authors need to enhance the data privacy and security of the framework. Transmitting data from the edge node to the cloud server raises concerns, as without robust encryption and security measures in place, sensitive healthcare data could be susceptible to unauthorized access or breaches. Li et al. [88] proposed a novel approach using deep learning in a video recognition IoT application with edge computing to ensure privacy and minimize network traffic and data processing load in the central cloud. Multiple wireless video cameras monitor the environment and identify objects. The wireless cameras record video at a resolution of 720p with a bitrate of 3000 kilobits per second. the collected video's raw data is offloaded to the edge layer (which is deployed in an IoT Gateway) via standard Wi-Fi connections to be pre-processed. Then, the edge nodes would upload the reduced intermediate data to the cloud servers for further processing using CNN models. In this

study, the CNN model is divided into two parts, the edge servers computed the lower layers and the cloud computed the higher layers. Nevertheless, the system's scalability may be limited by the capacity of edge nodes and cloud servers to handle increasing data volumes and computational tasks. As the number of IoT devices and video streams grows, the ability of the system to effectively scale to accommodate these demands may be limited. Sivaganesan et al. [89] developed an innovative approach "semi self-learning farm management" based on edge computing instead of cloud processing paradigm to improve speed, reduce security risks, costs and latency. The wireless environmental sensor collects data on temperature, humidity, soil moisture, wind speed, and pressure, while the Tetracam-ADC Lite camera mounted on the drone monitors crop growth and pest attacks. This data is then transmitted to the edge layer via Wi-Fi Max for processing by the H_2O deep learning model. H_2O uses the ISTAT (national institute of statistics) dataset for training, then it predicts the right time to sow, to reap, to water and fertilize the crops to avoid any damages. All processed information from the edge layer is transmitted back to the farmer's portable device via Wi-Fi. Sajjad et al. [90] created a face recognition algorithm for police departments in a smart city. A mobile wireless camera attached on a police officer's uniform is utilized to capture images, which are then transmitted to a Raspberry Pi3 for facial recognition. The Viola-Jones algorithm is employed for face detection in images, followed by the ORB algorithm (Oriented FAST and Rotated BRIEF), which extracts peculiarities from the faces and offloads the data to an SVM classifier in the cloud to identify people and potential criminals. Yet, the accuracy of facial recognition algorithms can be affected by various factors such as lighting conditions, image quality, and occlusions. Errors in face detection and feature extraction may lead to misidentification or false positives, which could have serious consequences in law enforcement applications. These authors proposed an online water-quality monitoring system within the water distribution system utilizing both edge and cloud computing. Through testing various scenarios including edge computing alone and cloud computing alone, they introduced a hybrid edge-cloud framework. This hybrid model demonstrated improved accuracy in classification models while maintaining low energy consumption rates. The research showcases the practical optimization and viability of integrating edge and cloud computing in water quality monitoring systems. However, there is potential for further reduction in power consumption [91]. Distributed edge-cloud computing paradigm has demonstrated its potential in a variety of IoT applications, delivering numerous benefits such as enhanced privacy, reduced network traffic, and improved real-time processing capabilities. The research studies discussed show that this architecture can be successfully implemented in video recognition, farm management, and Smart City face recognition, effectively addressing specific challenges in each domain. This approach has the potential to transform a wide range of IoT applications by delivering more efficient, secure, and scalable solutions.

4.3.4. Cloud-Device Joint Computation

While on-device computing, edge intelligence, and edge-cloud computing offer significant advantages, each approach has unique advantages and addresses specific challenges. However, in today's interconnected and data-driven world, the demand for seamless integration and efficient utilization of computing resources is more pressing than ever. This is where the concept of joint computing comes as a compelling solution. By combining the localized processing power of devices with the vast computational capabilities of the cloud, joint computing fills the gap between edge and cloud computing, unlocking several benefits. Several researchers have applied various techniques to make this approach feasible. In this paper, the authors used a combination of cloud and device intelligence in smart transportation [92]. They created an autonomous vehicle system by collected data from various sensors and implemented a motion control system on an STM32F10 microcontroller. They also developed an intelligent detection system in the same vehicle by deploying models on the embedded device EAIDK-310, which included a camera module. Additionally, they utilized cloud services for data transmission and information exchange,

enabling interaction with the collected data. Mrozek et al. [93] designed a new system called Whoops which consists of two main elements: an iPhone 8 mobile IoT device equipped with a triaxial-accelerometer, a triaxial-gyroscope and a local ML model (built using coreML library for iOS devices) that can sense the surroundings, detect falling in older people, pre-process the data and notify the caregiver in case of danger. then gather, analyse and transmit the data for further processing. Also, a datacenter where the collected data is analysed and processed in order to monitor numerous elderly people. After comparing these 4 classifiers: RF, ANN, SVM, and Boosted Decision Tree (BDT), the authors came across that BDT is the more suitable one due to its superior average accuracy, moderate precision, and minimal standard deviation. Which was chosen for both cloud and mobile processing. In order to validate these tests, the authors conducted a real-life experiments to confirm the accuracy of the classification. They conducted an experiment involving a participant, a 25-year-old equipped with a mobile device containing the Whoops app. The participant performed a series of falls in different directions, along with various daily activities such as walking, squats, and navigating slopes. The simulation involved falling into a pool of sponges to reduce the danger of injury. They recorded just one false positive in seventeen fall trials. In order to address the issue of air pollution. In this study, they proposed an IoT-Cloud-based model for forecasting the Air Quality Index [94]. The IoT devices were utilized to collect and preprocess real-time air pollution data, including various meteorological parameters such as temperature, humidity, and wind speed. These data were then sent to the cloud environment for further processing and analysis using LSTM and eXtreme Gradient Boosting. Barnawi et al. [95] introduced an IoT- UAV-based scheme that detects, track and notify covid- 19 cases using aerial thermal imaging. These authors used thermal onboard sensors to collect raw data (thermal images), captured in a real-time scenario from thermal infrared cameras in a large crowd or massive cities. The UAV performs preprocessing using the face recognition method to determine whether the person has an elevated temperature or not. The collected data are then transmitted to AI models (YOLO and DLiB) on a central server for temperature calculation, identification, and face mask detection. Five different machine learning classifiers SVM, KNN, XGB, LR, and MLP models were used to evaluate the proposed method.

Even though cloud-device joint computation provides numerous benefits, it also comes with its own drawbacks. One significant drawback is increased latency due to the need to offload data between devices and the cloud for processing. This latency can impact real-time applications that require immediate decision-making. Additionally, joint computing may raise concerns about data privacy and security, as sensitive information must be transmitted over networks to cloud servers. Furthermore, reliance on cloud infrastructure introduces dependency on external services, making the system vulnerable to outages or disruptions in internet connectivity. A full comparison of multiple types of machine learning methods and their applications in IoT is presented in Table 2.

Table 2. A Review of Machine Learning Algorithms and their Application in IoT Architectures.

Paper	Application Domain	ML Model	Framework / Hardware	Architecture	Benefits	Drawbacks
[74]	Smart Transportation	Random Forest	Raspberry Pi 3B + Smartphone	On-Device Computing	Reduced Delay	Low Privacy
[73]	Smart Health	Logistic Regression and XGBoost	MSP 432 and Smartphone	On-Device Computing	High Accuracy and Reduced Data Transmission	Requires Balancing model Complexity with Lightweight IoT Deployment
[77]	Smart Environment	CNN-LSTM	TensorFlow Lite with Raspberry Pi model B+ and Raspberry Pi 4 Model B	Edge Layer Computing	Lightweight Suitable for Edge Deployment	Accuracy Degradation

Table 2. *Cont.*

Paper	Application Domain	ML Model	Framework / Hardware	Architecture	Benefits	Drawbacks
[75]	Smart Agriculture	LSTM and Encoder Decoder	TensorFlow with Arduino (Mkr and Uno), Raspberry Pi 4 Model B and NVIDIA Jetson Nano	Edge Layer Computing	Higher Prediction and Low Time Consumption	High Complexity in Real-world Deployment
[76]	Smart Agriculture	Random Forest	Raspberry Pi	Edge Layer Computing	High Accuracy and Reduced Data Transmission Costs and Latency	Limited Scalability
[88]	Smart Environment	CNN	Caffe with Intel Core i7 7770 CPU and NVIDIA Geforce GTX 1080 graphic card	Distributed Edge and Cloud Computing	Enhanced Privacy and Reduced Network Traffic	High Computational Cost and Limited Scalability
[89]	Smart Agriculture	H20	Drone enabled with a Tetracam ADC lite camera	Distributed Edge and Cloud Computing	Reduced Security Risks and Low latency Compared to Centralized Systems	High Computational Cost
[90]	Surveillance Systems	SVM	Wireless camera on a Raspberry Pi	Distributed Edge and Cloud Computing	Enhanced Security	Accuracy my Degrade
[93]	Smart Health	Boosted Decision Tree using local ML model	Core ML with IPhone 8	Distributed Device and Cloud Computing	Enhanced Safety and Real-time detection	Further Validation is Required to Ensure Accurate Classification
[95]	Smart Health	SVM,KNN, Yolo3 and DLIB with ImageNet	keras library with UAV thermal and infrared cameras	Distributed Device and Cloud Computing	Detection with Reduced Response Time	Accuracy Could be Improved
[96]	Smart Health	ANN, CNN and RNN	Mobile Phone and sensors	On-Device Computing	High Accuracy and Real-Time Analysis	Does not Address IoT Device Diversity
[97]	Smart Health	Reinforcement Learning	Wearable devices and sensors	Edge Layer Computing	Improved Data Privacy	High Model Complexity
[78]	Smart Health	MobileNetV2 (CNN)	TensorFlow, Keras and OpenCV with NVIDIA Jetson Nano and Logitech USB camera C920e	On-Device Computing	Reduced Load to Cloud and Low System Cost	High Model Deployment Complexity
[98]	Smart Agriculture	LSTM	TensorFlow and Keras with Raspberry Pi 4 Model B	On-Device Computing	Load Reduction	Lower Accuracy
[81]	Smart Agriculture	ResNet-50 (CNN)	TensorFlow, Keras and OpenCV with NVIDIA Jetson Nano and Logitech WebCam	On-Device Computing	High Accuracy with Real-Time Detection	Does not Consider Data Diversity

Table 2. *Cont.*

Paper	Application Domain	ML Model	Framework / Hardware	Architecture	Benefits	Drawbacks
[91]	Smart Environment	Multilayer perceptron (MLP)	Simulation	Distributed Edge and Cloud Computing	High Accuracy and Low Energy Consumption Rates	Further Reduction in Power Consumption is needed
[10]	Smart Agriculture	CNN-SVM	TensorRT with NVIDIA Jetson TX1	On-Device Computing	Rapid Decision-Making and High Accuracy	Scalability Not Addressed
[83]	Smart Aquaculture	YOLOv4	Kubernetes and DeepStream with Nvidia Jetson Nx and Jetson Nano	Edge Layer Computing	Reduced latency and Enhanced Privacy	Heterogeneity of IoT Devices is Not Considered
[79]	Smart Agriculture	LSTM and GRU	TensorFlow Lite and Pytorch with Sensors and Raspberry Pi 3 B+	On-Device Computing	Improved Decision-Making and Enhanced Sustainability	High Model Complexity and Security Risks
[87]	Smart Health	RF	Azure IoT Edge with STM32	Distributed Edge and Cloud Computing	Enhanced Workload	Data Privacy and Security Concerns
[85]	Smart Environment	BPNN	Sensors	Edge Layer Computing	Efficient Decision Making	Accuracy Needs to be Improved
[92]	Smart Transportation	YOLOv4 and ORB	Tengine with EAIDK-310, STM-32 and various sensors	Distributed Device and Cloud Computing	Improved Safety	High Cost
[94]	Smart Environment	LSTM and eXtreme Gradient Boosting	Sensors	Distributed Device and Cloud Computing	High Accuracy and Real-Time Monitoring	High Model Cost

5. Open Challenges

Despite the advancements introduced by researchers in previous sections, deploying machine learning on the edge still faces significant challenges, with various factors influencing the performance of edge-based smart applications. In this section, we explore and outline the issues regarding the design and deployment of machine learning at the edge.

5.1. Security and Privacy Issues

Managing data processing at the edge layer remains a significant challenge for numerous global applications. While edge computing has effectively addressed various processing issues through preprocessing and decentralizing processing to the network's edge, instead of solely relying on central processing (cloud computing), several critical issues persist.

One primary challenge is the establishment of robust encryption mechanisms for both data in transit and data at rest to maintain security and privacy standards. Ensuring the confidentiality of data at rest is particularly essential in scenarios where a device could be compromised, such as theft or physical attacks, as this could expose sensitive user information connected to an edge device. Different heterogeneous edge devices and servers are required to run different AI models to provide computing power in order to implement edge intelligence. The extensive data generated by edge devices can pose privacy concerns as they may include sensitive information such as user location, health records, activity logs, or proprietary manufacturing data. Also, malicious data can be used to attack machine learning approaches for IoT, compromising the trust of IoT users. The privacy of user data is a fundamental concern of any machine learning scheme that must be considered during classification. Authentication and identification of each node within

the edge network pose significant challenges. It is crucial to guarantee that each device is granted only the necessary permissions to prevent scenarios where a compromised edge device could potentially access critical system components, including those not essential for regular operations. Privacy-preserving methodologies in machine learning, like federated learning [99,100] and secure model aggregation, enable the training of ML models on dispersed edge devices while safeguarding raw data from exposure. Access control measures, authentication protocols, and privacy policies are implemented to restrict data access and manipulation solely to authorized users and devices. Additionally, consent management frameworks empower users to maintain control over their personal data.

5.2. Resource Management

Edge devices and nodes that provide intelligence at the edge functionality are scattered across various geolocations and territories due to the decentralized nature of edge computing. Different AI models and specific AI tasks may be run on different edge devices and nodes. As a result, it's critical to create effective service discovery protocols to enable users to quickly identify and locate the appropriate EI service providers to meet their needs. Partitioning complex edge AI models into smaller subtasks and effectively distributing these tasks among the edge nodes and devices for collaborative executions are also required to fully utilize the disperse resource across edge nodes and devices [101]. Distributed learning comes as a solution to resource utilization challenges in edge computing. By distributing computational tasks across edge devices, it optimizes resource allocation, alleviating the burden on individual devices and enhancing overall system scalability and performance [102]. This approach not only leverages idle capacity efficiently, but also ensures optimal resource allocation based on workload demands, minimizing wastage and reducing costs. The flexibility and geographical distribution inherent in distributed computing further bolster system responsiveness, enabling organizations to adapt to evolving requirements while maximizing resource utilization. This ensures that AI tasks are executed in a timely and cost-effective manner, harnessing the collective computational power of edge resources to meet diverse user needs effectively.

5.3. Energy

Energy consumption presents a significant challenge in edge computing, particularly for devices reliant solely on batteries or lacking consistent access to power [103]. Both computation and communication processes consume considerable energy during decentralized DNN model training. Therefore, minimizing machine learning's energy consumption is crucial for battery-powered edge devices, which operate under strict energy constraints compared to cloud servers. It's imperative that both the training and inference processes of the model adhere to the energy limitations of the edge node. Achieving a balance between energy consumption and model accuracy is essential for prolonging battery life and ensuring the sustainability of edge devices in resource-constrained environments. Various ongoing efforts are aiming to address this pressing issue. Computation offloading is one way to address the issue of energy consumption. A number of computations are redirected to edge devices that are more suited to handle them, such as those with GPUs, or to devices with bigger energy reserves, or even directly to the cloud. These edge systems are equipped to monitor energy usage and can intelligently distribute tasks to suitable edge devices using offloading algorithms, often integrating machine learning methods for optimized decision-making [102,104].

6. Discussion of Published Papers

The goal of this study is to measure the number of research papers published as either conference papers or journal articles in the domains of machine learning and IoT. We opted to utilize the APIs provided by IEEE Xplore, an academic database, primarily due to their availability and open access to scholarly publications, making it an efficient choice for data collection. While other academic databases or search engines such as ACM Digital Library

or Elsevier could serve the same purpose, the accessibility of the IEEE API streamlined our data collection process. All records matching the keywords "Machine Learning" and "Internet of Things" or "IoT" were retrieved, ensuring comprehensive coverage of relevant research within the specified domains. From the result queries, we collected essential information, including title, abstract, date, and key terms. Despite the potential for utilizing alternative APIs, over 1000 publications containing both machine learning and IoT terms in their titles or abstracts were retrieved from IEEE Xplore. Notably, the publication year trends depicted in the graph below Figure 7 illustrate a significant increase in publications in the past few years, with the number of IoT publications growing faster than those related to machine learning.

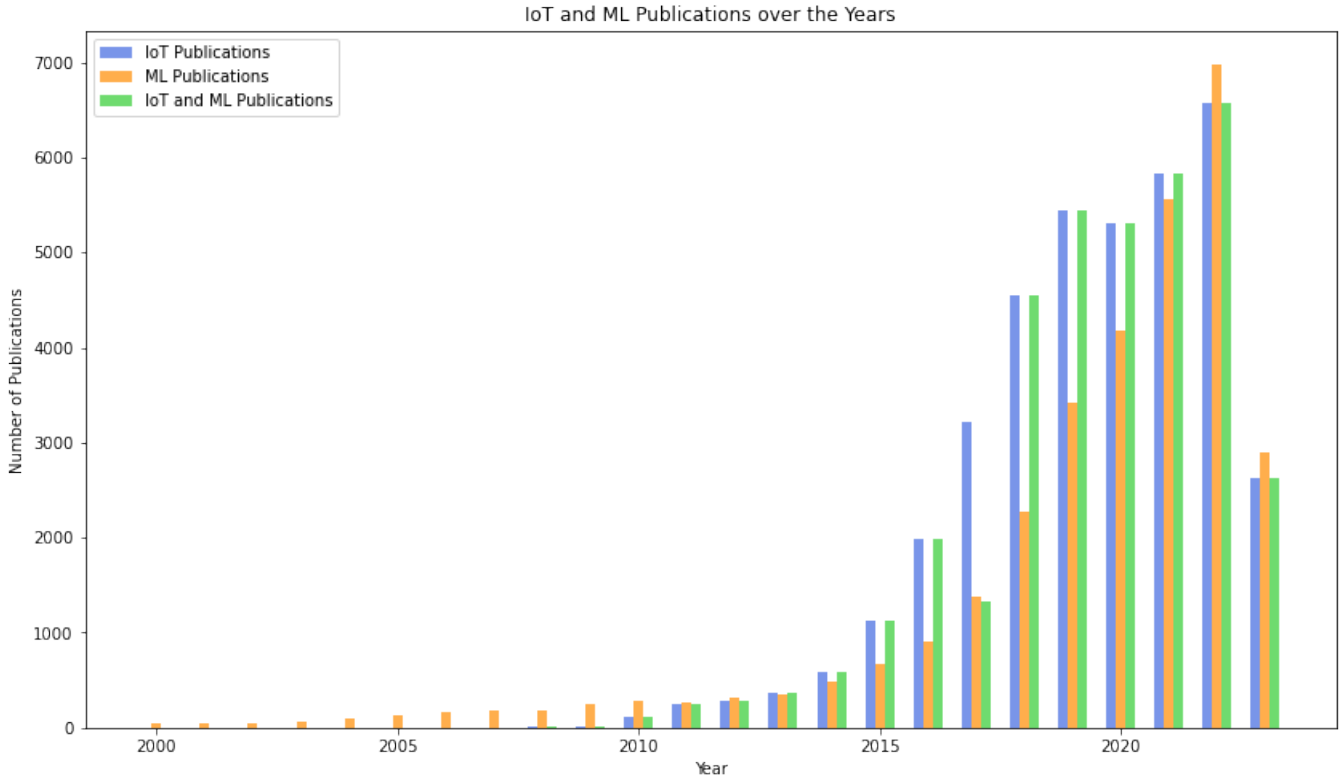

Figure 7. Analysis of published papers in machine learning and IoT.

6.1. The Classification Process

To efficiently handle the large number of papers retrieved from IEEE's Dynamic Query, we employed classification techniques using various ML algorithms. Initially, we manually labeled 450 papers from the query by carefully examining their titles, abstracts, and key terms if available. These labels encompassed various topics such as edge computing, cloud privacy, security, and activity recognition, etc. Reflecting the diverse research areas within the domains of machine learning and IoT. Subsequently, we divided the labeled data into two sets, for training our classifiers and for testing their performance (Figure 8). This approach allowed us to evaluate the effectiveness of our classification models accurately. Through iterative tuning and optimization of the classifiers using the training data, we aimed to enhance their predictive capabilities. Once our classifiers were adequately trained and validated, we utilized them to predict the field of the remaining 550 papers retrieved from the dataset. Leveraging features such as title, year, and abstract of each paper, our classifiers inferred the corresponding field or topic, providing a systematic way to categorize and analyze the extensive collection of research publications. This methodology not only enabled us to efficiently process a large volume of papers but also provided a structured approach to classify and interpret the content based on machine learning algorithms. By leveraging the power of automation and classification, we could uncover

patterns and insights within the dataset, ultimately contributing to a more comprehensive understanding of the research landscape in machine learning and IoT.

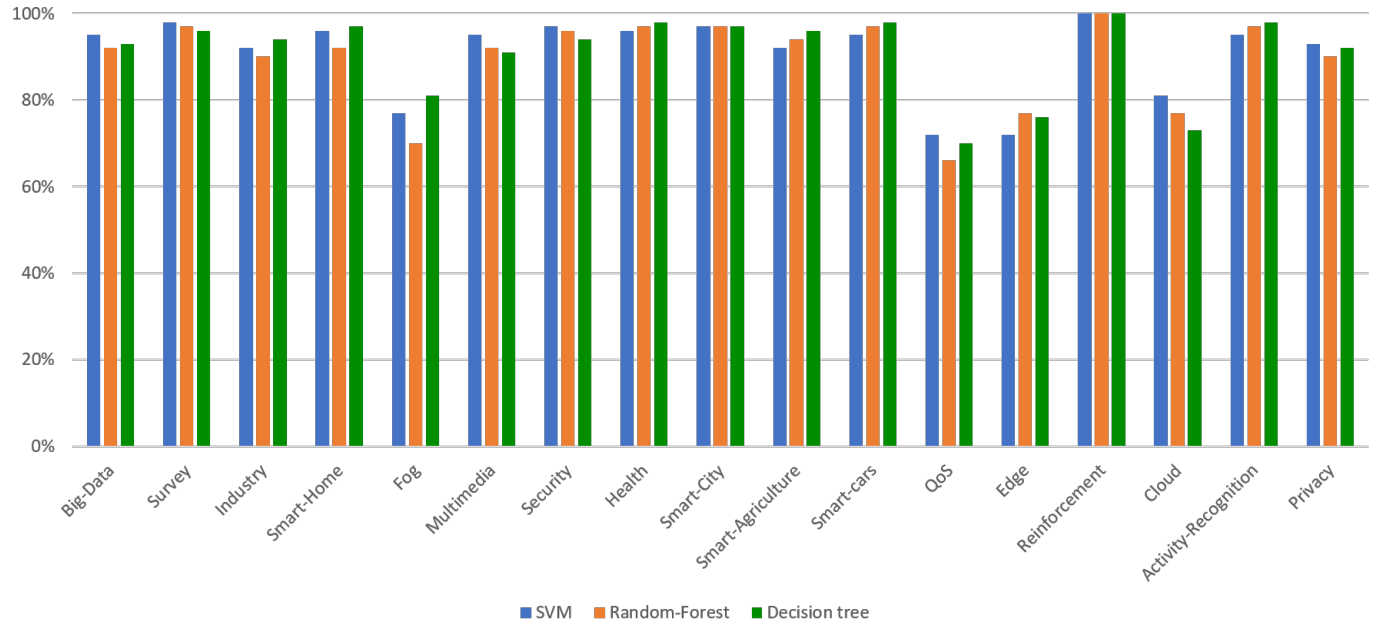

Figure 8. The accuracy predictions of each field using SVM, RF, and decision tree classifiers.

Feature Selection

To extract the necessary features for our classifiers, given that we are dealing with text data, we utilized several techniques in natural language processing. Specifically, we calculated the total occurrences of each keyword in both the title and abstract of the papers. This involved tokenization, stemming or lemmatization, and the use of TF-IDF vectorization. Tokenization involved breaking down the text into individual words or tokens, which served as the basis for further analysis. Stemming or lemmatization was employed to reduce words to their root form, thereby normalizing variations of the same word. TF-IDF vectorization, short for the term frequency-inverse document frequency, assigns weights to each word based on its frequency in a document relative to its frequency across all documents in the corpus. This technique helps to emphasize words that are more unique to a particular document while downplaying common words. Figure 9 illustrates our process of analyzing the published papers using multiple classifiers, including SVM, RF, and decision trees (DT). These classifiers were trained on the extracted features to predict the field or topic of each paper based on its textual content. By employing a combination of feature extraction techniques and machine learning algorithms, we aimed to accurately classify and interpret the content of the research papers, contributing to a deeper understanding of the topics within the domains of machine learning and IoT.

Figure 8 illustrates the accuracy achieved by each classifier across all created fields. It is evident that SVM and DT outperform RF, with an accuracy of 91%. We opted to use the accuracy metric to evaluate the classifiers' performance, as it is widely recognized and applied in assessing the efficacy of machine learning models. In Figure 9, we further delve into the classification accuracy for each class. Notably, the reinforcement class exhibits a remarkable accuracy of 100%, attributed to its specific and distinct index terms. Conversely, classes such as edge, fog, cloud, and QoS display lower accuracies at 72%, 77%, 81%, and 72%, respectively. This discrepancy arises from the varied usage and interpretation of terms like fog computing, edge computing, cloud computing, and QoS (sometimes used as a quality of service), which occasionally confound the classifiers' predictions. However, the remaining classes maintain an average accuracy of 95%.

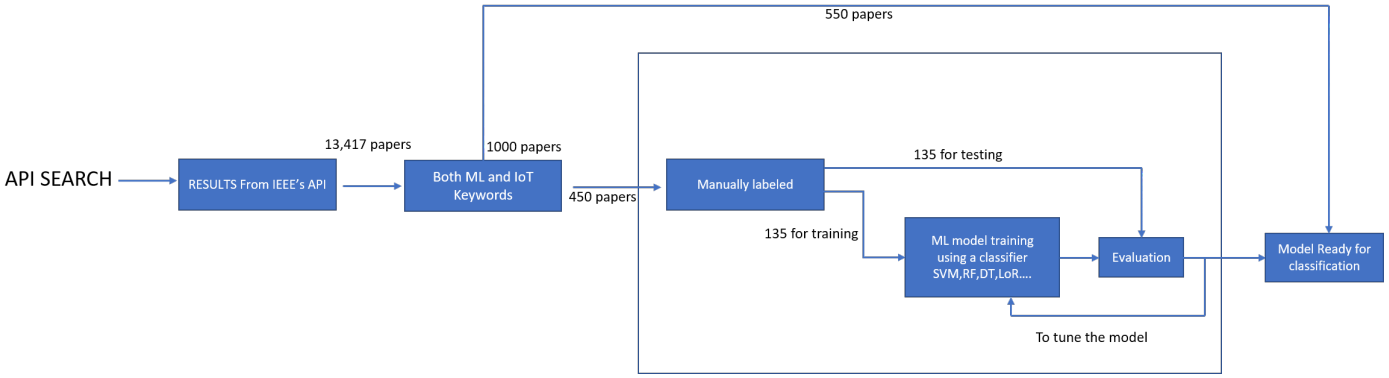

Figure 9. Using machine learning to classify The IEEE's published papers that include both ML and IoT.

These insights underscore the nuanced challenges inherent in classifying research papers within the domains of machine learning and IoT, particularly concerning terminology ambiguity and context sensitivity. Despite these challenges, our classifiers exhibit commendable accuracy across most classes, providing valuable tools for analyzing and categorizing scholarly publications in these dynamic fields.

6.2. Summary

The labels extracted for analysis encompass a broad range of topics, reflecting the diverse landscape of machine learning-enabled IoT research. These include Big Data, Survey (encompassing reviews, overviews, keynotes, and tutorials), Industry, Smart Home, Fog, Cloud, Edge, Multimedia (covering video, image, and sound), Security, Health, Smart City (encompassing traffic management, smart buildings, pollution monitoring, navigation systems, etc.), Smart Agriculture, Smart Cars, QoS, Reinforcement, Activity Recognition (including gesture recognition and assisted living), Privacy, and Device (encompassing papers that do not fit into any other class). To provide a focused analysis of trends in machine learning-enabled IoT literature, we specifically selected a subset of classes for examination. Our analysis encompasses papers published between 2018 and 2023, as the integration of machine learning in IoT has emerged as a prominent topic since approximately 2015, as depicted in Figure 7. Figure 10 illustrates the trends observed within these selected classes over the specified time frame. Notably, applications in Security, Cloud, and Edge have demonstrated substantial growth in recent years, with Security and Edge emerging as the highest trending areas. However, it is important to acknowledge fluctuations in interest observed in Smart City since 2020, contrasting with the relatively stable attention received by Smart Home. This analysis underscores the dynamic nature of research within the realm of machine learning-enabled IoT, highlighting areas of rapid growth and shifts in scholarly interest. By focusing on select classes and examining trends over time, we gain valuable insights into the evolving landscape of this interdisciplinary field, facilitating informed decision-making and future research directions.

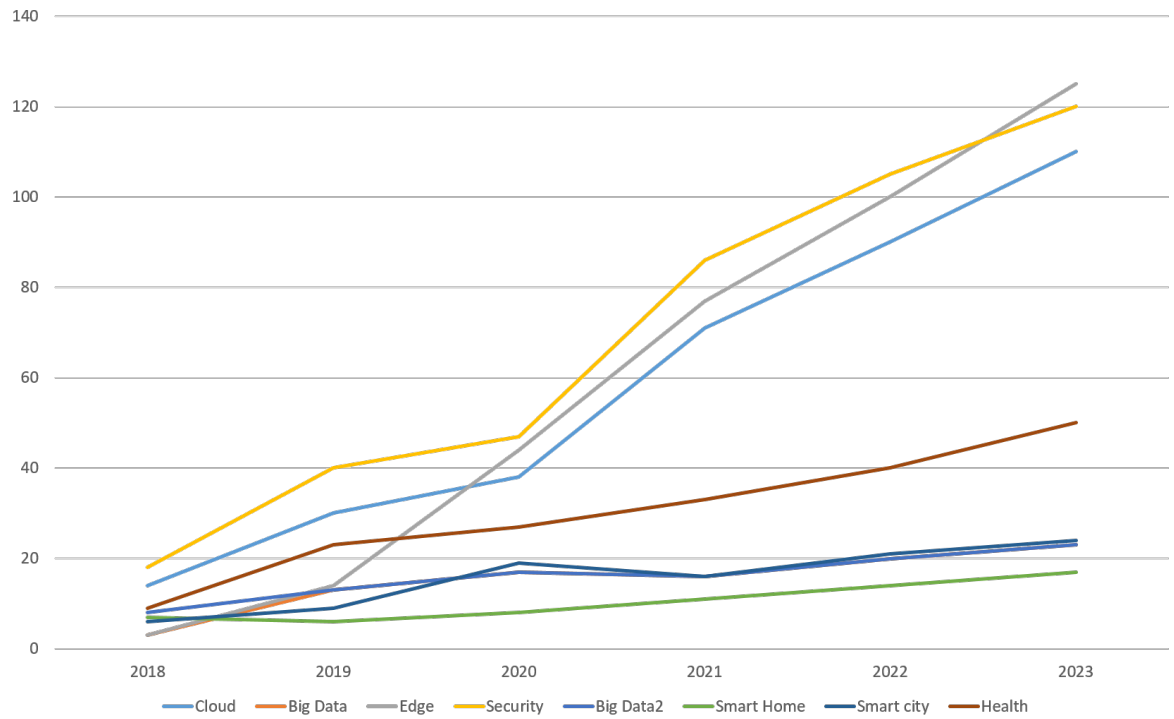

Figure 10. Applications trend of ML in IoT usage.

7. Future Research Directions

The landscape of edge computing is rapidly evolving, driven by a blend of technological advancements and evolving business needs. However, several open issues persist in the current research that need to be addressed to fully realize the potential of edge computing, especially in the context of IoT devices. The following section elaborates on these open issues and proposes directions for future research.

- The lack of standardized IoT architecture presents significant challenges in achieving interoperability, scalability, and security across diverse IoT systems. This fragmentation results in difficulties when integrating devices from different manufacturers and platforms. A Standardization effort should concentrate on defining common communication protocols, data formats, and security mechanisms to foster seamless integration and communication between diverse IoT devices and platforms.
- The advent of 5G technology promises ultra-low latency and high-bandwidth communication, which is crucial for enabling real-time applications such as autonomous vehicles and smart cities [105]. With the evolution towards 6G technology on the horizon, 5G/6G networks will enhance the capability of edge computing by providing faster and more reliable data transmission, thus supporting the real-time processing needs of critical applications [106]. Similarly, blockchain technology offers robust security and transparency, which can be leveraged to secure edge devices and ensure data integrity. Blockchain can enable decentralized and secure data management, which is essential for the growing number of interconnected IoT devices [107].
- The distributed nature of edge computing introduces significant security and privacy challenges. Ensuring that sensitive data remains protected while being processed at the edge is a complex task. Advanced encryption techniques and security frameworks that can be efficiently implemented on edge devices are needed. Additionally, privacy-preserving technologies such as homomorphic encryption and differential privacy should be investigated to secure data processing and sharing [108].
- Innovative solutions for efficient ML deployment in edge computing paradigms will play a crucial role. This involves exploring novel edge computing frameworks, algorithms, and architectures that optimize the distribution of computational tasks across

edge devices, fog nodes, and cloud servers. Advanced edge computing paradigms such as federated learning [109], distributed learning [110], and multi-agent reinforcement learning [111] are being investigated to enhance efficiency and effectiveness in edge computing environments. These paradigms enable collaborative learning across multiple devices without the need to centralize data, thus preserving privacy and reducing bandwidth consumption.

- The development of edge-native applications and services tailored for specific use cases and industries is set to accelerate. These applications are designed to leverage the unique advantages of edge computing, such as low latency and local data processing, to deliver high-performance experiences to end users. For instance, edge-native applications in healthcare can enable real-time monitoring and diagnostics, while those in manufacturing can support predictive maintenance and quality control. The customization of applications to meet the specific needs of different industries will drive the adoption of edge computing and unlock new opportunities for innovation.

8. Conclusions

The traditional cloud computing paradigm is no longer meeting the varying demands of IoT applications due to the massive growth of data collected from IoT devices. Deploying machine learning on different processing layers reduces network congestion, latency, and power consumption and secures the data by performing the computation closer to the source. The goal of this paper is to provide a comprehensive overview of all the key techniques for ensuring the successful execution of machine learning models, starting with the algorithms that can be used, frameworks, and even hardware selection. This paper presents a comprehensive review of machine learning algorithms, architectures, and criteria for solutions that implement ML on IoT devices at various processing layers, with the primary goal of defining the current state of the art and anticipating emerging needs. We have used a classification tool to identify the application field for each publication. According to the research results, the articles on security, edge computing, and cloud computing has increased recently, whereas smart city and smart home has decreased slightly. We also highlight the main research directions for effective machine learning deployment at the IoT edge.

Author Contributions: Conceptualization, O.J., K.S. and A.N.; methodology, O.J., K.S. and A.N.; software, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; validation, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; formal analysis, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; investigation, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; resources, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; data curation, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; writing—original draft preparation, O.J. and K.S.; writing—review and editing, O.J., K.S. and A.N.; visualization, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; supervision, O.J., K.S., A.N., N.A., M.H.A. and M.N.A.; project administration, O.J., K.S., A.N., N.A., M.H.A. and M.N.A. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their appreciation to the Deanship of Scientific Research at Northern Border University, Arar, KSA for funding this research work through the project number "NBU-FFR-2024-1180-10".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset available on request from the authors.

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

References

1. Statista. Number of Internet of Things (IoT) connected Devices Worldwide from 2019 to 2030. 2021. Available online: https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/ (accessed on 10 January 2022).
2. Burhan, M.; Rehman, R.A.; Khan, B.; Kim, B.S. IoT elements, layered architectures and security issues: A comprehensive survey. *Sensors* **2018**, *18*, 2796. [CrossRef] [PubMed]

3. Grzesik, P.; Mrozek, D. Combining Machine Learning and Edge Computing: Opportunities, Challenges, Platforms, Frameworks, and Use Cases. *Electronics* **2024**, *13*, 640. [CrossRef]
4. Chang, Z.; Liu, S.; Xiong, X.; Cai, Z.; Tu, G. A survey of recent advances in edge-computing-powered artificial intelligence of things. *IEEE Internet Things J.* **2021**, *8*, 13849–13875. [CrossRef]
5. Mitchell, T.M.; Carbonell, J.G.; Michalski, R.S. *Machine Learning*; Springer New York, NY, USA, 1986.
6. Devi, I.; Karpagam, G.; Kumar, B.V. A survey of machine learning techniques. *Int. J. Comput. Syst. Eng.* **2017**, *3*, 203–212. [CrossRef]
7. Sharma, K.; Nandal, R. A literature study on machine learning fusion with IOT. In Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23–25 April 2019; pp. 1440–1445.
8. Alsharif, M.H.; Kelechi, A.H.; Yahya, K.; Chaudhry, S.A. Machine learning algorithms for smart data analysis in internet of things environment: Taxonomies and research trends. *Symmetry* **2020**, *12*, 88. [CrossRef]
9. Somvanshi, M.; Chavan, P.; Tambade, S.; Shinde, S. A review of machine learning techniques using decision tree and support vector machine. In Proceedings of the 2016 International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 12–13 August 2016; pp. 1–7.
10. Mputu, H.S.; Mawgood, A.A.; Shimada, A.; Sayed, M.S. Real-Time Tomato Quality Assessment Using Hybrid CNN-SVM Model. In *IEEE Embedded Systems Letters*; IEEE: Piscataway, NJ, USA, 2024.
11. Yazici, M.T.; Basurra, S.; Gaber, M.M. Edge machine learning: Enabling smart internet of things applications. *Big Data Cogn. Comput.* **2018**, *2*, 26. [CrossRef]
12. Abdulkareem, N.M.; Abdulazeez, A.M. Machine learning classification based on Radom Forest Algorithm: A review. *Int. J. Sci. Bus.* **2021**, *5*, 128–142.
13. Xiao, Y.; Xia, R.; Li, Y.; Shi, G.; Nguyen, D.N.; Hoang, D.T.; Niyato, D.; Krunz, M. Distributed traffic synthesis and classification in edge networks: A federated self-supervised learning approach. *IEEE Trans. Mob. Comput.* **2023**, *23*, 1815–1829. [CrossRef]
14. Ongsulee, P. Artificial intelligence, machine learning and deep learning. In Proceedings of the 2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE), Bangkok, Thailand, 22–24 November 2017; pp. 1–6.
15. Ayodele, T.O. Machine learning overview. *New Adv. Mach. Learn.* **2010**, *2*, 16.
16. Zhu, X.; Goldberg, A.B. Introduction to semi-supervised learning. *Synth. Lect. Artif. Intell. Mach. Learn.* **2009**, *3*, 1–130. [CrossRef]
17. Zhu, X.J. *Semi-Supervised Learning Literature Survey*; Technical Report 1530; Department of Computer Sciences, University of Wisconsin-Madison: Madison, WI, USA, 2005.
18. Zhou, X.; Belkin, M. Semi-supervised learning. In *Academic Press Library in Signal Processing*; Elsevier: Amsterdam, The Netherlands, 2014; Volume 1, pp. 1239–1269.
19. Jindal, M.; Gupta, J.; Bhushan, B. Machine learning methods for IoT and their Future Applications. In Proceedings of the 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 18–19 October 2019; pp. 430–434.
20. Kaelbling, L.P.; Littman, M.L.; Moore, A.W. Reinforcement learning: A survey. *J. Artif. Intell. Res.* **1996**, *4*, 237–285. [CrossRef]
21. Yang, Z.; Nguyen, P.; Jin, H.; Nahrstedt, K. MIRAS: Model-based reinforcement learning for microservice resource allocation over scientific workflows. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 7–10 July 2019; pp. 122–132.
22. Tang, Q.; Xie, R.; Yu, F.R.; Chen, T.; Zhang, R.; Huang, T.; Liu, Y. Collective deep reinforcement learning for intelligence sharing in the internet of intelligence-empowered edge computing. *IEEE Trans. Mob. Comput.* **2022**, *22*, 6327–6342. [CrossRef]
23. Lin, T. Deep Learning for IoT. In Proceedings of the 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), Austin, TX, USA, 6–8 November 2020; pp. 1–4.
24. Chen, J.; Ran, X. Deep learning with edge computing: A review. *Proc. IEEE* **2019**, *107*, 1655–1674. [CrossRef]
25. Dhillon, A.; Verma, G.K. Convolutional neural network: A review of models, methodologies and applications to object detection. *Prog. Artif. Intell.* **2020**, *9*, 85–112. [CrossRef]
26. Torres, J.F.; Hadjout, D.; Sebaa, A.; Martínez-Álvarez, F.; Troncoso, A. Deep learning for time series forecasting: A survey. *Big Data* **2021**, *9*, 3–21. [CrossRef] [PubMed]
27. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef] [PubMed]
28. Islam, S.U.; Zaib, S.; Ferraioli, G.; Pascazio, V.; Schirinzi, G.; Husnain, G. Enhanced Deep Learning Architecture for Rapid and Accurate Tomato Plant Disease Diagnosis. *AgriEngineering* **2024**, *6*, 375–395. [CrossRef]
29. Escamilla-Ambrosio, P.; Rodríguez-Mota, A.; Aguirre-Anaya, E.; Acosta-Bermejo, R.; Salinas-Rosales, M. Distributing computing in the internet of things: Cloud, fog and edge computing overview. In *NEO 2016*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 87–115.
30. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 13–16.
31. Dolui, K.; Datta, S.K. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In Proceedings of the 2017 Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; pp. 1–6.
32. Singh, M.; Bharti, S.; Kaur, H.; Arora, V.; Saini, M.; Kaur, M.; Singh, J. A facial and vocal expression based comprehensive framework for real-time student stress monitoring in an IoT-Fog-Cloud environment. *IEEE Access* **2022**, *10*, 63177–63188. [CrossRef]
33. Raspberry Pi. Raspberry pi Products. 2021. Available online: https://www.raspberrypi.com/products/ (accessed on 10 May 2021).

34. Nvidia. Nvidia Jetson Nano Developer Kit. Available online: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (accessed on 1 February 2022).
35. Nguyen, H.H.; Tran, D.N.N.; Jeon, J.W. Towards real-time vehicle detection on edge devices with nvidia jetson tx2. In Proceedings of the 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Seoul, Republic of Korea, 1–3 November 2020; pp. 1–4. [CrossRef]
36. Store, A. Arduino Nano 33 BLE Sense. 2021. Available online: https://store-usa.arduino.cc/products/arduino-nano-33-ble-sense (accessed on 10 January 2022).
37. Waqar, D.M.; Gunawan, T.S.; Morshidi, M.A.; Kartiwi, M. Design of a Speech Anger Recognition System on Arduino Nano 33 BLE Sense. In Proceedings of the 2021 IEEE 7th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA), Virtual, 23–25 August 2021; pp. 64–69. [CrossRef]
38. Daskalos, A.C.; Theodoropoulos, P.; Spandonidis, C.; Vordos, N. Wearable Device for Observation of Physical Activity with the Purpose of Patient Monitoring Due to COVID-19. *Signals* **2022**, *3*, 11–28. [CrossRef]
39. ST. STM32. 2022. Available online: https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus/ (accessed on 10 February 2022).
40. SparkFun. 2021. SparkFun Edge Development Board-Apollo3 Blue. Available online: https://www.sparkfun.com/products/15170 (accessed on 10 January 2022).
41. Coral. Coral Dev Board. 2022. Available online: https://coral.ai/products/dev-board/ (accessed on 10 February 2022).
42. Warden, P.; Situnayake, D. *Tinyml: Machine Learning with Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers*; O'Reilly: Sebastopol, CA, USA, 2020.
43. Zaccone, G.; Karim, M.R.; Menshawy, A. *Deep learning with TensorFlow*; Packt Publishing Ltd.: Birmingham, UK, 2017.
44. Firmansyah, M.H.; Paul, A.; Bhattacharya, D.; Urfa, G.M. Ai based embedded speech to text using deepspeech. *arXiv* **2020**, arXiv:2002.12830. [CrossRef]
45. Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. {TensorFlow}: A System for {Large-Scale} Machine Learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), Savannah, GA, USA, 2–4 November 2016; pp. 265–283.
46. Zhang, X.; Wang, Y.; Lu, S.; Liu, L.; Shi, W. OpenEI: An open framework for edge intelligence. In Proceedings of the 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Savannah, GA, USA, 2–4 November 2019; pp. 1840–1851.
47. Apple. Core ML: Integrate Machine Learning Models Into Your App. 2019. (Online). Available online: https://developer.apple.com/documentation/coreml (accessed on 20 February 2022).
48. Thakkar, M. Introduction to core ML framework. In *Beginning Machine Learning in iOS*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 15–49. [CrossRef]
49. Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, 7 November 2014; pp. 675–678.
50. Liu, F.; Tang, G.; Li, Y.; Cai, Z.; Zhang, X.; Zhou, T. A survey on edge computing systems and tools. *Proc. IEEE* **2019**, *107*, 1537–1562. [CrossRef]
51. Stevens, E.; Antiga, L.; Viehmann, T. *Deep Learning with PyTorch*; Manning Publications: Shelter Island, NY, USA, 2020.
52. Gondi, S.; Pratap, V. Performance Evaluation of Offline Speech Recognition on Edge Devices. *Electronics* **2021**, *10*, 2697. [CrossRef]
53. Chen, T.; Li, M.; Li, Y.; Lin, M.; Wang, N.; Wang, M.; Xiao, T.; Xu, B.; Zhang, C.; Zhang, Z. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv* **2015**, arXiv:1512.01274. [CrossRef]
54. Abed, A.A.; Al-Ibadi, A.; Abed, I.A. Real-time multiple face mask and fever detection using YOLOv3 and TensorFlow lite platforms. *Bull. Electr. Eng. Inform.* **2023**, *12*, 922–929. [CrossRef]
55. Praneeth, R.S.; Akash, K.C.S.; Sree, B.K.; Rani, P.I.; Bhola, A. Scaling Object Detection to the Edge with YOLOv4, TensorFlow Lite. In Proceedings of the 2023 7th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 23–25 February 2023; pp. 1547–1552.
56. Kumar, B.A.; Bansal, M. Face mask detection on photo and real-time video images using Caffe-MobileNetV2 transfer learning. *Appl. Sci.* **2023**, *13*, 935. [CrossRef]
57. Azizan, M.A.; Zaini, N. Video Analysis to Detect Dress Code Violations in Laboratories. In Proceedings of the 2023 IEEE Symposium on Industrial Electronics & Applications (ISIEA), Kuala Lumpur, Malaysia, 15–16 July 2023; pp. 1–6.
58. Kumar, S.; Ratan, R.; Desai, J. Cotton disease detection using tensorflow machine learning technique. *Adv. Multimed.* **2022**, *2022*, 1812025. [CrossRef]
59. Stiller, S.; Duenas, J.F.; Hempel, S.; Rillig, M.C.; Ryo, M. Deep learning image analysis for filamentous fungi taxonomic classification: Dealing with small data sets with class imbalance and hierarchical grouping. *bioRxiv* **2023**. [CrossRef]
60. Chung, J.; Delteil, T. A computationally efficient pipeline approach to full page offline handwritten text recognition. In Proceedings of the 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), Sydney, Australia, 22–25 September 2019; Volume 5, pp. 35–40.

61. Sethia, D.; Singh, P.; Mohapatra, B. Gesture Recognition for American Sign Language Using Pytorch and Convolutional Neural Network. In *Intelligent Systems and Applications: Select Proceedings of ICISA 2022*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 307–317.
62. Khadija, M.A.; Widyawan, W.; Edi Nugroho, L. Deep learning Indonesian chatbot using PyTorch for customer support automation. In *Proceedings of the AIP Conference Proceedings*; AIP Publishing: Long Island, NY, USA, 2023; Volume 2654.
63. Lingudu, P.; Majji, N.; Sasala, B.; Nelli, V.V.; Rao, Y.S.; Battula, S. Edge Assisted Architecture for Performing Precision Agriculture. In Proceedings of the 2022 IEEE Region 10 Symposium (TENSYMP), Mumbai, India, 1–3 July 2022; pp. 1–5.
64. Cheng, W.K.; Ooi, B.Y.; Tan, T.B.; Chen, Y.L. Edge-Cloud Architecture for Precision Aquaculture. In Proceedings of the 2023 International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan), PingTung, Taiwan, 17–19 July 2023; pp. 355–356.
65. Gillam, L.; Katsaros, K.; Dianati, M.; Mouzakitis, A. Exploring edges for connected and autonomous driving. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, USA, 16–19 April 2018; pp. 148–153.
66. Premkumar, S.; Premkumar, V.; Dhakshinamurthy, R. Video Analytics on IoT devices. *arXiv* **2021**, arXiv:2102.07455.
67. Bishnoi, V.; Goel, N. Tensor-RT-Based Transfer Learning Model for Lung Cancer Classification. *J. Digit. Imaging* **2023**, *36*, 1364–1375. [CrossRef]
68. Joshi, P.; Hasanuzzaman, M.; Thapa, C.; Afli, H.; Scully, T. Enabling all in-edge deep learning: A literature review. *IEEE Access* **2023**, *11*, 3431–3460. [CrossRef]
69. Merenda, M.; Porcaro, C.; Iero, D. Edge machine learning for ai-enabled iot devices: A review. *Sensors* **2020**, *20*, 2533. [CrossRef] [PubMed]
70. Huang, Y.; Qiao, X.; Tang, J.; Ren, P.; Liu, L.; Pu, C.; Chen, J. An integrated cloud-edge-device adaptive deep learning service for cross-platform web. *IEEE Trans. Mob. Comput.* **2021**, *22*, 1950–1967. [CrossRef]
71. Wang, L.; Xiang, L.; Xu, J.; Chen, J.; Zhao, X.; Yao, D.; Wang, X.; Li, B. Context-aware deep model compression for edge cloud computing. In Proceedings of the 2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS), Singapore, 29 November–1 December 2020; pp. 787–797.
72. Yu, S.; Mazaheri, A.; Jannesari, A. Topology-aware network pruning using multi-stage graph embedding and reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Baltimore, MD, USA, 25–27 July 2022; pp. 25656–25667.
73. Samie, F.; Paul, S.; Bauer, L.; Henkel, J. Highly efficient and accurate seizure prediction on constrained iot devices. In Proceedings of the 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 955–960. [CrossRef]
74. Bansal, K.; Mittal, K.; Ahuja, G.; Singh, A.; Gill, S.S. DeepBus: Machine learning based real time pothole detection system for smart transportation using IoT. *Internet Technol. Lett.* **2020**, *3*, e156. [CrossRef]
75. Proietti, M.; Bianchi, F.; Marini, A.; Menculini, L.; Termite, L.F.; Garinei, A.; Biondi, L.; Marconi, M. Edge Intelligence with Deep Learning in Greenhouse Management. In Proceedings of the SMARTGREENS, Online, 28–30 April 2021; pp. 180–187.
76. Rumy, S.S.H.; Hossain, M.I.A.; Jahan, F.; Tanvin, T. An IoT based System with Edge Intelligence for Rice Leaf Disease Detection using Machine Learning. In Proceedings of the 2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), Toronto, ON, Canada, 21–24 April 2021; pp. 1–6.
77. Wardana, I.; Gardner, J.; Fahmy, S. Optimising deep learning at the edge for accurate hourly air quality prediction. *Sensors* **2021**, *21*, 1064. [CrossRef]
78. Hassan, N.F.A.; Abed, A.A.; Abdalla, T.Y. Face mask detection using deep learning on NVIDIA Jetson Nano. *Int. J. Electr. Comput. Eng. (2088-8708)* **2022**, *12*, 5427–5434.
79. Dahane, A.; Benameur, R.; Kechar, B. An IoT low-cost smart farming for enhancing irrigation efficiency of smallholders farmers. *Wirel. Pers. Commun.* **2022**, *127*, 3173–3210. [CrossRef]
80. Yang, Y.; Luan, T.; Yu, Z.; Zhang, M.; Li, F.; Chen, X.; Gao, F.; Zhang, Z. Technological Vanguard: The outstanding performance of the LTY-CNN model for the early prediction of epileptic seizures. *J. Transl. Med.* **2024**, *22*, 162. [CrossRef]
81. Shafi, U.; Mumtaz, R.; Qureshi, M.D.M.; Mahmood, Z.; Tanveer, S.K.; Haq, I.U.; Zaidi, S.M.H. Embedded AI for wheat yellow rust infection type classification. *IEEE Access* **2023**, *11*, 23726–23738. [CrossRef]
82. Semiconductors, L. Accelerating Implementation of Low Power Artificial Intelligence at the Edge. In *A Lattice Semiconductor White Paper*; 2018. Available online: https://file.elecfans.com/web1/M00/20/B4/ooYBAFmk0Z2ALJQbAAVfvxLevGo400.pdf (accessed on 28 May 2024).
83. Fathoni, H.; Yang, C.T.; Huang, C.Y.; Chen, C.Y. Empowered edge intelligent aquaculture with lightweight Kubernetes and GPU-embedded. *Wirel. Netw.* **2024**, 1–13. [CrossRef]
84. Zhang, S.; Guo, B.; Dong, A.; He, J.; Xu, Z.; Chen, S. Cautionary tales on air-quality improvement in Beijing. *Proc. R. Soc.* **2017**, *473*, 20170457. [CrossRef]
85. Ren, J.; Zhu, Q.; Wang, C. Edge Computing for Water Quality Monitoring Systems. *Mob. Inf. Syst.* **2022**, *2022*, 5056606. [CrossRef]
86. Liu, L.; Chen, C.; Pei, Q.; Maharjan, S.; Zhang, Y. Vehicular edge computing and networking: A survey. *Mob. Netw. Appl.* **2021**, *26*, 1145–1168. [CrossRef]

87. Islam, M.M.; Hasan, M.K.; Islam, S.; Balfaqih, M.; Alzahrani, A.I.; Alalwan, N.; Safie, N.; Bhuiyan, Z.A.; Thakkar, R.; Ghazal, T.M. Enabling pandemic-resilient healthcare: Narrowband Internet of Things and edge intelligence for real-time monitoring. *CAAI Trans. Intell. Technol.* **2024**. Available online: https://api.semanticscholar.org/CorpusID:268766083 (accessed on 28 May 2024). [CrossRef]
88. Li, H.; Ota, K.; Dong, M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
89. Sivaganesan, D. Design and development ai-enabled edge computing for intelligent-iot applications. *J. Trends Comput. Sci. Smart Technol. (TCSST)* **2019**, *1*, 84–94.
90. Sajjad, M.; Nasir, M.; Muhammad, K.; Khan, S.; Jan, Z.; Sangaiah, A.K.; Elhoseny, M.; Baik, S.W. Raspberry Pi assisted face recognition framework for enhanced law-enforcement services in smart cities. *Future Gener. Comput. Syst.* **2020**, *108*, 995–1007. [CrossRef]
91. Shahra, E.Q.; Wu, W.; Basurra, S.; Aneiba, A. Intelligent Edge-Cloud Framework for Water Quality Monitoring in Water Distribution System. *Water* **2024**, *16*, 196. [CrossRef]
92. Tong, R.; Jiang, Q.; Zou, Z.; Hu, T.; Li, T. Embedded system vehicle based on multi-sensor fusion. *IEEE Access* **2023**, *11*, 50334–50349. [CrossRef]
93. Mrozek, D.; Koczur, A.; Małysiak-Mrozek, B. Fall detection in older adults with mobile IoT devices and machine learning in the cloud and on the edge. *Inf. Sci.* **2020**, *537*, 132–147. [CrossRef]
94. Ansari, M.; Alam, M. An intelligent IoT-cloud-based air pollution forecasting model using univariate time-series analysis. *Arab. J. Sci. Eng.* **2024**, *49*, 3135–3162. [CrossRef]
95. Barnawi, A.; Chhikara, P.; Tekchandani, R.; Kumar, N.; Alzahrani, B. Artificial intelligence-enabled Internet of Things-based system for COVID-19 screening using aerial thermal imaging. *Future Gener. Comput. Syst.* **2021**, *124*, 119–132. [CrossRef]
96. Deebak, B.; Al-Turjman, F. EEI-IoT: Edge-Enabled Intelligent IoT Framework for Early Detection of COVID-19 Threats. *Sensors* **2023**, *23*, 2995. [CrossRef] [PubMed]
97. Arikumar, K.; Prathiba, S.B.; Alazab, M.; Gadekallu, T.R.; Pandya, S.; Khan, J.M.; Moorthy, R.S. FL-PMI: Federated learning-based person movement identification through wearable devices in smart healthcare systems. *Sensors* **2022**, *22*, 1377. [CrossRef]
98. Cordeiro, M.; Markert, C.; Araújo, S.S.; Campos, N.G.; Gondim, R.S.; da Silva, T.L.C.; da Rocha, A.R. Towards Smart Farming: Fog-enabled intelligent irrigation system using deep neural networks. *Future Gener. Comput. Syst.* **2022**, *129*, 115–124. [CrossRef]
99. Xiao, Y.; Zhang, X.; Li, Y.; Shi, G.; Krunz, M.; Nguyen, D.N.; Hoang, D.T. Time-sensitive learning for heterogeneous federated edge intelligence. *IEEE Trans. Mob. Comput.* **2023**, *23*, 1382–1400. [CrossRef]
100. Ying, C.; Jin, H.; Wang, X.; Luo, Y. Double insurance: Incentivized federated learning with differential privacy in mobile crowdsensing. In Proceedings of the 2020 International Symposium on Reliable Distributed Systems (SRDS), Shanghai, China, 21–24 September 2020; pp. 81–90.
101. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
102. Zhou, H.; Jiang, K.; Liu, X.; Li, X.; Leung, V.C. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. *IEEE Internet Things J.* **2021**, *9*, 1517–1530. [CrossRef]
103. Gowers, G.O.F.; Vince, O.; Charles, J.H.; Klarenberg, I.; Ellis, T.; Edwards, A. Entirely off-grid and solar-powered DNA sequencing of microbial communities during an ice cap traverse expedition. *Genes* **2019**, *10*, 902. [CrossRef]
104. Gao, Z.; Yang, L.; Dai, Y. Large-scale computation offloading using a multi-agent reinforcement learning in heterogeneous multi-access edge computing. *IEEE Trans. Mob. Comput.* **2022**, *22*, 3425–3443. [CrossRef]
105. Biswas, A.; Wang, H.C. Autonomous vehicles enabled by the integration of IoT, edge intelligence, 5G, and blockchain. *Sensors* **2023**, *23*, 1963. [CrossRef]
106. Chandra, S.; Arya, R.; Singh, M.P. Intelligent resource management in 5G/6G network by adopting edge intelligence for higher education systems. *E-Prime Electr. Eng. Electron. Energy* **2024**, *8*, 100517. [CrossRef]
107. Wang, X.; Shankar, A.; Li, K.; Parameshachari, B.; Lv, J. Blockchain-Enabled Decentralized Edge Intelligence for Trustworthy 6G Consumer Electronics. *IEEE Trans. Consum. Electron.* **2024**, *70*, 1214–1225. [CrossRef]
108. Yang, R.; Zhao, T.; Yu, F.R.; Li, M.; Zhang, D.; Zhao, X. Blockchain-Based Federated Learning with Enhanced Privacy and Security Using Homomorphic Encryption and Reputation. *IEEE Internet Things J.* **2024** . [CrossRef]
109. Hasan, M.K.; Jahan, N.; Nazri, M.Z.A.; Islam, S.; Khan, M.A.; Alzahrani, A.I.; Alalwan, N.; Nam, Y. Federated Learning for Computational Offloading and Resource Management of Vehicular Edge Computing in 6G-V2X Network. *IEEE Trans. Consum. Electron.* **2024**, *70*, 3827–3847. [CrossRef]
110. Yuan, Y.; Chen, S.; Yu, D.; Zhao, Z.; Zou, Y.; Cui, L.; Cheng, X. Distributed Learning for Large-scale Models at Edge with Privacy Protection. *IEEE Trans. Comput.* **2024**, *73*, 1060–1070. [CrossRef]
111. Liu, P.; An, K.; Lei, J.; Sun, Y.; Liu, W.; Chatzinotas, S. Computation Rate Maximization for SCMA-Aided Edge Computing in IoT Networks: A Multi-Agent Reinforcement Learning Approach. *IEEE Trans. Wirel. Commun.* **2024**. [CrossRef]