Article

A Survey of AI Techniques in IoT Applications with Use Case Investigations in the Smart Environmental Monitoring and Analytics in Real-Time IoT Platform

Yohanes Yohanie Fridelin Panduman 1, Nobuo Funabiki 1,*; Evianita Dewi Fajrianti 1,; Shihao Fang 1 and Sritrusta Sukaridhoto 2

1 Graduate School of Natural Science and Technology, Okayama University, Okayama 700-8530, Japan; p8f01q6f@s.okayama-u.ac.jp (Y.Y.F.); p2mu1tom@s.okayama-u.ac.jp (E.D.F.); pkpb8c9q@s.okayama-u.ac.jp (S.F.)
2 Department of Informatic and Computer, Politeknik Elektronika Negeri Surabaya, Surabaya 60111, Indonesia; dhoto@pens.ac.id
* Correspondence: funabiki@okayama-u.ac.jp

Abstract: In this paper, we have developed the SEMAR (Smart Environmental Monitoring and Analytics in Real-Time) IoT application server platform for fast deployments of IoT application systems. It provides various integration capabilities for the collection, display, and analysis of sensor data on a single platform. Recently, Artificial Intelligence (AI) has become very popular and widely used in various applications including IoT. To support this growth, the integration of AI into SEMAR is essential to enhance its capabilities after identifying the current trends of applicable AI technologies in IoT applications. In this paper, we first provide a comprehensive review of IoT applications using AI techniques in the literature. They cover predictive analytics, image classification, object detection, text spotting, auditory perception, Natural Language Processing (NLP), and collaborative AI. Next, we identify the characteristics of each technique by considering the key parameters, such as software requirements, input/output (I/O) data types, processing methods, and computations. Third, we design the integration of AI techniques into SEMAR based on the findings. Finally, we discuss use cases of SEMAR for IoT applications with AI techniques. The implementation of the proposed design in SEMAR and its use to IoT applications will be in future works.

Keywords: Internet of Things; AI; integration; survey; application server platform; SEMAR

1. Introduction

Nowadays, the Internet of Things (IoT) has attracted significant interest from both industrial and academic communities as an emerging technology designed to connect cyberspace with physical devices using Internet infrastructure [1]. As IoT infrastructures become common, an IoT application system can consist of various sensor devices and network connections across multiple domains [2]. In this context, developers need to efficiently design and implement the system adopting standards for heterogeneous device management and interoperability with other systems [3].

In this paper, we have developed the IoT application server platform called SEMAR (Smart Environmental Monitoring and Analytics in Real-Time). It can serve as a cloud server for integrating various IoT application systems. It offers various integration capabilities for the collection, display, and analysis of sensor data on a single platform [4], by providing built-in functions for data communications, aggregations, synchronizations, and classifications using machine learning algorithms in Big Data environments. It also supports the implementation of plug-in functions by allowing other systems to access data through the Representational State Transfer Application Programming Interface (REST API). Although
SEMAR has been efficiently used in several IoT applications, improvements to the platform will be expected to address the demands of IoT applications that may require more advanced data processing algorithms.

Recently, Artificial Intelligence (AI) has become very popular as a data processing algorithm. AI has been inspired by the thinking of the human brain [5]. It can create intelligent systems that may learn, operate, and respond intelligently like human behaviors [6,7]. In fact, AI is a field of intelligence systems that covers not only algorithms for data processing but also offers a wide range of tools and techniques that help computers perform specific tasks [8]. Machine learning, Natural Language Processing (NLP), deep learning, pattern recognition, optimization, robotics, and computer vision can be included as subfields of AI. Due to its ability to solve a lot of complex problems, AI is now widely used in various applications including IoT.

The integration of AI plays a critical role in the evolution of IoT technology. It enables advanced sensor data analysis by identifying data patterns, extracting valuable information, and making rapid decisions based on it [9]. Furthermore, the utilization of Big Data technologies enhances this integration by providing huge data sets for training AI models. This significantly increases the potential of AI implementations in various IoT applications. To support this growth, the integration of AI into SEMAR becomes essential to enhance its capabilities as an effective IoT application server platform.

The current AI studies are concerned with selecting appropriate techniques for specific cases of IoT applications. Each technique has its own characteristics and requirements. For instance, deep learning models require more computational resources, which increase with the computational complexity of the model. Therefore, it is important to identify the current trends of applicable AI techniques and their characteristics for effective integration.

In this paper, we present an overview of current AI techniques and their use cases in IoT applications. Our methodology explores the potential of AI integrations and how they can be implemented in IoT applications. First, we provide a comprehensive review of current studies on IoT applications using AI techniques. They include predictive analytics, image classification, object recognition, text spotting, auditory perception, NLP, and collaborative AI. Then, we identify the characteristics of each technique by considering the key parameters that play a critical role in integrations. These parameters include software requirements, input/output (I/O) data types, processing methods, and computations. Based on our findings, we design the seamless integration of AI capabilities into the SEMAR platform. Finally, we discuss use cases of IoT applications with AI techniques to illustrate how SEMAR can be used to support their developments.

This study contributes to the field of AI integration in the IoT domain. First, we present a comprehensive review of recent studies on AI technologies applied in various IoT application use cases. Second, we provide a detailed analysis of an AI algorithm with key parameters that are critical for effective AI integrations in IoT application developments. Third, we present the design of an IoT platform service with seamlessly integrated AI-driven capabilities that include both cloud and edge components. Through three IoT application use cases, we illustrate how our designed platform supports and enhances IoT application development with AI processes.

The rest of the paper is organized as follows: Section 2 briefly reviews our previous development of SEMAR. Section 3 presents the comprehensive literature review of AI techniques in IoT applications and highlights their use cases. Section 4 presents the design of AI techniques integrations in SEMAR. Section 5 describes the use cases of SEMAR in IoT applications. Finally, Section 6 concludes this paper with future works.

2. Review of SEMAR IoT Application Server Platform

In this section, we review the SEMAR IoT application server platform to facilitate the development of IoT applications in the cloud. In our previous studies [4], we presented the design and implementation of SEMAR with several IoT application systems. Currently, SEMAR provides integrated capabilities functions to collect, display, process, and analyze
sensor data. It provides built-in functions for data communication, aggregation, synchronization, and classification using machine learning algorithms that can be used without implementation or modification of the original source code. It also allows users to add plug-in functions by providing data access through the REST API. Furthermore, it was integrated with the edge device framework [10] that allows users to seamlessly create, update, and delete the edge configuration file for edge devices through the user interface of SEMAR. Figure 1 shows the system overview of SEMAR.

Figure 1. Design overview of SEMAR IoT application server platform.

The built-in functions can be divided into several components for data input, data processing, and data output. All the components are managed by the management system.

Collecting data from various IoT devices using network interfaces and communication protocols in SEMAR is handled by the data input components. They include the IoT cloud gateway, which provides services for data communications through MQTT, HTTP POST and KAFKA communication protocols, and the data aggregator, which collects raw sensor data and converts them to the consumable form according to the sensor format defined in the device management data. The data aggregator transmits the sensor data to the data processing components and stores them in the data storage. MongoDB [11] is implemented for the data storage service. The SEMAR platform allows for receiving various types of sensor data from devices in JavaScript Object Notation (JSON) format, as illustrated in Figure 1. They include image frames and common data types such as integer, float, string, boolean, date, and time.

The data processing components provide the functions for processing and analyzing the obtained data. The data filter allows to use several digital filtering techniques to reduce noise and inaccuracies in the data. The data synchronization allows the data from different resources to be synchronized into a single data record. It stores the result data in the schema data storage. The machine learning techniques can be used to provide real-time data classification services in the data analytics component. The current implementation covers the decision tree and Support Vector Machines (SVM) classification techniques.

According to the current trend of IoT applications, it is necessary to use several AI techniques in the data analytics component. In this paper, we first conducted a study of AI integration in IoT applications to identify the characteristics of each AI technique. Then,
based on the findings, we design the integration of AI techniques in SEMAR. In future works, we will implement the proposed design to improve the data analytics component.

The data output components provide the functions to visualize the obtained data and allow users to access them. A web-based user interface is prepared to visualize the data. The data export function is implemented in the user interface to allow the user to download the data at a specific time in a TXT, Excel, JSON, or CSV format. The notification function sends the message when the sensor value matches the user-defined threshold. The last component utilizes the REST API service for data sharing and integrations with the plug-in functions or other systems through HTTP POST communications in the JSON format.

In terms of the user model, the SEMAR platform is designed to support multi-user usage. In order to manage devices, user authorizations, communication protocols, and data in SEMAR, the management service is implemented. This service facilitates data sharing and access control among users through the user interface. First, the user is required to register a device on the platform, specifying the types of sensor data to be collected. Then, the user can send invitations to other registered users to grant them permission to access the sensor data. Once the invitation is accepted by the target user, access to the shared device data is granted. Finally, the user is able to view and download the data from the user interface of SEMAR.

In Ref. [10], we designed and built the edge device framework to optimize device utilizations by configuring it remotely through the SEMAR as the feature for device management. It allows users to create, update, and delete the edge configuration file by accessing the functions through the user interface.

The real-time system in the SEMAR platform processes data collected from devices and generates responses within a specified timeframe. This implies that the system is expected to meet deadlines to ensure time for data processing and response. For this purpose, the soft deadline approach has been implemented as the deadline constraint in our system. The soft deadline implies that completing a task within a certain time frame is preferable, but not mandatory. If a data processing task misses its soft deadline, it will trigger the timeout mechanism to avoid waiting indefinitely. However, this may result in data loss. To address this issue, we have implemented the data log function to store all the data sent by IoT devices as it is received by the IoT gateway. In addition, we have utilized the Network Time Protocol (NTP) to synchronize clocks between functions in the SEMAR platform.

3. Literature Review on Use Cases of AI Techniques in IoT Applications

In this section, we present a review of use cases of AI techniques in IoT applications as comprehensively as possible.

3.1. Methodology

The main purpose of this literature review section is to identify AI techniques that have been frequently used in IoT applications, including algorithms, characteristics, and how they can be implemented in IoT application use cases. For this purpose, we followed a structured research methodology. It consists of identifying the trends of applied AI techniques, finding the related literature, investigating characteristics, and analyzing necessary requirements for seamless integrations.

First, we identify the trends of applied AI techniques in the development of IoT application systems. For this purpose, we explored the surveyed papers that discuss applied AI techniques in IoT application systems with their potential. According to the findings in several studies [12–16], we selected predictive analytics, image classification, object detection, text spotting, auditory perception, NLP, and collaborative AI as the typical AI techniques to be explored in this paper.

In the next step, we systematically selected relevant papers for reviews from popular scientific databases, including Scopus, Elsevier, and IEEE. To capture the current state of each AI technique, we limited our literature review to publications that have been published between 2019 and 2023. Our literature selection was based on a combination of keywords
representing each AI technique identified in the previous step, as well as the domains of IoT application use cases. They included smart environments, smart manufacturing, smart cities, smart homes, smart buildings, smart healthcare, smart agriculture, smart farming, and smart laboratories.

For investigating the characteristics of each technique and its application use cases, we considered critical features such as software requirements, I/O data types, processing methods, and computations. Finally, we analyzed the unique strengths and requirements of each AI technique with the specific purpose of designing seamless integration. Following this insight, we designed the AI integration into the SEMAR platform.

3.2. Predictive Analytics

This subsection provides an overview of the current state of the art for integrating predictive analytics into IoT systems by reviewing papers with considering application use cases.

3.2.1. Overview

The integration of AI into an IoT application has changed the way how data is collected, processed, and visualized. As an IoT application requires the ability to rapidly extract meaningful information from data, a function or a system that enables the identification of data patterns and trends in a real-time manner becomes a critical issue. Predictive analytics is one of the AI techniques often used to solve this issue. It finds knowledge in current and past data to generate predictions of future events by using machine learning, statistics, and data mining techniques [17].

In the context of IoT, predictive analytics analyzes historical sensor data saved in the database to predict future events or data trends. It is often used to perform anomaly detection, predictive maintenance, optimization, and decision-making in a real-time or near-real-time manner.

3.2.2. Use Cases in IoT Applications and Characteristics Overview

Several papers discussed use cases that provide the potential for using predictive analytics techniques to improve IoT application systems. They include applications in smart environments [18–22], smart manufacturing [23,24], smart home [25], smart building [26], smart healthcare [27,28], smart farming [29], and smart agriculture [30].

Forecasting future environmental conditions based on historical data that were collected by sensors is one of the goals of smart environments. As a direction toward this goal, in [18], Imran et al. proposed an IoT-based simulation system that predicts fire spread and burned areas in mountainous areas. In Ref. [19], Hussain et al. used predictive analytics techniques to forecast the level of carbon monoxide (CO) concentration in the area around a garbage bin. Mumtaz et al. in [20] provided a system that can predict the concentration level of air pollutants in an indoor environment. Barthwal et al. in [21] proposed an IoT application to predict the air quality index (AQI) based on collected mobile sensor data. Jin et al. in [22] proposed a novel approach for predicting particulate matter (PM) 2.5 concentrations using a Bayesian network. This study introduced a Bayesian-based algorithm that provides a potential robust prediction for time-series data.

In Ref. [23], Bampoula et al. developed a system for predicting the remaining useful life of machinery in the steel industry. Teoh et al. in [24] developed an application focused on the predictive maintenance of manufacturing equipment. These use cases illustrate the effectiveness of integrating predictive analytics into IoT application systems to improve industrial asset management by accurately estimating machine or equipment conditions.

Predictive analytics techniques have also been applied to predict energy consumption in smart homes and smart building applications. In Ref. [25], Shorfuzzaman et al. presented the practical implementation for minimizing energy consumption of home appliances in a smart home context using predictive analytics. Then, Guo et al. in [26] provided a system
for predicting building electricity consumption based on a small-scale data set collected by sensors.

In Ref. [27], Nancy et al. offered the application of predictive analytics techniques into IoT cloud-based systems to forecast the risk of heart disease based on the medical history data of patients in the context of smart healthcare. Subahi et al. in [28] introduced a self-adaptive Bayesian algorithm for predicting heart disease based on medical data collected from patients. The medical data include several parameters related to the patient’s vital signs, such as blood pressure and heart rate.

In Ref. [29], Patrizi et al. demonstrated the implementation of a virtual-based soil moisture sensor in the context of smart farming applications. This can be achieved by estimating soil moisture using collected sensor data using predictive analytics techniques. Kocian et al. in [30] introduced an IoT system for smart agriculture. This system facilitated crop coefficient modeling and crop evapotranspiration (ET) prediction in a soilless substrate using a dynamic Bayesian approach. It utilized data collected from multiple sensors, including crop weight, global radiation, and temperature, to predict crop ET.

Table 1 summarizes the characteristics of predictive analytics techniques that were adopted in the papers discussed in this subsection. Long Short-Term Memory (LSTM) is the widely used algorithm for predicting future events. It belongs to the variant of Recurrent Neural Network (RNN) architecture, which effectively learns and retains information over a long period using cell states [31]. The works by Barthwal et al. in [21], Shorfuzzaman et al. in [25], and Guo et al. in [26] utilized the capabilities of the Autoregressive Integrated Moving Average (ARIMA) algorithm [32] to construct robust data models for predicting future values in time series data. Taking it one step further, Guoh et al. in [26] combined Support Vector Regression (SVR) with ARIMA to predict energy consumption. Then, Imran et al. in [18] integrated Principal Component Regression (PCR) and Artificial Neural Network (ANN) for an effective predictive model in their application system.

Table 1. Key characteristics of predictive analytics technique in current studies.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Algorithms</th>
<th>Software Requirements</th>
<th>Data Types</th>
<th>Processing Methods</th>
<th>Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[18]</td>
<td>ANN with PCR and Kalman filter</td>
<td>Python</td>
<td>Time-series data</td>
<td>Predicted area</td>
<td>Filtering and real-time data processing</td>
</tr>
<tr>
<td>[19]</td>
<td>LSTM</td>
<td>Python, TensorFlow</td>
<td>Time-series data</td>
<td>Predicted CO level</td>
<td>Real-time data processing</td>
</tr>
<tr>
<td>[20]</td>
<td>LSTM</td>
<td>-</td>
<td>Time-series data</td>
<td>Predicted air pollutants level</td>
<td>Real-time data processing</td>
</tr>
<tr>
<td>[21]</td>
<td>ARIMA</td>
<td>-</td>
<td>Time-series data</td>
<td>Predicted AQI</td>
<td>Real-time data processing</td>
</tr>
<tr>
<td>[22]</td>
<td>Bayesian Network</td>
<td>Python</td>
<td>Time-series data</td>
<td>Predicted PM2.5 level</td>
<td>Missing data handling, data normalization, and data correlations</td>
</tr>
<tr>
<td>[23]</td>
<td>LSTM</td>
<td>Python, TensorFlow</td>
<td>Time-series data</td>
<td>Predicted machine states</td>
<td>Data transformation and real-time data processing</td>
</tr>
<tr>
<td>[25]</td>
<td>LSTM and ARIMA</td>
<td>Python, TensorFlow</td>
<td>Time-series data</td>
<td>Predicted energy consumption</td>
<td>Missing data handling, outlier detection, data transformation</td>
</tr>
</tbody>
</table>
In addition, the works by Kocian et al. in [30], Jin et al. in [22], and Subahi et al. in [28] used Bayesian-based approaches. While ARIMA focuses on constructing models of the time-series data, Bayesian approaches take a different approach by generating prior knowledge in the form of a probability distribution for predicting the data. This prior knowledge represents initial beliefs. It is then updated with observed data using Bayes’ theorem, allowing for more flexible and robust modeling. Moreover, Bayesian approaches can be implemented in scenarios with limited datasets, as mentioned in the work by Kocian et al. in [30].

Predictive analytics is essential to estimate future values or labels in real-time scenarios of IoT applications. The effective implementation of this technique requires consideration of several key elements. First, a database system that can handle time-series data is critical for efficient data storage and retrieval. Second, the pre-processing capabilities must be implemented to prevent potential error data in the collected data and improve reliability. Third, the IoT system must have real-time data processing capabilities to perform immediate analysis for rapid forecasting and decision-making. Fourth, Python, with its extensive support for algorithms for predictive analytics such as LSTM, ARIMA and Bayesian, becomes the suitable option for the software environment. As shown in Table 1, the predictive analytics can be deployed on either servers or edge devices such as Raspberry Pi.

### 3.3. Image Classification

In this subsection, we review papers emphasizing IoT application use cases for image classifications.

#### 3.3.1. Introduction

Computer vision is the field of AI that mimics human intelligence to understand image data. It enables machines to see and recognize objects from visual images to facilitate decision-making [33]. Techniques such as image classification and object detection are part of the fields in computer vision, where image classification refers to the ability to identify categories of images. In the IoT domain, image classification plays an important role in recognizing visual data using a classification model. Typically, the data model is trained using labeled image datasets, where each image is assigned to a specific category.

#### 3.3.2. Use Cases in IoT Applications and Characteristics Overview

The implementation of an image classification in an IoT application has been explored in numerous papers. Each paper demonstrated the effectiveness of image classification algorithms in addressing vision-based use cases in a variety of applications.

## Table 1. Cont.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Algorithms</th>
<th>Software Requirements</th>
<th>Data Types</th>
<th>Processing Methods</th>
<th>Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[26]</td>
<td>ARIMA and SVR</td>
<td>Python</td>
<td>Time-series data</td>
<td>Predicted electric consumption</td>
<td>Missing data handling, data normalization, and data correlations</td>
</tr>
<tr>
<td>[27]</td>
<td>Bidirectional LSTM</td>
<td>Python, TensorFlow</td>
<td>Time-series data</td>
<td>Predicted diagnosis of heart disease</td>
<td>Data Filtering</td>
</tr>
<tr>
<td>[28]</td>
<td>Self-Adaptive Bayesian</td>
<td>-</td>
<td>Time-series data</td>
<td>Predicted diagnosis of heart disease</td>
<td>Data normalization</td>
</tr>
<tr>
<td>[29]</td>
<td>LSTM</td>
<td>Python, TensorFlow</td>
<td>Time-series data</td>
<td>Predicted soil moisture</td>
<td>Data correlations and data synchronization</td>
</tr>
<tr>
<td>[30]</td>
<td>Dynamic Bayesian</td>
<td>MATLAB™</td>
<td>Time-series data</td>
<td>Predicted ET value</td>
<td>Real-time data processing</td>
</tr>
</tbody>
</table>
For IoT applications in agriculture, image classification is a valuable technique for monitoring crops and detecting plant diseases. In Ref. [34], Chouhan et al. introduced a system for detecting galls, a plant disease that affects leaves, using captured images. This technique is also suitable for drone-based IoT applications in environmental monitoring systems. In a separate study [35], Munawar et al. developed a system to detect flooding from aerial images captured by drones using image classifications.

Image classification is a proven AI technique for supporting diagnostic processes through visual data analysis. In Ref. [36], Abd Elaziz et al. proposed a deep learning model incorporating MobileNet and DenseNet architectures for medical image classifications to achieve rapid diagnostic results. In Ref. [37], Saleh et al. employed a hybrid approach of combining Convolutional Neural Network (CNN) and SVM to classify lung cancers based on computed tomography (CT) scan images. These studies demonstrated the effectiveness of image classifications in improving diagnostic capabilities within smart healthcare use cases.

In Ref. [38], Iyer et al. explored the application of image classifications in transportation monitoring. This study demonstrated the detection of rail fractures by analyzing images captured by a mobile robot. Similarly, Medus et al. in [39] introduced a vision-based system for detecting leakage in food tray seals on the production line, using a similar concept of image classifications with the CNN algorithm. These use cases demonstrated the versatility of image classifications and extended its benefits to diverse areas beyond healthcare, such as transportation infrastructure and quality control in food production.

Table 2 shows the overview of the characteristics of image classification techniques in the literature discussed in this paper. The CNN algorithm [39] has been widely used for image classifications in various applications. The architecture of the CNN algorithm allows it to be integrated with other algorithms, such as SVM. In Ref. [37], Saleh et al. presented a hybrid algorithm with CNN and SVM to achieve robust performance. SVM is used to generate the classification result using features extracted by CNN. This approach leverages the strengths of both models and enhances accuracy in classification tasks.

Several researchers have proposed alternative approaches to classifying images instead of the CNN algorithm. Abd Elaziz et al. in [36] have developed a deep learning model that combines MobileNet and DenseNet architectures to extract medical image representation. This model is able to extract complex features from medical images, making it useful for better understanding and diagnosing medical conditions. Medus et al. in [39] presented the implementation of the Fuzzy-Based Functional Network (FBFN) algorithm that integrates fuzzy logic with function network capabilities. This approach allows the user to apply the image classification process in real time.

There are several key elements to be considered for implementing image classification algorithms. As the programming language, Python is often used. Then, libraries such as TensorFlow, Keras, and OpenCV are installed to implement various deep learning-based image processing algorithms. Since the input data includes image files, storage capacity becomes necessary. To achieve high performance, additional functions such as noise reductions in images are applied. Hyperparameter optimization is also applied to optimize the performance of the model under different input data. Finally, for the computation device, researchers often use GPU-integrated and memory-optimized approaches. For example, Saleh et al. in [37] improved the performance by adding GPUs, accelerating the training phase, and reducing the processing time during the detection phase.
### Table 2. Key characteristics of image classification techniques.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Algorithms</th>
<th>Software Requirements</th>
<th>Data Types</th>
<th>Processing Methods</th>
<th>Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[34] FBFN</td>
<td>Python and OpenCV</td>
<td>Captured images</td>
<td>Leaf gall detection (Boolean)</td>
<td>Image pre-processing, feature extraction, hyperparameters optimization, and real-time data processing</td>
<td>-</td>
</tr>
<tr>
<td>[35] CNN</td>
<td>Python, OpenCV, and TensorFlow</td>
<td>Captured images</td>
<td>Flooded detection (Boolean)</td>
<td>Image pre-processing</td>
<td>Intel Core i7 CPU</td>
</tr>
<tr>
<td>[36] Deep learning (MobileNetV2 and DenseNet169)</td>
<td>Python, OpenCV, and TensorFlow</td>
<td>Medical images</td>
<td>Medical diagnostic classes</td>
<td>Feature extraction, feature selection, and REST API services</td>
<td>-</td>
</tr>
<tr>
<td>[37] The hybrid of CNN and SVM</td>
<td>Python, OpenCV, and TensorFlow</td>
<td>Medical images</td>
<td>Lung cancer classes</td>
<td>Hyperparameters optimization</td>
<td>Intel Core i5 CPU with 16.00 GB of RAM and NVIDIA GeForce RTX 2060 GPU</td>
</tr>
<tr>
<td>[38] CNN</td>
<td>Python, OpenCV, and TensorFlow</td>
<td>Captured images</td>
<td>Fracture detection (Boolean)</td>
<td>Image pre-processing and feature extraction</td>
<td>Raspberry Pi 3</td>
</tr>
<tr>
<td>[39] CNN</td>
<td>Python, OpenCV, TensorFlow, and Keras</td>
<td>Captured images</td>
<td>Failure detection (Boolean)</td>
<td>Hyperparameters optimization</td>
<td>Intel Core i7 CPU with 8.00 GB of RAM</td>
</tr>
</tbody>
</table>

3.4. Object Detection

In this subsection, we provide an overview of integrating object detection techniques into IoT applications.

3.4.1. Introduction

Object detection is one of the AI techniques in the field of computer vision. While image classification focuses on categorizing entire images, object detection goes a step further by recognizing both the categories and the precise locations of specific objects within the images. This technique is often used as the first step to perform other tasks, including recognizing faces, estimating poses, and analyzing human activity.

In the context of IoT, object detection plays a critical role in various applications, such as autonomous video surveillance, smart cities, and manufacturing. Its integration into IoT devices facilitates real-time analysis of video sequences, which is essential for ensuring safety and efficiency in various environments. However, this integration brings new challenges in detecting moving objects and rapidly extracting their features. Addressing these challenges requires consideration of computational resources to efficiently manage the huge amount of IoT data, especially in use cases of intelligent surveillance systems.

3.4.2. Use Cases in IoT Applications and Characteristics Overview

In this section, we explore the papers that discuss the integration of object detection in the IoT domain. They presented how object detection algorithms are applied in various application scenarios, such as smart cities with drones, smart manufacturing, smart buildings, and smart laboratories. In the context of smart cities, object detection helps to improve...
urban management. It allows the detection and localization of various entities in urban environments, such as vehicles, pedestrians, and objects, for intelligent transportation, intelligent surveillance, and drone monitoring.

In Ref. [40], Zhou et al. presented a multi-target detection system for real-time surveillance using IoT sensors. By following this concept, Abdellatif et al. in [41] advanced the implementation of object detection by developing a server framework that uses a flying drone and data stream communication to detect multiple objects. They used the You Only Look Once (YOLO) algorithm for this purpose. To address a similar use case, Lee et al. in [42] proposed a cloud computing service for detecting multiple objects in drone-captured images by using the Faster Region-based CNN (Faster R-CNN) algorithm.

In another study, Meivel et al. in [43] presented a mask detection and social distance measurement system using drones. These literature studies highlight the effectiveness of integrating object detection and IoT devices, such as drones, to enhance urban surveillance and security management.

The concept of Industry 4.0 brings manufacturing processes to be monitored and controlled virtually. Recent research in this area has focused on the detection of intelligent small objects to build a digital twin environment. As an example, Yao et al. in [44] proposed a small object detection model in a manufacturing workshop use case using YOLOX.

In the context of smart laboratories, object detection contributes to efficient equipment monitoring and helps ensure compliance with safety protocols. Ali et al. in [45] introduced an IoT-based monitoring system for detecting compliance with personal protective equipment (PPE) guidelines to improve laboratory safety.

With the growth of communication protocol technology, object detection can be seamlessly performed on cloud servers in real-time scenarios. Baretto et al. in [46] demonstrated an application for person detections with CCTV cameras on cloud servers using WebRTC technology [47]. As the technology continues to evolve, the integration of real-time object detection on cloud servers opens up new possibilities for improved monitoring and decision-making.

Table 3 summarizes the characteristics of the object detection techniques used in the papers discussed in this subsection.

YOLO and Faster R-CNN are popular algorithms for their computation speed and accuracy in detecting objects in image data. The architecture of YOLO processes entire images in a single forward pass through the neural network to enable real-time object detection. On the other hand, Faster R-CNN uses a two-stage process, where the first stage proposes regions of interest, and the second stage classifies these regions and refines the bounding boxes to achieve better accuracy. This difference in concepts contributes to the different characteristics of the two algorithms, with YOLO being highly efficient for real-time processing, while Faster R-CNN focuses on improving accuracy by using a two-stage approach.

Implementing YOLO and Faster R-CNN typically involves using deep learning frameworks such as TensorFlow and PyTorch. These frameworks are commonly employed within Python environments that seamlessly integrate with CUDA for GPU acceleration and supporting libraries such as Keras and OpenCV. In order to accommodate the high demand for computing resources, a physical server is deployed along with GPUs. This hardware setup ensures more efficient processing and optimization of the algorithms.

According to use cases of IoT applications, object detection processes a captured image to obtain the detected objects in an image file. The detected objects are annotated with bounding boxes, class labels, and confidence scores. Similar to image classification, these techniques require a significant storage capacity for dataset storage. In addition, the data management approach is a critical aspect. The users need to carefully consider whether the results will be stored on temporary or permanent storage mechanisms.
Table 3. Key characteristics of object recognition techniques.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Algorithms</th>
<th>Software Requirements</th>
<th>Data Types</th>
<th>Processing Methods</th>
<th>Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[40]</td>
<td>Integration of YOLOv3 and Multitask CNN (MTCNN)</td>
<td>Python, TensorFlow, CUDA, and OpenCV</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>Real-time data processing</td>
</tr>
<tr>
<td>[41]</td>
<td>YOLOv5</td>
<td>Python with PyTorch, Apache Kafka, Apache Flink and CUDA</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>Real-time data processing</td>
</tr>
<tr>
<td>[42]</td>
<td>Faster R-CNN</td>
<td>Python with PyTorch and OpenCV</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>Real-time data processing</td>
</tr>
<tr>
<td>[43]</td>
<td>Faster R-CNN and YOLOv3</td>
<td>Python with PyTorch, TensorFlow, CUDA, Keras, and OpenCV</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>Image pre-processing</td>
</tr>
<tr>
<td>[44]</td>
<td>YOLOX</td>
<td>Python with PyTorch, CUDA, and OpenCV</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>Image enhancement and feature enhancement</td>
</tr>
<tr>
<td>[45]</td>
<td>YOLOv5</td>
<td>Python, TensorFlow, CUDA, and OpenCV</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>Real-time data processing</td>
</tr>
<tr>
<td>[46]</td>
<td>YOLOv3</td>
<td>Python, OpenCV, CUDA, and WebRTC</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>Real-time data processing</td>
</tr>
</tbody>
</table>

3.5. Text Spotting

This subsection presents an overview of the papers that focus on application use cases of text-spotting techniques in IoT systems.

3.5.1. Introduction

In AI, text spotting refers to the ability to detect and recognize texts within an image [12]. This technique is closely related to object detection, as it includes the recognition and localization of the text regions within images. However, text spotting extends beyond object detection by further extracting the textual content presented in the identified regions. The objective of this technique is to automate the extraction of meaningful information from images containing texts. This is particularly important for real-world applications such as mapping, document analysis, and augmented reality.

Text spotting has a significant role in IoT by enabling the extraction of valuable information in texts from visual data collected by sensors. This capability is particularly valuable for the tasks such as recognizing street signs, license plates, and product labels. By effectively identifying and extracting texts from images, text spotting enhances the intelligence of IoT systems, enabling them to derive meaningful insights and support diverse application use cases. Nevertheless, the implementation of text spotting involves numerous challenges. The wide variety of text appearances, including variations in sizes, lengths, widths, and orientations, poses a significant challenge to the development of effective text-spotting techniques.
3.5.2. Use Cases in IoT Applications and Characteristics Overview

The implementation of text spotting in IoT has been thoroughly explored in numerous literature studies. They illustrated the effectiveness of text-spotting algorithms in detecting and recognizing textual information from images for various application use cases.

The seamless integrations of IoT and text-spotting techniques in smart cities play a critical role in improving the efficiency, safety, and functionality of urban environments. The integrations are able to optimize parking management, ensure city safety, and improve public services.

In Ref. [48], Bassam et al. presented an IoT system designed to detect available parking spaces in urban areas. Following this concept, Wu et al. in [49] proposed a vehicle localization system to detect parking space numbers in real time. Both studies employed cameras placed in parking lots and implemented an Optical Character Recognition (OCR) model to extract the textual information about available parking spaces. This innovative approach highlights the effectiveness of integrating IoT and text-spotting techniques for efficient parking management in urban environments.

The works of Glasenapp et al. in [50] and Tham et al. in [51] proposed innovative systems to improve public safety. Their works focus on the development of IoT solutions for license plate recognitions from video streams. In another study, Ktari et al. in [52] presented an IoT-based system for monitoring water consumption. This helps staff to efficiently recognize water meters and improve smart city services, especially in the area of water consumption management.

This technique has also been applied to assist in recognizing medicine labels in smart healthcare applications. In Ref. [53], Abdullah et al. presented an IoT device for recognizing medicine names to manage the medicine consumption of elderly people by using the OCR model and Bidirectional LSTM (BiLSTM) algorithm. Chang et al. in [54] demonstrated the implementation of the OCR model in an intelligent medicine dispensing system to detect medicine bag information. These applications collectively highlighted the significant role of text recognition research in advancing healthcare applications.

In Ref. [55], Dilshad et al. applied an OCR model to determine the location of a UAV by analyzing visual data from its surroundings. Meanwhile, Promsuk et al. in [56] implemented a neural network to recognize numbers in seven-segment displays of industrial instruments. Extending this concept, Meng et al. in [57] developed early warning systems for cold chain logistics using text spotting to detect labels on goods. In addition, Cao et al. in [58] demonstrated the application of an OCR model in an infrastructure management scenario. They proposed systems for identifying irregular components on terminal blocks of electrical power equipment cabinets. These studies clearly demonstrate the versatility and practicality of text spotting in various domains.

Table 4 presents the characteristics overview of text-spotting techniques applied in the literature studies discussed in this paper.

Among the applied text-spotting algorithms, OCR models have proven their effectiveness in extracting textual information from vision-based data. First, these models identify regions within the image where text exists. Then, characters within each identified region are recognized and converted to machine-readable texts. Finally, they output the recognized texts and the image with the bounding boxes indicating the text locations.

Currently, a variety of OCR models are available to address different use cases, such as TesseractOCR, EasyOCR, PaddleOCR, and PaddlePaddle OCR (PP-OCR). These models offer different capabilities and should be selected based on characteristics such as performance, ease of use, and suitability for specific applications. In addition, OCR models can be effectively combined with other algorithms to improve the accuracy. In Ref. [53], Abdullah et al. demonstrated the integration of EasyOCR models with the BiLSTM algorithm to improve text recognition results. These integrations implement the adaptability of OCR models and the ability to benefit from complementary algorithms in specialized use cases.
Table 4. Key characteristics of text-spotting techniques.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Algorithms</th>
<th>Software Requirements</th>
<th>Data Types</th>
<th>Processing Methods</th>
<th>Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Input</td>
<td></td>
<td>Output</td>
<td></td>
</tr>
<tr>
<td>[48]</td>
<td>OCR model</td>
<td>LabView</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Image pre-processing, segmentation, and morphology filters</td>
</tr>
<tr>
<td>[49]</td>
<td>ABCNet OCR</td>
<td>Python, PyTorch, and OpenCV</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Object detection, anomaly filter module, and real-time data processing</td>
</tr>
<tr>
<td>[50]</td>
<td>OCR model by OpenALPR API</td>
<td>Python, OpenCV, and OpenALPR API</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Object Detection, Image pre-processing, feature extraction, segmentation, and real-time data processing</td>
</tr>
<tr>
<td>[51]</td>
<td>Tesseract OCR</td>
<td>Python, OpenCV, CUDA, and TensorFlow</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Object Detection, Image pre-processing, geofencing, segmentation, and real-time data processing</td>
</tr>
<tr>
<td>[52]</td>
<td>Tesseract OCR</td>
<td>Python, OpenCV, CUDA, and TensorFlow</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Object detection and real-time data processing</td>
</tr>
<tr>
<td>[53]</td>
<td>EasyOCR with BiLSTM</td>
<td>Python, OpenCV, and TensorFlow</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Image pre-processing and real-time data processing</td>
</tr>
<tr>
<td>[54]</td>
<td>PP-OCR</td>
<td>Python and OpenCV</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Image pre-processing and parameters optimization</td>
</tr>
<tr>
<td>[55]</td>
<td>EasyOCR</td>
<td>Python, OpenCV, and PyTorch</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Object detection, image pre-processing, and real-time data processing</td>
</tr>
<tr>
<td>[56]</td>
<td>Neural Network</td>
<td>Python</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Image pre-processing, feature extraction, and real-time data processing</td>
</tr>
<tr>
<td>[57]</td>
<td>OCR model</td>
<td>Python and OpenCV</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Video pre-processing and real-time data processing</td>
</tr>
<tr>
<td>[58]</td>
<td>Paddle OCR</td>
<td>Python, PyTorch, and OpenCV</td>
<td>Captured images</td>
<td>Recognized text</td>
<td>Object detection, feature extraction, and segmentation</td>
</tr>
</tbody>
</table>

Although existing OCR models are still the leading algorithm for text spotting, researchers have also explored alternative approaches. For instance, Promsuk et al. introduced a novel neural network algorithm in [56]. The algorithm was designed to recognize numbers in seven-segment displays.

Several aspects should be considered for effective implementations of text spotting. First, an appropriate software environment is required. Python environments are commonly used because of the key libraries such as OpenCV, CUDA, and TensorFlow. To achieve optimal performance, object recognition techniques often require a lot of computational resources. The addition of hardware accelerations is one solution to meet this demand. While the algorithms can be adapted for deployments on edge devices, it is noteworthy that additional GPUs will be required to achieve better performances, as shown in the studies of Glasenapp et al. in [50], Tham et al. in [51], Abdullah et al. in [53], Dilshad et al. in [55] and Cao et al. in [58].
On the data storage aspect, an effective data management system becomes critical. Due to the possibility of storing the input data, it is necessary to have a system for handling and organizing the image data. In addition, the implementation of additional processing methods in the algorithm is essential to improve accuracy and prevent errors in text recognition.

3.6. Auditory Perception

In this subsection, we provide an overview of the integration of auditory perception techniques into IoT through a review of the current literature studies that include application use cases.

3.6.1. Introduction

The motivation behind the development of auditory perception in AI is to mimic the human ability to understand and interpret sound. While computer vision enables machines to “see” by recognizing objects from visual information, auditory perception enables machines to “hear” and understand auditory information [33]. This capability expands AI applications to perform tasks that involve processing and extracting meaningful information from audio signals. Speech recognition, speaker recognition, sound classification, and environmental sound analysis are integral components of auditory perception. By applying these techniques, AI systems are able to extract and identify the auditory environment. This enables the development of more immersive and interactive applications.

In the IoT context, auditory perception is essential for extracting valuable information from the audio data gathered by IoT devices. In general, this technology allows the devices to analyze the sounds in their environments to identify certain patterns, events, and irregularities. This feature enhances the cognitive capabilities of IoT devices, where the IoT systems are able to trigger automated responses and actions based on the results of this auditory analysis.

3.6.2. Use Cases in IoT Applications and Characteristics Overview

In this section, we review the literature studies that presented how algorithms in auditory perception are applied in various application scenarios, such as smart cities [59–61], smart homes [62], and smart environments [63].

The application of auditory perception techniques in smart cities refers to the implementation of audio analysis algorithms for urban security. In this case, IoT devices were used to collect audio data in the urban environment. In Ref. [59], Balia et al. introduced an IoT system designed to enhance urban security by identifying potential threats through audio-based analysis on roads. They used the Short-Time Fourier Transform (STFT) algorithm to extract spectrograms as features of audio data. Following this approach, Yan et al. in [60] developed a sound detection system to identify traffic accidents in tunnels by using Deep Neural Network (DNN) for classifier and Mel-Frequency Cepstral Coefficients (MFCCs) for feature extraction. In another use case, Ciaburro et al. in [61] proposed a UAV presence detection system using sound analysis. This system addresses the increasing use of UAVs in urban areas.

The auditory perception has been employed to monitor environmental sounds in homes and buildings. Polo et al. in [62] combined MFCCs with CNN to detect activities of daily living based on sound cues. This research demonstrates the effectiveness of the integrated approach in identifying various activities of daily living through auditory analysis. Chhaglani et al. in [63] presented the application of auditory perception techniques in an intelligent environment. This study focused on predicting the airflow rate in a building by analyzing the sounds associated with air flowing through a duct in a building.

Table 5 summarizes the characteristics of the auditory perception techniques applied in the literature studies discussed in this subsection.
In the field of auditory perception, neural network-based algorithms are commonly used. In computer vision applications, features extracted from an image are typically used as the input. However, to process audio data, CNN analyzes spectrograms of audio data as features. Spectrograms refer to visual representations of the frequency variations of a sound signal over time. They consist of coefficients that capture the spectral characteristics of an audio signal. To obtain spectrograms of audio data, feature extraction algorithms such as MFCCs and STFT algorithms are integrated. The effectiveness of this integration was demonstrated in works by Balia et al. in [59] and Polo et al. in [62]. In addition, similar to the CNN algorithm, other approaches such as DNN, Fully Connected Neural Network (FCNN), and Bi-LSTM algorithms also require spectrograms of audio data as the input. The selection among these algorithms depends on several factors, such as the complexity of the auditory task, the size and nature of the dataset, and the desired level of abstraction for feature extraction.

Currently, auditory perception algorithms are implemented using deep learning frameworks along with Python programming environments. Through integration with supporting libraries such as Keras, the process of building, training, and deploying algorithms based on neural networks can be simplified. As a result, algorithms for auditory perception have lower computational requirements compared to computer vision applications. Researchers are potentially using edge computing devices such as the Raspberry Pi to implement the algorithms.

As we explore the characteristics of auditory perception, the implementation requires specific pre-processing steps before audio data can be analyzed. The steps include filtering, data transformation, and feature extraction, because algorithms cannot directly handle raw sensor data. They need transformed representations of audio data, such as spectrograms, to perform auditory perception effectively. Thus, we highlight that selecting feature extraction approaches offers the potential for significantly improving IoT applications in the auditory perception field.

3.7. Natural Language Processing

This subsection provides an overview of papers on implementations of NLP techniques in IoT systems considering application use cases.

3.7.1. Introduction

In the field of AI, NLP refers to the ability of computers to understand and interact with human language [66]. The objective of this technique is to enhance the efficiency
of communications between humans and computers. This involves computers not only understanding human language but also recognizing the contextual details involved in human communication. Through this process, computers are able to perform actions and generate responses that are associated with human language and communication patterns. NLP techniques are mainly divided into Natural Language Understanding (NLU) and Natural Language Generation (NLG). NLU is concerned with understanding and recognizing the linguistic aspects of natural language, while NLG is concerned with generating clear responses in the form of words or sentences to facilitate efficient communication. NLP integrates speech recognition, particularly in specific scenarios such as voice control systems, to extend its functionality and potential.

Currently, NLP has attracted widespread attention from researchers due to its capabilities. In the IoT context, NLP plays an important role in enabling users to control and interact with IoT systems using human language. The integration of NLP into IoT applications enables more instinctive and interactive connections between humans and computers. It facilitates voice control, text data analysis, and intelligent assistants in IoT environments.

3.7.2. Use Cases in IoT Applications and Characteristics Overview

The implementation of NLP in IoT applications has been thoroughly explored by several researchers. They demonstrated the effectiveness of NLP approaches in improving communications and interactions between humans and IoT systems.

In smart home applications, ongoing developments use NLP techniques to recognize and understand natural language commands spoken by humans accurately. In Ref. [67], Ismail et al. developed an IoT system that provides speech recognition to control home appliances. This system enables users, especially elderly patients and people with disabilities, to effortlessly control home appliances using voice commands. They adopted a robust combination of the SVM algorithm and Dynamic Time Warping (DTW) to accurately interpret commands from users’ voice audio. Following this work, Froiz et al. in [68] incorporated advanced technologies such as Wav2vec2 and Whisper models for speech recognition and the Bidirectional Encoder Representations from Transformers (BERT) model for NLP to enable seamless control of IoT devices through voice commands. Furthermore, Ali et al. in [69] combined the Google Speech API, NLP model, and logistic regression to enable the recognition and execution of both structured and unstructured voice commands.

In the context of smart buildings, Dweik et al. in [70] presented a significant step forward by introducing a voice control system designed to manage devices in buildings autonomously. These smart home and smart building use cases demonstrate the benefits of NLP techniques in diverse domains.

Table 6 presents the characteristics overview of the NLP techniques applied in the literature studies discussed in this paper.

The main purpose of NLP is to extract information from transcribed spoken sentences. To achieve this task, researchers often employ NLP models, as illustrated in the works of Ali et al. in [69] and Dweik et al. in [70]. Typically, an NLP model executes multiple steps, such as sentence segmentation, word tokenization, prediction of parts of speech, lemmatization, identification of stop words, definition of relationships between tokens, and recognition of named entities. Nevertheless, it is necessary to emphasize that different NLP models, such as BERT and BoW models, may contain different procedures. For instance, a BoW model involves steps including tokenization, stop word removal, token normalization, and vocabulary generation [71].

According to the use cases of IoT applications, the effective implementation of this technology requires the consideration of several key elements. Firstly, the software requirements should be considered. Currently, Python programming provides Natural Language Toolkit (NLTK) libraries to perform algorithms in NLP techniques. The NLTK libraries work together with TensorFlow to produce the desired results.
Table 6. Key characteristics of NLP techniques.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Algorithms</th>
<th>Software Requirements</th>
<th>Data Types</th>
<th>Processing Methods</th>
<th>Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[67]</td>
<td>SVM with a Dynamic Time Warping (DTW) algorithm</td>
<td>Python and Java</td>
<td>User’s speech audio</td>
<td>Command recognized (string)</td>
<td>Speech recognition and device control</td>
</tr>
<tr>
<td>[68]</td>
<td>Wav2vec2, Whisper, and BERT models</td>
<td>Python and TensorFlow</td>
<td>User’s speech audio</td>
<td>Command recognized (string)</td>
<td>Speech recognition, device control, model optimization</td>
</tr>
<tr>
<td>[69]</td>
<td>Google Speech API, NLP model, and Logistic Regression</td>
<td>Python, NLTK, and TensorFlow</td>
<td>User’s speech audio</td>
<td>Command recognized (string)</td>
<td>Speech recognition and device control</td>
</tr>
<tr>
<td>[70]</td>
<td>Google Speech API and NLP model using Bag-of-Words (BoW) approach</td>
<td>Python, NLTK, and TensorFlow</td>
<td>User’s speech audio</td>
<td>Command recognized (string)</td>
<td>Speaker verification, speech recognition, and device control.</td>
</tr>
</tbody>
</table>

As we explore the characteristics of NLP in IoT systems, implementing the related processes will significantly contribute to achieving optimal results. A speech recognition algorithm becomes an important process. For this purpose, researchers can use existing models or third-party platforms such as Whisper models and the Google Speech API. Then, a device control process is required to allow an IoT application to perform the control systems. Finally, the model optimization is able to enhance the trained models. For computational hardware, Table 6 shows that NLP models can be deployed and executed on either servers or edge devices such as Raspberry Pi.

3.8. Collaborative AI

In this subsection, we provide an overview of the practical use cases of the collaborative AI approach in IoT systems in existing literature studies.

3.8.1. Introduction

The development of collaborative AI in IoT systems is driven by the demand to address challenges associated with traditional cloud-based processing. In the traditional model, AI algorithms are executed on centralized cloud servers. As a result, cloud servers require high computational resources to cover all the AI processes. This raises many issues related to latency, communication, connectivity, and privacy concerns [12]. To address these challenges, collaborative AI aims to distribute computational tasks effectively by using both edge computing and cloud resources.

In the IoT context, collaborative AI extends data processing at the edge to reduce latency and bandwidth consumption. The purpose of this technique is not only to optimize resources but also to enable real-time or near-real-time analysis of IoT data. This is important in applications where rapid decision-making is required. This paradigm emphasizes the balanced and efficient use of local and cloud resources. Local processing involves the execution of lightweight AI algorithms for fast analysis, addressing the need for reduced latency. At the same time, the centralized cloud servers handle heavier data processing tasks, ensuring a comprehensive and robust approach to AI computation. The implementation of this collaborative approach indicates improvements towards an AI application in IoT systems, which is more adaptable and efficient.

3.8.2. Use Cases in IoT Applications and Characteristics Overview

In this section, we conducted a thorough review of the literature studies that explored applications of collaborative AI in IoT systems. They demonstrated the effectiveness of
collaborative AI in improving system performance across various application use cases. By leveraging the processing capabilities of both the edge and the cloud, collaborative AI has become a powerful paradigm for improving the efficiency and effectiveness of IoT systems.

In Ref. [72], Song et al. presented the implementation of collaborative AI in a monitoring system using Unmanned Aerial Vehicles (UAVs) as edge computing and cloud servers. The UAV used a faster R-CNN model to detect insulator strings. Then, the images of the insulator strings were transmitted to the cloud server. Finally, the cloud-based system identified defects in the insulator strings using the Up-Net model. This collaborative approach effectively optimizes the resources of the UAV and the cloud, which improves the efficiency of the monitoring system.

In Ref. [73], Li et al. introduced an edge/cloud collaborative architecture designed for efficient image recognition in the smart agriculture use case. The system employs a lightweight DNN model for object detection at the edge. If the object is not successfully recognized, the image is then transmitted to a server for further processing using more powerful DNNs.

In addition, this technique has been developed for further distributed AI architectures. In Ref. [74], Chen et al. introduced a distributed real-time object detection framework for video surveillance systems. This approach allows edge nodes to perform object detection using a YOLO model. Then, images of the detected objects are sent to the server to be used to generate a new model. Once the model is generated, it is sent back to the edge nodes. Following this concept, in [75], Loseto et al. proposed edge intelligence components that allow edge devices to perform data training using local data to generate models for early prediction. Using data collected from multiple edge devices, the cloud performs more advanced data training to generate highly accurate models and sends them to the edge devices. These scenarios illustrate the use of collaborative AI to continuously improve AI capabilities at the edge.

Table 7 presents the characteristics overview of the collaborative AI applied in the literature studies discussed in this paper.

Table 7. Key characteristics of collaborative AI techniques.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Algorithms</th>
<th>Software Requirements</th>
<th>Data Types</th>
<th>Processing Methods</th>
<th>Computations</th>
</tr>
</thead>
<tbody>
<tr>
<td>[72]</td>
<td>Faster RCNN (UAV), Up-Net Model (cloud)</td>
<td>Python, TensorFlow, Keras, OpenCV and Caffe</td>
<td>Captured images (UAV and cloud)</td>
<td>Image pre-processing, image rotation and segmentation on the cloud, image detection on UAVs, and real-time data processing</td>
<td>PC Server with NVIDIA GeForce RTX 2080 Ti (Server)</td>
</tr>
<tr>
<td>[73]</td>
<td>DNN</td>
<td>Python and PyTorch</td>
<td>Captured images</td>
<td>Image classes</td>
<td>-</td>
</tr>
<tr>
<td>[74]</td>
<td>YOLOv3</td>
<td>Python, PyTorch, and OpenCV</td>
<td>Captured images</td>
<td>Images with bounding box, class labels, and confidence scores</td>
<td>NVIDIA Jetson Xavier NX (Edge) and Intel Core i7 CPU with 32.00 GB of RAM and Nvidia RTX 2080 GPU (Cloud)</td>
</tr>
<tr>
<td>[72]</td>
<td>Multi-layer perceptron regressor</td>
<td>Python, Apache Kafka, TensorFlow and Keras</td>
<td>Time-series data</td>
<td>Predicted amount of silica</td>
<td>Raspberry Pi 4 Model B with 4.00 GB of RAM (Edge) and Intel Xeon CPU E5-2673 with 32.00 GB of RAM (Cloud)</td>
</tr>
</tbody>
</table>
The effective integration of collaborative AI into IoT applications involves several elements. First, the software requirement plays an important role in designing lightweight algorithms specifically for resource-constrained edge devices. Due to its integration capabilities with popular AI frameworks and supporting libraries, Python is often preferred for this application.

The integration architecture between the edge and the cloud is also necessary, as highlighted in the work of Chen et al. in [74] and Loseto et al. in [72]. The proposed distributed AI architecture allows local data training and model updates directly from the cloud server. To achieve this, well-designed communication methods with an emphasis on efficient protocols are required.

3.9. Integration of AI in IoT Platforms

This section presents a comparison between the SEMAR platform and the current state-of-the-art research on the integration of AI in IoT platforms to highlight our proposed ideas. We identified eight literature studies that have similar approaches to our design. In Ref. [76], Bu et al. proposed a platform for integrating AI with Industrial Internet of Things (IIoT) technologies to monitor and optimize manufacturing processes. In Ref. [77], Seshan et al. demonstrated the integration of the FIWARE framework [78] with AI capabilities, specifically for anomaly detection in wastewater monitoring applications. The FIWARE framework was chosen for device management and data collection processes. In another study presented in [79], Ramallo-Gonzalez et al. proposed an IoT platform for smart healthcare that leverages the FIWARE framework, big data technologies, and AI-based data analysis support. Both studies demonstrate the use of existing open-source frameworks to build an IoT platform service.

In Ref. [80], Raj et al. developed a framework called EdgeMLOps to deploy AI models directly at the edge. In Ref. [81], Li et al. introduced an Artificial IoT (AIoT) platform for smart agriculture applications that supports the addition of AI models on the edge device. In another study by Rong et al. in [82], the Sophon Edge platform was designed for collaborative computing between the cloud and edge devices. This platform enables the updating of AI models. In Ref. [83], Liang et al. developed an AIoT platform that facilitates the implementation of various AI models. Utilizing a micro-service architecture, each AI model runs concurrently. It allows the platform to support the combination of multiple AI models into a single dataflow process. Then, in [84], Stavropoulos et al. introduced the integration of machine learning into the IoT platform to create virtual sensors to replace physical sensors.

Table 8 shows the comparison of our proposed idea with the eight literature studies. Several parameters are considered, as follows:

- **IoT applications**: represents the use cases of the IoT applications that are implemented in each work.
- **Device management**: represents the ability to allow users to dynamically manage devices connected to the platform (Yes or No).
- **Model management**: represents the ability to manage multiple AI models, including adding and updating models (Yes or No).
- **Support various AI techniques**: indicates that the platform supports AI-driven capabilities across several techniques (Yes or No).
- **Edge device integration**: refers to the ability to deploy AI models to edge device systems (Yes or No).
- **Data types**: represents the specific types of data that can be handled by the platform.
Table 8. State-of-the-art comparison between the existing related studies and the proposed solution.

<table>
<thead>
<tr>
<th>Ref.</th>
<th>IoT Application</th>
<th>Device Management</th>
<th>Model Management</th>
<th>Support Various AI Techniques</th>
<th>Edge Devices Integration</th>
<th>Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>[76]</td>
<td>Smart Manufacturing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Common data types</td>
</tr>
<tr>
<td>[78]</td>
<td>Smart Environments</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Common data types</td>
</tr>
<tr>
<td>[79]</td>
<td>Smart Healthcare</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✗</td>
<td>Common data types</td>
</tr>
<tr>
<td>[80]</td>
<td>Various IoT applications</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>Common data types</td>
</tr>
<tr>
<td>[81]</td>
<td>Smart Agriculture</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Common data types and image</td>
</tr>
<tr>
<td>[82]</td>
<td>Various IoT Applications</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>Common data types and image</td>
</tr>
<tr>
<td>[83]</td>
<td>Various IoT Applications</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>Common data types, image and audio</td>
</tr>
<tr>
<td>[84]</td>
<td>Smart Homes and Environments</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>Common data types</td>
</tr>
<tr>
<td>Our Work</td>
<td>Various IoT Applications</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>Common data types, image and audio</td>
</tr>
</tbody>
</table>

Regarding the covered IoT application use cases, the works of Raj et al. in [80], Rong et al. in [82], and Liang et al. in [83] have the potential to be used in various IoT applications. This is similar to our SEMAR platform, which has been implemented and integrated into various IoT application use cases.

All of the mentioned works, including our IoT platform, provide device management capabilities for adding and removing connected IoT devices. These works also allow for receiving common data types from sensors, such as integer, float, string, boolean, date, and time. However, since IoT devices have a variety of data types, an IoT platform should be able to handle media file data types, including images and audio. The works by Li et al. in [81] and Rong et al. in [82] have demonstrated the ability to receive image frame data. Furthermore, the work of Liang et al. in [83] extends this capability by providing a system that can process both image and audio data. Our proposed design for the SEMAR platform addresses these concerns by facilitating the collection of both image and audio data types.

An IoT platform may require the ability to manage a large number of AI models and flexibly deploy different models across its processes. For this purpose, robust AI model management capabilities are used to enable users to handle AI models in order to achieve optimal results. The works of Raj et al. in [80], Li et al. in [81], Rong et al. in [82], and Liang et al. in [83] exemplify these capabilities in managing AI models within the flow of IoT data processes. The AI integration in the SEMAR platform follows this approach. We develop the functionalities that enable users to manage the versioning of AI models, store them in data storage, and deploy them in the flow of data processing.

Several works have emphasized the importance of AI techniques in IoT platforms by designing systems that support multiple AI capabilities. The works of Bu et al. in [76], Ramallo-Gonzalez et al. in [79], and Liang et al. in [83] have demonstrated that their proposed platform is able to accommodate different types of AI techniques. By supporting multiple AI techniques, an AIoT platform can effectively handle the variety of data generated by IoT devices. In line with this concept, the AI capabilities in the SEMAR platform are designed to facilitate the seamless integration of multiple AI techniques. With these capabilities, our platform is able to process different types of data.

In the field of IoT, the current approach involves the utilization of edge computing to perform data processing close to the device. The work by Raj et al. in [80], Li et al. in [81], and Rong et al. in [82] introduced a platform that facilitates the implementation of AI models on edge devices. This approach involves utilizing AI models that are designed to be lightweight in order to accommodate the limited computational resources available on the edge device. Following this idea, we also provide the capability to allow users to deploy their AI model to the edge device through the cloud environment.
4. Design of AI Techniques Integration in SEMAR

In this section, we present the design of integrating AI techniques in the SEMAR IoT server platform.

4.1. System Overview

The section presents the design for the seamless integration of AI techniques into SEMAR with considering the key characteristics and analysis of each AI technique described in Section 3. Figure 2 shows the design overview of integrating AI techniques in SEMAR. It consists of AI Model Management, Real-Time AI, and Batch AI services. The AI Model Management feature is responsible for managing AI models. It allows users to generate models from datasets stored in the sensor data storage and to upload existing models from other machines. The Real-Time AI feature performs data processing using AI in real-time scenarios. The Batch AI feature enables users to apply AI models to process existing data in the data storage. Furthermore, we show how to implement AI on edge computing devices by integrating the Edge Device Framework [10] with SEMAR.

![Figure 2. Design overview of AI techniques in SEMAR IoT application server platform.](image)

The limitation of our proposed approach arises from the fact that the training of the AI model occurs in an environment external to the SEMAR platform. This implies that our platform is not able to directly influence or control the performance of the model. As a result, there may be variations in the performance of the model. To mitigate this limitation, the AI Model Management feature allows seamless versioning of AI models for deployments, enabling users to effectively track and manage different AI models. Another limitation is related to the compatibility of AI models supported by the SEMAR platform. Our approach restricts the platform’s support to Python-based models only.

4.2. AI Model Management

The implementation of AI requires a systematic approach to defining objectives, collecting data, building models, and deploying them for real-world applications. As an IoT development tool, SEMAR should perform the functions that allow users to implement AI techniques in the applications easily. The current implementation of SEMAR provides the ability to collect sensor data. To help the integration of AI models, we design the
implementation of the AI Model Management feature in SEMAR. It allows users to manage, add, remove, and deploy models through the user interfaces of SEMAR.

In this design, SEMAR enables users to utilize models generated on other machines. First, users can grab sensor data stored in the data storage of SEMAR. This provides easy access to relevant data sets and simplifies the data preparation process. Then, users engage in the data training process to generate an AI model. After obtaining a trained model, users upload it through user interfaces of SEMAR. Next, users define the properties of the model, including its name, version, inputs, outputs, and the type of AI techniques employed. Inputs represent the list of data to be processed, while outputs represent the list of results obtained after AI processing. The system simplifies the deployment process by automatically specifying the data types of inputs and outputs for image classification, object detection, text spotting, and NLP techniques, although users are still able to customize these settings. For predictive analytics, users manually define input and output elements. Once a user registers a new model, the system stores it in the AI model data store through file access. This ensures that the model can be easily accessed for future use. Finally, to bring the models into applications, users are allowed to deploy the models for Real-Time AI or Batch AI processing.

According to the software requirement characteristics identified in the literature review, we select Python to build the features. Then, we build new features in user interfaces and REST API services of the SEMAR that focus on seamlessly guiding users in managing AI models.

4.3. Real-Time and Batch AI Processing

This section introduces two features, namely Real-Time AI and Batch AI services, for applying AI models in SEMAR. The Real-Time AI service is specifically designed for scenarios that require immediate AI processing. As shown in Figure 2, this service is seamlessly integrated with IoT cloud gateway and the data aggregator to perform the data stream processing.

In object detection, image classification, text spotting, audio recognition, and NLP techniques, the IoT cloud gateway receives image frames or audio data. The data aggregator verifies the format of the data in following standards such as JPEG and WAV. The verified data are sent to the Real-Time AI service through established communication protocols for further processing by AI models. For this purpose, we utilize the Kafka communication protocols. Once the system receives the results, it stores them in the data storage. In the predictive analytics scenario, the data aggregator forwards the data to the data filter before reaching the Real-Time AI services, which perform data pre-processing. These services predict the future events of data using an AI model and the historical data collected from the data storage.

The Batch AI service provides a service designed for AI processing on existing data saved in the data storage. This service is particularly useful for dealing with large data sets that cannot be processed in real-time scenarios. Unlike the Real-Time AI services that process data automatically, users should first select specific data that will be processed through the user interface of SEMAR. Then, users select a suitable AI model from the storage. The system systematically applies the AI processes to all of the selected data, collects the results, and saves them back to the data storage. The implementation of Real-Time AI and Batch AI services in SEMAR enhances system flexibility by handling both immediate processing tasks and post-processing analysis requirements.

4.4. AI Implementation in Edge Devices

Previously, we introduced the edge device framework as one feature of SEMAR [10]. This framework allows users to optimize the functionality of IoT devices by allowing them to be configured remotely from the server. It provides the functions for connecting to multiple IoT sensors, processing data in standard formats, and using the collected data through multiple output components. These functions are organized into the input, processing, and
output components. As the insights from collaborative AI techniques in the literature review, we design a strategic extension by implementing AI models on edge devices. Figure 3 illustrates the design overview of the AI implementation in the edge device framework. We add the Real-Time AI function within the processing components to facilitate the immediate processing of sensor data using AI models.

Figure 3. Design overview of AI model implementation in edge device framework.

For the installation process, users need to select the appropriate AI model and define field output to store the result. Then, web services download the AI models from the SEMAR server through the HTTP-POST communication. After the download process is completed, the main service on the framework reads the AI model and runs the service to collect sensor data. Finally, once the Real-Time AI function receives sensor data from input components or the filtering function, it processes the data using this AI model. The generated results are forwarded to the output components or serve as the input to the ruling function. This interconnected workflow enables dynamic and responsive integration of AI models at the edge.

5. Use Cases of Integration AI and IoT Applications in SEMAR

In this section, we discuss IoT application use cases that are integrated with the AI-driven capabilities in SEMAR. Each use case presents the application overview, requirements, the AI algorithms being employed, and how SEMAR can be utilized to support them. They include the drone-based building monitoring system, the Indoor Navigation System Using Unity and Smartphone (INSUS), and the air-conditioning guidance system (AC-Guide).

5.1. Drone-Based Building Monitoring System

As the first application use case, we discuss the implementation of AI within a drone-based surveillance system. In the building inspection use case, drones emerge as powerful tools for rapid surveillance systems with the ability to navigate autonomously based on missions defined by the programs or to be operated under human control. Their versatility is advantageous for expanding the coverage areas of monitoring systems. By seamlessly integrating drones with AI technologies and an IoT system, drones are able to automatically detect defects in buildings such as cracks.

In the previous research, we introduced the concept of a drone-based building crack detection system [85]. This system consists of flying drones and edge devices connected to the SEMAR server. The flying drones capture the image data around the building, while the edge devices are placed in a specific area of the building to control the drones and facilitate the transfer of image data to the SEMAR server through a communication protocol.
service. Once the SEMAR server receives the image data, it processes the data using object detection models to detect cracks in the images. Finally, the user interface visualizes the crack detection results.

Several services are required to build this application, such as the communication protocol, object detection, data storage, and user interface services. The communication protocol service should be responsible for transmitting image data. Then, the object detection service provides the capability to identify the crack objects in the image.

To achieve an efficient implementation of this use case, we will use Confluent Kafka as the communication protocol service between SEMAR and edge devices. As mentioned in Section 3, the YOLO algorithm has gained popularity for its efficiency in real-time scenarios. Thus, the AI Model Training service generates a YOLO model using the open-source crack datasets available on the Internet in [86]. We then deploy the YOLO model to the Real-Time AI service to perform crack detections. This deployment step places the crack-detecting capabilities derived from the YOLO algorithm into the flow of data streams in SEMAR. Once the system detects crack objects in an image, it stores them in the data storage. Finally, the user interface visualizes the detected cracks for users by accessing the REST API service.

Figure 4 shows the preliminary implementation results of crack detections using the YOLOv7 algorithm. The crack images were captured using the Ryze Tello drone. The model was built using 3717 crack images. Then, we utilized an additional 200 images to validate the model with the Intersection over Union (IoU) threshold of 50%. The validation results show that the average model accuracy is up to 73% and the mean average precision (mAP) is up to 82%.

(a) Crack image captured by drone.  (b) Crack detection result.

Figure 4. Drone-based crack detection result.

5.2. Indoor Navigation System Using Unity and Smartphone

For the second application use case, we introduce the integration of the SEMAR server with the Indoor Navigation System Using Unity and Smartphone (INSUS). In this scenario, the primary requirement for the navigation system is to accurately determine the current position of the user. While Global Positioning System (GPS) is a commonly used technology in outdoor scenarios, its accuracy decreases in the indoor environment due to the difficulty of obtaining a stable signal. To address this limitation, indoor navigation systems use alternative approaches to determine the current position of the user.

In Ref. [87], the INSUS application introduced Augmented Reality (AR) technology for user guidance. This system utilizes the Simultaneous Localization and Mapping (SLAM) technique, which combines a 3D map of the building with a camera and inertial measurement unit (IMU) sensors of the smartphone. This system works by identifying the users’ initial position using the QR code technology. It then guides users from their starting position to their destination using the 3D arrows in the real view. However, a drawback of this method is the requirement of the QR code sheet in every room of the target building to achieve accurate initial positions. In order to solve this challenge, the current implementation of INSUS proposes a novel approach to identify the user’s position by recognizing room-
specific information that appears at the door, including floor levels, the number of rooms, and resident names.

Text spotting becomes a viable solution for this use case. It can extract room-specific information from the image captured by the smartphone through the INSUS system. In contrast to the first use case, where the application does not need any feedback from the SEMAR server after sending the image frame, this use case requires real-time communication that is able to send the image to the SEMAR server and receive the corresponding room information in real time.

For this purpose, the combination of HTTP-POST for the communication protocol and an OCR model for the text recognition can be a solution to achieve the efficient implementation of this use case. The process consists of several steps. First, we build the OCR model within AI Model Training Service. Next, the generated model is deployed to the Real-Time AI service. When the system receives input from the INSUS application, it employs the OCR model to recognize the room information in the image. Finally, the result is sent back to INSUS through the REST API services in JSON format to complete the real-time communication loop.

Figure 5 demonstrates the preliminary results of the text recognition using the PaddleOCR model. The results show that the room label was successfully identified from the image captured by the camera in the INSUS application.

5.3. Air-Conditioning Guidance System

As the last use case of IoT applications, we introduce the integration of AI in the air conditioning guidance system, namely AC-Guide. Recently, the development of smart homes has gained significant popularity and attention from both academia and industry. It focuses on technologies that enhance the convenience, efficiency, and comfort of users’ daily lives. In this use case, the environmental condition data are essential for the system to recognize the comfort or discomfort state of the specific area. Therefore, a smart home system utilizes IoT sensors to measure environmental conditions in order to fulfill this important requirement.

Previously, we integrated AC-Guide, an IoT system, to guide the use of AC by identifying the discomfort state of the room, with the SEMAR IoT application platform [4]. This system consists of a Raspberry Pi with attached sensors to measure the temperature and humidity of the room and a camera to collect images of the AC control panel. It accesses the OpenWeatherMap API [88] to collect the standard outdoor weather data. Then, it calculates the indoor and outdoor discomfort index (DI) to identify whether the current state conditions are comfort or discomfort. By considering the on/off state of AC, indoor DI and outdoor DI, it sends the proper guidance to users to turn on or turn off the AC through a message. Finally, it sends data to the server through MQTT communication and stores the data in the local log file.

Our implementation results demonstrate that the system has successfully identified the current state of DI. However, this system can be improved by using advanced analysis techniques to identify data patterns for predicting future events. This improvement is useful in preventing discomfort states.
One AI technique that can be applied to this application is predictive analytics. As mentioned in Section 3, predictive analytics allows the system to estimate future values and use the results to support decision-making. Therefore, we propose the application of predictive analytics techniques to perform early guidance systems for the use of AC by considering the prediction data.

For this purpose, we use the LSTM algorithm to generate AI models for predicting indoor temperature and humidity using collected sensor data. This process is performed by the AI Model Training service. After the models are generated, the system deploys them to the Real-Time AI service. Once it receives the sensor data, it predicts the future value by using the AI models and the stored data from the data storage. Then, it stores the data in the data storage. Finally, we build the plug-in function to calculate the DI state based on the predicted sensor values.

In this paper, we present the preliminary system to predict the humidity and temperature data collected by the AC-Guide system. We trained a model based on the LSTM algorithm for 100 epochs. The data sets contain 132 data for training and 68 data for validation. The validation results indicate that the model can predict the values with a root mean square error (RMSE) of 0.08 for humidity and 0.04 for temperature. Figure 6 shows the results of the predicted values compared to the real value.

![Figure 6. Predictive analytics results using LSTM algorithm in AC-Guide system.](image)

6. Conclusions

This paper presents the study of AI techniques and their implementation use cases for designing AI integrations in the SEMAR IoT application server platform. This study provides a comprehensive review of current research on implementing AI techniques in IoT applications, covering predictive analytics, image classification, object detection, text spotting, auditory perception, NLP, and collaborative AI. Software requirements, input/output (I/O) data types, processing methods, and computations are parameters used to identify the characteristics of each technique.

Based on the findings, we have designed the implementation for the seamless integration of AI techniques into SEMAR by incorporating new features such as AI Model Training, Real-Time AI, and Batch AI services. The paper also outlines the design of AI implementations in edge devices by utilizing the SEMAR and the edge device framework. Through the integration design with three IoT application use cases, the advantages of the proposed system are described.

In future works, we will continue to implement AI technologies in SEMAR to complete the proposed design in Figure 2, and will proceed with evaluations through integrations with other IoT application use cases.
Author Contributions: Conceptualization, Y.Y.F.P.; Methodology, Y.Y.F.P., E.D.F. and S.F.; Software, Y.Y.F.P., E.D.F. and S.F.; Writing—Original Draft Preparation, Y.Y.F.P.; Writing—Review and Editing, N.F.; Validation, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Dataset used in this work is available based on the request.

Acknowledgments: The authors thank the reviewers for their thorough reading and helpful comments.

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

References
2. Stankovic, J.A. Research Directions for the Internet of Things. IEEE Internet Things J. 2014, 1, 3–9. [CrossRef]
12. Zhang, J.; Tao, D. Empowering Things with Intelligence: A Survey of the Progress, Challenges, and Opportunities in Artificial Intelligence of Things. IEEE Internet Things J. 2021, 8, 7789–7817. [CrossRef]


54. Chang, J.; Ong, H.; Wang, T.; Chen, H.-H. A Fully Automated Intelligent Medicine Dispensary System Based on AIoT. IEEE Internet Things J. 2022, 9, 23954–23966. [CrossRef]


67. Ismail, A.; Abdlerazek, S.; El-Henawy, I.M. Development of Smart Healthcare System Based on Speech Recognition Using Support Vector Machine and Dynamic Time Warping. Sustainability 2020, 12, 4203. [CrossRef]

68. Garzón, A.; Abdlerazek, S.; El-Henawy, I.M. Development of Smart Healthcare System Based on Speech Recognition Using Support Vector Machine and Dynamic Time Warping. Sustainability 2020, 12, 4203. [CrossRef]


72. Song, C.; Xu, W.; Han, G.; Zeng, P.; Wang, Z.; Yu, S. A Cloud Edge Collaborative Intelligence Method of Insulator String Defect Detection for Power IoT. IEEE Internet Things J. 2021, 8, 7510–7520. [CrossRef]


Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.