


RESEARCH

Open Access



An edge-cloud collaborative computing platform for building AIoT applications efficiently

Guoping Rong^{1,2*} , Yangchen Xu¹, Xinxin Tong³ and Haojun Fan³

Abstract

The convergence of Artificial Intelligence (AI) and the Internet of Things (IoT), or AIoT, has breathed a new life into IoT operations and human-machine interactions. Currently, resource-constrained IoT devices usually cannot provide sufficient capability for data storage and processing so as to support building modern AI models. An intuitive solution is to integrate cloud computing technology into AIoT and exploit the powerful and elastic computing as well as the storage capacity of the servers on the cloud end. Nevertheless, the network bandwidth and communication latency increasingly become serious bottlenecks. The emerging edge computing can complement the cloud-based AIoT in terms of communication latency, and hence attracts more and more attention from the AIoT area. In this paper, we present an industrial edge-cloud collaborative computing platform, namely *Sophon Edge*, that helps to build and deploy AIoT applications efficiently. As an enterprise-level solution for the AIoT computing paradigm, *Sophon Edge* adopts a pipeline-based computing model for streaming data from IoT devices. Besides, this platform supports an iterative way for model evolution and updating so as to enable the AIoT applications agile and data-driven. Through a real-world example, we demonstrate the effectiveness and efficiency of building an AIoT application based on the *Sophon Edge* platform.

Keywords: AIoT platform, Edge-cloud collaboration, Pipeline

Introduction

Recently, the world has witnessed the arrival of the era of big data, along with two dominating technology trends, i.e., Artificial Intelligence (AI) and the Internet of Things (IoT). While the IoT creates an interconnected system of machines, AI gives machines the capability of simulating human intelligence. Obviously, AI and the IoT can complement each other, so as to enable a further promising technology, i.e., Artificial Intelligence of Things (AIoT). In general, AIoT aims to achieve more efficient IoT operations, improve human-machine interactions and enhance the capability for data management and analytics.

Thanks to recent advances in both hardware (e.g. smart sensors, actuators, and low-power consuming chips [1]) and software (e.g. embedded operating systems, virtualization technology, machine learning frameworks), AIoT is becoming into reality. Among many applicable areas, *Smart Home* [2], *Smart Factory* [3], and *Smart City* [4] are being at the forefront of AIoT adoption.

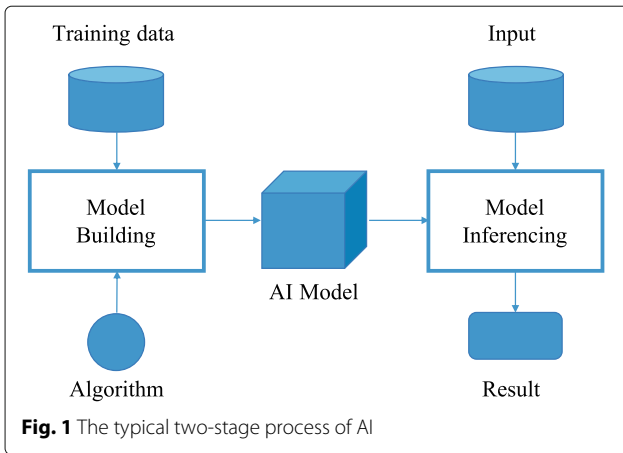
As an AI-enabled technology, AIoT also shares the typical two-stage process [5] shown in Fig. 1. In the first stage, AI models are built using various machine learning algorithms with training data. In the context of AIoT, the training data are commonly collected from various IoT devices. Then in the second stage, models are used to make inferences from certain input. Generally, the two stages can be called “Model Building” and “Model Inference”, respectively. A major challenge in AIoT is that the “Model Building” stage requires a significant amount of data as well as processing capability to obtain the best AI

*Correspondence: ronggp@nju.edu.cn

¹The Joint Laboratory of Nanjing University and Transwarp on Data Technology, Nanjing University, Nanjing, China

²The State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China

Full list of author information is available at the end of the article



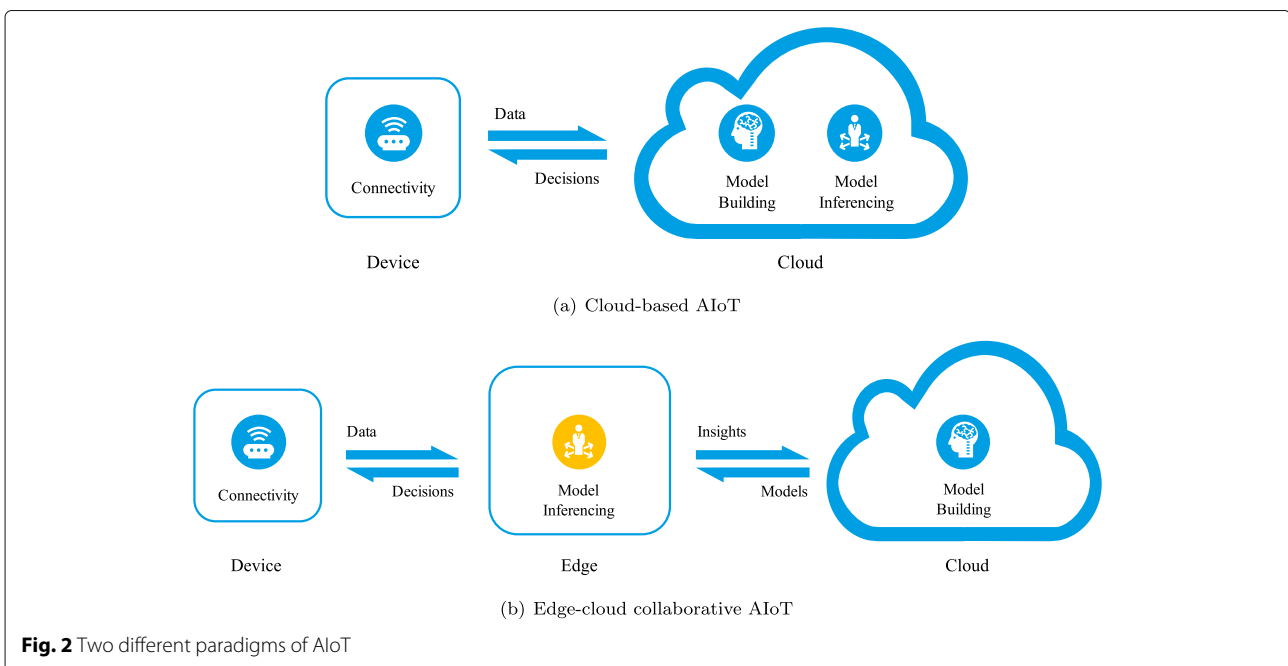
models [5], while most of the IoT devices cannot provide sufficient resources to support such computing or storage capability due to various constraints [6].

As depicted in Fig. 2a, a cloud-based paradigm seems to be an ideal solution. Both “Model Building” and “Model Inferencing” can be performed using the servers on the cloud end with powerful and elastic computing and storage resources, while the IoT devices only need to upload data and receive decisions from the cloud end. However, with the ever-growing amount of data produced by massive IoT devices, this cloud-based paradigm may encounter a great challenge caused by the existing network [7]. In other words, the network bandwidth and communication latency between the devices and the distant cloud may become serious bottlenecks for highly responsive AIoT applications [6]. To gain low latency,

modern AIoT starts embracing edge computing [8–13], an emerging paradigm that places computing and storage resources on the edge of the network, which is away from the servers on the cloud end, allowing a near real-time and localized decision making. Consequently, an edge-cloud collaborative paradigm of AIoT is formed, as depicted in Fig. 2b. In such a paradigm, devices are connected to the edge servers nearby rather than distant cloud servers. While “Model Building” is still performed in the cloud, “Model Inferencing” is now offloaded to the edge to achieve low-latency responses. Once AI models have been built in the cloud, they are delivered to the edge. The edge end then uses the models to make inferences from the data received from the IoT devices and returns back the decisions to the IoT devices. Meanwhile, it also sends significant insights to the cloud. Note that the edge still needs to send a large amount of device data to the cloud for “Model Building”, but in real-world applications, the frequency required for building models is usually much lower than that of using models to make inferences. Hence, the edge-cloud collaboration finally is able to achieve a better performance overall.

In this paper, we present an edge-cloud collaborative computing platform, named Sophon Edge, that facilitates building AIoT applications. The platform further addresses the following challenges faced by developers in practice:

- **Heterogeneity.** The inherent heterogeneity of the devices in a large-scale IoT system makes the connectivity and coordination process very difficult



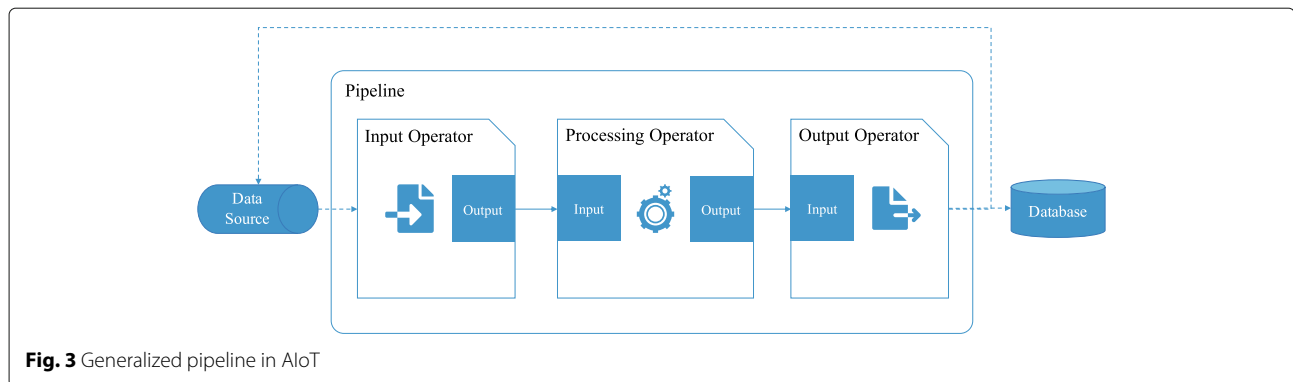


Fig. 3 Generalized pipeline in AIoT

[14]. Moreover, due to various protocols, the generated data usually have different formats, sizes, and timestamps, which form a challenging task for further processing, transmission, and storage [15]. Meanwhile, performing general computation on servers with heterogeneous operating systems or runtime also forms a non-ignorable aspect of the challenge.

- **Accuracy.** Algorithms used by AI need to be well developed and tuned to understand and interpret data so that more accurate decisions can be made [16]. Additionally, due to the highly dynamic nature of the physical world, a once trained AI model may not perform well all the time, i.e., the models need to be refined using the new-coming IoT data to achieve better performance.

Table 1 Pre-implemented operators for structured data (partial.)

Category	Name	Usage
Input	Time Series Database	The “Time Series Database” operator is used to observe certain data tables in a specified time series database (such as InfluxDB). When new data is inserted into the table, the operator will be able to obtain the data immediately and output it in a specific format to subsequent operators.
	Device Input	The “Device Input” operator provides real-time data access to a specified connected device on the Edge Node. The product type of the device should be specified in the operator, which determines the format of the data.
Processing	Filter	The “Filter” operator is used to filter the input data stream. The operator sets a conditional expression. Only when the expression is true, the corresponding data will be forwarded to subsequent operators, otherwise, it will be discarded.
	Sample	The “Sample” operator samples the input data stream according to a specific step size as well as a time interval. The operator can reduce the processing frequency of real-time data and thus optimize resource usage.
	Time Window	The “Time Window” operator provides the function of batch forwarding the data arriving in a specified period.
	Change Trigger	The “Change Trigger” operator monitors its input and only generates outputs when data changes occur. This operator is suitable for data streams that have frequent data duplication in a period.
	Aggregate	The “Aggregate” operator provides a variety of methods for aggregate computation on a data stream, such as sum, count, mean, mode, etc., which can complete some common statistical tasks.
	Custom Function	The “Custom Function” operator provides the ability to calculate data streams with user-defined functions. The operator supports multiple custom computational expressions for each piece of data flowing through the operator and merges the results of the computation into the final output data stream.
Output	Apply Model	The “Apply Model” operator calls a certain AI model to make inferences upon its input, and outputs the inferencing results. Note that the model to be called should have been already deployed on the Edge Node.
	Device Output	The “Device Output” operator provides the function of controlling devices, including setting the properties and calling the services of certain devices.
	Notification	The “Notification” operator provides a logging-like function that persists input data as messages on the Edge Nodes. These messages can be set to different levels, including INFO, WARN, and ERROR. The notification messages can be used as inputs for some operators.

Table 2 Pre-implemented operators for multimedia data (partial.)

Category	Name	Usage
Input	File Input	The “File Input” operator reads a video file from a specific device as the data source.
	Real-time Stream Input	The “Real-time Stream Input” operator binds a specific device using the RTSP or RTMP protocol and outputs the real-time data stream to subsequent operators.
	USB Camera Input	The “USB Camera Input” operator reads and outputs data from a specific USB camera.
Processing	Encode	The “Encode” operator encodes the original video stream into a specific format, e.g. H.264, H.265, JPEG.
	Decode	The “Decode” operator decodes the encoded video or JPEG images to obtain raw video information.
	Mux	The “Mux” operator encapsulates the encoded video into a specific format, e.g. FLV, MP4.
	Demux	The “Demux” operator parses the input data into encoded video.
	Resolution	The “Resolution” operator transforms the resolution of the input video stream into a specific setting, e.g. 1280*720.
	Frame Rate	The “Frame Rate” operator limits the maximum number of frames per second of the input video stream.
	Region Of Interest	The “Region Of Interest” operator specifies a region in the input image or video stream, so that subsequent “Apply Model” operators will only make inferences towards the specified region in the images. This avoids unnecessary processing and saves resources.
	Apply Model	The “Apply Model” operator binds a specific AI model, and the input video stream is filtered somehow to get the model input. Moreover, the results returned by the model are saved in the output video stream, such as drawing recognition frames.
Output	File Output	The “File Output” operator saves the input video stream as a local file.
	Alert	The “Alert” operator treats its inputs as alerts and sends them to the cloud.

Our contributions can be summarized as the following:

- We present the computing model and the architecture of *Sophon Edge*, an edge-cloud collaborative AIoT platform.
- We present the pre-implemented operators supported in *Sophon Edge*, which saves development and operation efforts for the user of the platform, i.e., the developers of various AIoT applications.
- We present the process of model evolution and updating using the *Sophon Edge* platform, which helps to build agile and data-driven AIoT applications [17].
- We implement the *Sophon Edge* platform and demonstrate its usability through a real-world use case.

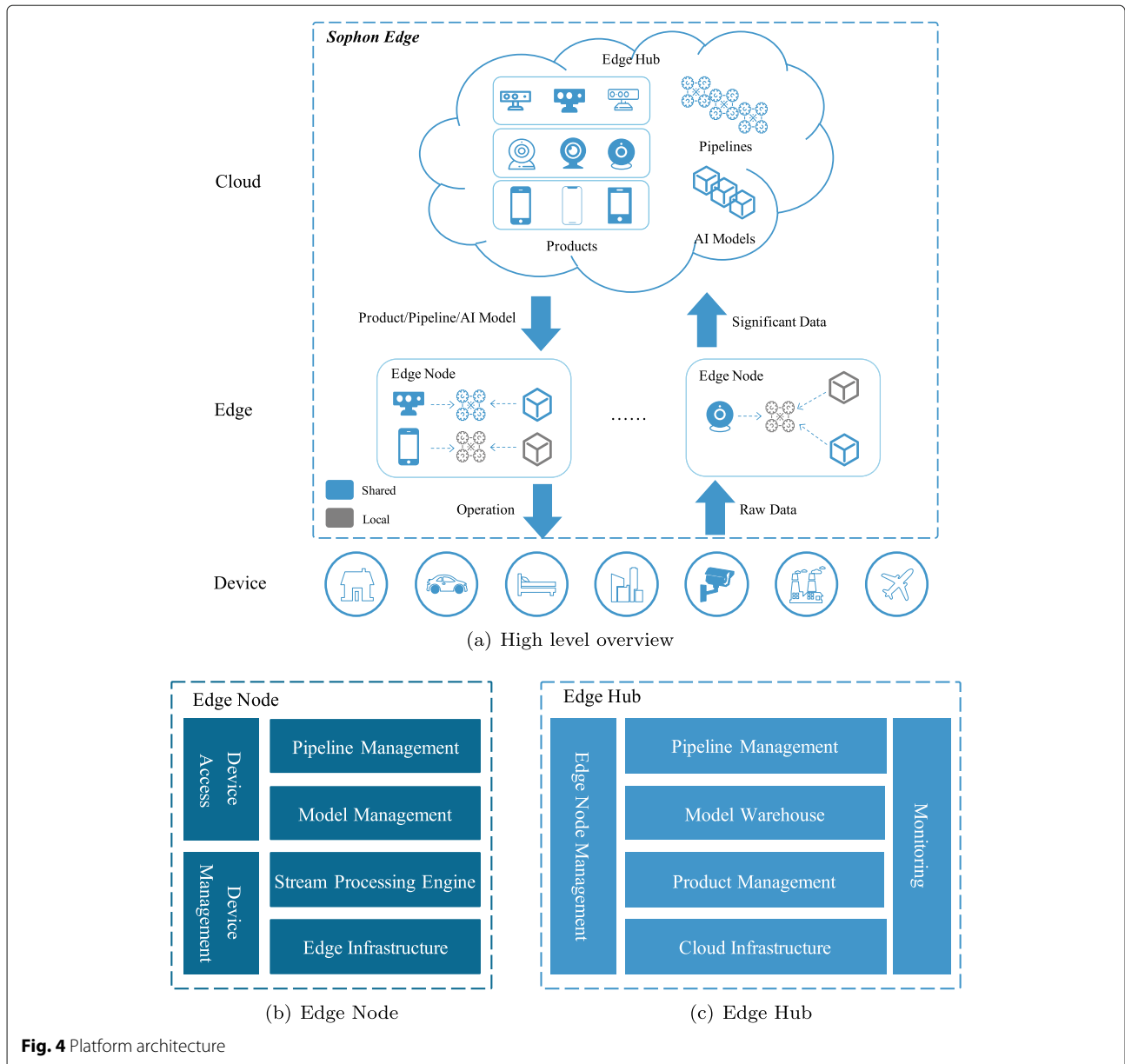
The rest of this paper is organized as follows: in “[Related work](#)” section, we present the related work. The design of *Sophon Edge* platform is elaborated in “[The sophon edge platform](#)” section, while some in-depth discussions are presented in “[An application example](#)” section. In “[Discussion](#)” section, we introduce a real-world use case and demonstrates the usability of the platform. Finally, we draw the conclusion of this work and discuss the future work in “[Conclusion](#)” section.

Related work

A lot of research efforts have been made towards the convergence of edge computing, AI, and the IoT. In this section, we present these work briefly.

Edge computing for IoT

Many comprehensive surveys on edge computing for IoT have been carried out in recent years. Yu et al. [7] discussed the potential ability for integrating IoT and edge computing as edge computing-based IoT, and illustrated the advantages and disadvantages of edge computing assisted IoT in transmission, storage, and computation. Pan et al. [18] presented discussions on the state-of-the-art efforts, key enabling technologies, research topics, and typical IoT applications benefiting from edge computing. PremSankar et al. [6] described key IoT application scenarios that benefit from edge computing, and carried out an experimental evaluation to demonstrate the necessity of edge computing to achieve a satisfactory quality of experience. Ai et al. [19] analyzed three different edge computing technologies for IoT, i.e. mobile edge computing (MEC), cloudlets, and fog computing. Ray et al. [1] presented a taxonomic classification of industrial edge-IoT computing. The authors also proposed a novel e-healthcare architec-



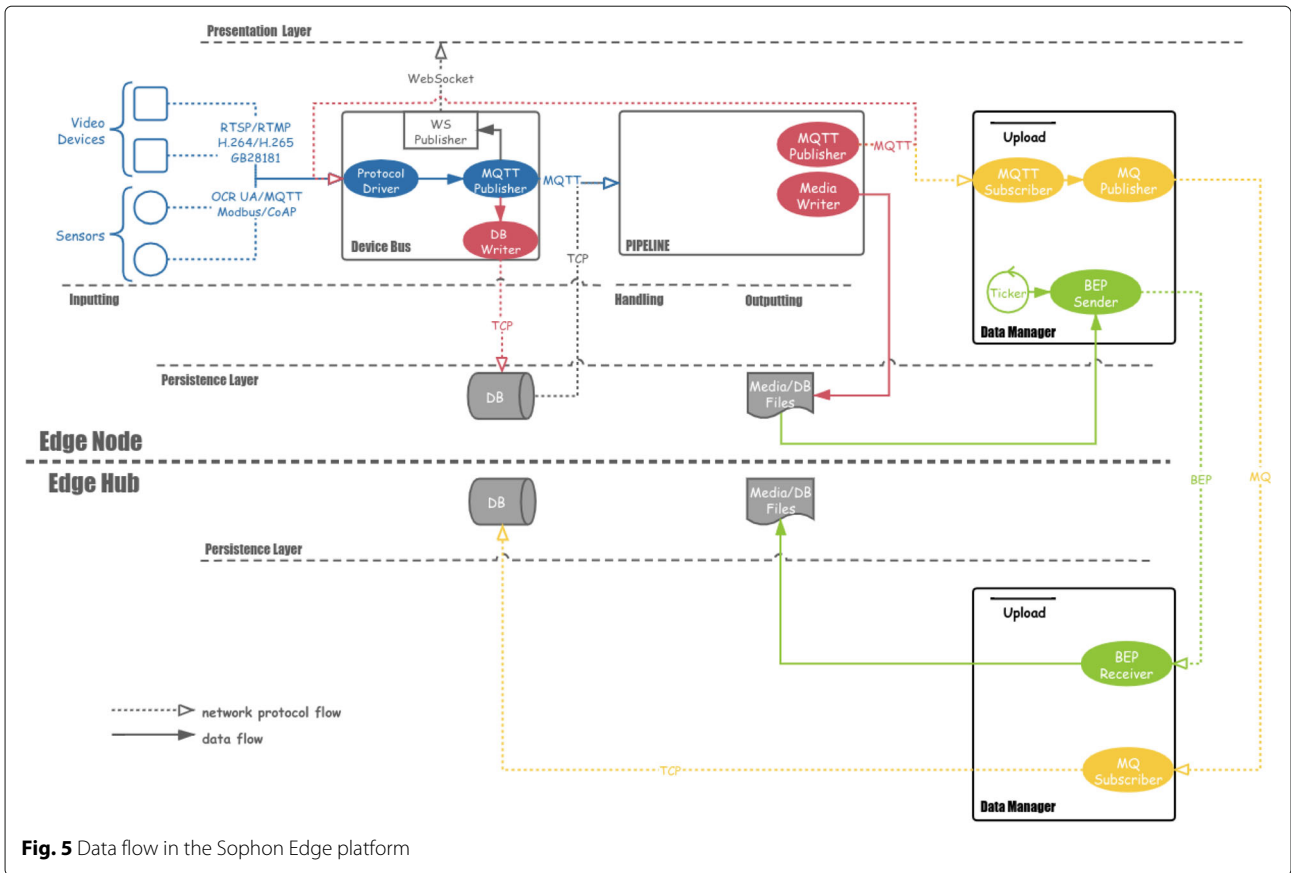
ture that relies upon industrial elements of the edge-IoT ecosystem.

Giang et al. [20] shared their experiences in building an edge computing platform for IoT applications. Notably, they used a distributed data flow programming model which is proved to map well to edge computing. Lertsinrubtavee et al. [21] also introduced a lightweight edge computing platform, addressing the computational needs for local applications like smart homes and applications with stringent Quality of Service (QoS) requirements.

Morabito et al. [22–24] conducted a series of studies inspecting the usage of lightweight virtualization (LV) technologies (i.e. containers, and unikernels) in edge computing for IoT. Due to the heterogeneous nature of the

edge devices and servers, it's crucial to provide simple but efficient configuration and instantiation methods, where the virtualization makes sense.

Typically, computation on the resource-constrained devices is offloaded to both the cloud and the edge. The offloading to the cloud allows complicated processing, while the offloading to the edge ensures lower latency and battery consumption. Hence the offloading should be performed in an appropriate manner to optimize the execution time and energy consumption. Chen et al. [25] proposed a framework that supports mobile applications with the context-aware computation offloading capability. Xu et al. [26] proposed a computation offloading method for IoT-enabled cloud-edge computing. Chen et al. [27]

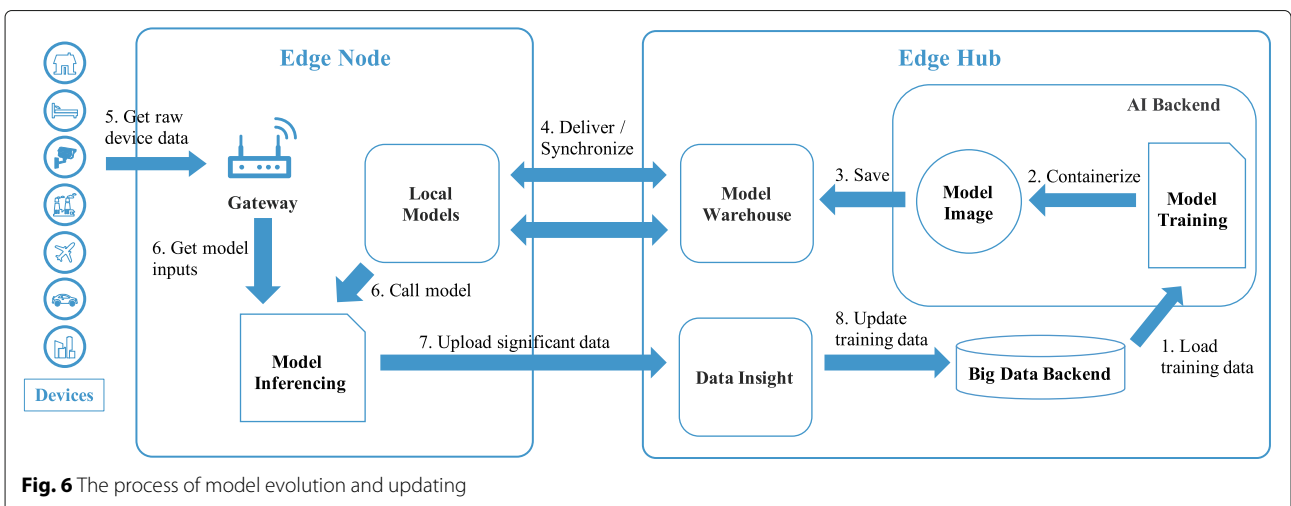


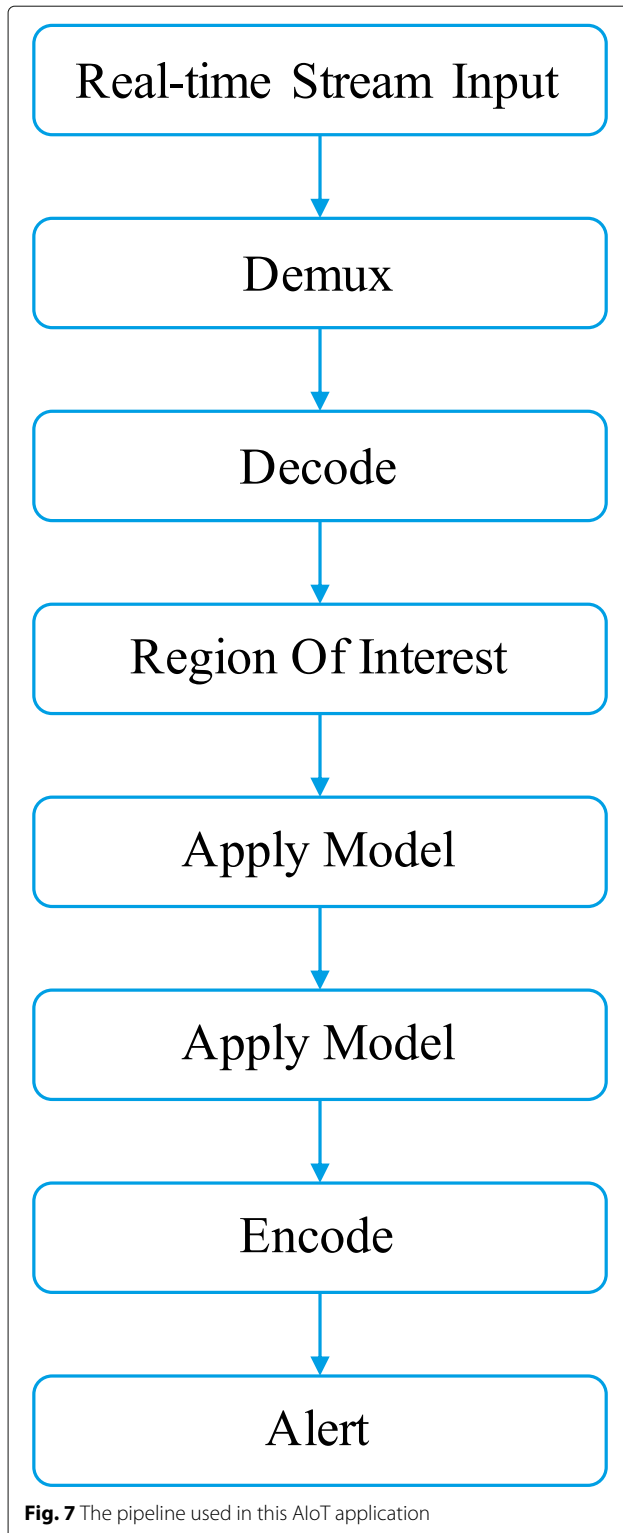
proposed a dynamic computation offloading algorithm (DCOA), which decomposes the optimization problem into a series of subproblems, and solves these subproblems concurrently in an online and distributed way. Shen et al. [28] used multiple deep reinforcement learning (DRL) agents to guide the deployed on IoT devices to

guide their own offloading decisions. Besides, federated learning is utilized to train the DRL agents.

Applying AI to IoT

Calo et al. [5] first illustrated that AI is being required by IoT applications, followed by a discussion of different





approaches for applying AI to IoT. They further proposed an architecture that preserves the advantages of both edge processing and server-based/cloud-centric computing for AI algorithms. Debauche et al. [29] proposed a new archi-

tecture used to deploy at edge level microservices and adapted artificial intelligence algorithms and models. Li et al. [30] introduced deep learning for IoT into the edge computing environment. Furthermore, the authors proposed an offloading strategy to optimize the performance of IoT deep learning applications with edge computing. Xiong et al. [31] discussed some design challenges for building a scalable and shared multi-tenant AIoT platform through two edge computing use cases. Chiu et al. [32] leveraged federated learning technologies to tackle the network bandwidth limitations and data privacy concerns in an AIoT platform.

The sophon edge platform

In this section, we first present the computing model adopted in the Sophon Edge platform. Then we elaborate on the platform's architecture. Finally, we illustrate how the platform supports the continuous evolution and updating of the AI models, i.e. the agile AIoT.

Unified device model

In consideration of the wide variety of existing IoT devices, the *Sophon Edge* platform introduces a unified device model, called *product*. A *product* is an "abstract class" of IoT devices, which declares certain properties or behaviors. Properties refer to the data generated on sensors, which are usually observations of their surroundings (e.g. temperature, humidity). Behaviors are the available actions of actuators. The *product* can also declare subscribable events that indicate some significant changes of a certain IoT device. Theoretically, the *product* is able to model any type of IoT device. Note that the *Sophon Edge* platform does not specify any "official" definition of *products*, which means *products* can be defined by users without any constraint. Nevertheless, devices that share the same or similar functionalities are usually defined as the same *product* in practice. For example, we can define a *product* named "webcam" to cover various types of cameras that can connect to the Internet.

With *Sophon Edge*, a user can first define several *products* and then bind each device to one or more *products*. Specifically, the properties, behaviors, and events defined in the *product* are bound to the corresponding data access or operation interfaces provided by the IoT device.

Pipelined computation task

Every computation task in the *Sophon Edge* platform is described as a *pipeline*, which consists of a series of data operation steps. Each operation step of the pipeline is called an *operator*. An operator is actually a function that implements certain business logic, along with the declaration of its input and/or output. In *Sophon Edge*, pipelines are built through assembling operators. Figure 3 shows a generalized pipeline in common AIoT applications. The

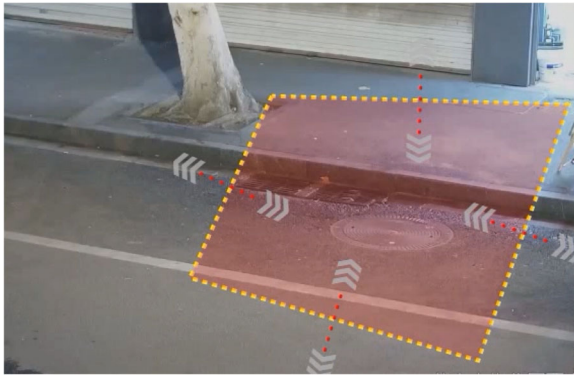


Fig. 8 Illustration of the “Region Of Interest” operator

pipeline starts with an “Input” operator that accesses data on a certain device. Then a “Processing” operator is used to process the data, e.g., using AI models to make inferences. Finally, an “Output” operator handles the results. Particularly, this “Output” operator may store the significant data, or insights, to a certain database, or operates certain devices according to the decisions made.

To help AIoT application developers to build their own pipelines efficiently, the *Sophon Edge* platform provides a programming framework that integrates many pre-implemented operators, i.e. general-purpose utility functions. Tables 1 and 2 list the major pre-implemented operators for structured and multimedia(unstructured) data, respectively. These operators can save lots of developers’ coding efforts and help them focus more on the construction of the overall computing procedure.

By adopting pipelines, not only the programming of complicated computation tasks can be simplified, but the parallelism of the computation can also be improved. Moreover, as the pipeline is actually an abstraction, multiple pipeline instances can be deployed through dynamic binding with different contexts (e.g. devices, AI models, databases), which is rather flexible. Remind that IoT

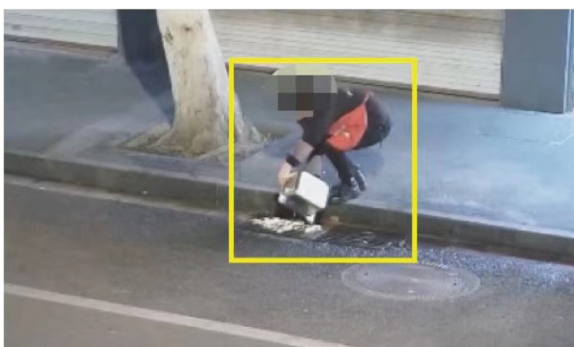


Fig. 9 A detected sewage dumping behavior

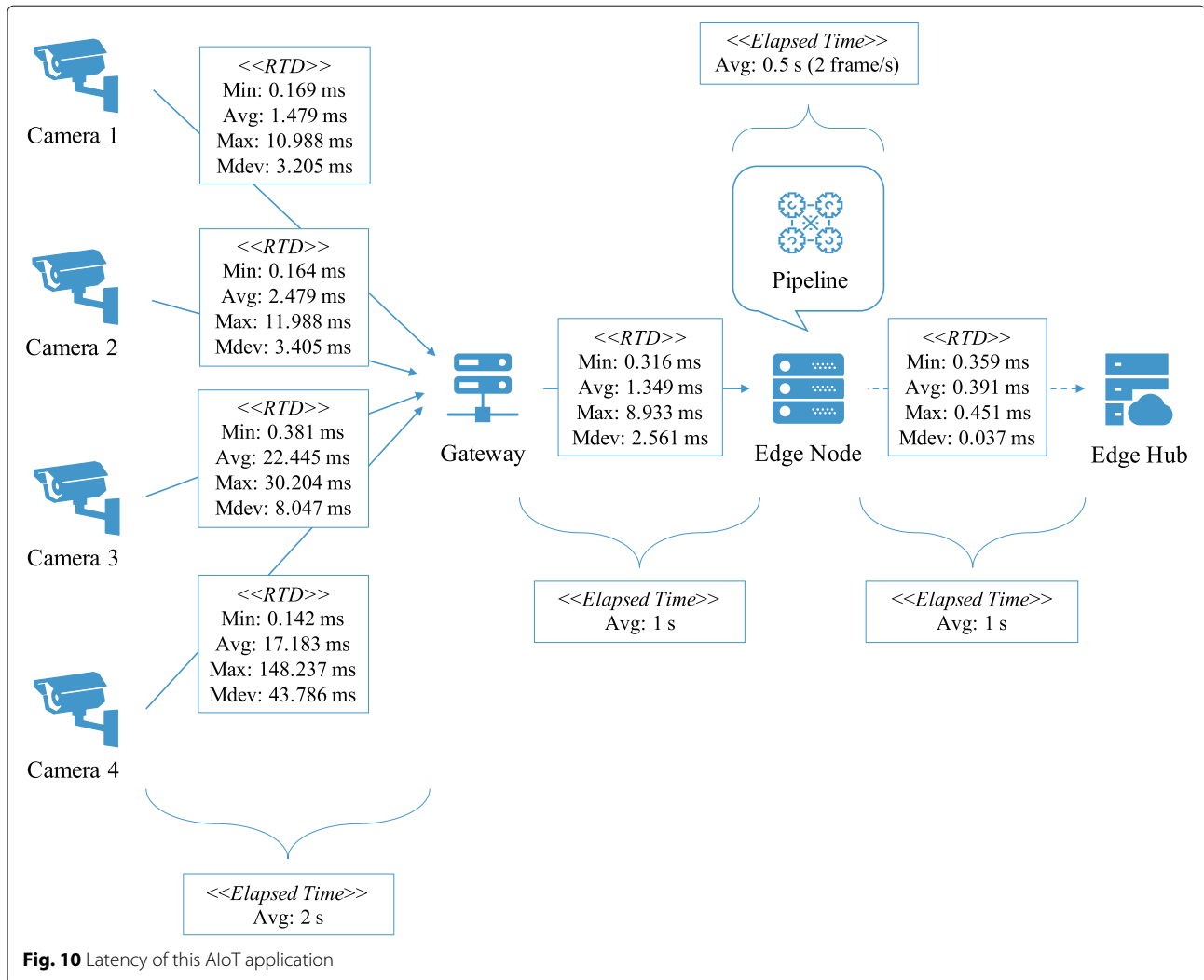
devices are abstracted into *products* on *Sophon Edge* platform. Such abstraction separates concrete devices from a specific pipeline, allowing a single pipeline to be deployed multiple times to access different devices under the same *product*, with the only effort of some simple configurations. Furthermore, if the input device needs to be replaced or upgraded, the pipeline can keep running after the device change as long as the new device belongs to the same *product* defined on the *Sophon Edge* platform.

Edge-cloud collaborative architecture

Figure 4a shows a high level overview of the architecture of the *Sophon Edge* platform. As can be seen, the platform consists of two major parts, i.e. Edge Nodes and Edge Hubs.

Edge Nodes are computing servers that are deployed at the network edge. The Edge Node is mainly responsible for managing the connection and operation on IoT devices, e.g., performing edge computing, forwarding significant data to the cloud, etc. The internal architecture of the Edge Node is portrayed in Fig. 4b. To facilitate device access, the platform supports a variety of communication protocols, including Modbus, MQTT, COAP, AMQP, ONVIF, RTSP, RTMP, UVC, OPCUA, HLS, GB28281, etc. which covers the current mainstream devices in the IoT field. Multiple drivers, brokers, and clients for these protocols are implemented in the *Sophon Edge* platform. The raw data of IoT devices are gathered and transformed into a data stream which facilitates subsequent data processing through the exploitation of a multimedia gateway or a message bus. The multimedia gateway collects real-time stream media data (e.g. images, videos) from multimedia devices, while the message bus collects structured data from ordinary IoT devices. In terms of computing, the Edge Node manages a certain number of pipelines that are either received from the cloud or created locally. The instances of the pipelines are executed by the stream processing engine. The Edge Node also manages several AI models received from the cloud, and these models can be used to make inferences in the deployed pipeline instances. The edge infrastructure mainly supports local data storage and transportation to the cloud end.

Edge Hubs are indeed cloud servers, which manage the Edge Nodes and perform computing-intensive tasks such as model training. Likewise, Fig. 4c shows the internal architecture of the Edge Hub. The Edge Hub also manages pipelines, which are called *shared* pipelines. As the scale of an AIoT application increases, the number of Edge Nodes could increase, making it possible for the same computation task to run on multiple Edge Nodes. It would be costly if we had to create at least one pipeline on each Edge Node. To address this issue, these shared pipelines are managed



on the Edge Hub and delivered to the Edge Nodes later on. The model warehouse stores all the trained models as *container images* that can be quickly deployed on the Edge Nodes. The aforementioned *products* are also managed on the Edge Hub since the definition of *products* are usually able to be shared by all the Edge Nodes in a single AIoT application. The cloud infrastructure takes the responsibility for computing, as it usually integrates a big data backend (e.g. Spark) for data processing as well as an AI backend (e.g. Tensorflow) for model training. In addition, a monitoring mechanism is also implemented to monitor the overall state of the AIoT application.

Figure 5 describes the data flow in the *Sophon Edge* platform. The original device data is sent to the Device Bus on the Edge Node and converted into MQTT messages. These MQTT messages are not only sent as input to the pipeline deployed on the Edge Node, but also being stored in the local database or file system in real-time. If nec-

essary, these MQTT messages will also be displayed to the user through WebSocket. Pipelines deployed on the Edge Node will process the MQTT messages and output the results as new MQTT messages. The output will be pushed to the message queue of the Edge Hub through an MQ publisher, and further stored on the cloud database. Besides, the non-real-time data (mainly media files, e.g. pictures, videos) will also be sent to the Edge Hub.

Agile AIoT

Since device data is being generated continuously in most AIoT applications, the data set used for model training thus should be frequently updated. Besides, as business requirements change, the performance of the model may also need to be improved, and even new models are required. Therefore, AIoT applications require continuous model evolution and updating, i.e. the agile AIoT.

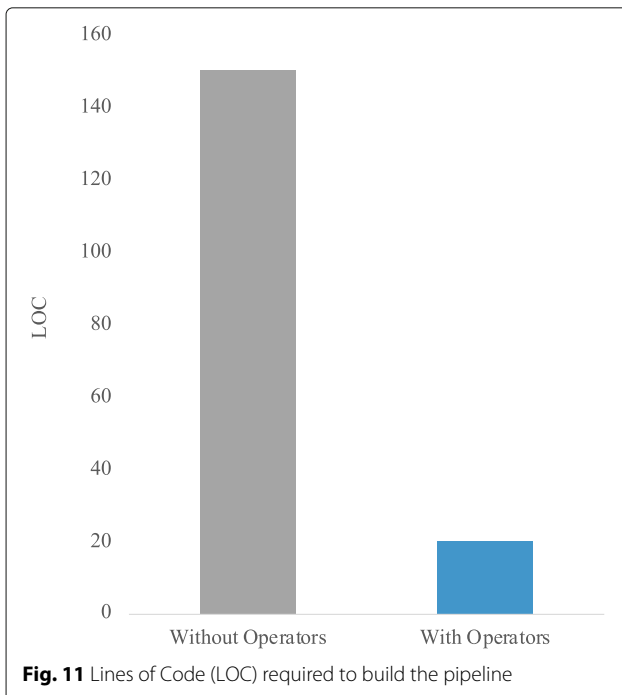


Figure 6 presents the entire process of model evolution and updating on the *Sophon Edge* platform. Suppose that some training data have been stored on the Edge Hub. First, the training data will be sent to the AI backend of the Edge Hub for model training. Once the training is completed, a model image will be built via the container technologies. Then the model image will be saved on the Edge Hub's model warehouse. Developers can deliver certain models to certain Edge Nodes on-demand, and the delivered models can be further updated through synchronization with the latest version of the models on the Edge Hub. After getting the model inputs through processing the raw data generated by IoT devices, the Edge Node can use certain models on the local to perform inferences. The inference results will be uploaded to the cloud together with the corresponding input data. After manual inspection, these data can be added to the training set as new labels and features. Through the continuous updates of training data, AI models are thus updated in an agile manner. As can be seen in Fig. 6, the entire process forms a closed loop, which allows the AIoT application to evolve in an iterative manner.

An application example

In this section, we present a real-world application example to demonstrate the effectiveness and efficiency of using the *Sophon Edge* platform.

Background

In many cities in China, diversion of rain and sewage water is required by the environment-related administrations. However, the implementation of this policy requires quick response to capture the undue behavior such as direct dumping sewage into the rainwater pipe network on the street, especially those sewage dumping behaviors conducted by the people in the shops nearby since the relatively large volume of sewage they dump in one time. The challenge is that such behavior normally happens in seconds, making it difficult to be observed and captured on the spot. Although surveillance devices on street may provide real-time video, manually capture the undue behavior (e.g., one staff staring at the screen all the time) of sewage dumping through video is not only boring but also inefficient.

To address the above problem, we built an AIoT application that automatically detects and alerts sewage dumping behavior based on the *Sophon Edge* platform.

Implementation

First, we set up the *Sophon Edge* platform and integrate the platform with the city's existing monitoring system. Usually, the street cameras have been connected to the gateways in the existing system, and these gateways provide interfaces to access the real-time video stream in different cameras. Then we deployed several Edge Nodes near these gateways, and one Edge Hub in the central server to support a new AIoT application to equip the existing cameras with the capability to detect sewage dumping behaviors in a timely manner. A noteworthy point is that the number of Edge Nodes to be deployed requires subtle consideration in practical applications, as there are normally a great number of street cameras in a city, it is important to keep a balance between cost and performance.

The next step is model training. We trained two AI models on the Edge Hub to detect the sewage dumping behavior (assuming that there was an available training set). Specifically, two computer vision (CV) models were trained to implement the task. The first model is used for "intrusion detection", i.e., to detect whether someone is approaching the drain. The second model is used for "activity recognition", i.e., to recognize the activity of sewage dumping. Note that these models can achieve better accuracy through model evolution and updating mentioned in the "Agile AIoT" section.

After the models are built, we defined the *product* that represents the street cameras, and built the pipeline shown in Fig. 7, utilizing the pre-implemented operators. The "Real-time Stream Input" operator is used to get access to the video stream. Through the "Demux" and "Decode" operators, images are retrieved from the video and fed to the input required by the

AI models. The ‘Region Of Interest’ operator specifies that only the area around the drain will be processed in subsequent operators, as illustrated in Fig. 8, which indeed clips the image and thus reduces the input size to improve efficiency and save resources. Two ‘Apply Model’ operators in which the two aforementioned AI models are used to make inferences from the clipped images. Only when both models output positive, that is, the sewage dumping behavior is recognized, the ‘Encode’ operator encodes the current image into JPEG format, and the ‘Alert’ operator further sends the image file to the cloud and raised a corresponding alert.

To provide a precise concept, we use pseudo-code to show the key elements of implementation to the AIoT application based on the *Sophon Edge* platform, which is presented in Code 1. The first part is to create a *product*, the second snippet is for binding the devices, and the last part is to create the corresponding pipeline.

```
public void main() {
    ...
    // Create the product
    Product cam = Product.name("camera")
        .property("video", Property.
            ↪ STREAM_DATA)
        .create();

    // Bind the cameras to the product
    for (String uri: cameraURIs) {
        cam.addPropertyBinding("video", uri);
    }

    // Create the pipeline
    Pipeline p = Pipeline.name("sewage dumping
        ↪ behavior detection")
        .streamInput(cam)
        .demux()
        .decode()
        .setProcessRegion(regions)
        .applyModel(intrusionDetectionModel)
        .applyModel(activityRecognitionModel)
        .encode()
        .alert(edgeHubURI)
        .create();
    ...
}
```

Code 1 Pseudo key implementation of the application

As the last step, we deployed instances of the pipeline on the Edge Nodes. By now this AIoT application has been successfully built and deployed. Figure 9 shows a detected example during the field test phase. With these captured images, administrators are able to capture such undue behaviors effectively. Moreover, manual efforts are reduced by large and the environment is thus well protected.

Results

As the result, the whole application reaches an accuracy of 85% in detecting sewage dumping behaviors.

To portray a concept of the latency of this AIoT application, we randomly selected 4 street cameras and the

corresponding gateway, Edge Node, and Edge Hub as well, i.e., 7 nodes in total to form a small network for evaluation. We measured the network latency in terms of Round-Trip Delay (RTD) as well as the elapsed time of data transmission between the nodes and pipeline processing on the Edge Node. The results are presented in Fig. 10. The average elapsed time of data transmission from the street cameras to the gateway, from the gateway to the Edge Node, and from the Edge Node to the Edge Hub is 2 seconds, 1 second, and 1 second, respectively. In this sense, the total elapsed time from a sewage dumping behavior being captured by a camera to a corresponding alert being raised on the cloud server (i.e. Edge Hub) is 4.5 seconds on average. To be specific, the pipeline shown in Fig. 7 takes about 0.5 seconds to process a single frame.

Regarding the efficiency, Fig. 11 shows that the pre-implemented operators help to reduce the lines of code (LOC) required to build the pipeline from 150 on average down to around 20, indicating that the *Sophon Edge* platform does help to save developers’ coding efforts.

Discussion

In this section, we present some in-depth discussions about several advantages and limitations of the *Sophon Edge* platform.

Advantages

The *Sophon Edge* platform was designed with the primary purpose of improving the efficiency of building AIoT applications. The platform provides a practical framework for developers to define computation tasks. The computation tasks are described as pipelines, which can be simply built through assembling several pre-implemented operators without having to write massive lines of source code on their own, which is rather efficient. Developers can concentrate on the overall computing procedure and better support business needs. Moreover, user-implemented operators are also supported to meet some advanced requirements in some special scenarios. The abstract pipelines also give the platform considerable flexibility, as a single pipeline can be instantiated multiple times with different configurations.

Managing devices is also efficient with the *Sophon Edge* platform. The mainstream IoT communication protocols are all covered so that most of the devices in current production environment can be quickly connected to the platform (see Table 3). The introduction of the *product* concept further facilitates data access to devices. Besides, the *product* allows a single pipeline to be deployed upon different individual devices with the same or similar function, and to keep working after device change under the same *product*. This promotes

Table 3 Comparison of Sophon Edge and other AIIoT platforms

		AWS IoT	Alibaba Cloud AIIoT Native	Tencent IoT Hub	Baidu Tiangong IoT Hub	Sophon Edge
Device Access	MQTT	✓	✓	✓	✓	✓
	COAP	-	✓	✓	✓	✓
	HTTP	✓	✓	✓	-	✓
	Others	LoRaWAN	-	-	-	Modbus/OPCUA/...
Computing	Rule-based	✓	✓	✓	✓	✓
	Complex	✓	✓	✓	✓	✓
Data Storage	Edge	-	-	-	✓	✓
	Cloud	✓	✓	✓	✓	✓
AI Model	Train	Cloud	Cloud	Cloud	Cloud	Cloud
	Inference	Cloud	Edge	Edge	Edge	Edge

the portability and availability of the pipeline, adapting to the heterogeneous and dynamic nature of the IoT environment.

In traditional programming, the same or similar code snippets commonly exist. Moving and merge these code into a single place can greatly benefit the maintainability of the software. Likewise, the same or similar pipelines could exist on multiple edge servers in an IoT environment. Thanks to the edge-cloud collaboration, pipelines can be shared in the cloud, and reused to different edge servers on-demand. This also improves efficiency in a sense.

Notably, as the “brain” of the overall application, the AI models can continuously evolve in the platform through a close loop of model training and performance feedback. In other words, the edge servers can always be equipped with the latest model to make better inferences in its changing environment.

Last but not least, the application can tolerate a certain degree of network disconnection between the devices and the cloud, as the edge can continue to perform localized decision making. After the network is restored, the data can be continuously transmitted and synchronized. In response to data privacy issues, the edge only uploads desensitized data to the cloud, reducing the risk of privacy data leakage.

The status of the *Sophon edge* platform

The *Sophon Edge* platform is still evolving currently. We provide a rough comparison of some similar platforms to show the general status of existing AIIoT platforms. Since it is not feasible to deploy and verify all the platforms, we extract the above information from technical reports and white papers as well. The results are presented in Table 3 based on four aspects, i.e., device access (communication protocols), computing, data storage and AI models.

In general, the *Sophon Edge* platform wins in its comprehensive functions, which is extremely valuable, given the heterogeneous nature of IoT environment.

Conclusion

The AIIoT combines the popular AI and IoT technologies, aiming to build a connected ecosystem with advanced artificial intelligence. Currently, the best solution for AIIoT seems to be the edge-cloud collaborative computing paradigm. While the cloud exploits its sufficient computing and storage resources to train sophisticated machine learning models, frequent but less complicated computation tasks are offloaded to the edge of the IoT network to provide low latency services, allowing the end users of the AIIoT application to get near real-time responses, which really improves the user experience. The rapid development of multiple enabling technologies also paves the way for the AIIoT applications. Nevertheless, developers face difficulties when building AIIoT applications in practice, due to the diversity of devices, the context-dependent characteristics of application logic, and some other reasons.

The *Sophon Edge* platform is proposed as a promising solution to these challenges. The pipeline-based computing model facilitates the programming of computation tasks, and allows the computing to scale up flexibly across multiple edge servers and even different IoT devices. The edge-cloud collaboration further allows the AI models to evolve with the continuously generated device data and quickly respond to the application logic change. The real-world Smart City application built on the *Sophon Edge* demonstrates the effectiveness and usability of the platform. A great deal of coding efforts has been saved in developing the application with the help of the platform’s abundant pre-implemented operators.

As for future work, an effective computation offloading scheme can be utilized to balance the computing and storage load on every edge server to maximally exploit the available resources. Distributed machine learning, especially federate learning in terms of data privacy-preserving, is another promising future work to achieve better efficiency, as currently the AI model is trained in a centralized manner on the *Sophon Edge* platform.

Abbreviations

AI: Artificial Intelligence; IoT: Internet of Things; AIoT: Artificial Intelligence of Things; MEC: Mobile Edge Computing; QoS: Quality of Service; LV: Lightweight Virtualization; DCOA: Dynamic computation offloading algorithm; DRL: Deep Reinforcement Learning; MQTT: Message Queuing Telemetry Transport; COAP: Constrained Application Protocol; AMQP: Advanced Message Queuing Protocol; ONVIF: Open Network Video Interface Forum; RTSP: Real Time Streaming Protocol; RTMP: Real-Time Messaging Protocol; UVC: USB Video device Class; OPCUA: OPC Unified Architecture; HLS: HTTP Live Streaming

Acknowledgments

The authors would like to express their gratitude to engineers in Transwarp Inc. for providing detailed information of the *Sophon Edge* platform and the presented real-world application.

Authors' contributions

GR (corresponding author): Contribution for this work includes research process planning and execution, supervision, writing-review, and polishing. YX: Contribution for this work includes conceptualization, investigation, writing-draft, and editing. XT & HF: Their contribution includes managing required software, and providing resources such as documentations and other related materials. All authors read and approved the final manuscript.

Funding

This work is jointly supported by the National Key Research and Development Program of China (No.2019YFE0105500) and the Research Council of Norway (No. 309494), as well as the National Natural Science Foundation of China (Grants No.62072227, 61802173), Intergovernmental Bilateral Innovation Project of Jiangsu Province (BZ2020017).

Availability of data and materials

Not applicable.

Declarations

Competing interests

The authors declare that they have no competing interests.

Author details

¹The Joint Laboratory of Nanjing University and Transwarp on Data Technology, Nanjing University, Nanjing, China. ²The State Key Laboratory of Novel Software Technology, Nanjing University, Nanjing, China. ³Transwarp Inc, Shanghai, China.

Received: 11 January 2021 Accepted: 7 June 2021

Published online: 02 July 2021

References

1. Ray PP, Dash D, De D (2019) Edge computing for internet of things: A survey, e-healthcare case study and future direction. *J Netw Comput Appl* 140:1–22. <https://doi.org/10.1016/j.jnca.2019.05.005>
2. Riquebourg V, Menga D, Durand D, Marhic B, Delafoche L, Loge C (2006) The smart home concept: our immediate future. In: 2006 1st IEEE International Conference on E-learning in Industrial Electronics. IEEE, pp 23–28. <https://doi.org/10.1109/ICELIE.2006.347206>
3. Lucke D, Constantinescu C, Westkämper E (2008) Smart factory—a step towards the next generation of manufacturing. In: *Manufacturing Systems and Technologies for the New Frontier*. Springer, London, pp 115–118. https://doi.org/10.1007/978-1-84800-267-8_23
4. Schaffers H, Komninos N, Pallot M, Trousse B, Nilsson M, Oliveira A (2011) Smart cities and the future internet: Towards cooperation frameworks for open innovation. In: *The Future Internet Assembly*. Springer, Berlin, Heidelberg, pp 431–446. https://doi.org/10.1007/978-3-642-20898-0_31
5. Calo SB, Touna M, Verma DC, Cullen A (2017) Edge computing architecture for applying ai to iot. In: 2017 IEEE International Conference on Big Data (Big Data). IEEE, pp 3012–3016. <https://doi.org/10.1109/BigData.2017.8258272>
6. Premsankar G, Di Francesco M, Taleb T (2018) Edge computing for the internet of things: A case study. *IEEE Internet Things J* 5(2):1275–1284. <https://doi.org/10.1109/JIOT.2018.2805263>
7. Yu W, Liang F, He X, Hatcher WG, Lu C, Lin J, Yang X (2017) A survey on the edge computing for the internet of things. *IEEE Access* 6:6900–6919. <https://doi.org/10.1109/ACCESS.2017.2778504>
8. Shi W, Dustdar S (2016) The promise of edge computing. *Computer* 49(5):78–81. <https://doi.org/10.1109/MC.2016.145>
9. Shi W, Cao J, Zhang Q, Li Y, Xu L (2016) Edge computing: Vision and challenges. *IEEE Internet Things J* 3(5):637–646. <https://doi.org/10.1109/JIOT.2016.2579198>
10. Varghese B, Wang N, Barbhuiya S, Kilpatrick P, Nikolopoulos DS (2016) Challenges and opportunities in edge computing. In: 2016 IEEE International Conference on Smart Cloud (SmartCloud). IEEE, pp 20–26. <https://doi.org/10.1109/SmartCloud.2016.18>
11. El-Sayed H, Sankar S, Prasad M, Puthal D, Gupta A, Mohanty M, Lin C-T (2017) Edge of things: The big picture on the integration of edge, iot and the cloud in a distributed computing environment. *IEEE Access* 6:1706–1717. <https://doi.org/10.1109/ACCESS.2017.2780087>
12. Satyanarayanan M (2017) The emergence of edge computing. *Computer* 50(1):30–39. <https://doi.org/10.1109/MC.2017.9>
13. Khan WZ, Ahmed E, Hakak S, Yaqoob I, Ahmed A (2019) Edge computing: A survey. *Future Generation Computer Systems* 97:219–235. <https://doi.org/10.1016/j.future.2019.02.050>
14. Atlam HF, Walters RJ, Wills GB (2018) Intelligence of things: opportunities & challenges. In: 2018 3rd Cloudification of the Internet of Things (ClOT). IEEE, pp 1–6. <https://doi.org/10.1109/CIOT.2018.8627114>
15. Zhang J, Tao D (2020) Empowering things with intelligence: A survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet Things J*. <https://doi.org/10.1109/JIOT.2020.3039359>
16. Katare G, Padihar G, Quereshi Z (2018) Challenges in the integration of artificial intelligence and internet of things. *Int J Syst Softw Eng* 6(2):10–15
17. Wang S, Hu Y, Wu J (2020) Kubeedge: ai: Ai platform for edge devices. *arXiv preprint arXiv:2007.09227*
18. Pan J, McElhannon J (2017) Future edge cloud and edge computing for internet of things applications. *IEEE Internet Things J* 5(1):439–449
19. Ai Y, Peng M, Zhang K (2018) Edge computing technologies for internet of things: a primer. *Digit Commun Netw* 4(2):77–86. <https://doi.org/10.1016/j.dcan.2017.07.001>
20. Giang NK, Lea R, Blackstock M, Leung VC (2018) Fog at the edge: Experiences building an edge computing platform. In: 2018 IEEE International Conference on Edge Computing (EDGE). IEEE, pp 9–16. <https://doi.org/10.1109/EDGE.2018.00009>
21. Lertsinsrubtavee A, Ali A, Molina-Jimenez C, Sathiseelan A, Crowcroft J (2017) Picasso: A lightweight edge computing platform. In: 2017 IEEE 6th International Conference on Cloud Networking (CloudNet). IEEE, pp 1–7. <https://doi.org/10.1109/CloudNet.2017.8071529>
22. Morabito R, Beijar N (2016) Enabling data processing at the network edge through lightweight virtualization technologies. In: 2016 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops). IEEE, pp 1–6. <https://doi.org/10.1109/SECONW.2016.7746807>
23. Morabito R (2017) Virtualization on internet of things edge devices with container technologies: A performance evaluation. *IEEE Access* 5:8835–8850. <https://doi.org/10.1109/ACCESS.2017.2704444>
24. Morabito R, Cozzolino V, Ding AY, Beijar N, Ott J (2018) Consolidate iot edge computing with lightweight virtualization. *IEEE Netw* 32(1):102–111. <https://doi.org/10.1109/MNET.2018.1700175>
25. Chen X, Chen S, Zeng X, Zheng X, Zhang Y, Rong C (2017) Framework for context-aware computation offloading in mobile cloud computing. *J Cloud Comput* 6(1):1. <https://doi.org/10.1186/s13677-016-0071-y>
26. Xu X, Liu Q, Luo Y, Peng K, Zhang X, Meng S, Qi L (2019) A computation offloading method over big data for iot-enabled cloud-edge computing.

Futur Gener Comput Syst 95:522–533. <https://doi.org/10.1109/JIOT.2017.2767608>

27. Chen Y, Zhang N, Zhang Y, Chen X (2018) Dynamic computation offloading in edge computing for internet of things. *IEEE Internet Things J* 6(3):4242–4251. <https://doi.org/10.1109/JIOT.2018.2875715>
28. Shen S, Han Y, Wang X, Wang Y (2019) Computation offloading with multiple agents in edge-computing-supported iot. *ACM Trans Sensor Netw (TOSN)* 16(1):1–27. <https://doi.org/10.1145/3372025>
29. Debauche O, Mahmoudi S, Mahmoudi SA, Manneback P, Lebeau F (2020) A new edge architecture for ai-iot services deployment. *Procedia Comput Sci* 175:10–19. <https://doi.org/10.1016/j.procs.2020.07.006>
30. Li H, Ota K, Dong M (2018) Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Netw* 32(1):96–101. <https://doi.org/10.1109/MNET.2018.1700202>
31. Xiong J, Chen H (2020) Challenges for building a cloud native scalable and trustable multi-tenant aiot platform. In: 2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD). IEEE. pp 1–8
32. Chiu T-C, Shih Y-Y, Pang A-C, Wang C-S, Weng W, Chou C-T (2020) Semi-supervised distributed learning with non-iid data for aiot service platform. *IEEE Internet Things J*. <https://doi.org/10.1109/JIOT.2020.2995162>

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Submit your manuscript to a SpringerOpen[®] journal and benefit from:

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

Submit your next manuscript at ► [springeropen.com](https://www.springeropen.com)
