electronics

MDPI

Review

Combining Machine Learning and Edge Computing: Opportunities, Challenges, Platforms, Frameworks, and Use Cases

Piotr Grzesik and Dariusz Mrozek *

Department of Applied Informatics, Silesian University of Technology, 44-100 Gliwice, Poland; pj.grzesik@gmail.com
* Correspondence: dariusz.mrozek@polsl.pl; Tel.: +48-32-237-13-39

Abstract: In recent years, we have been observing the rapid growth and adoption of IoT-based systems, enhancing multiple areas of our lives. Concurrently, the utilization of machine learning techniques has surged, often for similar use cases as those seen in IoT systems. In this survey, we aim to focus on the combination of machine learning and the edge computing paradigm. The presented research commences with the topic of edge computing, its benefits, such as reduced data transmission, improved scalability, and reduced latency, as well as the challenges associated with this computing paradigm, like energy consumption, constrained devices, security, and device fleet management. It then presents the motivations behind the combination of machine learning and edge computing, such as the availability of more powerful edge devices, improving data privacy, reducing latency, or lowering reliance on centralized services. Then, it describes several edge computing platforms, with a focus on their capability to enable edge intelligence workflows. It also reviews the currently available edge intelligence frameworks and libraries, such as TensorFlow Lite or PyTorch Mobile. Afterward, the paper focuses on the existing use cases for edge intelligence in areas like industrial applications, healthcare applications, smart cities, environmental monitoring, or autonomous vehicles.

Keywords: machine learning; edge computing; smart cities; healthcare; predictive maintenance; Internet of Things

Citation: Grzesik, P.; Mrozek, D. Combining Machine Learning and Edge Computing: Opportunities, Challenges, Platforms, Frameworks, and Use Cases. *Electronics* **2024**, *13*, 640. https://doi.org/10.3390/electronics13030640

Academic Editor: Palden Lama

Received: 31 December 2023
Revised: 25 January 2024
Accepted: 1 February 2024
Published: 3 February 2024

1. Introduction

In recent years, we have experienced the rapid growth and adoption of IoT-based systems in multiple areas of our lives. It is common to see IoT devices improving healthcare systems [1], being used in autonomous vehicles [2], enabling smart city solutions [3], enhancing environmental monitoring [4,5], or powering smart wearable equipment [6], just to name a few scenarios. In all mentioned cases, the IoT devices are responsible for generating massive volumes of data, either in numerical form for various sensor readings or images and videos from camera-equipped devices. Processing such amounts of data in data centers has become a big challenge, especially in situations where the Internet connection is unreliable. In order to address these issues, a new computing paradigm, called edge computing [7], has started to emerge. The main idea of edge computing is to bring the processing units closer to the end devices and end users. By doing that, it is possible to process the data streams without the need to send them to a centralized service, which improves the reaction time of such systems, saves network bandwidth, improves scalability by reducing dependency on centralized services, and improves data privacy, as not everything has to be sent over to the cloud or data center for further processing. We have seen the growing adoption of the edge computing paradigm in various areas, such as smart cities [8], where it powers distributed processing on multiple devices across the city; healthcare systems [9], where one of the most significant benefits of edge computing is data privacy, achieved by keeping sensitive medical records at the edge instead of sending them to the cloud [10]; and in industrial applications [11], where it is used for predictive

maintenance, quality control monitoring, or enabling fleets of autonomous robots that carry out various tasks within factories, among others. Figure 1 presents a common edge computing architecture. It is important to note that the edge computing layer can be very varied, with multiple types of devices such as dedicated edge servers, satellites, or UAVs, performing edge computations for other devices in the network. In addition to that, we have been observing artificial intelligence (AI) and machine learning (ML) techniques being used for more and more use cases in the past years, often in the same context as the mentioned IoT devices [12]. AI covers a set of technologies that allow a computer system to learn, reason, and perform sophisticated tasks like a human [13,14]. As frequently mentioned as a sub-technique of AI, ML uses various self-learning algorithms to build reasoning models based on available data [15]. However, many machine learning workflows are still computationally expensive and executed on centralized servers, making the adoption of machine learning at the edge more challenging [16]. The same applies to deep learning techniques that rely on artificial neural networks (AANs) with many layers that usually learn based on larger data sets, automate feature extraction, and require less human intervention than typical ML models [17].

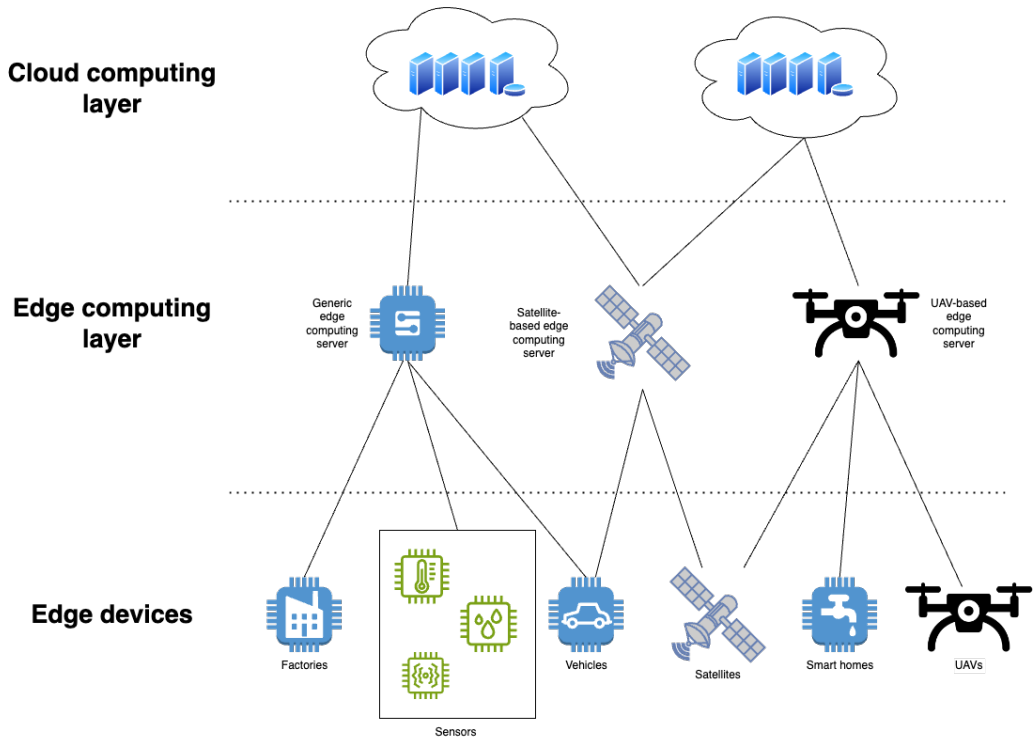

Figure 1. Illustration of an edge computing architecture.

The motivation for our work is to highlight the opportunities that the combination of machine learning and edge computing brings, what challenges are associated with it, what tools are currently available for building edge intelligence solutions, and, most importantly, we highlight existing use cases that highly benefit from edge intelligence. The paper effectively integrates the knowledge acquired in recent years and technical concepts with real-world applications, making it accessible to both technical and non-technical audiences. It elucidates complex topics, such as edge computing architectures and machine learning frameworks while grounding them in practical application areas like smart cities, healthcare systems, and industrial applications that benefit from these concepts. It also showcases a balanced perspective by addressing both the opportunities and challenges associated with edge computing and machine learning integration. It acknowledges the potential benefits of reduced latency while also highlighting concerns such as energy consumption and security risks. The paper focuses on specific examples, platforms, and frameworks

relevant to edge intelligence. By reviewing specific tools, the paper provides actionable insights for readers interested in implementing edge computing solutions.

The paper is organized as follows. In Section 2, we review the existing challenges for edge computing, focusing on aspects that are especially important for machine learning tasks. Section 3 highlights the motivations for combining machine learning and edge computing, namely, more powerful edge devices, reducing reliance on centralized services, and improving the privacy of users' data. In Section 4, we describe edge computing platforms, noting how they can enable the easier deployment of edge intelligence workflows. Section 5 presents state-of-the-art frameworks and libraries that are dedicated to building and training machine learning models that will run on constrained edge devices. In Section 6 of the article, we go over existing use cases in which we already observe the successful combination of edge computing and machine learning in areas such as industrial applications, autonomous vehicles, healthcare applications, and environmental monitoring. Section 7 briefly reviews trends and future developments in edge computing. Section 8 concludes the paper with a discussion.

2. Challenges of Edge Computing

While edge computing has a lot of benefits, it is also associated with downsides and challenges. This section presents the major challenges for edge computing and highlights which of them are especially important to consider when it comes to running machine learning workloads.

2.1. Constrained Devices and Computation Offloading

One of the main challenges of edge computing stems from the fact that computation is now being carried out by computing boards with inferior CPUs, less memory, and less disk space than machines running in cloud data centers. Additionally, the edge devices vary a lot across different applications due to external constraints such as size or energy consumption. In certain applications, an edge device could be a relatively powerful industrial PC. In another case, it could be a single-board computer such as Jetson Nano or Raspberry Pi, which offers less computing power than the mentioned industrial PCs. In other cases, the possibilities might be limited to microcontrollers such as STM32F411VE or GAP8, often powering wearable devices such as smartwatches where a small form factor is very important. Due to these variable capabilities of certain devices, the edge computing workloads must be heavily optimized and adjusted to the computing capabilities of the selected edge device. Often, it comes at the price of the accuracy or speed of processing. Another technique that is used to overcome the limited computing capabilities of edge devices is computation offloading. It works by decomposing workloads into multiple steps or tasks that can be processed by other edge nodes or in a cloud environment in order to speed up or enable certain workloads [18–20]. It is also a method to ensure better scalability and mitigate bottlenecks in the system by enabling the offloading of tasks from the nodes that are used more heavily than others in the edge-based system.

2.2. Security and Privacy

While edge computing can improve data privacy in certain scenarios, it also creates new challenges when it comes to ensuring the security of the whole system. The first challenge in that regard is the authentication and identification of each node in the edge network. It is also essential to ensure that each device has only a minimal set of permissions to avoid situations where a compromised edge device can be used to gain access to critical parts of the system, even the parts that are not required for regular operations. Another crucial challenge regarding security and privacy is proper encryption mechanisms, both for data in transit and at rest. Ensuring data privacy at rest is especially important in situations where the device can be stolen or accessed physically in an attempt to access (e.g., the private data of users connected to an edge device). Anonymization and pseudonymization techniques are utilized to remove or replace personally identifiable information (PII) from data sets,

preventing the identification of individuals while allowing for data analysis [21,22]. Data minimization strategies such as aggregation and summarization help reduce the amount of data processed at the edge, minimizing privacy risks [16,23]. Privacy-preserving machine learning techniques such as federated learning [24,25] and secure model aggregation [26] allow for training machine learning models on distributed edge devices without exposing raw data. Access control mechanisms, authentication protocols, and privacy policies ensure that only authorized users and devices can access and manipulate data, while consent management frameworks empower users to control their personal data. Secure data sharing and collaboration [27,28], along with privacy impact assessments, promote transparency and accountability in privacy practices, fostering trust and compliance with privacy regulations. An in-depth overview of various security threats in mobile edge computing has been presented by Roman et al. [29], while Yahuza et al. [30] developed a systematic review of all security and privacy requirements for edge-based systems.

2.3. Energy Consumption

One of the largest challenges related to edge computing is energy consumption, especially for devices that are only battery-powered or devices that do not have constant and reliable access to a power outlet, e.g., during expeditions [31]. Addressing this problem becomes even more critical for workloads that require significant computing power or need to use additional accelerators such as onboard GPUs (Graphics Processing Units) as is the case with machine learning applications. The algorithms used at the edge must be optimized to maintain a reasonable ratio between performance and energy efficiency to address that challenge. There are multiple ongoing works that aim to improve this issue. One of the approaches involves energy harvesting techniques based on solar panels, wind mills, or electromagnetic signals that allow the edge devices to recharge their batteries without external intervention, making these systems more resilient and autonomous [32–34]. Another way of addressing the energy consumption issue is computation offloading. In that scenario, certain computations are offloaded to other edge devices that are better equipped to deal with them (e.g., edge device with GPUs), or to devices that currently have access to more energy [35], or directly to the cloud. Such edge systems are energy-aware and, thanks to offloading algorithms, often taking advantage of machine learning techniques, are able to intelligently route tasks to the proper edge devices [36].

2.4. Device Fleet Management

Another important challenge related to edge computing is the requirement to manage a heterogeneous fleet of distributed devices. In the case of cloud computing, all nodes frequently exhibit uniform characteristics, are often placed in the same geographical region, or even housed within the same data center. In the case of edge computing, there is often a need to accommodate various device types, operating systems, and software executed on these devices, sometimes running across the globe. One way to mitigate issues that might span from that is the use of standardized communication protocols such as MQTT, CoAP, AMQP, DDS, or OPC UA, which enable interoperability between different devices and systems. There are also ongoing studies [37] that research how these existing protocols can be optimized with the use of machine learning to mitigate the downsides of specific protocols. Separately, it is also more difficult to deal with hardware failures than in the cloud environment, as edge devices might be located in places that are hard to access. The same applies to software failures, especially with the unstable network connection between edge and cloud devices. The deployment of new software to the edge fleet is also more complex than in the cloud environment, and there are ongoing works on developing more robust deployment approaches dedicated to edge-based scenarios [38–40]. In Section 4, there are several edge computing platforms presented that aim to improve the process of device and application management. These platforms often combine the capability to manage the fleet of devices in a centralized system; maintain their operating systems, firmware, and runtime for applications running on the edge devices; integrate with standardized communication

protocols; and manage orchestration algorithms that distribute and deploy the computing workloads across devices in the system.

3. Motivations for Combining Machine Learning and Edge Computing

This section focuses on the benefits that the combination of machine learning and edge computing brings. It is becoming especially important with the growing number of smart devices distributed across the whole globe and with the growing number of use cases for both ML and edge computing.

3.1. More Powerful Devices Available at the Edge

One of the biggest opportunities is the fact that nowadays, there are multiple mobile platforms available that offer acceleration for machine learning tasks, thanks to built-in hardware accelerators. Devices like Jetson Nano, Jetson Xavier NX, or the recently announced Jetson Orin Nano are equipped with GPUs, and smartphones like the iPhone have dedicated NPUs (Neural Processing Units) in the form of Apple Neural Engine. Alternatively, there are also add-on devices like Intel Neural Compute Stick, which is equipped with VPUs (Visual Processing Units) in order to improve the performance of inference tasks. Another such accelerator is the Google Coral USB Accelerator, which serves as a TPU (Tensor Processing Unit) coprocessor that speeds up inference workflows. Both Intel NCS and Google Coral USB Accelerator can turn popular low-cost edge devices such as Raspberry Pis into powerful machine learning inference engines.

3.2. Reducing Reliance on Centralized Services and Decreasing Latency

One of the challenges with intelligence servers placed in data centers or in the cloud computing environment is the requirement to send data over the network in order to process them and react to the changing state of the system. It becomes especially problematic in secluded areas with unreliable Internet connections or in situations where there are multiple people accessing the centralized service at the same time. By moving the intelligence modules closer to the end user, it is possible to reduce the amount of data that need to be sent over to the centralized service, which in turn allows the whole system to react faster to the changing state, decreasing the latency of machine learning workloads. It is especially important in systems that need to make decisions in real time, e.g., autonomous vehicles or health monitoring devices.

3.3. Improving Privacy of Personal Data

Another great benefit of combining machine learning with edge computing is the potential to improve the privacy of users' personal data. When using techniques such as federated learning, the personal data do not have to leave users' devices or the edge server in order to be used in the model training process, which reduces the impact of potential security breaches, e.g., in the data center.

4. Edge Computing Platforms

Edge computing has become an attractive alternative for centralized computations performed in on-premise data centers or directly on cloud platforms. This fact was noticed quite early by the providers of cloud platforms, third-party companies, and open-source societies, leading to the emergence of several frameworks that operate according to industrial standards. This section characterizes the most important of them.

4.1. Microsoft Azure IoT Edge

Microsoft Azure IoT Edge [41] is an open-source engine, edge computing runtime, and a set of integrated cloud services that enable building, managing, and deploying various edge computing workloads. It seamlessly integrates with other Microsoft Azure services, supports Linux and Windows operating systems, and offers dedicated modules directly from Microsoft or its partners in the form of Docker containers that can efficiently run at

the edge devices. It also seamlessly integrates with Microsoft Azure Machine Learning to run ML inference directly at the edge. It can operate without constant Internet connectivity. Additionally, it handles the authentication and authorization of all connected devices and ensures secure communication between all parts of the system, both locally and in the cloud. It is also supported to run stream analytics directly at the edge devices thanks to integration with the Azure Stream Analytics service. Microsoft Azure additionally offers Azure SQL Edge, which is a robust SQL database dedicated to handling data storage on edge devices.

4.2. AWS IoT Greengrass

AWS IoT Greengrass [42] is an open-source, edge computing runtime and a corresponding cloud-based service that allows building, managing, and deploying edge computing workloads. It is available as a part of the Amazon Web Services platform and supports running local containers, local AWS Lambda functions, and local messaging for IoT devices. AWS IoT Greengrass can also operate without any Internet connectivity. By supporting custom containers, it is capable of running virtually any workload directly on the edge device, regardless of the used programming language. It can access local devices such as GPU, sensors, actuators, and so on. It also seamlessly integrates with machine learning models training in the cloud thanks to the AWS IoT Greengrass ML Inference offering, which allows for running machine learning inference directly on edge devices, enabling edge intelligence. It can be successfully used to implement a federated learning approach [43]. It also has a dedicated Stream Manager module that allows organizing data streams to collect and process incoming data from local IoT devices. It can also integrate directly with cloud services such as AWS Kinesis or AWS S3. AWS IoT Greengrass is also responsible for ensuring the security, authentication, and authorization of all services and devices communicating locally and with the cloud-based components.

4.3. Balena

Balena [44] is a suite of tools dedicated to building, managing, and provisioning IoT devices. The core part of the platform is BalenaCloud, which is used for managing the fleet of IoT devices. Another part is BalenaOS, which is an operating system based on Yocto Linux, optimized for edge devices, that provides balenaEngine, an optimized Docker-compatible container engine for running custom containers on these devices. Thanks to that, it is possible to use Balena to manage and deploy edge intelligence applications in a secure and automated manner. They also offer an extensive repository of examples on how to run various workloads with the use of Balena services [45].

4.4. KubeEdge.AI

KubeEdge [46] is an open-source, edge computing platform built based on Kubernetes that provides support for application deployment, synchronization, and networking across deployments in the cloud and at the edge. It is optimized to run successfully even on low-powered devices, simplifies the device communication stack, and allows the edge components to operate without issues, even without an Internet connection. Out of the box, KubeEdge does not offer any dedicated solutions for running edge intelligence workloads. That is where KubeEdge.AI [47] intends to improve the base capabilities of KubeEdge. KubeEdge.AI is an edge intelligence framework that introduces a few dedicated modules dedicated to working with machine learning workloads directly at the edge. One of the modules is a data handling and processing engine built on top of TSDB (Time-Series Database), Apache Flink, and Apache Spark. Another key module is a dedicated AI engine that helps with model deployment and model refreshment, ensures the privacy and security of the model, and includes an optimized runtime for performing machine learning workloads. In addition to that, KubeEdge.AI also consists of a decision engine and distributed query interface.

4.5. EdgeX Foundry

EdgeX Foundry [48] is an open-source, highly scalable, full-featured framework for the management and orchestration of edge computing services primarily dedicated to industrial environments. EdgeX Foundry is designed with cloud-native principles, making it vendor- and operating system-agnostic. It supports all major platforms and operating systems. EdgeX Foundry consists of loosely coupled microservices that are organized into multiple layers. A core services layer hosts services like core data, registry and config, command, and metadata. The second layer is the device services layer, which is responsible for direct interaction with various IoT devices. An application services layer also manages integration between the edge and cloud environment by handling data transmission and processing. Another layer is the supporting services layer. It includes services like the rules engine that can monitor incoming data from sensors and react to it. There is a scheduling service that can, for example, aggregate data in predefined time intervals. Another service in the supporting services layer is the alerts and notifications engine that handles alerting and sending notifications to external systems. The next layer is a management services layer that implements an API (Application Programming Interface) to control all the microservices that build the whole EdgeX Foundry deployment. The last layer is the security services layer which includes functionalities like a secret store service for handling secrets and an API gateway service for external clients. It was also designed with interoperability in mind. It provides implementations for all popular IoT protocols, such as MQTT, REST, or CoAP.

5. Edge Intelligence Frameworks and Libraries

In recent years, we have observed the development of more tools and libraries dedicated to building and deploying machine learning models directly to edge devices. Some of them are growing out of existing mature ML frameworks, and some are built from the ground up for edge-based cases. This section provides an overview of selected frameworks and libraries that help build edge intelligence applications. Table 1 presents a summary of the discussed frameworks and libraries for edge intelligence.

5.1. TensorFlow Lite

The first presented machine learning framework is TensorFlow Lite [49]. It is an open-source framework developed by Google that grew from the original TensorFlow offering. It is specifically optimized to run machine learning workloads directly on constrained devices. It supports various platforms, such as iOS and Android mobile operating systems, single-board computers like Jetson Nano or Raspberry Pi, and even microcontrollers, particularly from the ARM Cortex-M series. It also supports multiple programming languages, including Python, Java, Swift, and C++. It can also take advantage of hardware acceleration, e.g., GPU on Jetson Nano boards or TPU from Coral USB Accelerator. It also has an extensive community and multiple working examples available in the documentation.

5.2. edge-ml

Another framework dedicated to edge intelligence is edge-ml. It was developed by Röddiger et al. [50], is open-source, and aims to allow the building of robust machine learning models for microcontrollers using a browser-based environment. It supports the whole model lifecycle, from data recording, labeling, and model training to deploying the trained model to edge devices. It speeds up the entire process by automatically selecting the best neural network architecture for the use case at hand. It then optimizes the trained models for the chosen platform. Currently, it supports Arduino Nicla Sense ME, Arduino Nano 33 BLE, all boards that are based on the ESP32 SOC microcontroller, as well as devices based on the Android operating system.

Table 1. Summary of frameworks and libraries for edge intelligence.

Library	Supported Devices or Systems	Open Source	First Released	Edge Intelligence Features
TensorFlow Lite	iOS, Android, Linux-based SBC	Yes	2019	GPU/TPU acceleration Optimization of TensorFlow models for edge devices
edge-ml	Arduino Nicla Sense ME, Arduino Nano 33 BLE, ESP32, Android	Yes	2020	End-to-end ML workflows for edge devices. Automatic detection of optimal NN architecture
TinyDL	NVIDIA Jetson TK1	Yes	2017	Optimization of deep learning models based on available hardware on the edge device
PyTorch Mobile	iOS, Android, Linux-based SBC	Yes	2019	GPU/NPU acceleration (in beta) Optimization of PyTorch models for edge devices
CoreML	iOS, WatchOS, MacOS	No	2017	GPU/NPU acceleration
ML Kit for Firebase	iOS, Android	No	2018	Unified SDK supporting multiple ML APIs
Apache MXNet	iOS (conversion), Android, Linux-based SBC	Yes	2015	GPU/TPU acceleration Amalgamation support Model conversion e.g., to CoreML
EEL	Raspberry Pi, Arduino, micro:bit	Yes	2017	Support for workflow: compilation-training-deployment to edge
DeepThings	Raspberry Pi	Yes	2018	Improves running inference on a cluster of edge devices
DeepIoT	Intel Edison	Yes	2017	Neural network compression for edge devices

5.3. TinyDL

TinyDL is a framework dedicated to the end-to-end integration of deep learning models into edge-based systems. It was introduced by Rouhani et al. [51]. The TinyDL framework improves the training and execution of the models by profiling and optimizing the computation for the detected platform. It does so by introducing a signal transformation algorithm to better accommodate available resources. In the experiments carried out on the NVidia Jetson TK1 board, the authors validated that TinyDL allows for reducing the training time while keeping the inference accuracy the same for speech recognition, indoor localization, and smart sensing workflows.

5.4. PyTorch Mobile

Another framework that is a mobile version of a popular full-fledged machine learning framework is PyTorch Mobile [52], based on the PyTorch framework. At the time of writing, it is still the beta version, but it is already functional. It aims to train and deploy ML models to edge devices while using a similar toolchain and APIs as when building regular PyTorch projects. It also focuses on privacy preservation by offering dedicated support to federated learning. Currently, it is available for Android, iOS, and Linux operating systems. Support for hardware acceleration is still in development. Currently, there are only nightly releases of the framework that offer support for GPU acceleration on iOS with Metal, GPU support on Android with Vulkan, and NPU and DSP support on Android with NNAPI. While, according to the documentation, the framework is already in wide use, the APIs still might change before reaching a stable release.

5.5. CoreML

CoreML [53] is a closed-source, machine learning framework developed by Apple, Inc. It supports multiple mobile devices, but due to being tied to a single company, it only supports devices such as iPhones, iPads, and Apple Watches. It helps design and train the models from scratch, as well as converting and optimizing models from other frameworks. It is optimized to take advantage of the CPU, GPU, and Neural Engine available on the selected device and also can integrate with existing applications within the Apple ecosystem. It offers Vision, Natural Language, Speech, and Sound Analysis APIs. While not being a multiplatform, it has the benefit of being tightly integrated with the whole Apple ecosystem, both from a software and hardware perspective, which makes it an excellent choice for building models that will run on Apple devices.

5.6. ML Kit for Firebase

ML Kit for Firebase [54] is a library that combines support for Google's ML APIs and custom TensorFlow Lite models into one easy-to-consume SDK (Software Development Kit), allowing developers to use ML-based workflows with little or zero knowledge about machine learning. It is dedicated to Android and iOS operating systems. Without using custom models, it offers support for workflows such as text recognition, face detection, image labeling, object detection, object tracking, barcode scanning, or translation. It can also take advantage of cloud services to improve the performance or accuracy of selected workflows.

5.7. Apache MXNet

Apache MXNet [55,56] is a general-purpose, deep learning framework focused on efficiency and flexibility. It can be used across cloud and edge devices and is designed to be portable and lightweight, supporting acceleration with TPUs and GPUs. It supports multiple programming languages, including Python, Java, Go, Javascript, Perl, Julia, C++, Scala, and Clojure. While being mostly used in cloud environments, it also has support for amalgamation, which is a process for combining all required code along with minimal dependencies into a single file that is easier to deploy and use on smart edge devices.

5.8. Embedded Learning Library (ELL)

The Embedded Learning Library (ELL) [57] is an open-source library and set of tools for designing, developing, and deploying machine learning models to constrained devices such as microcontrollers or single-board computers such as Raspberry Pi. It is currently an early preview version and is being developed by the Microsoft Research team. It is written in C++. However, it also offers an optional interface in Python programming language. The ELL running on a laptop or a virtual machine takes care of the compilation, training, and deployment of models to an edge device, which can later run the prepared model fully independently. It is important to note that the ELL is still under development, and the APIs and functionalities it offers might change in the future.

5.9. DeepThings

DeepThings is a deep learning framework introduced by Zhao et al. [58]. It aims to improve the performance of running CNN-based inference workloads on clusters of resource-constrained devices in edge-based architecture. In order to achieve that, it introduces the Fused Tile Partitioning method that divides convolutional layers into separate tasks that can be distributed across devices in an edge cluster. It also offers a dedicated runtime system for such clusters to better distribute the partitions, especially in dynamic systems. It is accompanied by a reference implementation in C [59], which was implemented for Raspberry Pi 3 single-board computers.

5.10. DeepIoT

DeepIoT is a library developed and presented by Yao et al. [60] in their paper. The authors proposed a solution for compressing the neural networks by reducing the number of required hidden elements without compromising the accuracy of the network. The compressed models can then be successfully deployed and used on constrained devices without additional modifications and adjustments. The authors also performed experiments on platforms such as Intel Edison and confirmed that the use of DeepIoT helped reduce the execution time and energy consumption while maintaining the same accuracy. The authors also provided a reference implementation based on the TensorFlow framework [61].

6. Use Cases

The inherent features of edge computing make this approach an important option when designing the architecture of IT systems built in various areas. However, some areas especially benefit from the advantages of this computing model. These areas include manufacturing, healthcare, smart cities, the environment, and autonomous and unmanned devices. We will focus on these areas in this section.

6.1. Industrial Applications

A combination of machine learning and edge computing has been highlighted by Hu et al. [62] in their work, where the authors described the "iRobot-Factory", which is an intelligent robot factory that takes advantage of edge computing techniques. In the proposed system, the edge node can carry out analysis with machine learning/deep learning models, which, according to the authors, improves service scheduling and helps with the efficient management of edge computing resources. The authors also highlighted that offloading some analytical operations to the edge tier reduces the delay in executing real-time monitoring and allows for achieving autonomous monitoring and more dynamic feedback.

Another industrial application has been presented by Boguslawski et al. [63] in their paper. The authors described an analytical solution for the predictive maintenance of rod pumps with machine learning models deployed directly at the edge. The proposed solution is capable of working fully offline, without any Internet access, while at the same time preserving data security requirements specific to the gas and oil industry. In order to deploy prepared intelligence models, the authors used Microsoft IoT Edge, Docker, and Ubuntu Core. It is worth noting that the authors decided that using multiple models instead of

one is feasible and additionally improves the performance of the system. Thanks to the proposed solution, it was possible to detect such failures as a fluid pound or gas interference. Edge intelligence can also help ensure the reliability and resilience of the power grid. One such example has been proposed and described by Matthews et al. [64]. In their works, the authors proposed an edge-based system for anomaly detection in synchrophasor data, allowing the system to detect anomalies quicker and be more resilient to network outages.

The following example of using machine learning algorithms directly on edge devices in an industrial environment has been presented by Don et al. [65]. In their works, the authors highlighted the issue of reliable video streaming and proposed an algorithm to improve that reliability by intelligently processing the stream and discarding the video frames that are considered unnecessary, given current network conditions. Using an edge server for that processing enables better detection of the network quality from the perspective of a user that is receiving the video stream.

Another architecture that combines edge and machine learning techniques in an industrial setting has been proposed by Zhang et al. [66], where the authors presented an architecture for federated learning across multiple devices based on edge computing and blockchain technology for authentication. In the proposed architecture, each factory has a dedicated edge server responsible for training models on local data from the factory before sending them to a centralized server, where the data are combined with models from other factories and distributed back to the edge servers and end devices. Using that technique allows for achieving better accuracy and preservation of the security and privacy of the data of individual factories. Another use of federated learning in the industrial application has been recently presented by Shubyn et al. [67]. The authors introduced a federated learning-based algorithm for anomaly detection for autonomous guided vehicles in factories. Using that technique improves prediction accuracy and also helps preserve the security of local data. Yet another use of federated learning techniques in the industry has been proposed and described by Liu et al. [68]. The authors introduced a framework for anomaly detection for time-series data in the Industrial Internet of Things based on federated learning and deep anomaly detection. In the experiments, the authors proved that the proposed approach could be used for processing power demand or engine data sets.

Another promising use case in the industry for the edge computing and machine learning combination is visual inspection of conveyor belts. Zeng et al. [69] introduced "Boomerang", which is a framework for cooperative deep neural network inference for edge devices in industrial settings. In the proposed approach, the model deployed on each edge device is adjusted and optimized for the specific inference scenario, further improving accuracy. The inference is run both on the IoT device and edge server in a cooperative manner for further improvement in the performance of the whole system. The authors cited that such an approach can result in improved manufacturing productivity and reduced operational costs. A similar system has also been proposed by Li et al. [70] in their work. The authors introduced a manufacturing visual inspection system for defect detection. However, they used fog intelligence nodes instead of edge servers and cooperative inference between IoT and the edge server. Moreover, sensors only pass raw data directly to intelligence nodes for processing.

Another important use case is predictive maintenance and real-time fault detection. In their works, Park et al. [71] presented a lightweight, edge-based fault detection system for industrial robot manipulators called "LiRed". The system has two parts: an edge device and a central server. The central server is responsible for ingesting data and training LSTM-based inference models, while the edge device runs the inference and raises alarms via the fault monitor interface. According to the authors, using an edge-based approach improves the analysis speed and reduces data processing costs.

6.2. Healthcare Applications

A combination of the edge computing paradigm with machine learning techniques can also significantly improve healthcare systems. One such example has been presented

by Mohan et al. [72]. The authors proposed a Convolutional Neural Network architecture optimized for resource-constrained devices capable of detecting whether people are wearing medical face masks. This is a very fitting use case, given the COVID-19 pandemic. The authors were able to optimize the model built with the TensorFlow Lite framework in a way that it can run successfully on a low-powered ARM Cortex-M7 microcontroller. While the model runs on a microcontroller, the authors opted to train the models in the cloud, using Kaggle kernels with TPU acceleration to shorten the training time. The authors concluded the paper with experiments that show over 99 percent accuracy for the tested data set.

A similar approach has been proposed by Faleh et al. [73]. The authors introduced a system based on a more powerful device, the Jetson Nano board, capable of performing mask recognition in real time directly at the edge. The authors used the MobileNetV2 classifier that was trained on a central server, and then the model was deployed to be run in an edge manner, connected to an alarm system when a missing mask was detected. Based on the experimental results, the authors were able to achieve over 99% accuracy.

Another use case aimed at improving the ability to detect and diagnose COVID-19 has been presented by Qayyum et al. [74]. The authors proposed a system based on a clustered federated learning approach to perform multi-modal COVID-19 diagnosis directly at the edge, processing X-ray and ultrasound images. Using that approach reduces the data transmission, reduces the processing time, and ensures the privacy of local patients' data. The experiments carried out by the authors confirm that such a system could be successfully used for diagnosis by remote healthcare units.

Still, within the topic of COVID-19, Adhikari and Munusamy [75] present research that proposes an intelligent health monitoring system based on edge computing and machine learning called "iCovidCare". While the two previously presented solutions were based on image processing, "iCovidCare" evaluates patients' health using readings from wearable temperature sensors and fuses them with previous medical records. The application performs classification using the ensemble random forest technique, running directly on edge devices. Based on the experimental results obtained by the authors, the proposed approach achieves over 95% accuracy. The authors highlighted that an edge-based solution allows immediate risk assessment for monitored patients.

Another solution that aims to improve the response to the COVID-19 pandemic has been proposed by Velichko [76]. In his paper, the author presents a novel method for analyzing medical data based on the LogNNet neural network, which allows for calculating risk factors for selected diseases based on medical data. The proposed system has been implemented for COVID-19 risk assessment, as well as for perinatal risk assessment. While the model training was carried out on a centralized server, the final model was optimized to the extent that it was possible to run it even on such a low-powered device as an Arduino microcontroller, being able to achieve over 91% of accuracy while requiring as low as 3 kB of RAM. The presented results clearly highlight that even low-powered devices can be very valuable for clinical decision systems.

Yang et al. [77], in their paper, proposed a robust, featureful, visual healthcare platform based on an intelligent, multi-tiered architecture. The proposed architecture consists of three layers: end devices, edge, and cloud. While the cloud is the central part of the system, the edge nodes are responsible for reducing the amount of data that need to be sent to the central system and improving the efficiency of the whole system. It is achieved by offloading part of the computation directly to the edge nodes, which includes running machine learning-based analytics.

Another use case for intelligent edge devices has been implemented and presented by Mrozek et al. [78] in their paper. The authors introduced a scalable system for large-scale fall detection based on a central system deployed in data centers and mobile phones that serve as edge devices. The edge devices are the key to the system, as they are responsible for data collection and processing. They used machine learning models directly at the edge, allowing them to detect falls more quickly. An edge-based approach also reduces the amount of data that need to be sent to the cloud and enables fall detection to function even

without constant Internet access. Experiments carried out by the authors also confirm that an edge-based model can be successfully used to detect such dangerous events reliably.

Ahmed et al. [79], in their paper, presented another innovative use of edge computing. The authors proposed a smart healthcare monitoring system that allows for the noninvasive detection of patients' discomfort based on the analysis of camera output and the patient's posture and position. The whole analysis is performed directly on the edge device, using a deep learning method called Mask-RCNN. After detection, data are also forwarded to a central system for further processing. Experiments carried out by the authors prove that the system can operate with success, recording 94% true-positive records.

Another interesting healthcare solution has been developed and described by Liu et al. [80]. In their works, the authors proposed a food recognition system for dietary assessment, using deep learning, edge devices in the form of smartphones, and a centralized server. The authors used the edge-based approach to reduce the amount of data that need to be sent to the cloud, improve the response time, and lower energy consumption. However, the actual detection is still performed in the cloud using a CNN-based algorithm. The authors plan to improve their solution to complete the detection fully offline on the edge device, which will make the proposed solution more robust.

In order to enhance the performance of healthcare-related wearables, Xu et al. [81] proposed "DeepWear", which is a deep learning framework dedicated to wearable devices that allows them to offload some of the computational tasks to edge devices such as smartphones. This technique allows for improvement in their performance and reduces energy consumption. It also allows running computations that require more computing power than it can achieve directly on the wearable device. The DeepWear uses TensorFlow and has been evaluated on Android and Android Wear OS systems. The experiments confirm that combining the wearable-smartphone approach allows for lower reaction time and energy consumption compared to fully on-wearable and fully on-smartphone approaches.

Another example of combining wearable devices with edge intelligence has been presented by Pramukantoro and Gofuku [82]. The authors, in their paper, introduced a wearable system based on a Polar H10 device and personal computer that is able to perform real-time heartbeat monitoring, thanks to the use of machine learning algorithms. In the proposed system, Polar H10 communicates via Bluetooth Low Energy with a personal computer that receives readings from the wearable device and evaluates them against a designed machine learning model. In the considered case, the SVM-based model turned out to offer the best accuracy while being able to perform classification in less than one second.

The topic of wearable devices has also been explored by Zanetti et al. [83]. The authors, in their paper, introduced a cognitive workload monitor based on a wearable device running a machine learning workload. The authors introduced several optimizations in order to reduce the model size and the memory requirements in order to run the model directly on the e-Glass wearable device, equipped with a low-power microcontroller from the ARM Cortex-M4 family. The authors concluded the paper with experiments that confirm that the proposed solution can be successfully implemented on the proposed device.

Another platform that takes advantage of edge intelligence has been proposed and presented by Puerta et al. [84]. In their works, the authors implemented and evaluated several machine learning models for seizure detection in cloud, fog, and edge computing scenarios. While training has to be carried out in the cloud environment, the detection can successfully be executed on fog or edge devices, with the Multi-layer Perceptron neural network being most fitting for edge nodes due to its relatively high accuracy and low computational cost.

A common use case for wearable devices is their ability to perform human activity recognition, such as classifying if a person is running or walking. A dedicated framework for deploying human activity recognition tasks directly on wearable devices with deep learning models has been proposed by Coelho et al. [85]. The use of the edge computing paradigm allows to enable real-time detection, reduces energy consumption, and improves the privacy of the user's data. The implemented human activity detection model was

evaluated on the STM32F411VE microcontroller, performing detection with over 97% accuracy, confirming that it would be the suitable choice for running such detection on wearable devices in an energy-efficient manner.

Another use case where the federated learning approach shines is a smart healthcare system responsible for person movement identification based on signals from wearable sensors. Arikumar et al. [86] proposed such a solution, in which they employ edge computing to auto-label the data incoming from sensors, and then use that data to train models using federated learning. This approach allows the processing of all the sensor data on edge servers, without the need to send them to the cloud for processing. Based on the carried out experiments, the taken approach resulted in lowered memory usage and computation costs and allowed to reduce the volume of transmitted data by over 36%.

6.3. Smart Cities and Environmental Applications

Edge computing combined with machine learning is also becoming ubiquitous when it comes to environmental applications. One such example has been highlighted by Zhang et al. [87]. In their article, they presented a microseismic monitoring platform. In the proposed solution, called "Edge-to-Center LearnReduce", the authors took advantage of the edge computing paradigm to resolve the problem of real-time data transmission, which is cited as one of the main challenges. The authors used a solution where the neural network models are trained in a data center and then deployed to edge nodes in order to perform event detection and reduce the data volume before they are sent back to the data centers. Based on the presented experimental results, the use of the described approach resulted in high detection accuracy and significantly reduced data transmission.

Another example has been presented by Kumar et al. [88], who proposed and described a water quality monitoring system based on edge devices. The authors initially deployed edge devices that were responsible for collecting data and forwarding them to the data center where the analysis was performed. In the second step, the machine learning model was trained in a data center and embedded on an edge device to enable sensing and reasoning without any communication with central servers.

The intelligent edge is also important when it comes to enabling smart cities. One such use case has been described in detail by Liu et al. [89]. The authors, in their article, described energy management platforms for smart cities. The architecture consists of energy edge servers and energy cloud servers and aims to solve the problem of efficient energy scheduling. The proposed approach takes advantage of edge servers to run neural network models that, according to the authors, reduce the energy cost and the observed delay compared to the solution that uses only centralized servers.

Another case in which edge intelligence is used to improve energy management has been presented by Cicirelli et al. [90]. In their work, they proposed an edge-based system for reducing the energy cost incurred by in-home appliances. The proposed solution allows users to request a schedule for running predefined tasks on home appliances. To do so, it takes advantage of reinforcement learning algorithms and the COGITO platform on Raspberry Pi devices. The experimental results obtained by the authors prove the feasibility of the proposed architecture.

Another class of use cases in smart city environments where edge-based analytics shines is real-time object detection, identification, and classification. In the paper [91], Ali and Ishak presented a real-time object detection system running on edge IoT devices that can detect ambulances, fire engines, or police cars on the city streets. Their solution is based on the Microsoft Azure Custom Vision model, trained in the cloud environment. The model was then deployed directly on edge devices thanks to Microsoft Azure IoT Edge. The authors highlighted that the selected approach allows for the reduction in overall latency and improves the reaction times of the system. Additionally, by reducing the dependency on centralized cloud services, the potential scalability of the system is also improved, as the devices can operate independently.

An aspect that is also important in the environment of smart cities is the ability to detect dangerous events such as gunshots, sirens, screaming, or shattering glass sounds. In their article, Janjua et al. [92] presented a system called IRESE, which stands for "Intelligent Rare-Event Detection System", that is capable of capturing and identifying such dangerous events. The authors identified that the main issue in such systems is the amount of data that have to be transferred and processed. They decided that using intelligent edge modules would allow for a significant reduction in data volume and reduce the time it takes to detect and identify certain events. In their system, the authors decided to use unsupervised learning algorithms that were implemented directly on the edge devices, in this case, Raspberry Pis, that operate on incoming sound data streams.

Another approach to an emergency detection system has been proposed by Orfanidis et al. [93]. The authors, in their paper, introduced a long-range emergency system based on wearable devices capable of performing edge intelligence tasks with communication relying on a Low-Power Wide Area Network. The designed machine learning model is intended to run directly on an edge device in the form of a microcontroller placed in a shoe. The algorithm is able to detect specific foot gestures and trigger an alarm in case of a detected emergency. The authors experimentally confirmed that the implemented neural network classifier is able to perform detection directly on an ESP32 MCU with 98% accuracy, proving that the system can be used effectively in the defined scenarios.

The following use case in the context of smart cities has been presented by Nikouei et al. [94]. In their research, the authors described an edge-based system for real-time human detection in video streams for surveillance purposes. In the presented works, the authors decided to use a lightweight Convolutional Neural Network that was trained on cloud servers. The network was then optimized and deployed on edge servers. It was highlighted that moving the computation to the edge devices improves communication time and reduces data volume overhead, resulting in reduced latency and quicker system reaction.

A similar use case has been presented by Pang et al. [95]. In their paper, the authors presented a surveillance system that can identify the same person on multiple cameras. The proposed solution is based on a Convolutional Network Architecture that was trained on a dedicated machine but is lightweight enough to run detection routines on edge devices successfully.

Another scenario in which machine learning performed directly at the edge is beneficial is the security of houses and residences. A system for home intrusion monitoring has been introduced by Dhakal and Ramakrishnan in their work [96]. They proposed a fully edge-based system that can operate without a centralized cloud server and can train machine learning models directly at the edge of the network. The experiments carried out by the authors also suggest that a single edge node could support monitoring even thousands of homes at a close distance.

A more complex smart city system called "UrbanEdge" has been proposed by Fan et al. [97]. They proposed a multi-layered architecture that is capable of running prediction algorithms on various time-series data, such as Air Quality Index, traffic volume, electricity usage, or building occupancy data. In their proposal, they suggested a solution with two types of edge servers, one responsible for data preprocessing and basic analytics and another one, more powerful, that can run deep learning-based prediction algorithms.

Yet another surveillance use case has been presented by Sabella [98] in their thesis. The author proposed and implemented a fire and smoke detection system for smart cities based on edge computing, video processing, and deep neural networks. In the implemented system, learning happens on a centralized server, while inference is performed on the edge, using the OpenVINO toolkit, as well as Intel NCS2 and Movidius NCS accelerators. According to the author, using an edge-based solution reduced the time from sending the alert from 30 to 20 s compared to a cloud-based platform.

An innovative approach to taking advantage of edge intelligence has been presented by Silva et al. [99]. Their paper introduced a wearable device in the form of a helmet, equipped

with a camera and an additional edge server that is capable of running edge intelligence workloads for performing leaf disease detection in forests. The authors tested the workflows on various platforms and concluded that Jetson Nano offers the best performance. It also has been confirmed that the proposed device is capable of classifying leaf diseases in the field, with over 90% accuracy.

6.4. Satellite–Terrestrial Integrated Networks

Another area where the combination of edge computing and machine learning is gaining popularity is satellite–terrestrial integrated networks. Such a use case has been presented by Zhu et al. [100] in their works. The authors considered a scenario of task offloading in satellite–terrestrial computing networks. In their article, they described a task-offloading algorithm, based on deep reinforcement learning. The proposed algorithm achieves a performance cost similar to that of existing algorithms. However, it significantly reduces runtime consumption, making it more suitable for satellite–terrestrial edge computing networks.

Another example of edge computing and machine learning in the context of satellite–terrestrial integrated networks has been presented by Jiang et al. [25]. In their work, the authors proposed a split-then-federated learning framework dedicated to satellite–terrestrial networks. It is optimized for handling sequential data and is based on LSTM cells. The authors also evaluated the proposed algorithm based on a specific case study of electricity theft detection. The carried out experiments highlight that the proposed model is an effective solution, offering comparable performance metrics to centralized models.

Yet another case for satellite–terrestrial edge networks has been presented by Zhang et al. [101]. In their paper, the authors introduced a novel, double-edge architecture called DILIGENT, with edge computing devices being deployed across ground stations and satellites. The proposed approach allows for cooperative learning of the system. It also allows for intelligent task offloading, cooperative resource allocation, and more efficient cache and content delivery. In their experimental simulation, the authors highlighted a reduction in the average delay of a processing task of about 40%.

6.5. Autonomous and Intelligence-Assisted Vehicles

A combination of edge computing and machine learning is becoming very important for autonomous and intelligence-assisted vehicles. One such use case has been described by de Prado et al. [102]. In their work, the authors proposed and implemented several machine learning models for steering autonomous mini vehicles. Such an approach aims to reduce reaction time, preserve energy, decrease the volume of transmitted data, and reduce dependency on unreliable connections to remote servers that usually perform such computations. To achieve that, the authors took advantage of tinyML for building machine learning models that are highly optimized for low-powered devices. The created models use data from a linear camera mounted on the devices. While the training of models happens on centralized servers, the prepared models could run entirely offline on edge devices.

Another solution aimed at autonomous vehicles has been presented by Kocić et al. [103]. In their article, the authors focused on an autonomous driving intelligence module that can be successfully used directly by embedded automotive platforms. The implementation uses real-time input from the camera and outputs a steering wheel angle to ensure autonomous driving capabilities. To make it possible to run on embedded devices, the authors proposed a novel architecture called J-Net. During the experiments, it turned out to be faster by up to 280 times than the AlexNet architecture used for comparison. The authors concluded that using lightweight solutions, such as the one developed as a part of their works, enables direct deployment on low-powered embedded devices.

One of the most important objectives when building autonomous cars is ensuring the safety of pedestrians and other vehicles. The system that allows for pedestrian detection for autonomous vehicles has been developed and presented by Navarro et al. [104]. In

their article, the authors described a machine learning approach, using the LIDAR sensor's data, running in an edge computing manner.

A more advanced system aimed at detecting all kinds of road anomalies for autonomous vehicles has been developed by Bibi et al. [105]. The authors implemented a vehicular ad hoc network where autonomous cars can share detected anomalies with each other, further improving road safety. Each autonomous vehicle is equipped with an edge intelligence module that processes camera data and detects anomalies such as potholes, road bumps, and cracks. The deep learning models are trained on centralized servers, but the inference is made directly at the edge.

In their article, Ferdowsi et al. [106] focused on a more complex Intelligent Transportation System that takes advantage of multiple edge analytics nodes equipped with deep learning modules. According to the authors, the use of deep learning techniques allows for the optimization of latency, robustness, and reliability of the proposed system. In the paper, the authors proposed algorithms that help with path planning and controlling autonomous vehicles, or ones that improve the security and privacy of data.

Object detection is essential not only for autonomous cars but also for smaller vehicles like autonomous robots. An approach for object detection for such small robots has been introduced by Hu et al. [107]. In their article, the authors presented a novel object detection algorithm based on YOLOv4 and GhostNet, performing training on a centralized server and then deploying the trained model to run directly on an edge device, in this case, Jetson TX2. In their work, Febbo et al. [108] also focused on small autonomous robots. In their paper, the authors described how they designed both the robot and the control system for it, based on a deep neural network model, running directly on board the vehicle, in this case on Jetson Nano. The authors also took advantage of the transfer learning technique to speed up the learning process.

Another class of autonomous vehicles is drones, which also require edge-based intelligence for navigation purposes. One such system has been presented by Palossi et al. [109]. In their paper, the authors described a deep neural network navigation solution for autonomous nano-drones. Due to their size, more powerful edge devices such as Jetson Nano or Raspberry Pi cannot be used, which poses an even more significant challenge regarding implementation. The author's implementation can execute on COTS Crazyflie 2.0 nano-drone while consuming as low as 64 [mW], processing data from the onboard ultralow-power camera. The author's development enables such nano-drones to be autonomous and carry out various tasks such as sensing and data collection.

Drones are also often used to help manage various disasters by assessing risks, providing real-time data about emergencies, and helping to understand better the scope of the issues, which in the end, often can result in saving people's lives. Alsamhi et al. [110] presented their research on a robust system for data sharing between intelligent drones and smart wearables used during disasters, highlighting how valuable such drones are in various environmental disasters due to their capabilities to carry out further analysis directly on the device, in an edge-based manner, including running machine learning models. The authors evaluate the impact of network connectivity in the case of such events and highlight that properly optimized transmission is crucial in enabling search–rescue teams to operate efficiently.

One of the most critical tasks drones can carry out, especially in disastrous scenarios, is surveillance with onboard cameras. However, doing so in real time can be challenging due to unreliable network connections. To tackle that issue, Wang et al. [111] proposed an architecture that allows for bandwidth-efficient real-time video analytics in an edge-based manner. The authors proposed a system where a swarm of drones can store all needed video footage and preprocess it in a way that would significantly reduce the amount of data that need to be sent for further processing. The preprocessing pipeline can also take advantage of deep learning techniques. In the paper, the authors introduced the use of just-in-time learning directly on the drones to optimize the preprocessing pipeline and further reduce the amount of data that need to be sent for additional processing. Then,

the prepared payload is sent to an edge server capable of performing full analytics of the footage, taking advantage of deep neural network models.

7. Trends and Future Developments in Edge Computing

The landscape of edge computing is evolving rapidly, propelled by technological advancements and changing business needs. Key trends and future developments include the widespread expansion of edge infrastructure, driven by the proliferation of edge nodes and micro data centers at the network edge. This expansion is closely intertwined with the convergence of edge computing and 5G networks, which promise ultra-low latency and high-bandwidth communication [112], enabling real-time applications like autonomous vehicles and smart cities [113]. Additionally, the integration of artificial intelligence (AI) and machine learning (ML) at the edge will enable faster decision-making and improved operational efficiency [114]. The emergence of hybrid cloud–edge architectures will facilitate the seamless integration and orchestration of computing resources across distributed environments, optimizing workload placement and resource utilization. There will also be a heightened focus on enhancing security and privacy measures at the edge, including advanced encryption and authentication mechanisms [115]. The development of edge–native applications and services tailored for specific use cases and industries will accelerate, delivering low-latency, high-performance experiences to end users. Moreover, edge orchestration platforms and management tools will become more sophisticated, enabling the efficient provisioning, monitoring, and management of edge resources [116]. Edge computing will play a crucial role in IoT and industrial IoT (IIoT) deployments, enabling real-time data processing, analysis, and control [117]. Efforts to establish industry standards and promote interoperability between edge devices and platforms will intensify, fostering collaboration and innovation. With these trends, edge computing is poised to drive innovation, transform industries, and unlock new opportunities for businesses and organizations to leverage its power for competitive advantage and value creation.

8. Discussion

Our review shows that edge computing has gained much popularity in the last decade and, next to cloud computing, is now an important computing model that should be considered while planning the construction of modern sensor-based and IoT-based systems. This approach can be supportive for centralized computations and data processing, offloading the data centers from performing time-consuming operations for large numbers of data batches generated by thousands of IoT devices, industrial, institutional, social, and personal. The constantly increasing compute capabilities of edge devices make them useful platforms not only for storing and processing local data but also for inferencing with sophisticated machine learning models. Local data processing combined with the central one increases the possibilities for scaling the solution proportionally to the IoT installations mounted in a particular area. Locally performed computations with edge devices may also increase the speed of decision-making while decreasing the intensity of communication and reducing data transfers. This leads to significant savings in energy consumption, which is especially necessary for edge devices working in difficult environmental conditions. Finally, several use cases have confirmed that edge computing plays a vital role in data security and privacy, which is critical not only in smart home solutions but also in sensitive areas of manufacturing and extraction of natural resources.

Table 2 summarizes the application areas of edge computing and the benefits achieved from using it in particular use cases. Based on our survey, we can see that edge intelligence already plays an important role in healthcare systems, enables autonomous vehicles, enables smart cities, and enhances industrial applications.

Our review has also revealed that there are still several challenges for edge computing. IoT units working as edge devices still have limited computing capabilities compared to traditional workstations and even virtual machines in the cloud that can be scaled vertically and horizontally. This requires the prepared implementations to be tailored to the

capabilities of a particular edge device, sometimes requiring optimizations and trade-offs that might result in reduced accuracy or degraded performance. The energy consumption of edge devices is one of the main challenges that should be considered due to limited battery capacities or power constraints flowing from the larger platforms the edge devices are mounted on (e.g., autonomous guided vehicles). Finally, management of the fleet of edge devices requires well-designed frameworks that implement industry standards and can register, identify, and monitor the state of the whole fleet of connected edge devices.

Table 2. Summary of use cases for edge computing and machine learning

Paper	Use Case	Devices/Platforms	Benefits
Hu et al. [62]	Intelligent robot factory	Robot with Android Edge server CentOS 7 Cloud Server Ubuntu 16.04	Reduced delay in real-time monitoring Improved recognition accuracy Improved resource management
Boguslawski et al. [63]	Predictive maintenance of rod pumps	Microsoft IoT Edge Docker, Ubuntu Core	Ability to operate offline
Matthews et al. [64]	Synchrophasor data analysis	Jetson Nano	Improved performance Reduced energy consumption
Don et al. [65]	Video streaming	Edge server with Tesla V100 GPU	Improved detection of network quality Improved reliability of video stream
Zhang et al. [66]	IIoT authentication	Simulation	Improved data security and privacy
Shubyn et al. [67]	Predictive maintenance	Simulation	Improved prediction accuracy Improved data security
Liu et al. [68]	IIoT anomaly detection	Virtual edge server with Ubuntu 18.04	Reduced communication overhead
Zeng et al. [69]	IIoT image recognition	Raspberry Pi 3 Desktop PC edge server	Reduced latency Improved performance
Li et al. [70]	IIoT defect detection	N/A	Improved computing efficiency
Park et al. [71]	Predictive maintenance Real-time fault detection	Raspberry Pi 3	Improved performance Reduced data analysis costs
Mohan et al. [72]	Face mask detection	ARM Cortex M7 microcontroller	Ability to run on a microntroller Over 99% accuracy
Faleh et al. [73]	Face mask detection	Jetson Nano	Ability to run on an edge device Over 99% accuracy
Qayuum et al. [74]	COVID-19 diagnostics	N/A	Reduced data transmission Reduced processing time Improved data privacy
Adhikari et al. [75]	COVID-19 health monitoring system	N/A	Improved accuracy Allows for immediate risk assessment
Velichko [76]	Diseases risk assessment	Arduino Uno Arduino Nano	Ability to run detection on low-powered devices
Yang et al. [77]	Visual healthcare platform	N/A	Reduced amount of data sent Improved efficiency
Mrozek et al. [78]	Fall detection system	iPhone 8	Reduced latency Reduced data transfer
Ahmed et al. [79]	Monitoring patients' discomfort	N/A	Allows for noninvasive monitoring

Table 2. *Cont.*

Paper	Use Case	Devices/Platforms	Benefits
Liu et al. [80]	Food recognition for dietary assessment	Android 6.0.1 device Edge server with CentOS 7	Improved response time Reduced data transfer
Xu et al. [81]	Healthcare-related wearable devices	Nexus 6 with Android 7 LG Watch Urbane	Improved reaction time Reduced energy consumption
Pramukantoro et al. [82]	Real-time heartbeat monitoring	Polar H10 Desktop PC edge server	Ability to perform classification at the edge
Zanetti et al. [83]	Cognitive workload monitoring	eGlass ARM Cortex-M4	Ability to perform classification at the edge
Puerta et al. [84]	Seizure detection	N/A	Low computational cost High accuracy
Coelho et al. [85]	Human activity recognition	STM32F411VE	Ability to perform classification at the edge Energy efficiency
Arikumar et al. [86]	Person movement identification	N/A	Reduced computation cost Reduced memory usage Reduced data transmission
Zhang et al. [87]	Microseismic monitoring platform	Xilinx FPGA Intel-based edge server	Reduced data transmission
Kumar et al. [88]	Water monitoring	AquaSense Sensor Arduino Uno	No need to communicate with central servers
Liu et al. [89]	Smart city energy management	N/A	Reduced cost Reduced latency
Cicirelli et al. [90]	Home energy management	Raspberry Pi	Reduced energy consumption
Ali et al. [91]	Real-time object detection	Raspberry Pi Azure IoT Edge	Reduced latency Improved scalability
Janjua et al. [92]	Dangerous event detection in smart cities	Raspberry Pi	Reduced data transmission Reduced latency
Orfanidis et al. [93]	Long-range emergency system	ESP32	Ability to run detection on an edge device
Nikouei et al. [94]	Real-time human detection in video streams	Raspberry Pi 3	Reduced data transmission Reduced latency
Pang et al. [95]	Surveillance	N/A	Improved performance
Dhakal et al. [96]	Home intrusion monitoring	OpenNet VM	Ability to operate without central server
Sabella [98]	Fire and smoke detection	Intel NCS2 Movidius NCS Intel-based edge server	Improved response time
Silva et al. [99]	Leaf disease detection	Jetson Nano	Ability to operate without centralized service
de Prado et al. [102]	Steering mini autonomous vehicles	STM32L4 GAP8 NXP k64f	Reduced reaction time Reduced data transmission Reduced energy consumption
Kocic et al. [103]	Steering autonomous vehicles	Desktop PC edge server	Ability to execute on edge devices
Navarro et al. [104]	Pedestrian detection	N/A	Improved performance

Table 2. *Cont.*

Paper	Use Case	Devices/Platforms	Benefits
Bibi et al. [105]	Vehicular ad-hoc network for anomalies	Simulation	Improved road safety
Ferdowsi et al. [106]	Intelligent transportation system (ITS)	N/A	Reduced latency Improved reliability
Hu et al. [107]	Object detection for autonomous vehicles	Jetson TX2	Improved detection performance
Febbo et al. [108]	Autonomous robots	Jetson Nano	Ability to operate without centralized service
Palossi et al. [109]	Autonomous nano-drones	GAP8 COTS Crazyflie 2.0	Ability to operate autonomously
Alsamhi et al. [110]	Data sharing between drones and wearables	Simulation	Optimized data transmission Reduced latency
Zhu et al. [100]	Task offloading in satellite-terrestrial computing networks	Simulation	Reduced runtime consumption
Jiang et al. [25]	Electricity theft detection	Simulation	Ability to take advantage of federated learning
Zhang et al. [101]	Task offloading cache content delivery	Simulation	Reduced average delay of task processing
Wang et al. [111]	Drone surveillance	Intel Aero Drone Jetson TX2	Reduced data transmission

Author Contributions: Conceptualization, P.G. and D.M.; methodology, formal analysis, D.M.; investigation, P.G.; resources, P.G. and D.M.; writing—original draft preparation, P.G. and D.M.; writing—review and editing, P.G. and D.M.; visualization, P.G.; supervision, D.M.; project administration, D.M.; funding acquisition, D.M. All authors have read and agreed to the published version of the manuscript.

Funding: The research was supported by the Norway Grants 2014–2021 operated by the National Centre for Research and Development under the project Automated Guided Vehicles integrated with Collaborative Robots for Smart Industry Perspective (Project Contract no. NOR/POLNOR/CoBotAGV/0027/2019-00), partially by a pro-quality grant for highly scored publications or issued patents (grant No 02/100/RGJ23/0026), Statutory Research funds of Department of Applied Informatics, Silesian University of Technology, Gliwice, Poland (grants No. 02/100/BK_24/0035), Polish Ministry of Science and Higher Education as a part of the CyPhiS program at the Silesian University of Technology, Gliwice, Poland (Contract No. POWR.03.02.00-00-I007/17-00), and partially by the ReActive Too project that has received funding from the European Union's Horizon 2020 Research, Innovation and Staff Exchange Programme under the Marie Skłodowska-Curie Action (Grant Agreement No. 871163).

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

Abbreviations

The following abbreviations are used in this manuscript:

AGV	Autonomous Guided Vehicles
AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Programming Interface
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DL	Deep Learning

DSP	Digital Signal Processing
ELL	Embedded Learning Library
GPU	Graphics Processing Unit
IIoT	Industrial Internet of Things
IoT	Internet of Things
LSTM	Long Short-Term Memory
MES	Manufacturing Execution System
ML	Machine Learning
MLP	Multi-Layer Perception
MQTT	Message Queuing Telemetry Transport
NLP	Natural Language Processing
NPU	Neural Processing Unit
OPC UA	OPC Unified Architecture
PdM	Predictive Maintenance
PLC	Programmable Logical Controller
PSO	Particle Swarm Optimization
RFID	Radio-Frequency Identification
RNN	Recurrent Neural Network
SCADA	Supervisory Control And Data Acquisition
SDK	Software Development Kit
SQL	Structured Query Language
STFT	Short-Time Fourier Transform
SVM	Support Vector Machine
TPU	Tensor Processing Unit
TSDB	Time-Series Database
UML	Unified Modeling Language
VPU	Visual Processing Unit
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

References

1. Paul, A.; Pinjari, H.; Hong, W.H.; Seo, H.; Rho, S. Fog Computing-Based IoT for Health Monitoring System. *J. Sens.* **2018**, *2018*, 1386470. [CrossRef]
2. Krasniqi, X.; Hajrizi, E. Use of IoT Technology to Drive the Automotive Industry from Connected to Full Autonomous Vehicles. *IFAC-PapersOnLine* **2016**, *49*, 269–274. [CrossRef]
3. Renart, E.G.; Diaz-Montes, J.; Parashar, M. Data-Driven Stream Processing at the Edge. In Proceedings of the 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, 14–15 May 2017 ; pp. 31–40. [CrossRef]
4. Liu, X.; Nielsen, P. Air Quality Monitoring System and Benchmarking. In *Big Data Analytics and Knowledge Discovery*; Springer: Cham, Switzerland, 2017; pp. 459–470. [CrossRef]
5. Fadhel, M.; Sekerinski, E.; Yao, S. A Comparison of Time Series Databases for Storing Water Quality Data. In *Mobile Technologies and Applications for the Internet of Things*; Springer: Cham, Switzerland, 2019; pp. 302–313. [CrossRef]
6. Greco, L.; Ritrovato, P.; Xhafa, F. An edge-stream computing infrastructure for real-time analysis of wearable sensors data. *Future Gener. Comput. Syst.* **2019**, *93*, 515–528. [CrossRef]
7. Singh, S. Optimize cloud computations using edge computing. In Proceedings of the 2017 International Conference on Big Data, IoT and Data Science (BID), Pune, India, 20–22 December 2017; pp. 49–53. [CrossRef]
8. Khan, L.U.; Yaqoob, I.; Tran, N.H.; Kazmi, S.M.A.; Dang, T.N.; Hong, C.S. Edge-Computing-Enabled Smart Cities: A Comprehensive Survey. *IEEE Internet Things J.* **2020**, *7*, 10200–10232. [CrossRef]
9. Dong, P.; Ning, Z.; Obaidat, M.S.; Jiang, X.; Guo, Y.; Hu, X.; Hu, B.; Sadoun, B. Edge Computing Based Healthcare Systems: Enabling Decentralized Health Monitoring in Internet of Medical Things. *IEEE Netw.* **2020**, *34*, 254–261. [CrossRef]
10. Singh, A.; Chatterjee, K. Securing smart healthcare system with edge computing. *Comput. Secur.* **2021**, *108*, 102353. [CrossRef]
11. Stankovski, S.; Ostojić, G.; Baranovski, I.; Babić, M.; Stanojević, M. The Impact of Edge Computing on Industrial Automation. In Proceedings of the 2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH), East Sarajevo, Bosnia and Herzegovina, 18–20 March 2020; pp. 1–4. [CrossRef]
12. Benecki, P.; Kostrzewa, D.; Grzesik, P.; Shubyn, B.; Mrozek, D. Forecasting of Energy Consumption for Anomaly Detection in Automated Guided Vehicles: Models and Feature Selection. In Proceedings of the 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 9–12 October 2022.
13. Le, N.; Ou, Y. Incorporating efficient radial basis function networks and significant amino acid pairs for predicting GTP binding sites in transport proteins. *Bmc Bioinform.* **2016**, *17* (Suppl. S19), 183–192. [CrossRef]

14. Le, N.Q.K.; Nguyen, T.T.D.; Ou, Y.Y. Identifying the molecular functions of electron transport proteins using radial basis function networks and biochemical properties. *J. Mol. Graph. Model.* **2017**, *73*, 166–178. [CrossRef]
15. Cupek, R.; Drewniak, M.; Fojcik, M.; Kyrkjebø, E.; Lin, J.C.W.; Mrozek, D.; Øvsthus, K.; Ziebinski, A. Autonomous Guided Vehicles for Smart Industries–The State-of-the-Art and Research Challenges. In Proceedings of the Computational Science–ICCS 2020, Amsterdam, The Netherlands, 3–5 June 2020; pp. 330–343.
16. Grzesik, P.; Benecki, P.; Kostrzewa, D.; Shubyn, B.; Mrozek, D. On-Edge Aggregation Strategies over Industrial Data Produced by Autonomous Guided Vehicles. In Proceedings of the Computational Science–ICCS 2022, London, UK, 21–23 June 2022; Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A., Eds.; Springer: Cham, Switzerland, 2022; pp. 458–471.
17. Steclik, T.; Cupek, R.; Drewniak, M. Automatic grouping of production data in Industry 4.0: The use case of internal logistics systems based on Automated Guided Vehicles. *J. Comput. Sci.* **2022**, *62*, 101693. [CrossRef]
18. Wang, J.; Pan, J.; Esposito, F.; Calyam, P.; Yang, Z.; Mohapatra, P. Edge Cloud Offloading Algorithms: Issues, Methods, and Perspectives. *ACM Comput. Surv.* **2019**, *52*, 1–23. [CrossRef]
19. Grzesik, P.; Mrozek, D. Accelerating Edge Metagenomic Analysis with Serverless-Based Cloud Offloading. In Proceedings of the Computational Science–ICCS 2022, London, UK, 21–23 June 2022; Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A., Eds.; Springer: Cham, Switzerland, 2022; pp. 481–492.
20. Mach, P.; Becvar, Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1628–1656. [CrossRef]
21. Ribeiro, S.L.; Nakamura, E.T. Privacy Protection with Pseudonymization and Anonymization In a Health IoT System: Results from OCARIoT. In Proceedings of the 2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE), Athens, Greece, 28–30 October 2019; pp. 904–908. [CrossRef]
22. Silveira, M.M.; Portela, A.L.; Menezes, R.A.; Souza, M.S.; Silva, D.S.; Mesquita, M.C.; Gomes, R.L. Data Protection based on Searchable Encryption and Anonymization Techniques. In Proceedings of the NOMS 2023–2023 IEEE/IFIP Network Operations and Management Symposium, Miami, FL, USA, 8–12 May 2023; pp. 1–5. [CrossRef]
23. Ma, R.; Feng, T.; Fang, J. Edge Computing Assisted an Efficient Privacy Protection Layered Data Aggregation Scheme for IIoT. *Secur. Commun. Netw.* **2021**, *2021*, 7776193. [CrossRef]
24. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19. [CrossRef]
25. Jiang, W.; Han, H.; Zhang, Y.; Mu, J. Federated split learning for sequential data in satellite–terrestrial integrated networks. *Inf. Fusion* **2024**, *103*, 102141. [CrossRef]
26. Pasquini, D.; Francati, D.; Ateniese, G. Eluding Secure Aggregation in Federated Learning via Model Inconsistency. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022; CCS'22; pp. 2429–2443. [CrossRef]
27. Tao, Y.; Xu, P.; Jin, H. Secure Data Sharing and Search for Cloud-Edge-Collaborative Storage. *IEEE Access* **2020**, *8*, 15963–15972. [CrossRef]
28. Zheng, K.; Ding, C.; Wang, J. A Secure Data-Sharing Scheme for Privacy-Preserving Supporting Node–Edge–Cloud Collaborative Computation. *Electronics* **2023**, *12*, 2737. [CrossRef]
29. Roman, R.; Lopez, J.; Mambo, M. Mobile edge computing, Fog et al.: A survey and analysis of security threats and challenges. *Future Gener. Comput. Syst.* **2018**, *78*, 680–698. [CrossRef]
30. Yahuza, M.; Idris, M.Y.I.B.; Wahab, A.W.B.A.; Ho, A.T.S.; Khan, S.; Musa, S.N.B.; Taha, A.Z.B. Systematic Review on Security and Privacy Requirements in Edge Computing: State of the Art and Future Research Opportunities. *IEEE Access* **2020**, *8*, 76541–76567. [CrossRef]
31. 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] [PubMed]
32. Xu, J.; Chen, L.; Ren, S. Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing. *IEEE Trans. Cogn. Commun. Netw.* **2017**, *3*, 361–373. [CrossRef]
33. Mao, Y.; Zhang, J.; Letaief, K.B. Dynamic Computation Offloading for Mobile-Edge Computing With Energy Harvesting Devices. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 3590–3605. [CrossRef]
34. Ku, Y.J.; Chiang, P.H.; Dey, S. Quality of Service Optimization for Vehicular Edge Computing with Solar-Powered Road Side Units. In Proceedings of the 2018 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, 30 July 2018–2 August 2018; pp. 1–10. [CrossRef]
35. Li, B.; Fei, Z.; Shen, J.; Jiang, X.; Zhong, X. Dynamic Offloading for Energy Harvesting Mobile Edge Computing: Architecture, Case Studies, and Future Directions. *IEEE Access* **2019**, *7*, 79877–79886. [CrossRef]
36. Zhou, H.; Jiang, K.; Liu, X.; Li, X.; Leung, V.C.M. Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing. *IEEE Internet Things J.* **2022**, *9*, 1517–1530. [CrossRef]
37. Donta, P.K.; Dustdar, S. Towards Intelligent Data Protocols for the Edge. In Proceedings of the 2023 IEEE International Conference on Edge Computing and Communications (EDGE), Chicago, IL, USA, 2–8 July 2023. [CrossRef]
38. Li, B.; He, Q.; Cui, G.; Xia, X.; Chen, F.; Jin, H.; Yang, Y. READ: Robustness-Oriented Edge Application Deployment in Edge Computing Environment. *IEEE Trans. Serv. Comput.* **2022**, *15*, 1746–1759. [CrossRef]

39. Song, H.; Dautov, R.; Ferry, N.; Solberg, A.; Fleurey, F. Model-Based Fleet Deployment of Edge Computing Applications. In Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems, Virtual Event, 16–23 October 2020; MODELS'20; pp. 132–142. [CrossRef]
40. Wang, E.; Li, D.; Dong, B.; Zhou, H.; Zhu, M. Flat and hierarchical system deployment for edge computing systems. *Future Gener. Comput. Syst.* **2020**, *105*, 308–317. [CrossRef]
41. Microsoft Azure IoT Edge Documentation. Available online: https://azure.microsoft.com/en-us/products/iot-edge/ (accessed on 24 November 2022).
42. AWS IoT Greengrass Documentation. Available online: https://docs.aws.amazon.com/greengrass/index.html (accessed on 24 November 2022).
43. Applying Federated Learning for ML at the Edge. Available online: https://aws.amazon.com/blogs/architecture/applying-federated-learning-for-ml-at-the-edge/ (accessed on 24 November 2022).
44. Balena Documentation. Available online: https://www.balena.io/docs/learn/welcome/primer/ (accessed on 24 November 2022).
45. Balena Labs Projects Repository. Available online: https://github.com/balena-labs-projects (accessed on 24 November 2022).
46. KubeEdge Documentation. Available online: https://kubeedge.io/en/ (accessed on 24 November 2022).
47. Wang, S.; Hu, Y.; Wu, J. KubeEdge.AI: AI Platform for Edge Devices. *arXiv* **2020**, arXiv:2007.09227.
48. EdgeX Foundry Documentation. Available online: https://www.edgexfoundry.org/why-edgex/ (accessed on 24 November 2022).
49. TensorFlow Lite Documentation. Available online: https://www.tensorflow.org/lite (accessed on 24 November 2022).
50. Röddiger, T.; King, T.; Lepold, P.; Münk, J.; Du, S.; Riedel, T.; Beigl, M. edge-ml.org-End-To-End Embedded Machine Learning. Available online: https://edge-ml.org/ (accessed on 24 November 2022).
51. Darvish Rouhani, B.; Mirhoseini, A.; Koushanfar, F. TinyDL: Just-in-time deep learning solution for constrained embedded systems. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4. [CrossRef]
52. PyTorch Mobile Documentation. Available online: https://pytorch.org/mobile/home/ (accessed on 24 November 2022).
53. CoreML Documentation. Available online: https://developer.apple.com/documentation/coreml (accessed on 24 November 2022).
54. ML Kit for Firebase Documentation. Available online: https://firebase.google.com/docs/ml-kit. (accessed on 24 November 2022).
55. 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.
56. Apache MXNet Documentation. Available online: https://mxnet.apache.org/versions/1.9.0/api (accessed on 24 November 2022).
57. Microsoft Embedded Learning Library Documentation. Available online: https://microsoft.github.io/ELL/ (accessed on 24 November 2022).
58. Zhao, Z.; Barijough, K.M.; Gerstlauer, A. DeepThings: Distributed Adaptive Deep Learning Inference on Resource-Constrained IoT Edge Clusters. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2348–2359. [CrossRef]
59. Reference Implementation in C of DeepThigns Framework. Available online: https://github.com/zoranzhao/DeepThings (accessed on 24 November 2022).
60. Yao, S.; Zhao, Y.; Zhang, A.; Su, L.; Abdelzaher, T. DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. In Proceedings of the Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems, Delft, The Netherlands, 6–8 November 2017; SenSys'17. [CrossRef]
61. Reference Implementation of DeepIoT Framework. Available online: https://github.com/yscacaca/DeepIoT (accessed on 24 November 2022).
62. Hu, L.; Miao, Y.; Wu, G.; Hassan, M.M.; Humar, I. iRobot-Factory: An intelligent robot factory based on cognitive manufacturing and edge computing. *Future Gener. Comput. Syst.* **2019**, *90*, 569–577. [CrossRef]
63. Boguslawski, B.; Boujonnier, M.; Bissuel-Beauvais, L.; Saghir, F.; Sharma, R. IIoT Edge Analytics: Deploying Machine Learning at the Wellhead to Identify Rod Pump Failure. In Proceedings of the SPE Middle East Artificial Lift Conference and Exhibition, Manama, Bahrain, 28–29 November 2018. [CrossRef]
64. Matthews, S.J.; Leger, A.S. Energy-Efficient Analysis of Synchrophasor Data using the NVIDIA Jetson Nano. In Proceedings of the 2020 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 22–24 September 2020; pp. 1–7. [CrossRef]
65. Dou, W.; Zhao, X.; Yin, X.; Wang, H.; Luo, Y.; Qi, L. Edge Computing-Enabled Deep Learning for Real-time Video Optimization in IIoT. *IEEE Trans. Ind. Inform.* **2021**, *17*, 2842–2851. [CrossRef]
66. Zhang, P.; Sun, H.; Situ, J.; Jiang, C.; Xie, D. Federated Transfer Learning for IIoT Devices With Low Computing Power Based on Blockchain and Edge Computing. *IEEE Access* **2021**, *9*, 98630–98638. [CrossRef]
67. Shubyn, B.; Mrozek, D.; Maksymyuk, T.; Sunderam, V.; Kostrzewa, D.; Grzesik, P.; Benecki, P. Federated Learning for Anomaly Detection in Industrial IoT-Enabled Production Environment Supported by Autonomous Guided Vehicles. In Proceedings of the Computational Science–ICCS 2022: 22nd International Conference, London, UK, 21–23 June 2022; Proceedings, Part IV; Springer: Cham, Switzerland , 2022; pp. 409–421. [CrossRef]
68. Liu, Y.; Garg, S.; Nie, J.; Zhang, Y.; Xiong, Z.; Kang, J.; Hossain, M.S. Deep Anomaly Detection for Time-Series Data in Industrial IoT: A Communication-Efficient On-Device Federated Learning Approach. *IEEE Internet Things J.* **2021**, *8*, 6348–6358. [CrossRef]
69. Zeng, L.; Li, E.; Zhou, Z.; Chen, X. Boomerang: On-Demand Cooperative Deep Neural Network Inference for Edge Intelligence on the Industrial Internet of Things. *IEEE Netw.* **2019**, *33*, 96–103. [CrossRef]

70. Li, L.; Ota, K.; Dong, M. Deep Learning for Smart Industry: Efficient Manufacture Inspection System With Fog Computing. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4665–4673. [CrossRef]
71. Park, D.; Kim, S.; An, Y.; Jung, J.Y. LiReD: A Light-Weight Real-Time Fault Detection System for Edge Computing Using LSTM Recurrent Neural Networks. *Sensors* **2018**, *18*, 2110. [CrossRef] [PubMed]
72. Mohan, P.; Paul, A.; Chirania, A. A Tiny CNN Architecture for Medical Face Mask Detection for Resource-Constrained Endpoints. In *Innovations in Electrical and Electronic Engineering*; Springer: Singapore, 2021; pp. 657–670. [CrossRef]
73. Faleh, N.; Abdul Hassan, N.; Abed, A.; Abdalla, T. Face mask detection using deep learning on NVIDIA Jetson Nano. *Int. J. Electr. Comput. Eng.* **2022**, *12*, 5427–5434. [CrossRef]
74. Qayyum, A.; Ahmad, K.; Ahsan, M.A.; Al-Fuqaha, A.; Qadir, J. Collaborative Federated Learning for Healthcare: Multi-Modal COVID-19 Diagnosis at the Edge. *IEEE Open J. Comput. Soc.* **2022**, *3*, 172–184. [CrossRef]
75. Adhikari, M.; Munusamy, A. iCovidCare: Intelligent health monitoring framework for COVID-19 using ensemble random forest in edge networks. *Internet Things* **2021**, *14*, 100385. [CrossRef]
76. Velichko, A. A Method for Medical Data Analysis Using the LogNNet for Clinical Decision Support Systems and Edge Computing in Healthcare. *Sensors* **2021**, *21*, 6209. [CrossRef]
77. Yang, Z.; Liang, B.; Ji, W. An Intelligent End–Edge–Cloud Architecture for Visual IoT-Assisted Healthcare Systems. *IEEE Internet Things J.* **2021**, *8*, 16779–16786. [CrossRef]
78. 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]
79. Ahmed, I.; Jeon, G.; Piccialli, F. A Deep-Learning-Based Smart Healthcare System for Patient's Discomfort Detection at the Edge of Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 10318–10326. [CrossRef]
80. Liu, C.; Cao, Y.; Luo, Y.; Chen, G.; Vokkarane, V.; Yunsheng, M.; Chen, S.; Hou, P. A New Deep Learning-Based Food Recognition System for Dietary Assessment on An Edge Computing Service Infrastructure. *IEEE Trans. Serv. Comput.* **2018**, *11*, 249–261. [CrossRef]
81. Xu, M.; Qian, F.; Zhu, M.; Huang, F.; Pushp, S.; Liu, X. DeepWear: Adaptive Local Offloading for On-Wearable Deep Learning. *IEEE Trans. Mob. Comput.* **2020**, *19*, 314–330. [CrossRef]
82. Pramukantoro, E.S.; Gofuku, A. A real-time heartbeat monitoring using wearable device and machine learning. In Proceedings of the 2022 IEEE 4th Global Conference on Life Sciences and Technologies (LifeTech), Osaka, Japan, 7–9 March 2022; pp. 270–272. [CrossRef]
83. Zanetti, R.; Arza, A.; Aminifar, A.; Atienza, D. Real-Time EEG-Based Cognitive Workload Monitoring on Wearable Devices. *IEEE Trans. Biomed. Eng.* **2022**, *69*, 265–277. [CrossRef]
84. Puerta, G.; Le Mouël, F.; Carrillo, O. Machine Learning Models for Seizure Detection: Deployment Insights for e-Health IoT Platform. In Proceedings of the IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI'2021), Virtual, 27–30 July 2021.
85. Coelho, Y.L.; Santos, F.d.A.S.d.; Frizera-Neto, A.; Bastos-Filho, T.F. A Lightweight Framework for Human Activity Recognition on Wearable Devices. *IEEE Sen. J.* **2021**, *21*, 24471–24481. [CrossRef]
86. Arikumar, K.S.; 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] [PubMed]
87. Zhang, X.; Lin, J.; Chen, Z.; Sun, F.; Zhu, X.; Fang, G. An Efficient Neural-Network-Based Microseismic Monitoring Platform for Hydraulic Fracture on an Edge Computing Architecture. *Sensors* **2018**, *18*, 1828. [CrossRef]
88. Kumar, Y.; Udgata, S.K. Machine learning model for IoT-Edge device based Water Quality Monitoring. In Proceedings of the IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2–5 May 2022; pp. 1–6. [CrossRef]
89. Liu, Y.; Yang, C.; Jiang, L.; Xie, S.; Zhang, Y. Intelligent Edge Computing for IoT-Based Energy Management in Smart Cities. *IEEE Netw.* **2019**, *33*, 111–117. [CrossRef]
90. Cicirelli, F.; Gentile, A.F.; Greco, E.; Guerrieri, A.; Spezzano, G.; Vinci, A. An Energy Management System at the Edge based on Reinforcement Learning. In Proceedings of the 2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT), Prague, Czech Republic, 14–16 September 2020; pp. 1–8. [CrossRef]
91. Ali, O.; Ishak, M.K. Bringing intelligence to IoT Edge: Machine Learning based Smart City Image Classification using Microsoft Azure IoT and Custom Vision. *J. Phys. Conf. Ser.* **2020**, *1529*, 042076. [CrossRef]
92. Janjua, Z.H.; Vecchio, M.; Antonini, M.; Antonelli, F. IRESE: An intelligent rare-event detection system using unsupervised learning on the IoT edge. *Eng. Appl. Artif. Intell.* **2019**, *84*, 41–50. [CrossRef]
93. Orfanidis, C.; Hassen, R.B.H.; Kwiek, A.; Fafoutis, X.; Jacobsson, M. A Discreet Wearable Long-Range Emergency System Based on Embedded Machine Learning. In Proceedings of the 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Kassel, Germany, 22–26 March 2021; pp. 182–187. [CrossRef]
94. Nikouei, S.Y.; Chen, Y.; Song, S.; Xu, R.; Choi, B.; Faughnan, T.R. Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN. *arXiv* **2018**, arXiv:1805.00330.

95. Pang, S.; Qiao, S.; Song, T.; Zhao, J.; Zheng, P. An Improved Convolutional Network Architecture Based on Residual Modeling for Person Re-Identification in Edge Computing. *IEEE Access* **2019**, *7*, 106748–106759. [CrossRef]
96. Dhakal, A.; Ramakrishnan, K.K. Machine learning at the network edge for automated home intrusion monitoring. In Proceedings of the 2017 IEEE 25th International Conference on Network Protocols (ICNP), Toronto, ON, Canada, 10–13 October 2017; pp. 1–6. [CrossRef]
97. Fan, X.; Xiang, C.; Gong, L.; He, X.; Chen, C.; Huang, X. UrbanEdge: Deep Learning Empowered Edge Computing for Urban IoT Time Series Prediction. In Proceedings of the ACM Turing Celebration Conference-China, Chengdu, China, 17–19 May 2009; ACM TURC'19. [CrossRef]
98. Sabbella, S.R. Fire and Smoke Detection for Smart Cities Using Deep Neural Networks and Edge Computing on Embedded Sensors. Ph.D. Thesis, Sapienza University of Rome, Rome, Italy, 2020. [CrossRef]
99. Silva, M.C.; da Silva, J.C.F.; Delabrida, S.; Bianchi, A.G.C.; Ribeiro, S.P.; Silva, J.S.; Oliveira, R.A.R. Wearable Edge AI Applications for Ecological Environments. *Sensors* **2021**, *21*, 82. [CrossRef]
100. Zhu, D.; Liu, H.; Li, T.; Sun, J.; Liang, J.; Zhang, H.; Geng, L.; Liu, Y. Deep Reinforcement Learning-based Task Offloading in Satellite-Terrestrial Edge Computing Networks. In Proceedings of the 2021 IEEE Wireless Communications and Networking Conference (WCNC), Nanjing, China, 29 March–1 April 2021; pp. 1–7. [CrossRef]
101. Zhang, J.; Zhang, X.; Wang, P.; Liu, L.; Wang, Y. Double-edge intelligent integrated satellite terrestrial networks. *China Commun.* **2020**, *17*, 128–146. [CrossRef]
102. de Prado, M.; Rusci, M.; Capotondi, A.; Donze, R.; Benini, L.; Pazos, N. Robustifying the Deployment of tinyML Models for Autonomous Mini-Vehicles. *Sensors* **2021**, *21*, 1339. [CrossRef] [PubMed]
103. Kocić, J.; Jovičić, N.; Drndarević, V. An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms. *Sensors* **2019**, *19*, 2064. [CrossRef] [PubMed]
104. Navarro, P.J.; Fernández, C.; Borraz, R.; Alonso, D. A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data. *Sensors* **2017**, *17*, 18. [CrossRef] [PubMed]
105. Bibi, R.; Saeed, Y.; Zeb, A.; Ghazal, T.; Said, R.; Abbas, S.; Ahmad, M.; Khan, M. Edge AI-Based Automated Detection and Classification of Road Anomalies in VANET Using Deep Learning. *Comput. Intell. Neurosci.* **2021**, *2021*, 6262194. [CrossRef]
106. Ferdowsi, A.; Challita, U.; Saad, W. Deep Learning for Reliable Mobile Edge Analytics in Intelligent Transportation Systems. *IEEE Veh. Technol. Mag.* **2017**, *14*, 62–70. [CrossRef]
107. Hu, Y.; Liu, G.; Chen, Z.; Guo, J. Object Detection Algorithm for Wheeled Mobile Robot Based on an Improved YOLOv4. *Appl. Sci.* **2022**, *12*, 4769. [CrossRef]
108. Febbo, R.; Flood, B.; Halloy, J.; Lau, P.; Wong, K.; Ayala, A. Autonomous Vehicle Control Using a Deep Neural Network and Jetson Nano. In Proceedings of the Practice and Experience in Advanced Research Computing, Portland, OR, USA, 26–30 July 2020; PEARC'20; pp. 333–338. [CrossRef]
109. Palossi, D.; Loquercio, A.; Conti, F.; Flamand, E.; Scaramuzza, D.; Benini, L. A 64-mW DNN-Based Visual Navigation Engine for Autonomous Nano-Drones. *IEEE Internet Things J.* **2019**, *6*, 8357–8371. [CrossRef]
110. Alsamhi, S.; Almalki, F.; Al-Dois, H.; Shvetsov, A.; Ansari, S.; Hawbani, A.; Gupta, D.S.; Lee, B. Multi-Drone Edge Intelligence and SAR Smart Wearable Devices for Emergency Communication. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6710074. [CrossRef]
111. Wang, J.; Feng, Z.; Chen, Z.; George, S.; Bala, M.; Pillai, P.; Yang, S.W.; Satyanarayanan, M. Bandwidth-Efficient Live Video Analytics for Drones Via Edge Computing. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Seattle, WA, USA, 25–27 October 2018; pp. 159–173. [CrossRef]
112. Tsakanikas, V.; Dagiuklas, T.; Iqbal, M.; Wang, X.; Mumtaz, S. An intelligent model for supporting edge migration for virtual function chains in next generation internet of things. *Sci. Rep.* **2023**, *13*, 1063. [CrossRef]
113. Ju, Y.; Cao, Z.; Chen, Y.; Liu, L.; Pei, Q.; Mumtaz, S.; Dong, M.; Guizani, M. NOMA-Assisted Secure Offloading for Vehicular Edge Computing Networks With Asynchronous Deep Reinforcement Learning. *IEEE Trans. Intell. Transp. Syst.* 2023, *in press*. [CrossRef]
114. Yan, J.; Zhang, M.; Jiang, Y.; Zheng, F.C.; Chang, Q.; Abualnaja, K.M.; Mumtaz, S.; You, X. Double Deep Q-Network based Joint Edge Caching and Content Recommendation with Inconsistent File Sizes in Fog-RANs. *IEEE Trans. Veh. Technol.* **2023**, 1–14. [CrossRef]
115. Liu, J.; Fan, Y.; Sun, R.; Liu, L.; Wu, C.; Mumtaz, S. Blockchain-Aided Privacy-Preserving Medical Data Sharing Scheme for E-Healthcare System. *IEEE Internet Things J.* **2023**, *10*, 21377–21388. [CrossRef]
116. Guim, F.; Metsch, T.; Moustafa, H.; Verrall, T.; Carrera, D.; Cadenelli, N.; Chen, J.; Doria, D.; Ghadie, C.; Prats, R.G. Autonomous Lifecycle Management for Resource-Efficient Workload Orchestration for Green Edge Computing. *IEEE Trans. Green Commun. Netw.* **2022**, *6*, 571–582. [CrossRef]
117. Hanzel, K.; Grzechca, D.; Ziebinski, A.; Chruszczyk, L.; Janus, A. Estimating the AGV load and a battery lifetime based on the current measurement and random forest application. In Proceedings of the 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 15–18 December 2023; pp. 5057–5063. [CrossRef]