**SMART PARKING**

**Team Member:**

**622621121302:R.Ragavan**

**PHASE 4 SUBMISSION**

**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.
* 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 .
* 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.
* 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 reliabil- ity, 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 dedi- cated 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.

**Table 1.** A comparison of dew computing categories, based on [[5](#_bookmark10),[15](#_bookmark20),[20](#_bookmark25)–[23](#_bookmark28)].

**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 |
| STiD | Storage | PC Storage in dew has a cloud Dropbox  copy |
| DBiD | Database | PC Local database has a cloud  backup |
| SiD | Software | PC Software ownership and Apple App Store  settings have a cloud copy Google play |
| PiD | Platform/Library | PC SDK and projects have a GitHub  cloud copy |
| IaD | Computer/VM/Container | On-premise computer  PC settings and data have a |

Data in forms other than above

DiD

cloud copy Dew computing

PC applications not in above categories

Novell Groupwise 7

SaaDS/SaaDP Software suite/Product PC/VM Local and distributed

software access

Regeneration of cloud

IaaDS Virtualized system PC/VM

platform and relevant services

A proxy of cloud AI services on LAN

MLiD/DaaDS

Vehicular Fog

A lite cloud AI service on PC/VM/LAN server LAN without Internet

connection

Performs computations, communications, offloading,

Smart parking system

MMiD/MaaDS

Computing (VFC)

networksand resource utilization, considering the latency and energy consumption

**AIOT:**

Vehicular computing

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](#_bookmark29)]. Deep learning-based techniques for object segmentation, detection, and classification are used to analyze pest populations.

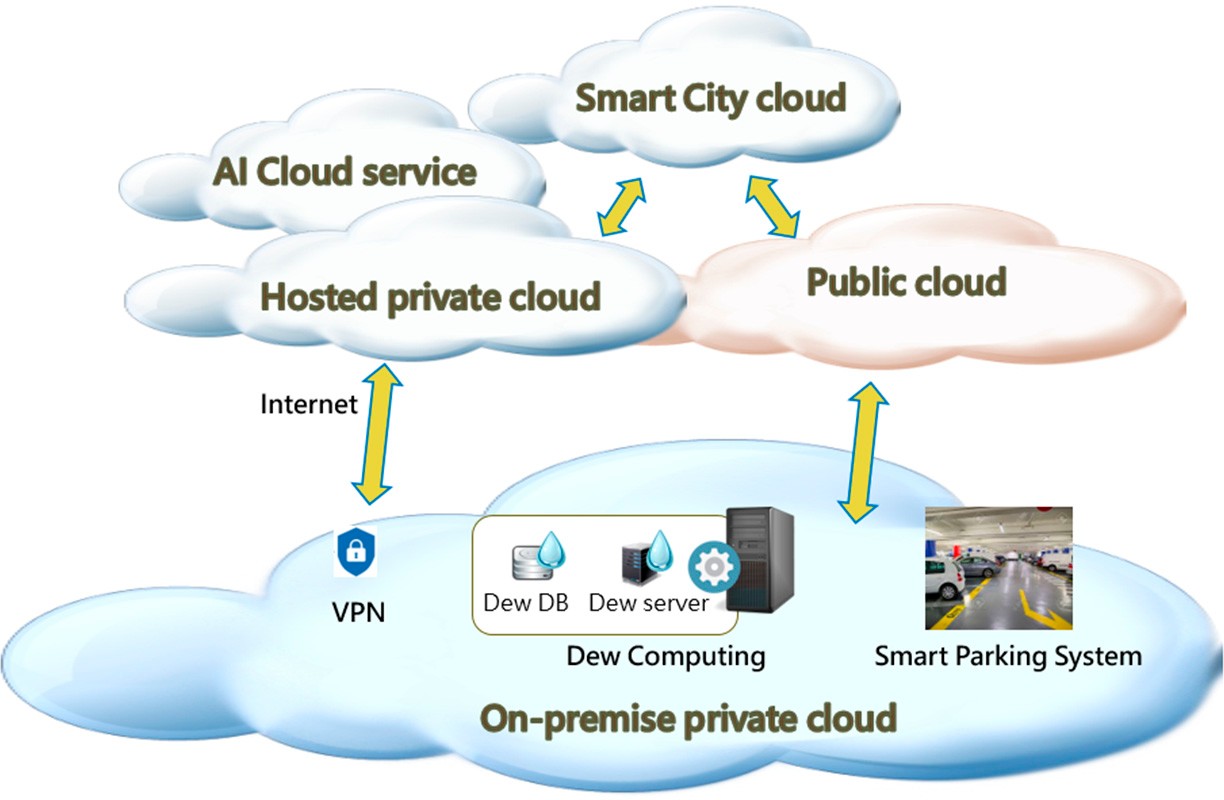
In our problem domain, the primary goal was to make the parking system smarter and more efficient through the additive effects of AIot and dew computing. Dew computing can help an AIoT device to remain available without an Internet connection. It not only detaches applications from the cloud, but also efficiently collaborates with cloud services [[25](#_bookmark30)].

# 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](#_bookmark21),[20](#_bookmark25)].

* 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](#_bookmark1) 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.

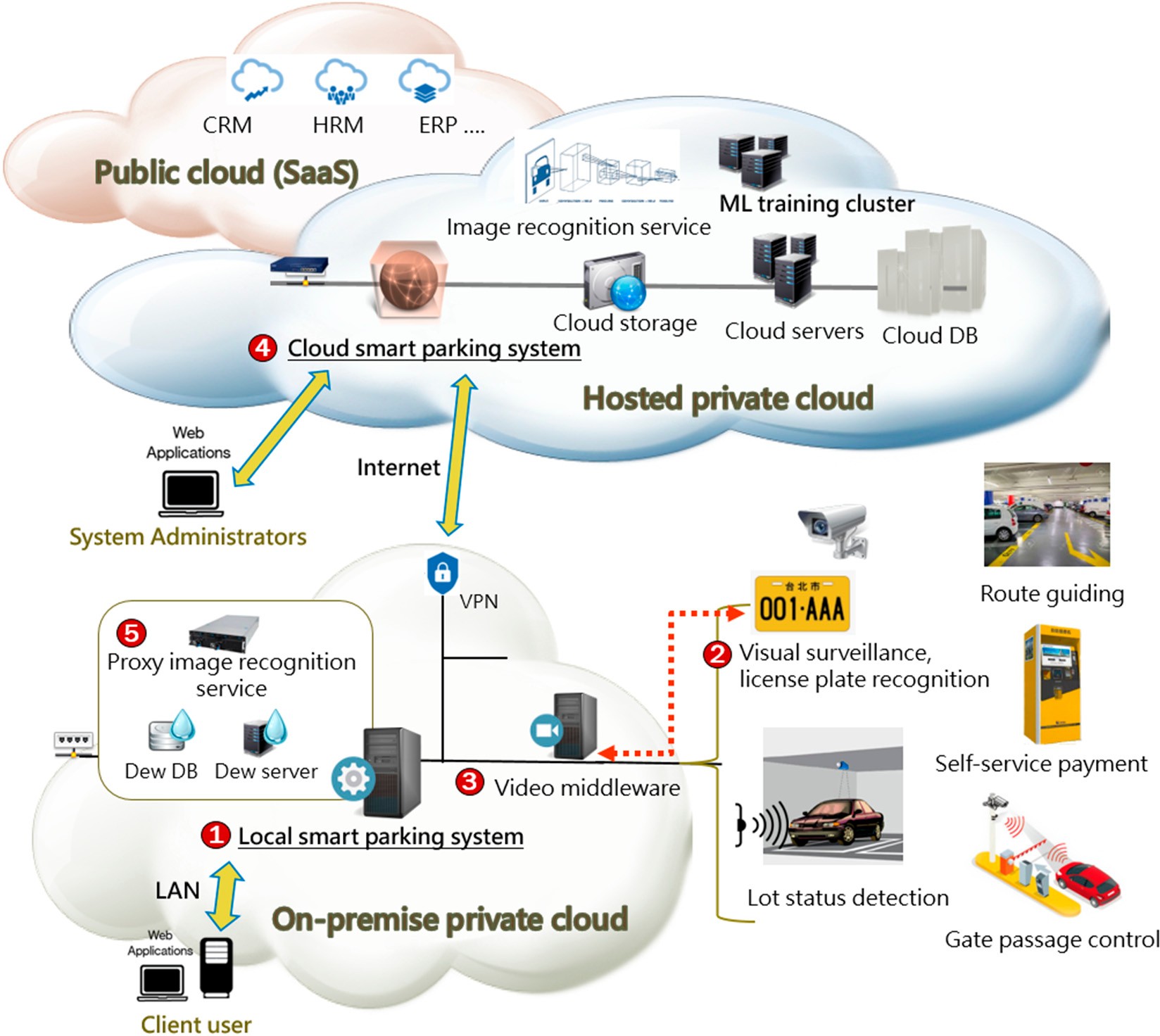


1. On-premise private cloud: Private clouds comprise two primary types: externally hosted cloud solutions and internally locally hosted (or on-premise hardware) so- lutions. The main differences are the overall cost, level of support required, and feature scaling. In this case, we adopt both types of 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.

1. 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.
2. 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 ad- vanced 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.
3. 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](#_bookmark2) as follows.



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 commu- nicating 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 recog- nition, 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, includ- ing 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 park- ing 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 admin- istrators. 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 unavail- able. 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.
   1. *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](#_bookmark3). 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:

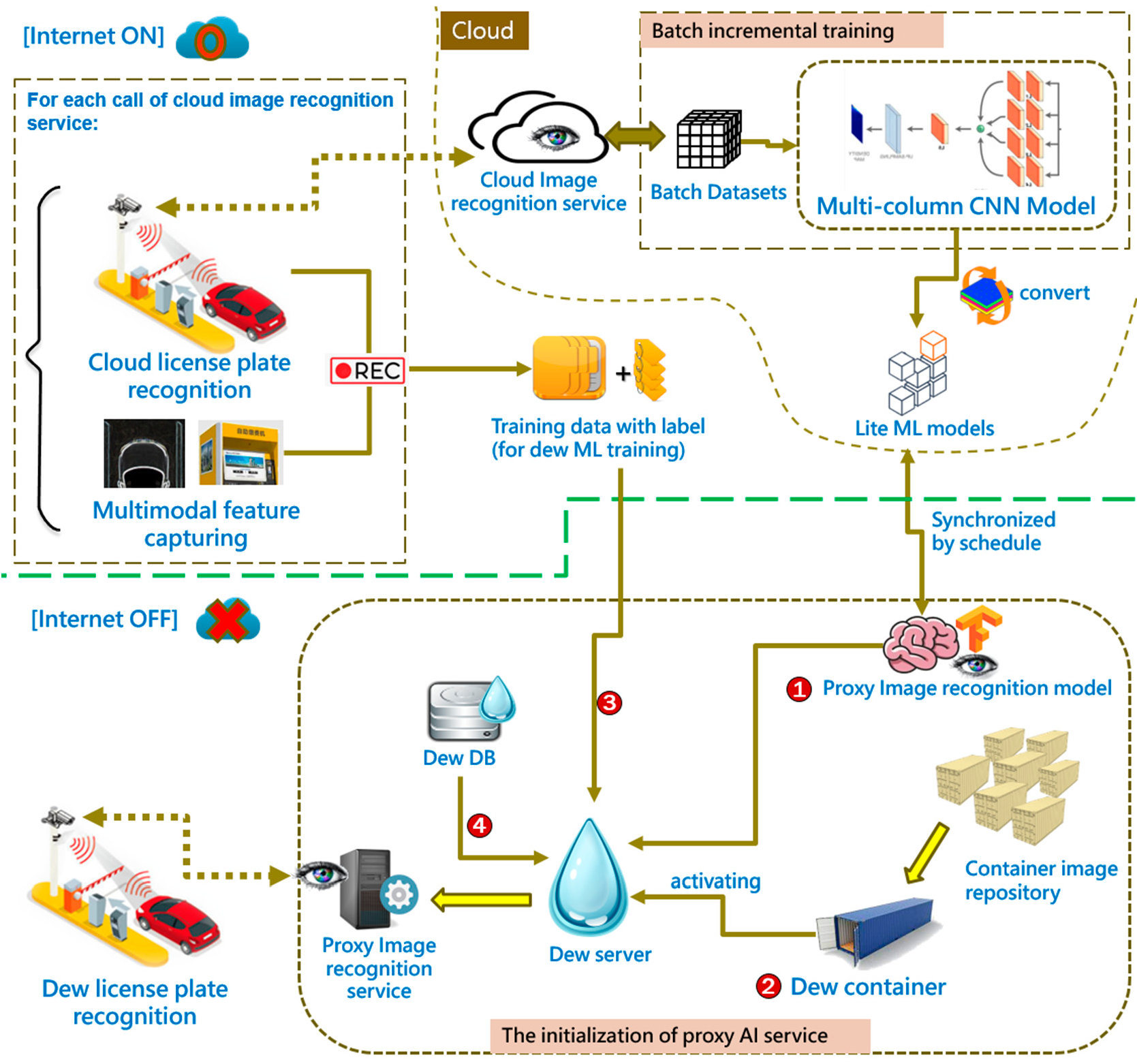
1. Proxy image recognition model: A lite version of the model is downloaded from the cloud. Since the model is always a copy of the latest cloud version, we do not need to spend effort on maintenance or adjustment. However, it may not fit with the dynamic changes in the parking lot environment. We can 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.

1. 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.
2. Dew ML training: We use the enhanced training data to train the proxy image recog- nition model. The whole process is intended to fulfill the function of the dew service; thus, we call it dew ML training.
3. 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.



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](#_bookmark31)], “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 applica- tion. 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 assump- tions are mistaken, cross-entropy will be greater to some extent— this extent is called the Kullback–Leibler (KL) divergence [[27](#_bookmark32)].

Kullback–Leibler divergence measures the difference between two probability dis- tributions 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](#_bookmark33)], 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](#_bookmark4).

**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 | X1 |
| Dew arch. (lite model) | 0.806 | 0.194-x | X2 |

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

*DKL*(*dew*||*cloud*) = −*Ex*

∼*dew*[ln *cloud*(*x*) − ln *dew*(*x*)] = −*Ex*

*cloud*(*x*)

∼*dew*[ln *dew*(*x*) ] (1)

*KLD cloud*, *dew*

log *σ*2

*σ*2 + (*µ*1 − *µ*2)2 1

2

(2)

( ) =

*σ*1 + 2*σ*2 − 2

Under perfect conditions, assuming that the design bias is zero, we can calculate the Kullback–Leibler (KL) divergence based on Table [2](#_bookmark4). 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.

1

# Discussion

* 1. *Architecture Quality*

Petrillo et al. [[29](#_bookmark34)] conducted a systematic literature review of software architecture metrics and revealed a list of the common quality features of software architecture, includ- ing 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](#_bookmark34)] 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 under- standable. 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.

•

* 1. *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](#_bookmark5).

**Table 3.** Cloud-dew service model.

**SaaDS PaaDS IaaDS**

Pros

Cons

Construction difficulty

Cloud computing on SaaS has the advantages of immediate access, no software management issues, and simple scalability.

Dew computing makes the software configuration available no matter whether the Internet is connected or not (cross-platform and cross-network).

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.

Dew computing makes software deployment more difficult (cloud and

on-premise) and the configuration experiences more frequent conflicts.

Dew services need a mechanism to add dew versions of software.

The difficulty is minor.

Cloud computing on PaaS has the advantages of simple management and development convenience.

Dew computing benefits the complicated and advanced function of PaaS, making it available when the internet is down.

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.

Dew computing makes the above problems more serious.

Dew services need mechanisms by which to add dew versions of platforms or platform functions.

The difficulty is significant, as the platform function is built from scratch again.

Cloud computing on IaaS has the advantages of cost control, a pay-as-you-go model, and horizontal scale-out.

Dew computing converts the IaaS into a dual-infrastructure framework (a fail-over feature)

Cloud computing on PaaS has the disadvantages of complex security controls and dependency on the IaaS provider.

Dew computing requires that the IaaS provider have the ability to construct a dew service.

Dew services need mechanisms to add dew versions of infrastructure or parts of it.

The difficulty is significant, but depends on the practice of the business domain.

# 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 recogni- tion) 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 compli- cated 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 com- puting 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.