electronics

MDPI

Article

Smart Parking System Based on Edge-Cloud-Dew Computing Architecture

Yuan-Chih Yu

Department of Information Management, Chinese Culture University, Taipei 111, Taiwan; yyz2@ulive.pccu.edu.tw

Abstract: In a smart parking system, the license plate recognition service controls the car's entry and exit and plays the core role in the parking lot system. When the Internet is interrupted, the parking lot's business will also be interrupted. Hence, we proposed an Edge-Cloud-Dew architecture for the mobile industry in order to tackle this critical problem. The architecture has an innovative design, including LAN-level deployment, Platform-as-a-Dew Service (PaaDS), the dew version of license plate recognition, and the dew type of machine learning model training. Based on these designs, the architecture presents many benefits, such as: (1) reduced maintenance and deployment issues and increased dew service reliability and sustainability; (2) effective release of the network constraint on cloud computing and increase in the horizontal and vertical scalability of the system; (3) enhancement of dew computing to resolve the heavy computing process problem; and (4) proposal of a dew type of machine learning training mechanism without requiring periodic retraining, but with acceptable accuracy. Finally, business owners can reduce their burdens when introducing machine learning technology. Our research goal is to make parking systems smarter in edge computing through the integration of cloud and dew architecture technology.

Keywords: dew computing; cloud service; image recognition service; smart parking system

Citation: Yu, Y.-C. Smart Parking System Based on Edge-Cloud-Dew Computing Architecture. *Electronics* **2023**, *12*, 2801. https://doi.org/10.3390/electronics12132801

Academic Editor: Bahman Javadi

Received: 13 May 2023
Revised: 16 June 2023
Accepted: 21 June 2023
Published: 25 June 2023

1. Introduction

The introduction of automated, digitized, and smart factors into parking systems can improve labor costs, user efficiency, and management. Among these improvements, smart factors are the most anticipated innovation. Equipped with network sensing and camera monitoring in the parking space, a smart parking management system (or so-called smart parking system) can transmit the on-site conditions instantly to the central control in order to sense various abnormalities, alleviate real-time problems, and guide vehicles, based on management viewpoints, to assist in the control of parking access [1]. Current smart parking management system trends employ innovative technologies to effectively manage and add value to the limited resources. Smart parking is an innovative parking technology that aims to use a smart strategy to deal with these limited resources more efficiently.

The smart parking system includes many basic functionalities: entrance and exit gate passage control, license plate recognition, movement line and parking space guidance, car search query, automatic billing, charging systems, etc. The system also has the ability to cooperate with the cloud and then be integrated as a cloud service [2]. All information is assembled on a central management platform, and more automated, digital, and intelligent functions are introduced through cloud-based intelligent computing and analysis services.

The most popular trend in smart parking systems presently is the use of image recognition for license plates. Its main purpose is to allow for control when entering and exiting the parking lot and to allow the driver to choose and pay when paying the toll. This not only makes driving more automated and convenient, but also greatly reduces the time needed for entering and exiting the traditional parking lot. In addition, with the concept of the Artificial Internet of Things (AIoT), the system can provide multiple advanced functionalities, including parking searching, reservation, payment, notifications,

statistics, monitoring, and even context awareness [3]. Another direction for developing advanced functionalities is via the cloud. Cloud services provide enterprises with more diverse and scalable solutions. For instance, using cloud services for image recognition may allow the heavy usage of local computing resources to be avoided and reduce the need for large image storage.

In terms of the above goals, we can develop a series of research questions. A cloud service system could be compromised when the network is unavailable or unstable; in addition, for operational auditing, there should also be a copy of the image recognition data available on the local site. After considering the importance of availability and reliability, building an on-premise service for image recognition may be an inevitable choice in terms of more realistic and robust features. Again, this raises next-level questions, even without considering the computing and storage issues previously mentioned. Building an on-premise image recognition service is not an easy task, and usually requires dedicated machine learning training and long-term performance adjustments; the parking lot operating office cannot often afford these improvements. Moreover, when many enterprise information systems, such as customer relationship management (CRM), human resource management (HRM), supply chain management (SCM), and messaging applications, have gradually been deployed in the cloud, re-embracing the on-premise design is not a good choice, because the collaboration with remote cloud enterprise systems requires the setup of a complex connection network.

Dew computing could be the answer, provided that such a specific hybrid infrastructure has on-premise/hosted private cloud features that can be independent of and collaborative with external cloud services [4]. This is not only relevant to the design of an on-premise private cloud dedicated to the parking lot system, but also to a proxy of a remotely hosted private cloud on the local site [5]. Specifically, the on-premises cloud gives us more leverage and control of our dew computing architecture to and increases the availability and reliability. Meanwhile, a proxy of a hosted private cloud allows us to conveniently access the cloud enterprise, as well as cloud resources and utilities, with better management and scalability.

Hence, utilizing the advantages of cloud-dew architecture to enhance the availability of smart parking systems has become the main motivation of this research. The problematic domain which we intend to frame is the real-time cloud image recognition service. We believe that an on-premise image recognition system is not suitable for the development of a smart parking system. In the future, most smart parking systems will be equipped with default cloud services. The reasons are as follows:

1. The trend in system development: The usage of cloud services is a global trend. It can effectively utilize other cloud resources and increase the horizontal and vertical scalability of the system.
2. Economic efficiency with scalability: Most smart parking systems rely heavily on image recognition to complete many innovation jobs, so the local build will experience the problem of relying heavily on computing and storage resource provision. This requires a great deal of expensive investment. However, the business demand of a parking lot fluctuates. When it comes to holidays or public events, parking services are usually very tight, and the quality of service is low. When the parking lot is in the off-season, its services are idle and a waste of investment. Hence, in the long run, using scalable cloud services is a better choice.
3. Issues of system maintenance: The machine learning model used for image recognition needs to be retrained regularly to reflect changes in the environment, such as license plate format revision, shifting of the camera's viewing angle, or fluctuations in lighting conditions. In these situations, business owners are usually unable to adjust or improve the system themselves, while cloud service providers can seamlessly adjust it remotely.

Hence, how to resolve the network's reliance on real-time cloud image recognition service is very important. Sooner or later, we will inevitably face the challenge of the

network of the smart parking system needing to be constantly connected to the internet. We hope that, through our proposed architecture, smart parking systems will still be able to operate their services all the time, with or without an internet connection. This paper is an extended version of conference paper published in COMPSAC 2021 [6]. We revised our original architecture design and significantly extend the study in the literature review, methodology, and evaluation.

2. Related Works

There are many domain concepts related to this research. We will review them in order as follows to clarify the related definitions.

2.1. Smart Parking System

The parking area of a large city often occupies 31% of its land use [7], so the management of parking areas affects the operation of a city enormously every day. If the driver could obtain parking information in real-time, they would be able to adjust their schedules and activities efficiently. Hence, many city officers are eager to introduce smart parking systems, hoping to create more convenient parking services and to help drivers to find parking spaces more effectively [8]. In general, the parking system has some common business problems that need to be solved; thus, it needs to be upgraded to become more automatic and intelligent. The common business problems are:

1. The significant manual workload of parking lot inspection and ticketing;
2. The inefficiency of parking space usage;
3. Inability to allocate the optimal location and layout of parking space for new incoming cars [9].

Smart parking systems (SPSs), combined with the artificial intelligence Internet of Things (AIoT), have been widely used in Europe and North America. Smart parking systems (or smart parking management systems) refer to parking management systems that combine cloud computing technology, databases, and mobile devices in a parking environment [9]. In a smart parking system, a "parking space" is the basic logical unit of all service and management. Knowing the real-time status of all parking spaces can resolve many service problems. In most cases, a well-deployed sensor network can fulfill the requirement of real-time parking space detection. The construction of a sensor network for a parking system involves three different aspects: sensors, computing, and communications. The first is sensors, such as instruments, detectors/actuators, or monitors, that interact with the environment. These multiple sensors respond to changes in the environment and generate signal data. The advantage of using multiple sensors lies in multi-faceted perception; they complement each other to detect blind spots and can be used for sensor fusion to form a multi-dimensional data viewpoint with time series correction, which can be used for back-end spatial–temporal data mining and big data analysis to gain deeper insight.

Compared with the complexity of the sensor network, using image recognition to detect the state of a parking space is simpler and more flexible. As long as some cameras are placed in the parking lot, an intelligent image analysis system can replace the role of the parking lot sensor. Like the popular facial recognition technology used by the surveillance system to control human access, license plate recognition is used to control cars' access to the entrance of the parking lot in order to effectively control the entry and exit of vehicles. It can also be used in public places where trucks often enter and exit in the absence of supervision, such as communities or commercial buildings.

Currently, a large number of high-definition cameras, such as 1440 p (4 MP), 1080 p (2 MP), and 720 p cameras, are widely used to fit security needs regarding camera resolution. Therefore, based on the availability of abundant high-quality images, the development of visual recognition technology using deep learning has gradually attracted attention in various fields of application. For instance, parking systems use wide-angle fisheye lenses or catadioptric cameras to realize visual recognition. This method uses a simple camera

module to calibrate a space line without knowing its position and direction. It automatically extracts the parking space's boundary line from the input wide-angle image, and can easily change its direction. The parking space boundary is used to divide the parking grid and to detect empty parking spaces through background subtraction [10,11].

However, visual recognition using a deep learning model with full-field camera identification relies heavily on CPU and GPU for real-time computing, so it is very dependent on hardware and electric power. Many systems use cloud image identification services to relieve the burden of the local computing requirements. However, the customization of cloud services is not easy, and the fees are expensive, which is disadvantageous in scenarios which heavily use computing in smart parking systems. Therefore, some studies have proposed an energy-saving algorithm architecture (such as YOLO), which would use agile object detection algorithms to detect the availability in parking lots using embedded artificial intelligence edge devices (such as TX2) [12].

2.2. Edage-Cloud-Dew Computing Architecture

The definition of dew computing and its applications are still under development and need to be further discussed [13]. Here, we only list the published literature which does not involve business research. Dew computing works together with cloud services and controls edge/fog devices, including sensors, routers, and IoT devices. Initially, the cloud-dew architecture can be viewed as an extension of the existing client–server architecture, which forms the three-level communication link pipe (client–dew server–server) [14,15]. With the mediation of a dew server, a client can still access website services without an Internet connection, using a local form of an actual website. The reason for this is that the related data are stored in the dew server as a local copy, which is synchronized with the master copy in the cloud server. A duplicate (local copy) of the visited website on a user's local computer is called a "Dew Site" [15]. Once such a dew site exists in the local machine, the user can browse the web content from the dew site and perform the necessary operations. Moreover, through the dew domain naming system, name mapping between different local dew sites has become possible [15]. As a result, novel concepts of services can be proposed, such as infrastructure-as-a-dew, a software-as-a-dew service, and software-as-a-dew products [15,16].

Although dew devices are often commodity computers (desktop, laptop, tablet, smartphone, etc.), with appropriate software support, they can provide functionality independently of cloud services and work as mediators between fog and cloud. Dew computing is an additional layer between end-user devices to process and coordinate with the Edge/Fog/Cloud layers, offering autonomy, independence, and collaboration features [17].

Dew computing, such as fog computing closer to the user, also seeks to optimize resource allocation before processing and before the analysis has been transferred into the cloud, hence reducing the complexity and improving the productivity of distributed computing architecture [18]. Dew computing provides a solution to organizing non-cloud components when enterprises are required to upgrade IT infrastructure with the cloud. Its main goal is to access a pool of raw data/metadata that can be rapidly created, edited, stored, and deleted offline with minimal internetwork management effort [14]. Dew computing cam accelerate computing services across devices. Hence, some research has applied dew computing to a 5G IoT coalition formation game [19].

Dew computing has two key features—independence of external systems and collaboration with cloud servers—and it works in two modes [4,20]:

- Localized mode: where all the services are provided within the internal local network perimeter;
- Global mode: where it functions as an intermediate device in the client–server cloud model.

In addition, the dew computing model is composed of six essential characteristics: rule-based data collection, synchronization, scalability, re-origination, transparency, and anytime/anyhow accessibility [15].

Cloud-dew computing architecture enables the personal cloud data to be continuously accessible by a new type of application without an Internet connection [14]. Some of the obvious examples of dew applications are Dropbox, OneDrive, and Google Drive Offline. Users can use these services regardless of their Internet connection, and they can be synchronized with cloud services [18]. Dew computing accelerates computing services across devices for 5G IoT using a coalition formation game.

Dew computing speeds up computing services across devices and allows data output to be generated very quickly.

To employ cloud-dew architecture on a local computer, a dew virtual machine is required (DVM). The DVM is an isolated environment for implementing the dew server on a local computer [15]. While a cloud server is considered as an orating cloud in the sky, a dew server is thought of as dew upon a plant leaf [4]. The dew server (DS) acts like the cloud service on the local computer, which can work in a closed environment without the use of external server systems [20]. When Internet is available, it interacts with and periodically synchronizes content with the cloud service.

Dew servers store user data/metadata with capacities smaller than those of cloud servers. Generally, a dew server serves only one client with cloud services, and the database of the dew server takes care of the synchronization with the cloud database. Dew servers can be reinstalled with the help of cloud data backup and can be accessed independently of an Internet connection, as they run on local computers [18].

Based on the application field, Wang [4] proposed categories of dew computing, including Web in Dew (WiD), Storage in Dew (SiD), Database in Dew (DBiD), Software in Dew (SiD), Platform in Dew (PiD), Infrastructure as Dew (IaD), and Data in Dew (DiD) [4]. All application fields run on applications in a distributed and hierarchical environment, without requiring continuous intervention from a remote central control server [15]. In SiD, the client's local computer has a cloud copy of the software ownership and settings. The existence of the software on the client's local computer and the setting information recorded in the cloud service both reflect the ownership of the client. The client can re-download the software when necessary [4]. Apple's App Store and Google Play are examples of SiD [21]. However, this use case does not exactly fit our situation.

The most relevant categories to our research are Platform in Dew (PiD) and Software in Dew (SiD), as the response of image recognition services can be viewed as a collection of software supporting the operation for specific purposes, and also the settings of image recognition services can be considered as software settings for the dew site [16]. In our case, the situation is more complicated, and Wang's classification [4] may not exactly cover it. The image recognition service is not just a dataset or piece of software; a machine learning model is embedded within it, constantly requiring training. When it is woven into dew computing, the synchronized mechanism is more complicated. We will explain this in the section focusing on the research method.

In Ray's research [15], he proposed an extended version of the dew cloud definition. The local machine is comprised of four components: (1) the dew server, (2) the dew DBMS, (3) the dew client program, and (4) the dew client service application. Corresponding to the cloud computing service models, he presented the following viewpoints:

1. He described the intersection of Software as a Service (SaaS) with the dew computing paradigm as Software-as-a-Dew Service (SaaDS) and Software-as-Dew Product (SaaDP). When one goes offline on SaaS, the other continually holds the business under the SaaS service model. Related data will be stored partly in cloud servers and partly in the local machine. The sew server can access and update the information stored in the local machine and also that synchronized with the cloud server.
2. He also described the intersection of Infrastructure-as-a-Service (IaaS) with the dew computing paradigm as Infrastructure-as-a-Dew Service (IaaDS). The dew server ensures that the local infrastructure is safely recovered from the cloud when infrastructure-related data is lost or destroyed. Hence, the dew server must store all settings related to the infrastructure's data as a backup copy into the cloud.

Like the traditional cognition on dew computing, Ray's introduction to dew computing [15] focused on defining dew components in terms of an internal local network of a single machine. However, for some advanced applications, it is impractical and unreliable to use a single-client machine to deploy the entirety of the dew architecture, because enterprise business operations often rely on resources of the local area network (LAN). Specifically, for machine learning techniques, the image recognition service relies heavily on computing resources and data processing space. Hence, our dew architecture was deployed on the LAN-level, and the dew server was a real local server dedicated to performing dew service tasks. Furthermore, the position of the image recognition service here was the cloud API used remotely in smart parking applications. It, like Microsoft's Cognitive Services APIs, is a cloud platform-based service (PaaS: Platform as a Service) that leverages the latest AI technologies in the cloud. When it is woven into dew computing architecture, it becomes a new type of category: ML API in Dew (MLiD) or Platform-as-a-Dew Service (PaaDS), as we refer to it. This is not similar to the "an API for dew computing services" study [22], which provided services to other users by means of a local device.

In the study by Mishra et al. [23], they proposed a dew-enabled vehicular fog computing model dedicated to dealing with the disaster challenge of latency-sensitive and computing-intensive tasks of vehicular networks. Their work consisted of a practical application, like our research focus, in parking lot management, and dew-enabled AIoT, which adapts Deep Q-Learning-based (FedDQL) techniques for the optimal offloading of tasks in a collaborative computing paradigm, was also facilitated in their architecture. However, due to different application domain, they emphasized fog computing on a vehicular network, which would take place further away from the sensors generating the data. On the other hand, in our application domain, we emphasized the edge computing taking place directly on the AIoT device, which was physically connected to the camera monitoring sensors. We can refer to this as a new category—an ML model in dew (MMiD) or a Model-as-a-Dew Service (MaaDS). To clarify the aforementioned points, we summarize these related discussions in Table 1.

Table 1. A comparison of dew computing categories, based on [5,15,20–23].

Category	Resource in Dew	Default Carrier	Key Function	Applications (Use Case)
WiD	A fraction of web content	PC	Access a fraction of the web without Internet connection	Dropbox
STiD	Storage	PC	Storage in dew has a cloud copy	
DBiD	Database	PC	Local database has a cloud backup	
SiD	Software	PC	Software ownership and settings have a cloud copy	Apple App Store Google play
PiD	Platform/Library	PC	SDK and projects have a cloud copy	GitHub
IaD	Computer/VM/Container	PC	On-premise computer settings and data have a cloud copy	
DiD	Data in forms other than above	PC	Dew computing applications not in above categories	Novell Groupwise 7
SaaDS/SaaDP	Software suite/Product	PC/VM	Local and distributed software access	
IaaDS	Virtualized system	PC/VM	Regeneration of cloud platform and relevant services	
MLiD/DaaDS	A proxy of cloud AI services on LAN	PC/VM/LAN server	A lite cloud AI service on LAN without Internet connection	Smart parking system
MMiD/MaaDS	Vehicular Fog Computing (VFC) networks	Vehicular MCUs	Performs computations, communications, offloading, and resource utilization, considering the latency and energy consumption	Vehicular computing

2.3. AIoT

For parking systems, a changing number of cars competing for a limited number of parking spaces is always a major concern. Through technological innovation, AIot (Artificial Intelligence of Things) presents a viable method for alleviating the inefficient usage of existing parking spaces. AIot, a combination of AI and IoT, helps the system to achieve ambient automation, which gives it self-healing and self-management properties, by leveraging IoT sensing and AI decision-making capabilities. IoT sensors collect spatial–temporal data and response information to the cloud service. In the cloud, the AI deep learning pipeline incrementally trains the model and uses prescriptive and predictive analytics for advanced applications. For instance, a smart pest monitoring system uses the environment and wireless image sensors installed inside greenhouses to monitor the population density of pests in hotspots [24]. Deep learning-based techniques for object segmentation, detection, and classification are used to analyze pest populations.

In our problem domain, the primary goal was to make the parking system smarter and more efficient through the additive effects of AIot and dew computing. Dew computing can help an AIoT device to remain available without an Internet connection. It not only detaches applications from the cloud, but also efficiently collaborates with cloud services [25].

3. Materials and Method

Since the theoretical framework of dew computing is still being formed, we proposed a dedicated form of dew computing architecture for the smart parking system without considering its generalization or standardization [16,20].

3.1. The Cloud-Dew Architecture for Smart Parking System

In this research, the cloud-dew architecture was deployed as an on-premise/hosted hybrid cloud environment. As Figure 1 depicts, when an Internet connection is available, the dew computing at the local site can replace the role of remote cloud services. We explain the deployment as follows.

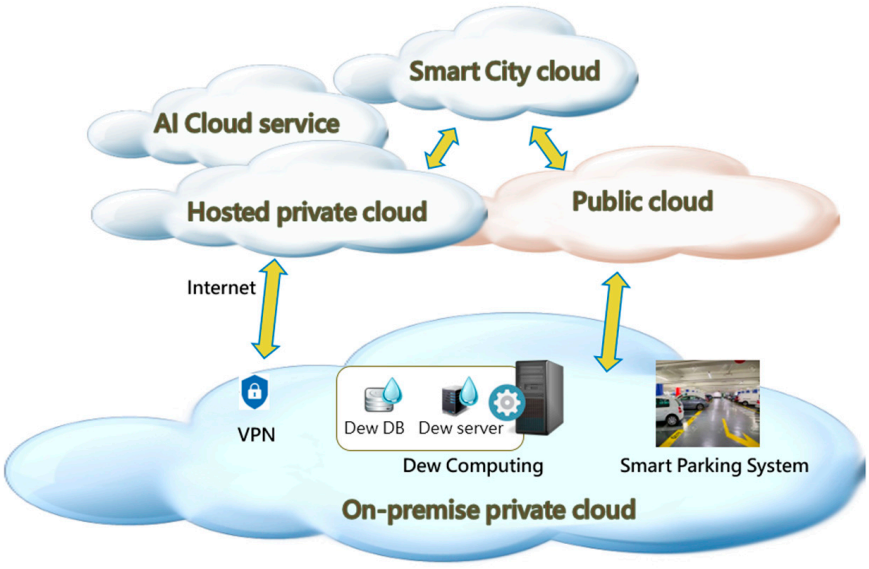

Figure 1. Hybrid-cloud environment for the deployment of dew computing.

1. On-premise private cloud: Private clouds comprise two primary types: externally hosted cloud solutions and internally locally hosted (or on-premise hardware) solutions. The main differences are the overall cost, level of support required, and feature scaling. In this case, we adopt both types of private cloud. The on-premise one can provide the components which require performance-critical missions, and the external cloud can provide the components which require massive batch computing

or long-term persistence. Thus, in the on-premise private cloud, we had the client side of the smart parking and dew computing systems, and on the hosted private cloud, we had the server side of the smart parking system.

2. Hosted private cloud: This is the core version of the smart parking system when an Internet connection is available. The dew server synchronizes with this version of the parking system.
3. AI service cloud (third-party AI cloud service): Cloud service providers (CSP) provide services dedicated to specific tasks: object detection via video, face recognition of celebrities, and speech-to-text conversion. Some of these providers even offer advanced computing platforms (e.g., AI Platform as a Service (AI PaaS)). Instead of using a customized machine software package, some of the in-operation parking systems have begun to use third-party AI cloud services to assemble their image recognition systems. In our case, for the purpose of reducing the complicity of dew computing architecture, we built our own image recognition service from scratch.
4. Smart city cloud (community cloud): By means of a smart city cloud, smart city services can be provided. For instance, when a vehicle departs from a parking space, city-wide IoT sensors notify the driver of available parking spaces via a smartphone app or messaging. Also, by using the cloud in the field of smart parking, stakeholders and governments can use the data for storage, analysis, and sharing in order to take the most effective actions.

To present the operating mechanism of this cloud-dew architecture, we explain the involved components in Figure 2 as follows.

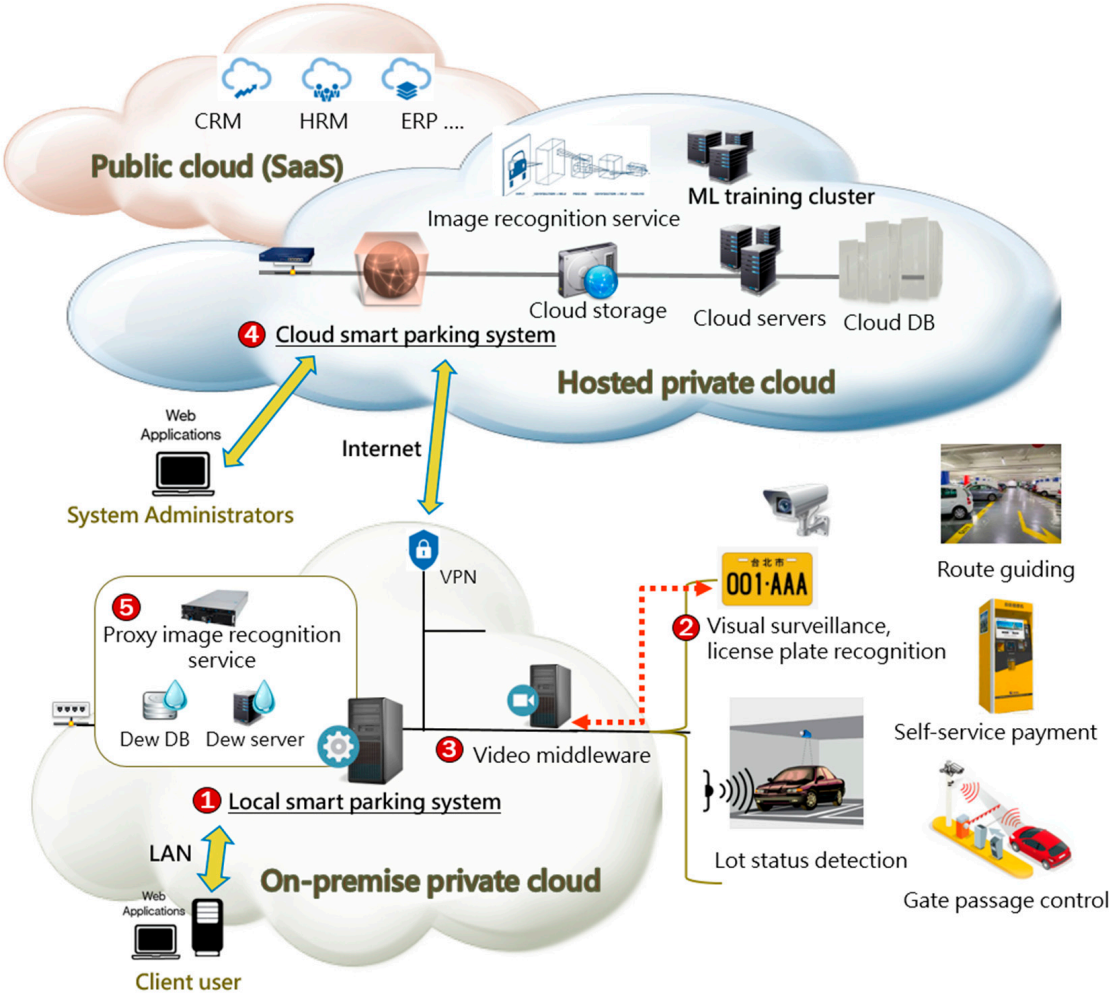

Figure 2. The cloud-dew architecture for a smart parking system.

1. Local smart parking system: In the LAN environment, the client user interacts with the local smart parking system (the client-side agent). The agent takes care of communicating with the server side of the smart parking system for the purpose of system coordination. Also, the dew server periodically synchronizes with the server-side system in the background to prepare the dew service's functions. When executing the image recognition function, it uses the cloud version of the image recognition service by default. However, when the Internet is unavailable, it functions as a proxy of the cloud image recognition service in order to execute the car plate recognition task.
2. Image recognition tasks: The tasks include visual surveillance, license plate recognition, parking space detection, etc. These computing processes need to stay local for real-time performance. In our case, only the license plate recognition task was executed for simplicity.
3. Video management middleware: For large-scale video surveillance, the middleware management system provides a consolidated hub of management capabilities, including system management, user management, and various extensive services. It also provides the buffer for image data processing.
4. Cloud smart parking system: The server-side smart parking system collects the parking transaction data for operational analysis and auditing. It provides the cloud service/API for the client-side agent to use. In addition, it acts as a portal for administrators. By combining with the public cloud, it can collaborate with the enterprise resource planning system.
5. Proxy image recognition service: This service is activated when the Internet is unavailable. The dew server is responsible for initializing and running it, and then shutting it down when the Internet is restored. It uses a lite version of the machine learning model to perform image recognition. A lite version of the model is downloaded in advance and updated at every scheduled time.

3.2. The Operational Flow of Dew License Plate Recognition Service

The local smart parking system has a dew server to ensure that the proxy image recognition service is working perfectly under a situation of unavailable Internet access, as shown in Figure 3. Under normal circumstances, the system fully delegates cloud services for image recognition. The local system does not truly operate a machine learning model using proxy image recognition services. Instead, it simply routes the request to the cloud, then copies the request and response as a record of enhanced training data which the dew server can use to prepare its training for the initialization of proxy image recognition. Each record of enhanced training data consists of the image plus context features (request) and a label (response). The enhanced data denote the most recent recognition events, reflecting the current entrance/exit status of the parking lot. It highly corresponds with the current parking conditions, and can increase the performance of the server in terms of proxy image recognition.

In the cloud, the system administrator collects several batches of new training data in order to incrementally train the machine learning (ML) model to maintain its performance. Also, the system automatically converts the original model into a lite version of the model, which acts as an on-device ML solution for mobile and edge use cases.

Therefore, the ML model of proxy image recognition is a lightweight deep learning model, and the dew server initializes it when the Internet is interrupted. The initialization process requires several preparation steps:

1. Proxy image recognition model: A lite version of the model is downloaded from the cloud. Since the model is always a copy of the latest cloud version, we do not need to spend effort on maintenance or adjustment. However, it may not fit with the dynamic changes in the parking lot environment. We can enhance it using the recently collected dataset, which consists of the aforementioned enhanced training data. Although the initial performance of the lite proxy model may not be of high enough quality for

real-time operation, after enhanced training, it approaches the performance level of the cloud image recognition model.

2. Dew container: The proxy image recognition service is an API program hosted in a container environment. The dew container takes care of the initialization/destruction of the container from the image repository. After the container is prepared, the service can be run on it.
3. Dew ML training: We use the enhanced training data to train the proxy image recognition model. The whole process is intended to fulfill the function of the dew service; thus, we call it dew ML training.
4. Dew DB: The dew DB stores the configuration of the dew service, copies of pieces of cloud software, dew hyper-parameters, and fast training strategies and policies for dew ML training. The dew server uses them for the initialization of the proxy service.

The steps of the initialization of the dew service are:

1. From the dew DB, gather the initialization-related data for the following process;
2. Initial the dew container to prepare and activate the executive platform for the proxy image recognition service program;
3. Fetch the lite version of the proxy image recognition model and the enhanced dataset from local storage for the following enhanced training;
4. Train the proxy image recognition model using the enhanced dataset. Then, deploy the model and optimize the model into the proxy image recognition service;
5. Operate the dew version of the license plate recognition service.

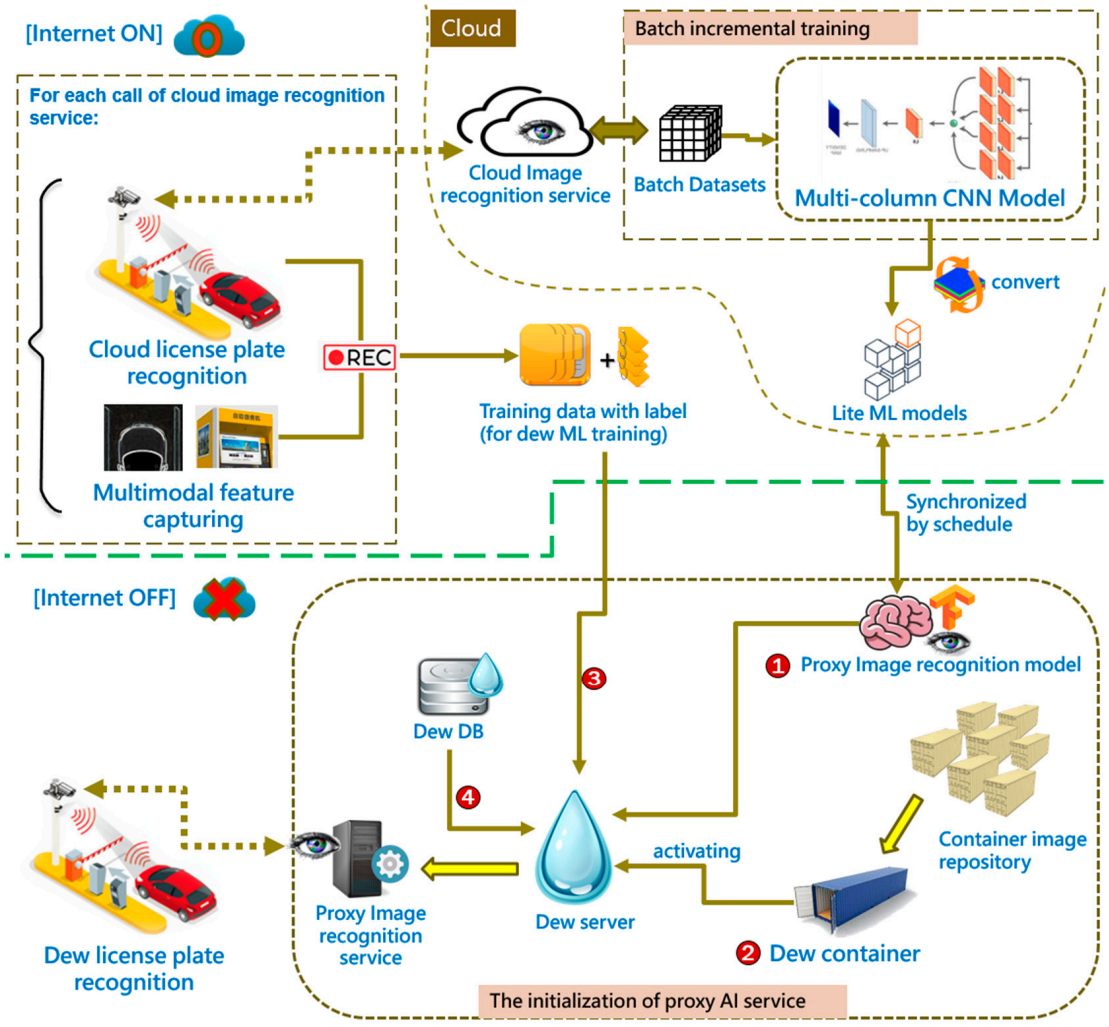

Figure 3. The working model of dew computing.

Moreover, for the purpose of increasing the training performance, we can combine sensor network or payment information to provide the collection of enhanced data with more features in order to obtain a better outcome. Enhance training does not need to be conducted on high-performance computers, as it is a lightweight and limited training dataset. The results of its implementation can be simplifying and cost-cutting.

4. Evaluation

Full Internet reliance is the main disadvantage of cloud computing. When the Internet connection goes down, everything is out. This drawback has been amplified, especially when relying on cloud image recognition services in smart parking systems. Hence, our research proposed a type of cloud-dew architecture dedicated to tackling this problem.

To present the practicality of our solution, our evaluation focuses on analyzing the performance of the license plate recognition service using our cloud-dew architecture. As described in the definition of dew computing [26], "computer functionality can run independently of services in cloud computing, but also can collaborate with services in cloud computing." We measured the performance of the license plate recognition service both with and without an Internet connection.

The first step is to define the measuring method according to our real-world application. In information theory, cross-entropy measures the average number of bits by actually encoding the information. Assuming that what we observe for the classification is perfect, the cross-entropy will be equal to the entropy of the real world itself. But if our assumptions are mistaken, cross-entropy will be greater to some extent— this extent is called the Kullback–Leibler (KL) divergence [27].

Kullback–Leibler divergence measures the difference between two probability distributions of the same variable, or, more simply, it calculates the difference between the actual distribution and the observed distribution. KL divergence may not be interpreted as a "distance measure" between two distributions, but rather as a measure of an increase in entropy due to the use of an approximation of the true distribution rather than the true distribution itself. KL divergence (or "relative entropy") "measures the inefficiency caused by the approximation".

Suppose our proxy dew image recognition model has 84% accuracy, which is what a typical simple CNN (convolutional neural network) network can achieve, and the original cloud image recognition model has 98% accuracy. Between two architectures, there is a gap (x) with which dew architecture can never catch up ("architecture bias"). Also, when converted to the lite version, we can assume that the performance drop will be about 4%. The performance drop refers to the article cited in [28], in which the researchers converted Tensor-Flow models to the Tensor-Flow Lite format, and a mean performance loss of around 3.x% accuracy was achieved. The uncertainty condition could be encoded as Table 2.

Table 2. An example of uncertainty condition encoding.

	Uncertainty		
	Recognizable	**Unrecognizable**	**Arch. Bias**
Cloud arch.	0.98	0.02	NA
Dew arch. (normal model)	0.84	0.16-x	X_1
Dew arch. (lite model)	0.806	0.194-x	X_2

With KL divergence (as in Equation (1)), we can calculate exactly how much information is lost when we approximate one architecture with another.

$$D_{KL}(dew||cloud) = -E_{x\sim dew}[\ln cloud(x) - \ln dew(x)] = -E_{x\sim dew}[\ln \frac{cloud(x)}{dew(x)}] \quad (1)$$

$$KLD(cloud, dew) = \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \quad (2)$$

Under perfect conditions, assuming that the design bias is zero, we can calculate the Kullback–Leibler (KL) divergence based on Table 2. If we use mean = "recognizable rate" and variance = 5% based on Equation (2), the KL divergence for the dew architecture (normal model) is 14.288 and that for the dew architecture (lite model) is 22.577. Although these values do not represent quantified information, we can optimize our dew architecture by minimizing the KL divergence in the testing mode. Based on the KL divergence, we can slightly change the architecture in order to judge whether the performance bottleneck is due to the dew architecture itself or the machine learning model conversion.

5. Discussion

5.1. Architecture Quality

Petrillo et al. [29] conducted a systematic literature review of software architecture metrics and revealed a list of the common quality features of software architecture, including maintainability, extensibility, simplicity, re-usability, and performance. We will discuss each perspective as it relates to our design, respectively.

However, as the authors of [29] emphasized, the quality of an architectural design cannot be quantified, as a qualitative evaluation in terms of the project's requirements is required. Also, most architecture metrics are not designed to measure one quality feature, and a quantitative metric often combines many perspectives of many quality features. For this reason, we only discuss our solutions regarding the common quality features, and do not go through the metric level.

- Maintainability: In cloud-dew architecture, change may occur in the cloud or on-premise. Making changes in software, whether for improvement or for the debugging process, is not an easy task. In our architecture, the dew server is responsible for the core task of providing dew services. As we constructed the dew server on the LAN-level, the deployment issue can be reduced to once rather than be installed again each time a new client machine uses the system. Moreover, every client's machine has a different operating environment, and we cannot ensure that the deployment of dew software can avoid software conflict problems. Although LAN-level dew architecture may suffer the heavy-weight of architectural problems, it can largely reduce maintenance and deployment issues and increase the dew service's reliability and sustainability. It is very important that software be easy to maintain, as it affects whether the system is acceptable for the user or not.
- Extensibility: Similarly, LAN-level dew architecture has a better ability to handle the addition of new functionalities and components through dew function decoupling in LAN design.
- Simplicity and understandability: Like the principle of Occam's razor—"the simplest explanation is usually the best one"—making architecture as simple as possible usually makes it more explainable and understandable for the following missions. Although our architecture may be large and heavy, the mechanism is transparent and understandable. The fact that it is not a black box brings more flexibility and reduces its complexity. An understandable form of architecture is more maintainable, extensible, and reliable.
- Re-usability: Reusing software components makes development and design more responsive. Making a software architecture reusable is often very valuable to that software's robustness. Herein, we reused the machine learning model from the cloud site and converted it into a lite version, resulting in a more robust architecture with exceptional handling.
- Performance: As the previous evaluation section presented, our smart parking system emphasized the performance of the license plate recognition service. Thus, cloud-dew architecture is able to keep the performance stable when the Internet is interrupted.

5.2. Cloud-Dew Service Model

Cloud computing defines three service models, known as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). When the three cloud service models join with dew computing to form a cloud-dew architecture, there is a need to discuss the new service model type. As mentioned in the previous section, we refer to them as the Software-as-a-Dew Service (SaaDS), Platform-as-a-Dew Service (PaaDS), and Infrastructure-as-a-Dew Service (IaaDS). These new service models inherit some of the advantages and disadvantages of cloud computing, but not necessarily identical ones. We list the key points for comparison in Table 3.

Table 3. Cloud-dew service model.

	SaaDS	**PaaDS**	**IaaDS**
Pros	Cloud computing on SaaS has the advantages of immediate access, no software management issues, and simple scalability.	Cloud computing on PaaS has the advantages of simple management and development convenience.	Cloud computing on IaaS has the advantages of cost control, a pay-as-you-go model, and horizontal scale-out.
	Dew computing makes the software configuration available no matter whether the Internet is connected or not (cross-platform and cross-network).	Dew computing benefits the complicated and advanced function of PaaS, making it available when the internet is down.	Dew computing converts the IaaS into a dual-infrastructure framework (a fail-over feature)
Cons	Cloud computing on SaaS has the disadvantages of a lack of control over the underlying infrastructure and the fact that integration with internal applications is difficult.	Cloud computing on PaaS has the disadvantages of difficulty meeting instant and unpredictable spikes in demand, unique services and configurations, and difficulty changing the PaaS platform.	Cloud computing on PaaS has the disadvantages of complex security controls and dependency on the IaaS provider.
	Dew computing makes software deployment more difficult (cloud and on-premise) and the configuration experiences more frequent conflicts.	Dew computing makes the above problems more serious.	Dew computing requires that the IaaS provider have the ability to construct a dew service.
Construction difficulty	Dew services need a mechanism to add dew versions of software.	Dew services need mechanisms by which to add dew versions of platforms or platform functions.	Dew services need mechanisms to add dew versions of infrastructure or parts of it.
	The difficulty is minor.	The difficulty is significant, as the platform function is built from scratch again.	The difficulty is significant, but depends on the practice of the business domain.

6. Conclusions

In this paper, we proposed an experimental concept of cloud-dew architecture for smart parking systems. Our main concern was regarding cloud services (ex: license plate recognition) and their dependence on Internet connection. When cloud services are unavailable, there is a need to ensure that parking lot businesses can continue to service their customers. In particular, license plate recognition services control cars' entries and exits, and plays the core role in a parking lot system. Regardless of outages or Internet interruption, these exceptions should not destroy or compromise the system necessary to control a parking business. Hence, we propose an innovative form of cloud-dew architecture to tackle this problem. The features of this architecture are LAN-level deployment, Platform-as-a-Dew Service (PaaDS), a dew version of license plate recognition, and a dew type of machine learning model training. The proposed solution was tailor-made to the smart

parking domain, and it allowed the business owner to optimize their parking system, enabling them to maximize business hours while also taking system stability into account.

From an architecture theory perspective, we present a new type of Edge-Cloud-Dew computing architecture which weaves the "Cloud" service (cloud license plate recognition) into the "Dew" architecture (Internet resilience smart parking system) on an "Edge" deployment environment (AIoT license plate recognition). The boundaries among Edge, Cloud, and Dew became more blurred, but on the other hand, service orchestration was executed smoothly and efficiently. For example, our smart parking system architecture was able to effectively release the network constraint on cloud computing and increase the horizontal and vertical scalability of the local system. Meanwhile, we introduced a form of LAN-level dew service architecture which was able to reduce maintenance and deployment issues and increase the dew service's reliability and sustainability.

Similarly, from a business owner's perspective, we propose a training mechanism for dew type machine learning models that does not require periodic retraining and that performs with acceptable accuracy. Business owners of parking lots can reduce their burdens of system management issues when introducing machine learning technology. This feature can draw stakeholders to invest more resources into the innovation of smart parking systems.

However, the limits of the proposed architecture need a more technical and complicated cross-platform implementation to overcome. This also made our experiments on this system more unrealistic. Hence, we introduce an alternative metrics method to evaluate the performance of our proposed architecture. Currently, we have only adopted our cloud-dew architecture design in prototype experiments. Real-world parking lot operations tend to be massive and run 24/7, so implementing our proposed architecture into production mode will require caution and will take some time.

Due to the micro-service property, dew computing often fails to cope with heavy computing processes. Based on the proxy design of the image recognition service, we proposed a dew version of a license plate recognition service, which can act as a reference solution for other domains' dew computing applications with machine learning requirements. There is still a long way to go before practical online realization. In addition, how to make use of the advantages of the Artificial Internet of Things (AIoT) to combine dew computing architecture has yet not been explored, and requires further research in the future.

Funding: This research received no external funding.

Data Availability Statement: No new data were created.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Lin, T.; Rivano, H.; Mouël, F.L. A Survey of Smart Parking Solutions. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 3229–3253. [CrossRef]
2. Pham, T.N.; Tsai, M.; Nguyen, D.B.; Dow, C.; Deng, D. A Cloud-Based Smart-Parking System Based on Internet-of-Things Technologies. *IEEE Access* **2015**, *3*, 1581–1591. [CrossRef]
3. Farooqi, N.; Alshehri, S.; Nollily, S.; Najmi, L.; Alqurashi, G.; Alrashedi, A. UParking: Developing a Smart Parking Management System Using the Internet of Things. In Proceedings of the 2019 Sixth HCT Information Technology Trends (ITT), Ras Al Khaimah, United Arab Emirates, 20–21 November 2019.
4. Wang, Y. Definition and Categorization of Dew Computing. *Open J. Cloud Comput.* **2016**, *3*, 1–7. [CrossRef]
5. Fox, R.; Hao, W. *Internet Infrastructure: Networking, Web Services, and Cloud Computing*; CRC Press: Boca Raton, FL, USA, 2017.
6. Yu, Y.-C. A Dew Computing Architecture for Smart Parking System with Cloud Image Recognition Service. In Proceedings of the 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 12–16 July 2021; pp. 1805–1809. [CrossRef]
7. Manville, M.; Shoup, D. Parking, people, and cities. *J. Urban Plan. Develop.* **2005**, *131*, 233–245. [CrossRef]
8. Naji, B.; Abdelmoula, C.; Masmoudi, M. A Real Time Algorithm for Versatile Mode Parking System and Its Implementation on FPGA Board. *Appl. Sci.* **2022**, *12*, 655. [CrossRef]
9. Mufaqih, M.S.; Kaburuan, E.R.; Wang, G. Applying smart parking system with internet of things (IoT) design. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *725*, 012095. [CrossRef]

10. Shih, S.E.; Tsai, W.H. A Convenient Vision-Based System for Automatic Detection of Parking Spaces in Indoor Parking Lots Using Wide-Angle Cameras. *IEEE Trans. Veh. Technol.* **2014**, *63*, 2521–2532. [CrossRef]
11. Ashqer, M.I.; Bikdash, M. Parking Lot Space Detection Based On Image Processing. In Proceedings of the 2019 SoutheastCon, Huntsville, AL, USA, 11–14 April 2019.
12. Anggawijaya, Y.M.; Weng, T.; Herawati, R. Energy Aware Parking Lot Availability Detection Using YOLO on TX2. In Proceedings of the 2019 3rd International Conference on Informatics and Computational Sciences (ICICoS), Semarang, Indonesia, 29–30 October 2019.
13. Yi, P.; Guangchun, L. Cloud computing, fog computing, and dew computing. *ZTE Commun.* **2019**, *15*, 1–2.
14. Wang, Y. Cloud-dew Architecture. *Int. J. Cloud Comput.* **2015**, *4*, 199–210. [CrossRef]
15. Ray, P.P. An Introduction to Dew Computing: Definition, Concept and Implications. *IEEE Access* **2018**, *6*, 723–737. [CrossRef]
16. Wang, Y.; Leblanc, D. Integrating SaaS and SaaP with Dew Computing. In Proceedings of the 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), Atlanta, GA, USA, 8–10 October 2016; pp. 590–594. [CrossRef]
17. Gushev, M. Dew Computing Architecture for Cyber-Physical Systems and IoT. *Internet Things* **2020**, *11*, 100186. [CrossRef]
18. Loncar, P. Data-Intensive Computing Paradigms for Big Data. In Proceedings of the 29th International DAAAM Symposium, Zadar, Croatia, 24–27 October 2018; pp. 1010–1018.
19. Ghosh, S.; De, D. DewGame: D2D communication enabled dew computing for 5G IoT using coalition formation game. *J. Supercomput.* **2023**, *in press*. [CrossRef]
20. Gusev, M.; Wang, Y. Formal Description of Dew Computing. In Proceedings of the 3rd International Workshop on Dew Computing, Riverton, NJ, USA, 29–31 October 2018; pp. 8–13.
21. Martin, W.; Sarro, F.; Jia, Y.; Zhang, Y.; Harman, M. *A Survey of App Store Analysis for Software Engineering*; University College London Research Note RN/16/02; UCL Department of Computer Science: London, UK, 2016.
22. Wang, Y. An API for Dew Computing Services. In Proceedings of the 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), Madrid, Spain, 12–16 July 2021; pp. 1801–1804. [CrossRef]
23. Mishra, K.; Rajareddy, G.N.; Ghugar, U.; Chhabra, G.S.; Gandomi, A.H. A Collaborative Computation and Offloading for Compute-intensive and Latency-sensitive Dependency-aware Tasks in Dew-enabled Vehicular Fog Computing: A Federated Deep Q-Learning Approach. *IEEE Trans. Netw. Serv. Manag.* **2023**, *in press*. [CrossRef]
24. Lin, T.-T.; Rustia, D.J.A. Trends of AIoT Application in Smart Agriculture. 2019. Available online: https://ap.fftc.org.tw/article/1636 (accessed on 16 June 2023).
25. Leveraging the Upcoming Disruptions from AI and IoT. Available online: https://www.pwc.es/es/publicaciones/digital/pwc-ai-and-iot.pdf (accessed on 16 June 2023).
26. Utomo, P.; Falahah. Dew Computing: Concept and Its Implementation Strategy. In Proceedings of the 2020 Fifth International Conference on Informatics and Computing (ICIC), Gorontalo, Indonesia, 3–4 November 2020; pp. 1–6. [CrossRef]
27. Géron, A. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*; O'Reilly Media: Sebastopol, CA, USA, 2019.
28. Benchmarking TensorFlow and TensorFlow Lite on the Raspberry Pi. Available online: https://www.hackster.io/news/benchmarking-tensorflow-and-tensorflow-lite-on-the-raspberry-pi-43f51b796796 (accessed on 16 June 2023).
29. Coulin, T.; Detante, M.; Mouchère, W.; Petrillo, F. Software Architecture Metrics: A literature review. *arXiv* **2019**, arXiv:1901.09050.