Journal of
Sensor and Actuator Networks

MDPI

Article

Virtualizing AI at the Distributed Edge towards Intelligent IoT Applications

Claudia Campolo [1], Giacomo Genovese [1], Antonio Iera [2] and Antonella Molinaro [1,3,*]

[1] The Department of Information, Infrastructures and Sustainable Energy (DIIES) Department, University Mediterranea of Reggio Calabria, 89100 Reggio Calabria, Italy; claudia.campolo@unirc.it (C.C.); giacomo.genovese@unirc.it (G.G.)
[2] The Department of Information, Infrastructures and Sustainable Energy (DIIES) Department, University of Calabria, 87036 Rende, Italy; antonio.iera@dimes.unical.it
[3] Laboratoire des Signaux et Systémes (L2S), CentraleSupélec, Université Paris-Saclay, 91190 Gif-sur-Yvette, France
* Correspondence: antonella.molinaro@unirc.it

Abstract: Several Internet of Things (IoT) applications are booming which rely on advanced artificial intelligence (AI) and, in particular, machine learning (ML) algorithms to assist the users and make decisions on their behalf in a large variety of contexts, such as smart homes, smart cities, smart factories. Although the traditional approach is to deploy such compute-intensive algorithms into the centralized cloud, the recent proliferation of low-cost, AI-powered microcontrollers and consumer devices paves the way for having the intelligence pervasively spread along the cloud-to-things continuum. The take off of such a promising vision may be hurdled by the resource constraints of IoT devices and by the heterogeneity of (mostly proprietary) AI-embedded software and hardware platforms. In this paper, we propose a solution for the AI distributed deployment at the deep edge, which lays its foundation in the IoT virtualization concept. We design a *virtualization layer* hosted at the network edge that is in charge of the semantic description of AI-embedded IoT devices, and, hence, it can expose as well as augment their cognitive capabilities in order to feed intelligent IoT applications. The proposal has been mainly devised with the twofold aim of (i) relieving the pressure on constrained devices that are solicited by multiple parties interested in accessing their generated data and inference, and (ii) and targeting interoperability among AI-powered platforms. A Proof-of-Concept (PoC) is provided to showcase the viability and advantages of the proposed solution.

Keywords: Internet of Things; edge computing; virtualization; edge AI; artificial intelligence; TinyML; 6G

Citation: Campolo, C.; Genovese, G.; Iera, A.; Molinaro, A. Virtualizing AI at the Distributed Edge towards Intelligent IoT Applications. *J. Sens. Actuator Netw.* **2021**, *10*, 13. https://doi.org/10.3390/jsan10010013

Received: 12 December 2020
Accepted: 1 February 2021
Published: 8 February 2021

1. Introduction

Today, an ever growing market of Internet of Things (IoT) applications, such as video surveillance, intelligent personal assistants, smart home appliances, and smart manufacturing, requires advanced Artificial Intelligence (AI) capabilities, including computer vision, speech recognition, and natural language processing. Such intelligent applications are traditionally implemented using a centralized approach: raw data collected by IoT devices are streamed to the remote cloud, which has virtually unlimited capabilities to run compute-intensive tasks, such as AI model building/training and inference. Centralization has the further advantage of storing the output of the inference stage which can be requested later by other applications [1], with no need to re-run the computation.

Notwithstanding, there is a potential downside in leveraging the remote cloud for running cognitive components. First, uploading a massive amount of input data to the cloud consumes network bandwidth and energy of the IoT devices. Second, the latency to the cloud may be prohibitive for delay-sensitive applications. Third, transferring sensitive data retrieved by IoT devices may raise privacy issues [2].

Edge computing, considered as one of the main pillars of fifth generation (5G) systems [3], can be used to augment the capabilities of IoT devices by pushing computing and storage resources to close servers in several smart environments. Relying on the local edge capabilities, instead of moving data to powerful remote data centers, can boost the system performance by reducing the data traffic load traversing the core network and thus ensuring low-latency access to context-aware cloud-like services. Recently, the availability of edge computing solutions, coupled with the need to build inference and make intelligent decisions as fast as possible, contributed to the marriage of edge computing and AI, leading to the so-called edge AI paradigm [2,4]. Based on this innovative approach, computation-intensive and latency-critical AI-based applications can be executed in a real-time responsive manner by pushing the AI model computation close to the requesting users, while also achieving a reduced pressure on the network and the protection of the user privacy. Context-awareness is a further benefit of deploying cognitive components at the edge [5].

Edge and cloud are not mutually exclusive but rather complementary, with the former taking care of delay-sensitive tasks and the latter in charge of extremely computationally intensive tasks. By applying such a split to the AI lifecycle workflow, model training can be deployed into the cloud while inference can be performed at the edge. Similarly, some of the Deep Learning (DL) model layers can be deployed at the edge, close to the input data, while the extracted features, whose size is smaller than that of the input data, can be transferred to the cloud [6].

Bringing the edge AI vision to the extreme, a true revolution will be achieved when AI will be spread at the *deep* edge, being embedded into IoT devices, e.g., running on micro-controllers units (MCUs), such as wearable devices, surveillance cameras, and smartphones [1,7]. Of course, straightforwardly moving algorithms, which originally run on powerful platforms, into IoT devices is not feasible, due to the relatively limited computing power, memory footprint, and energy of such devices. Approaches like parallel acceleration and AI model compression (e.g., weight pruning, data quantization) are necessary to enable DL on IoT devices, while keeping the same accuracy as the original models as much as possible [2,4,8].

The new trend, referred to as *TinyML* [9], of equipping IoT end-devices with capabilities to execute Machine Learning (ML) algorithms paves the way for a wide plethora of innovative intelligent applications and services, and contributes to the radical IoT shift from *connected things* to *connected intelligent things*, which is at the basis of future sixth generation (6G) systems [10]. TinyML is getting close to reality thanks to recent advancements in the miniaturization of AI-optimized processors and the definition of extremely lightweight ML inference frameworks. Examples are the STM32Cube.AI [11] by STMicroElectronics, which embeds human activity recognition and audio scene classification capabilities in micro-controllers; the Snapdragon Neural Processing Engine (SNPE) Software Development Kit (SDK) [12] released by Qualcomm, which enables DL into smartphones; and uTensor [13], built on Tensorflow and optimized for ARM processors.

More distributed AI approaches can be also foreseen, which do not only spread learning and inference tasks between the cloud and the (deep) edge, but also enforce inference model splitting between device-edge, edge-edge, and edge-cloud tiers on a per-application basis [14].

Having AI pervasively deployed over a *distributed cloud* along the cloud-to-things continuum [7,15] would not come without challenges, as summarized in Table 1. Unlike in centralized deployments, distributed AI solutions may suffer from interoperability issues, due to fragmented and mainly application-specific solutions [1]. AI algorithms are typically tightly coupled to the application that exploits them, so hindering the provisioning of the same offered service to other applications. To circumvent this issue, it is crucial to set up mechanisms to *identify and discover AI components* and build intelligent applications upon them, while efficiently using network and computing resources. Indeed, solutions should be devised to make interactions with resource-constrained IoT devices *as light as possible*

and reduce the pressure on them for inference computation. Finally, the heterogeneity of AI accelerators and chipsets calls for robust platform abstractions, which can ensure *transparent access to AI components* by hiding the specific low-layer implementation details to the upper layers that exploit them.

Table 1. AI at the deepest edge: main issues and our proposal.

Issue	Description	Proposed Solution
Interoperability	Fragmented and mainly application-specific AI solutions	Uniform semantic description of AI components
Platform heterogeneity	AI-enabled chips and compilers with different features	Hardware- and software-agnostic abstraction
Pressure on constrained devices	Multiple applications requesting the same inference results to IoT devices	Caching of inference results and lightweight messaging protocols

In this paper, we propose leveraging the IoT virtualization concept [16] and apply it to AI-powered IoT devices in order to tackle the aforementioned issues. IoT virtualization has the ability to make heterogeneous objects interoperable through the use of semantic descriptions coupled to the digital counterpart of any real entity in the IoT. This approach makes the discovery of IoT services easier, since metadata are used to index the virtual objects. We couple the virtualization concept with edge computing to ensure quicker interactions between the twinned physical devices and their virtual counterparts. Being hosted at the edge facilities, the digital counterpart can augment the typically constrained capabilities of the corresponding physical device, e.g., by caching inference results.

The main contributions of this paper can be summarized as follows:

- We propose to leverage the concept of IoT virtualization for the semantic description of AI-empowered IoT devices being part of the distributed cloud and for the augmentation of their capabilities. The ultimate goal is to make their resources to be discovered and accessed by different stakeholders *as-a-Service*, while ensuring interoperability.
- We provide the semantic description of the AI-empowered IoT devices through the well-known Open Mobile Alliance (OMA) Lightweight Machine-to-Machine (LwM2M) resource description model [17] proposed in the IoT domain. Conceived extensions to specifically deal with AI components embedded in IoT devices are detailed.
- We promote the usage of the Constrained Application Protocol (CoAP) [18] to allow lightweight interactions between an AI-empowered IoT device and its virtual counterpart at the edge.
- We realize a Proof-of-Concept (PoC) to showcase the viability of the conceived proposal when referring to an object detection application and leveraging the Leshan implementation of OMA LwM2M. We also measure the data footprint in terms of exchanged bytes to retrieve the output of an object detection inference task.

The paper is structured as follows: Section 2 introduces the OMA LwM2M protocol as the enabler for IoT virtualization as well as CoAP to facilitate message exchange. Our proposal is discussed in Section 3 and the devised PoC is presented in Section 4. Final remarks and conclusions are drawn in Section 5, by providing hints on future works.

2. Internet of Things (IoT) Virtualization

2.1. The VO Concept

Virtualization typically refers to the logical abstraction of underlying hardware devices, through a software implementation/description. In the IoT context, it can either impact the network and its functions [19] or the devices [16]. Notably, device virtualization has become a key pillar of many reference IoT platforms (e.g., iCore [20], IoT-A [21]) and commercial implementations (e.g., Amazon Web Services IoT). It is intended to make heterogeneous

objects plug-and-playable: this means that, as soon as a device joins a network, it can be immediately provided with mechanisms that enable its interaction with the external world [16].

The Virtual Object (VO) represents the digital counterpart of the physical IoT device. The most appropriate manner to represent IoT devices is by using semantic technologies [16]. Hence, the VO provides the semantic enrichment of data and functionalities provided by the IoT device. The result of the semantic description is the VO model which includes, for instance: objects' characteristics, objects' location, resources, services, and quality parameters provided by objects. The VO model, intended as a software built for such a service, is independent from a specific device; it is initialized at startup according to the properties of the physical homologous it is going to represent thanks to a configuration file built on purpose.

The semantic description copes with heterogeneity and provides interoperability in the IoT domain eliminating vertical silos. In addition, it is very powerful in supporting search and discovery operations. Indeed, search and discovery mechanisms allow for finding the device that is most appropriate to perform a given application's task.

The VO can also augment the physical counterpart with storage and computing capabilities, by providing caching and preliminary filtering/aggregation/processing of raw data streamed by the corresponding IoT device, before feeding IoT applications building upon them. Caching data provided by the physical device would also avoid overwhelming it with the same requests coming from multiple remote applications, which is particularly helpful in case of resource-constrained IoT devices.

Although VOs were initially conceived to be deployed in the remote cloud, recent literature solutions have disclosed the benefits of edge networks to satisfactorily meet the latency constraints on pairing a physical device and its corresponding VO [22–24]. In particular, in ref. [23], a proxy Virtual Machine (VM) is considered to be hosted at the edge, and containers are instead considered to create virtualized cameras in ref. [25].

In the same context of abstractions for IoT, other approaches are advocating the agent concept, as extensively surveyed in ref. [26]. Agents seem to have found a wide use in the implementation of vertical IoT solutions within the same specific domain, for instance, integration of multiple heterogeneous systems belonging to the same holder. Nevertheless, agents actually look more viable for specific micro-operations or platform-to-platform interconnection and, unlike VOs, not yet ready to boost up SOs' connectivity and interoperability [26].

2.2. The OMA LwM2M Protocol

Several semantic models exist for IoT device discovery and uniform data format. Semantic technologies are largely leveraged in the web domain and extend the Web with machine interpretable meaning, thus enabling data integration and sharing, and interoperability amongst interconnected machines [27]. A well established Semantic Web standard is the Web Ontology Language (OWL), developed by the World Wide Web Consortium (W3C). The application of semantic technologies to the IoT domain has been largely advocated in the literature [28–30]. More recently, such techniques have been properly extended to match IoT peculiarities. The first attempt of standardization in IoT semantic description was the Semantic Sensor Networks (SSN) ontology [31] which is an OWL ontology for describing sensors developed by W3C. More recently, the W3C Web of Things (WoT) has proposed Things description (TD) [32] that specifies a semantic way to map IoT devices in the physical world to virtual things [33]. A proprietary ontology is considered in ref. [25]. In addition, the popular IoT implementation, oneM2M, has its ontology, the *base ontology*, which defines a device as a derivation of a generic thing designed to accomplish a particular task through functions of the device [34].

In this work, we leverage OMA LwM2M [17], which provides a simple object-based resource model for Machine-to-Machine (M2M) and IoT device management [35]. It has been used in commercial implementations [36] and within the FIWARE initiative [37]. In

ref. [38], OMA LwM2M is leveraged as a key pillar for the VO implementation. It is also considered for several vertical markets, e.g., industry 4.0 [39] and automotive [40].

In OMA LwM2M, the device is represented by a collection of *Objects* and each object is composed of *Resources*, as shown in Figure 1.

The Resource specifically identifies the elementary accessible entity which can define, for instance, the information that a device can transmit [41]. It defines a specific resource related to the OMA Object itself. For instance, a Resource could be the *Value* for a temperature sensor, the *Latitude and Longitude* values for a positioning equipment, as well as the *Memory Free* and the *Battery Level* for a device [42].

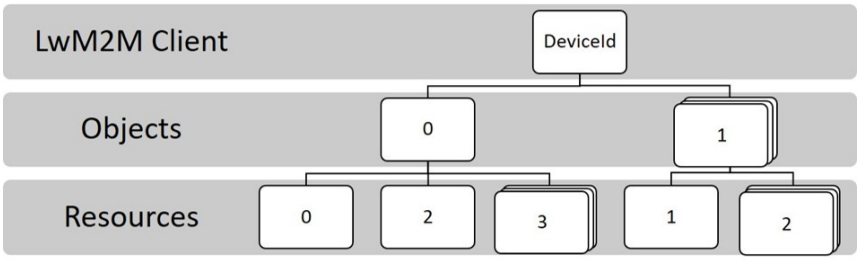

Figure 1. OMA LwM2M data model.

In particular, Objects and Resources are represented through a Uniform Resource Identifier (URI) path hierarchy, where each URI path component sequentially represents: the *Object Type Identifier (ID)*, the *Object Instance ID*, and the *Resource Type ID*. For instance, the URI path for *Latitude* coming from a geo-localization sensor is */6/0/0*. The component *6* identifies the object *Location*, *0* identifies the instance, and it is used to differentiate the presence of multiple objects of the same type into the Device; *0* represents the *Latitude* value (i.e., 38, 120766) of the sensor. The *Longitude* value, instead, is represented by a different resource with path *6/0/1*. All the resources included in the *Location* object are reported in Table 2.

Table 2. OMA-LwM2M location object's resources definition.

Resource Name	Object ID	Object Instance	Resource ID
Latitude	6	0	0
Longitude	6	0	1
Altitude	6	0	2
Radius	6	0	3
Velocity	6	0	4
Timestamp	6	0	5
Speed	6	0	6

Objects defined by OMA as well as standard objects produced by third-party organizations are both provided by the public registry [42]. By following the technical specifications, customized objects can be further defined.

Objects and resources are hosted by a data producer, which is referred to as the *OMA LwM2M client*, and they are consumed by the so-called *OMA LwM2M server*. The object structure and its resource data are defined within an eXtensible Markup Language (XML) configuration file. The same configuration file must be kept by both client and server to serialize/de-serialize the exchanged information.

The LwM2M Enabler interface provides access to resources through the use of CREATE, READ, WRITE, DELETE, EXECUTE, WRITE-ATTRIBUTE, or DISCOVER operations.

2.3. The CoAP Protocol

OMA LwM2M leverages CoAP as a messaging protocol. CoAP has been proposed within the Internet Engineering Task Force (IETF) to allow Internet Protocol (IP)-enabled

IoT devices to work in a Web-like fashion [18]. This protocol provides discovery mechanisms, resource abstraction, URIs, and request/response methods.

Although built upon the well-known Hyper Text Transfer Protocol (HTTP), it is specifically customized to incur a low footprint in terms of bandwidth consumption and implementation complexity, and, hence, to be deployed by constrained devices.

At the transport layer, it relies on User Datagram Protocol (UDP), instead of the heavier Transport Control Protocol (TCP), and implements retransmissions at the application layer.

Besides request/response methods, it provides the asynchronous monitoring of IoT resources through the OBSERVE extension. Such feature is particularly beneficial for those resources that do not change with a fixed periodicity and for which, instead, a periodical request/response approach would waste network bandwidth and device battery for exchanging unchanged values of the resource.

3. Proposal

3.1. Reference Architecture

Our proposal builds upon the successful IoT virtualization concept that is extended to the case of upcoming AI-empowered IoT devices. The resulting reference architecture is reported in Figure 2.

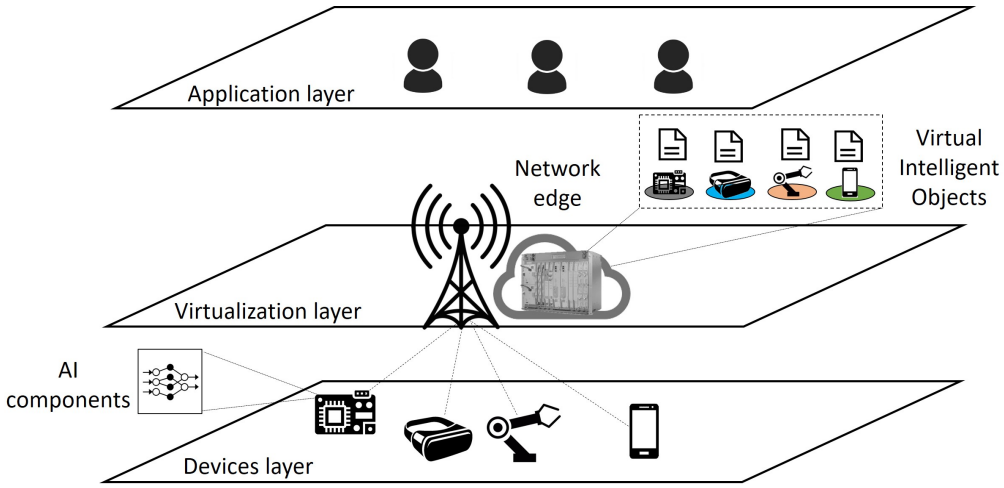

Figure 2. Reference architecture.

At the bottom of the architecture, we have intelligent IoT devices, i.e., devices equipped with AI capabilities. Through embedded sensing capabilities, they can collect data feeding the on-board inference engine. The latter one mainly consists of models pre-trained on massive datasets by more powerful platforms, e.g., the remote cloud. A typical pre-trained ML inference model cannot be run on constrained IoT devices as it is, and must be converted to fit the target limited device resources. Quantization and pruning [43] are just a few examples of the techniques to be deployed by an ML compiler to build the optimized model for the specific software and hardware platform that the device is featuring. Such model can be installed (and also modified) into devices on-the-fly. Once the optimized model is deployed into the device, the latter one can start inference.

At this stage of research, without loss of generality, we assume that ML models deployed at physical devices are Artificial Neural Networks (ANNs) (also encompassing deep learning, hence ANNs with complex multilayers [44]), which are widely leveraged to accurately classify and recognize patterns. Hence, they can be particularly helpful to support IoT applications by processing large amounts of unstructured data provided by physical devices. Examples are the recognition of objects, traffic signs, speech as well as obstacle avoidance, see, e.g., [45] and references therein. Moreover, the proposal is intended to specifically support solutions already available on the market that foresee the

implementation of pre-trained ANN models into constrained platforms; see, for instance, the solution provided by STMicroelectronics [46].

In our proposal, the virtualization layer represented by the digital counterparts of the physical devices is hosted at the edge. In particular, each physical device is associated with what we refer to as *Virtual Intelligent Object* (VIO).

At the top of the architecture, we have intelligent IoT applications, which may request inputs from cognitive components hosted in IoT devices, through the VIO. Such consumer applications can be either hosted remotely (e.g., remote surveillance) or located close to the intelligent IoT devices (e.g., augmented reality).

3.2. The VIO Design

The VIO represents the key novelty of our proposal. Similarly to the VO initially conceived in IoT, its presence targets the following crucial objectives: (i) overcoming platform heterogeneity, (ii) ensuring interoperability, (iii) improving search and discovery, and *(iv)* reducing the pressure on constrained devices. In addition, in our proposal, its design is enhanced to specifically support the augmentation of the physical AI-powered device with additional functionalities detailed as follows:

- It *provides the semantic description of the physical AI-empowered counterpart* so to ensure a common understanding of its features and capabilities among all potential consumer applications. Specifically, it describes the cognitive embedded components by abstracting the specific hardware and software platform implementation. Hence, the VIO exposes the capabilities of the relevant physical device for interested applications, managing transparent access to the intelligent heterogeneous resources. Such a feature is particularly beneficial for sophisticated applications relying on AI inference capabilities. Indeed, the semantic description of AI-empowered IoT devices can facilitate search and discovery procedures in order to identify the AI components that are the most appropriate, according to the demands of the requesting application (e.g., in terms of accuracy, expected inference latency), to perform a given inference task. Moreover, in so doing, the conceived abstraction of the AI capabilities of IoT devices makes the latter ones available to all interested applications in an interoperable manner, by overcoming fragmentation.
- It *acts as a proxy between the physical device and the consumer applications*. It is in charge of replying to the requesting applications, on behalf of the physical device.
- It *caches the output of inference procedures* performed by the physical device. Such cached results can feed multiple consumer applications issuing multiple requests, which may potentially overwhelm the constrained IoT device. It could happen, for instance, that users within the same area request recognition tasks related to it [2]. As a result, resources of the physical device will be saved, since there would be no need to re-run the inference task to reply to each request issued by different applications.
- It *is in charge of issuing the update of the ANN* inference model on the physical device. This can result either in the update of the weight parameters or in the modifications of the model itself. The update can be issued for instance by monitoring the accuracy levels achieved in performed inference procedures or upon feedback received by the consumer applications.
- It *can train the ANN model*, on behalf of the cloud, by ensuring a higher proximity to the physical device where it should be injected.
- It *can optimize the pre-trained ANN model* before its injection into the device. This is more convenient than what is currently assumed, i.e., a remote server playing this role. Indeed, the VIO knows the capabilities of the device, according to which it can modify the model for a proper fitting.

3.3. OMA Object and Relevant Resources

In this work, we propose the use of a new OMA LwM2M object, named *OMA-TinyML*, for the semantic description of the physical device which is kept by the VIO. Such object

defines the semantic representation of an ML capability embedded in an IoT device and allows for exposing the capabilities of the device to external applications. We assign it the OMA ID of 20,000, according to object classes defined by OMA [47]. The following resources are defined for it:

- *AI application*: this resource describes the type of inference that can be performed by the physical device, e.g., object detection, face recognition, and audio classification.
- *Model*: it describes the type of ANN that the device runs locally and for which it can provide an inference, e.g., Convolutional Neural Network (CNN);
- *CPU*: it provides details about the processing capabilities of the device. It is expressed in GHz.
- *Start inference*: it triggers the execution of the inference task by a consumer application.
- *Output*: it provides the output of the inference, e.g., the set of detected objects in a picture or in video source, along with the measured accuracy and the coordinates of the bounded box of the detected object.

The first three resources play a crucial role in the discovery procedure. In particular, once an application identifies a given IoT device for an inference task, the parameter about the CPU on board can provide some hints about the expected inference latency. The latter information can be leveraged together with the residual battery level (exposed by the legacy OMA LwM2M Device object, ID 3, at resource ID 9) and the free memory (exposed by the legacy OMA LwM2M Device object, ID 3, at resource ID 10) to understand whether the device can successfully accomplish the inference task.

It is worth noting that a consumer could leverage the OMA LwM2M *OBSERVE* method in order to be updated on each output of performed inferences. In other words, instead of explicitly requesting each output of the inference, some logic can be defined upon which the physical device pushes updates on the performed inference to the VIO. For instance, in case of a surveillance camera with an embedded face recognition engine deployed in an office environment, the OMA client can issue an update on the *Output* resource whenever an unrecognized individual is detected at closing hours.

Table 3 reports the OMA-TinyML Object and its resources.

Table 3. OMA-TinyML object resources.

Name	Resource ID	OMA LwM2M Resource URI Path
AI application	0	/20000/0/0/
Model	1	/20000/0/1/
CPU	2	/20000/0/2/
Start inference	3	/20000/0/3/
Output	5	/20000/0/4/

4. Proof-of-Concept

In this section, we aim to assess the viability of our proposal, by showcasing how the VIO can be deployed to augment an IoT device running an ML algorithm for the sake of object detection. Moreover, measurements concerning the incurred traffic footprint as well as the inference latency when compared to the case in which inference is performed into the edge are reported.

4.1. Experimental Set-Up

The experimental set-up for our study is shown in Figure 3. The OMA LwM2M client component runs on the AI-powered device, i.e., a low-cost Rasbperry Pi, and provides the set of resources feeding the corresponding digital counterpart.

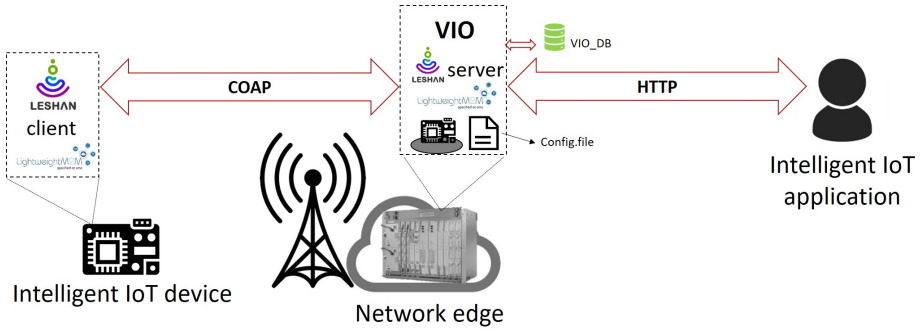

Figure 3. PoC set-up.

4.2. Results

For the OMA LwM2M implementation, we leverage Leshan [48], which is written in Java and is provided by the Eclipse foundation. Leshan provides a set of libraries supporting the development of OMA LwM2M-compliant server and clients. Such implementation covers most of the OMA LwM2M specifications [49].

In order to implement the described features, the Leshan client has been overhauled to include the new object created and the relevant resources. The new client differs from the vanilla Leshan one for the implementation of different classes that allow for connecting, managing and controlling the ML components through the objects and related OMA LwM2M resources exposed.

The Leshan server core is incorporated in the VIO as an interface to the physical counterpart, the southbound interface, managing connection to the client and the OMA LwM2M layer. The remaining architectural VIO levels are used for the implementation of enriched functionalities that will be provided to consumers through more cloud-oriented interfaces. Moreover, a database is associated with the VIO which stores the history of all the data (e.g., inference outputs) received over a short term period (e.g., a day), from the physical device. For the sake of the PoC, a laptop is leveraged as a network edge device hosting the VIO.

4.2.1. The VIO Web Interface

Figure 4 shows the VIO web interface inherited from the Leshan server. The interface enables users to issue OMA LwM2M methods like READ, OBSERVE, and EXECUTE. The same interface can be reached using HTTP GET, PUT, POST, etc., which are bound to a CoAP request. The bold text in the right side of the figure is the result of queries on resources. The user can choose to query the single resource or the entire instance. In the second case, it will receive the available data of all resources with *READ* functionality. In particular, the result of resource *Output* is a JavaScript Object Notation (JSON) representation of the inference result provided by an object detection algorithm.

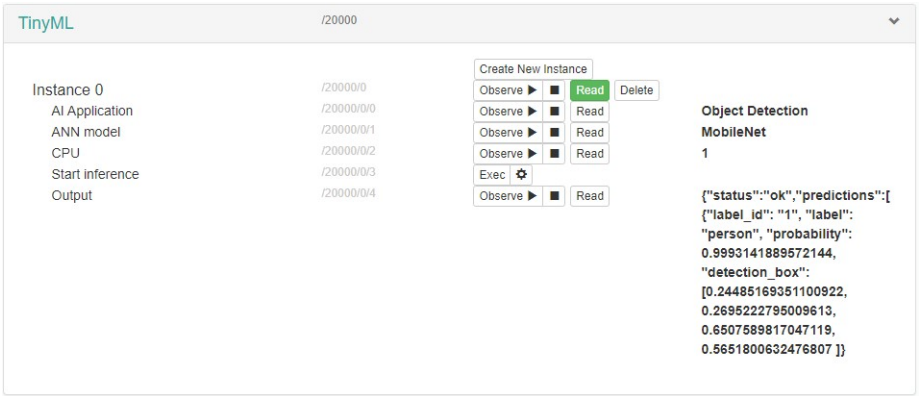

Figure 4. The VIO web interface.

4.2.2. Exchanged Data Traffic

We measure the number of exchanged bytes to retrieve an inference result upon a request issued by a remote consumer application. In particular, results reported in Figure 5 refer to the following cases: (i) the request issued by the remote application is forwarded by the VIO to the physical device, since there is no cached inference matching the request (curve labeled as "No caching at the VIO") and (ii) the inference requested by the remote application is cached by the VIO (curve labeled as "Caching at the VIO"). To enable caching at the VIO, whenever a new inference result is received by the physical device, data are stored in the local database and sent to the requesting application. CoAP is leveraged over the link between the Leshan client and the VIO to better match network and device constraints. Instead, intelligent IoT applications consuming data can access the VIO through HTTP interfaces adding the device name to the resource URI path defined by OMA LwM2M. The metric is derived as the number of bytes composing GET requests and replies exchanged between the OMA LwM2M client and the VIO (for the CoAP protocol), and between the VIO and the remote application (for the HTTP protocol).

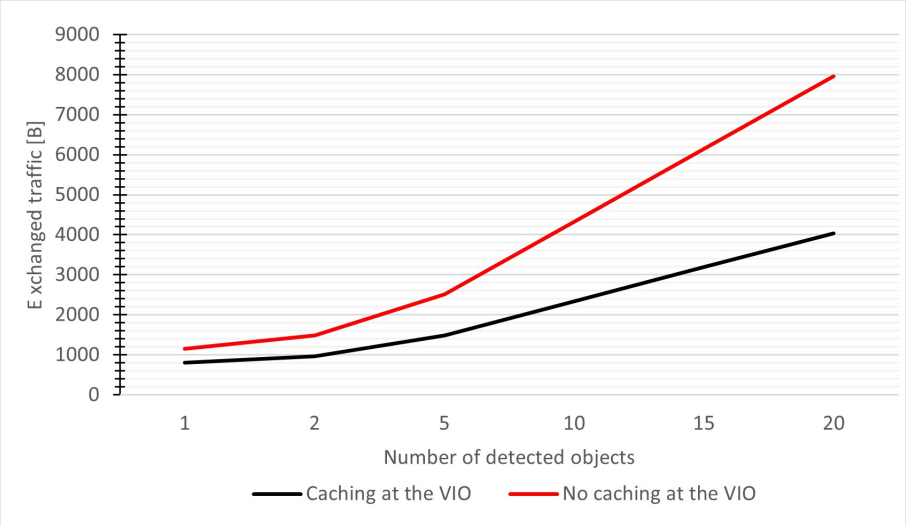

Figure 5. Exchanged traffic when varying the number of detected objects.

The measurement has been performed for different numbers of recognized objects (from 1 to 20, as in the *x*-axis of Figure 5), as returned through the *Output* resource, and performed through the Wireshark [50] protocol analyzer.

Figure 5 shows that the presence of the VIO allows for reducing the amount of exchanged data traffic. This is more true as the number of detected objects increases. Besides reducing the interactions with the physical device, the caching of inference results at the VIO has the additional benefit of avoiding the physical device to re-run the inference, by saving precious (limited) resources.

Although the overall amount of exchanged data is not significant, in the near future, we expect massively deployed intelligent IoT devices. Hence, reducing the exchanged data traffic would overall relieve the pressure on the network.

It is worth remarking that, even in case of no caching at the VIO, the exchanged traffic with the physical device is limited thanks to the usage of CoAP, instead of HTTP. The amount of transferred bytes incurred by the two protocols for the request of an inference resulting in a single detected object as a reply is reported in Table 4.

Table 4. Number of exchanged bytes per a single detected object.

Method	HTTP	CoAP
GET request	295	54
GET reply	497	249

4.2.3. TinyML vs. Edge

Before concluding, we report results measuring the performance achieved when running an object detection inference task. In particular, we leverage two different off-the-shelf objection detection models to match the computation capabilities of different hosting platforms considered as benchmarks, i.e., a more capable edge node and a constrained Raspberry Pi device.

The Faster R-CNN ResNet-101 algorithm [51] has been run in an edge device with 2.1 GHz-CPU and 8 GB-RAM. Instead, the MobileNet object detection model [52] has been deployed over the constrained Raspberry Pi device, being representative of the TinyML approach.

The chosen models are widely used in the literature. Faster R-CNN ResNet-101 is a region-based CNN. MobileNet is notoriously faster but less accurate than Faster R-CNN ResNet 101 [53,54]. Indeed, MobileNet is designed for efficient inference in various mobile and embedded vision applications. To effectively reduce both computational cost and number of parameters, it builds upon depthwise separable convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution. The focus of this work being on the design of the virtualization layer, we leave as a future work the adaptation of the same model used at the edge to a constrained platform, e.g., through quantization and pruning techniques.

Table 5 reports the metrics of interest (i.e., transferred bytes, latency, accuracy) for the detection of objects within two (input) images of different sizes. Our aim is not to support real-time inference but to analyze the sources of latency in the entire inference process. In the edge case, the inference is performed after the input data (an image) is transferred from the IoT device to the edge. In the TinyML case, instead, the inference is performed over the locally available image; hence, no data are exchanged over the network.

Table 5. TinyML vs. Edge.

Image Size	TinyML			Edge		
	Transferred Bytes	Latency (s)	Accuracy	Transferred Bytes	Latency (s)	Accuracy
127 kB	-	4.15	0.9	140 kB	9.2	0.998
2.2 MB	-	24.29	0.91	2.5 MB	25.5	0.997

We can observe that, as expected, the faster R-CNN ResNet-101 model deployed at the edge achieves higher accuracy compared to the lighter (and simplified) model running on the constrained device. Regarding latency, it encompasses the following contributions: (i) the input image transfer delay, (ii) the processing delay for running the inference task, and (iii) the delay for delivering the output (i.e., the indication of the set of detected objects within the image, along with the measured accuracy and the coordinates of the bounded box of each detected object). The first and latter contributions apply only to the edge case. For the small image, the latency experienced by the TinyML approach is smaller compared to the edge solution. Latency values for the two cases, instead, are close for the larger image. This is because the latter one entails heavier computations, which are slower in the constrained device.

Such a result would suggest to investigate the feasibility to offload the inference (or part of it) to more powerful platforms at the edge as the computations get heavier. This would be possible for instance, by equipping the VIO with inference capabilities

complementing the corresponding physical device. The decision about whether to offload the inference task mainly depends on the application demands in terms of latency and accuracy and should be made according to (i) the network conditions experienced over the link between the physical device and its counterpart and (ii) the computation capabilities of both [55]. The design of effective offloading decision algorithms is outside the scope of this work.

5. Conclusions and Future Works

In this paper, we have presented a novel solution to enable the vision of AI deployed also at the deep edge in order to support intelligent IoT applications. The proposal relies on the virtualization concept, we borrowed from the IoT literature, and we specifically customized to meet the demands of emerging AI-powered IoT devices. We have designed a VIO, as a virtual counterpart of constrained IoT devices equipped with AI inference engines. For the semantic description of the cognitive device capabilities at the VIO, we relied on OMA LwM2M, to ensure interoperability and facilitate the discovery of AI capabilities by interested third-party applications requesting them. The conceived VIO also augments devices with storage capabilities, by caching inference results that may serve multiple consumer applications, as well as by pre-training and optimizing the pre-trained models to be injected into the constrained physical device. We develop a PoC to showcase the viability of the proposal. Results confirm a low pressure in terms of exchanged data on constrained devices, thanks to the usage of CoAP as a messaging protocol as well as to the caching of inference results at the VIO.

The proposal is intended to enable the semantic description of AI-powered (potentially constrained) devices and favor the transparent access to the output of the inference engine, regardless of the specific hardware/software implementation, while hiding details about the ANN model (and relevant settings) in charge of the inference task. Notwithstanding, the proposed VIO has been conceived with modularity in mind, and its usage can be extended to support additional functionalities, besides the abstraction for the consumer applications accessing AI resources as-a-service.

Hence, as a future work, we plan to apply the devised solution to specific distributed ML contexts, e.g., federated learning, where the workers and the aggregator node may need to interact for the interoperable exchange of models and relevant updates achieved through local training.

More in general, through proper extensions, the conceived proposal can be leveraged also to facilitate the orchestration of AI capabilities and resources along the cloud-to-things continuum, e.g., the chaining of cognitive components which are split among multiple (edge) devices with heterogeneous capabilities, as well as between the physical device and the VIO.

Author Contributions: Conceptualization and methodology, C.C., A.I. and A.M.; software, C.C. and G.G.; supervision, C.C., A.I. and A.M.; writing—original draft preparation, C.C.; writing—review and editing, C.C., G.G., A.I. and A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This publication is co-financed with the support of the European Commission, the European Social Fund, and the Calabria Region. The authors are solely responsible for this publication, and the European Commission and the Region of Calabria decline any responsibility for the use that may be made of the information contained therein.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

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

References

1. Ramos, E.; Morabito, R.; Kainulainen, J.P. Distributing Intelligence to the Edge and Beyond [Research Frontier]. *IEEE Comput. Intell. Mag.* **2019**, *14*, 65–92. [CrossRef]
2. Wang, X.; Han, Y.; Leung, V.C.; Niyato, D.; Yan, X.; Chen, X. Convergence of edge computing and deep learning: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 869–904. [CrossRef]
3. Taleb, T.; Samdanis, K.; Mada, B.; Flinck, H.; Dutta, S.; Sabella, D. On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 1657–1681. [CrossRef]
4. Zhou, Z.; Chen, X.; Li, E.; Zeng, L.; Luo, K.; Zhang, J. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proc. IEEE* **2019**, *107*, 1738–1762. [CrossRef]
5. Rausch, T.; Dustdar, S. Edge intelligence: The convergence of humans, things, and AI. In Proceedings of the 2019 IEEE International Conference on Cloud Engineering (IC2E), Prague, Czech Republic, 24–27 June 2019; pp. 86–96.
6. Li, H.; Ota, K.; Dong, M. Learning IoT in edge: Deep learning for the Internet of Things with edge computing. *IEEE Netw.* **2018**, *32*, 96–101. [CrossRef]
7. Doyu, H.; Morabito, R.; Höller, J. Bringing Machine Learning to the Deepest IoT Edge with TinyML as-a-Service. *IEEE IoT Newsl.* **2020**. Available online: https://www.researchgate.net/profile/Roberto_Morabito/publication/342916900_Bringing_Machine_Learning_to_the_Deepest_IoT_Edge_with_TinyML_as-a-Service/links/5f0d54f592851c38a51ce4d0/Bringing-Machine-Learning-to-the-Deepest-IoT-Edge-with-TinyML-as-a-Service.pdf (accessed on 12 December 2020).
8. Qi, X.; Liu, C. Enabling deep learning on iot edge: Approaches and evaluation. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 367–372.
9. Sanchez-Iborra, R.; Skarmeta, A.F. TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities. *IEEE Circuits Syst. Mag.* **2020**, *20*, 4–18. [CrossRef]
10. Peltonen, E.; Bennis, M.; Capobianco, M.; Debbah, M.; Ding, A.; Gil-Castiñeira, F.; Jurmu, M.; Karvonen, T.; Kelanti, M.; Kliks, A.; et al. 6G White Paper on Edge Intelligence. *arXiv* **2020**, arXiv:2004.14850
11. AI Expansion Pack for STM32CubeMX. Available online: https://www.st.com/en/embedded-software/x-cube-ai.html (accessed on 4 February 2021).
12. Snapdragon Neural Processing Engine SDK. Available online: https://developer.qualcomm.com/docs/snpe/overview.html (accessed on 4 February 2021).
13. Available online: https://github.com/uTensor/uTensor (accessed on 4 February 2021).
14. Liang, Q.; Shenoy, P.; Irwin, D. AI on the Edge: Rethinking AI-based IoT Applications Using Specialized Edge Architectures. *arXiv* **2020**, arXiv:2003.12488
15. ITU. *FG NET-2030-Additional Representative Use Cases and Key Network Requirements for Network 2030*; Technical Report; ITU: Geneva, Switzerland, 2020.
16. Nitti, M.; Pilloni, V.; Colistra, G.; Atzori, L. The virtual object as a major element of the internet of things: A survey. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1228–1240. [CrossRef]
17. Open Mobile Alliance, Lightweight Machine to Machine Technical Specification Core. V1_1-20180612-C. 2018. Available online: https://openmobilealliance.org/RELEASE/LightweightM2M/V1_1-20180612-C/OMA-TS-LightweightM2M_Transport-V1_1-20180612-C.pdf (accessed on 4 February 2021)
18. Bormann, C.; Castellani, A.P.; Shelby, Z. CoAP: An application protocol for billions of tiny internet nodes. *IEEE Internet Comput.* **2012**, *16*, 62–67. [CrossRef]
19. Alam, I.; Sharif, K.; Li, F.; Latif, Z.; Karim, M.M.; Nour, B.; Biswas, S.; Wang, Y. IoT virtualization: A survey of software definition & function virtualization techniques for internet of things. *arXiv* **2019**, arXiv:1902.10910
20. Giaffreda, R. iCore: A cognitive management framework for the Internet of Things. In *The Future Internet Assembly*; Springer: Berlin, Germany,2013; pp. 350–352.
21. Weyrich, M.; Ebert, C. Reference architectures for the internet of things. *IEEE Softw.* **2015**, *33*, 112–116. [CrossRef]
22. Fan, Q.; Ansari, N. On cost aware cloudlet placement for mobile edge computing. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 926–937. [CrossRef]
23. Sun, X.; Ansari, N. EdgeIoT: Mobile edge computing for the Internet of Things. *IEEE Commun. Mag.* **2016**, *54*, 22–29. [CrossRef]
24. Chukhno, O.; Chukhno, N.; Araniti, G.; Campolo, C.; Iera, A.; Molinaro, A. Optimal Placement of Social Digital Twins in Edge IoT Networks. *Sensors* **2020**, *20*, 6181. [CrossRef]
25. Jang, S.Y.; Lee, Y.; Shin, B.; Lee, D. Application-aware IoT camera virtualization for video analytics edge computing. In Proceedings of the 2018 IEEE/ACM Symposium on Edge Computing (SEC), Bellevue, WA, USA, 25–27 October 2018; pp. 132–144.
26. Savaglio, C., G.M.P.M.B.C.I.M.F.G. Agent-based Internet of Things: State-of-the-art and research challenges. *Future Gener. Comput. Syst.* **2020**, *102*, 1038–1053. [CrossRef]
27. Bădică, C.; Braubach, L.; Paschke, A. Rule-based distributed and agent systems. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*; Springer: Berlin, Germany, 2011; pp. 3–28.
28. Barnaghi, P.; Wang, W.; Henson, C.; Taylor, K. Semantics for the Internet of Things: Early progress and back to the future. *Int. J. Semant. Web Inf. Syst.* **2012**, *8*, 1–21. [CrossRef]
29. Vermesan, O.; Friess, P.; Guillemin, P.; Sundmaeker, H.; Eisenhauer, M.; Moessner, K.; Le Gall, F.; Cousin, P. Internet of things strategic research and innovation agenda. *River Publ. Ser. Commun.* **2013**, *7*, 56–80.

30. Maarala, A.I.; Su, X.; Riekki, J. Semantic reasoning for context-aware Internet of Things applications. *IEEE Internet Things J.* **2016**, *4*, 461–473. [CrossRef]
31. Semantic Sensor Network Ontology. Available online: https://www.w3.org/TR/vocab-ssn/ (accessed on 4 February 2021).
32. Available online: https://www.w3.org/TR/wot-thing-description/introduction (accessed on 4 February 2021).
33. Muralidharan, S.; Yoo, B.; Ko, H. Designing a Semantic Digital Twin model for IoT. In Proceedings of the 2020 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 4–6 January 2020; pp. 1–2.
34. oneM2M Partners. oneM2M Base Ontology. Available online: http://www.onem2m.org/technical/latest-drafts (accessed on 26 November 2020).
35. Lakka, E.; Petroulakis, N.E.; Hatzivasilis, G.; Soultatos, O.; Michalodimitrakis, M.; Rak, U.; Waledzik, K.; Anicic, D.; Kulkarni, V. End-to-End Semantic Interoperability Mechanisms for IoT. In Proceedings of the 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Limassol, Cyprus, 11–13 September 2019; pp. 1–6.
36. LwM2M Client-Anjay-Open-Source Software Development Kit. Available online: https://www.avsystem.com/products/anjay/ (accessed on 4 February 2021).
37. OMA Lightweight M2M IoT Agent: User and Development Guide. Available online: https://fiware-iotagent-lwm2m.readthedocs.io/en/latest/userGuide/index.html (accessed on 4 February 2021).
38. Atzori, L.; Bellido, J.L.; Bolla, R.; Genovese, G.; Iera, A.; Jara, A.; Lombardo, C.; Morabito, G. SDN&NFV contribution to IoT objects virtualization. *Comput. Netw.* **2019**, *149*, 200–212.
39. Karaagac, A.; Verbeeck, N.; Hoebeke, J. The integration of LwM2M and OPC UA: An interoperability approach for industrial IoT. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 313–318.
40. Choi, D.K.; Jung, J.H.; Kim, J.I.; Gohar, M.; Koh, S.J. IoT-Based Resource Control for In-Vehicle Infotainment Services: Design and Experimentation. *Sensors* **2019**, *19*, 620. [CrossRef] [PubMed]
41. Klas, G.; Rodermund, F.; Shelby, Z.; Akhouri, S.; Holler, J. *OMA Whitepaper LightweightM2M*; OMA SpecWorks: San Diego, CA, USA, 2014.
42. OMA LightweightM2M (LwM2M) Object and Resource Registry. Available online: www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.html (accessed on 4 February 2021).
43. Banbury, C.R.; Reddi, V.J.; Lam, M.; Fu, W.; Fazel, A.; Holleman, J.; Huang, X.; Hurtado, R.; Kanter, D.; Lokhmotov, A.; et al. Benchmarking TinyML Systems: Challenges and Direction. *arXiv* **2020**, arXiv:2003.04821
44. Abiodun, O.I.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Mohamed, N.A.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. *Heliyon* **2018**, *4*, e00938. [CrossRef] [PubMed]
45. De Coninck, E.; Verbelen, T.; Vankeirsbilck, B.; Bohez, S.; Leroux, S.; Simoens, P. Dianne: Distributed artificial neural networks for the internet of things. In Proceedings of the 2nd Workshop on Middleware for Context-Aware Applications in the IoT, New York, NY, USA, 7–11 December 2015; pp. 19–24.
46. STM32 Solutions for Artificial Neural Networks. Available online: https://www.st.com/content/st_com/en/stm32-ann.html (accessed on 4 February 2021).
47. Available online: http://www.openmobilealliance.org/wp/OMNA/LwM2M/LwM2MRegistry.htmlresources (accessed on 4 February 2021).
48. OMA Lightweight M2M Server and Client in Java. Available online: https://www.eclipse.org/leshan/ (accessed on 4 February 2021).
49. LWM2M Supported Features. Available online: https://github.com/eclipse/leshan/wiki/LWM2M-Supported-features (accessed on 4 February 2021).
50. Wireshark. Go Deep. Available online: https://www.wireshark.org/ (accessed on 4 February 2021).
51. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
52. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
53. Cao, M.T.; Tran, Q.V.; Nguyen, N.M.; Chang, K.T. Survey on performance of deep learning models for detecting road damages using multiple dashcam image resources. *Adv. Eng. Inf.* **2020**, *46*, 101182. [CrossRef]
54. Wang, Y.; Liu, M.; Zheng, P.; Yang, H.; Zou, J. A smart surface inspection system using faster R-CNN in cloud-edge computing environment. *Adv. Eng. Inf.* **2020**, *43*, 101037. [CrossRef]
55. Sun, Y.; Shi, W.; Huang, X.; Zhou, S.; Niu, Z. Edge Learning with Timeliness Constraints: Challenges and Solutions. *IEEE Commun. Mag.* **2020**, *58*, 27–33. [CrossRef]