2023 IEEE 29th International Conference on Parallel and Distributed Systems (ICPADS) | 979-8-3503-3071-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICPADS60453.2023.00336

# A Fog-based Smart Agriculture System to Detect Animal Intrusion

Jinpeng Miao[1], Dasari Rajasekhar[2], Shivakant Mishra[1], Sanjeet Kumar Nayak[2], Ramanarayan Yadav[3]

[1]University of Colorado Boulder, USA, [2]IIITDM Kancheepuram, India, [3]IITRAM, Ahmedabad, India

***Abstract*—Smart agriculture is one of the most promising areas where IoT-enabled technologies have the potential to substantially improve the quality and quantity of the crops and reduce the operational cost. However, building a smart agriculture system presents several challenges, including high latency and bandwidth consumption associated with cloud computing, Internet disconnections in rural areas, and the need to keep costs low for farmers. To address these issues, this paper proposes a fog-based smart agriculture infrastructure with edge computing and LoRa communication. We address the top concern of farmers - animals intruding - by proposing a solution that detects animal intrusion using low-cost PIR sensors, cameras, and computer vision and predicts animal locations using a novel algorithm. Our system can detect animals before the intrusion, identify them, predict their future locations, and alert farmers promptly. The paper proposes three sensor layouts, and experiments confirm the system's effectiveness and lower cost compared to state-of-the-art systems.**

***Index Terms*—smart agriculture, animal intrusion detection, LoRa, fog computing**[1]

## I. Introduction

Smart agriculture applies modern information technology, integrates big data, mobile Internet, cloud computing and IoT technologies relying on various sensing nodes to achieve precise tracking, monitoring, automating and analyzing operations. At present, cloud-based infrastructures are being utilized to support various smart agriculture applications and data processing. Data from smart sensors in the agricultural field is transmitted to the cloud over the Internet, and then stored and processed in the cloud for decision making. While cloud-based infrastructures certainly offer enormous processing power and storage capacity, there are two key limitations that need to be addressed when used in the context of smart agriculture [1]: (i) Sensor data transmitted over the Internet requires continuous Internet connectivity, consumes high bandwidth and incurs delays, which is not feasible in rural areas where the internet connectivity is unstable. (ii) Since IoT devices must transmit large volumes of data to the cloud for storage and processing, the energy of battery-powered IoT devices is quickly drained. These limitations make cloud-based infrastructure ill-suited for smart agriculture. To address these limitations, we propose a LoRa-enabled, fog-based smart agriculture infrastructure that distributes computation workload to Raspberry Pi intelligently so as to reduce the quantity of data transferred to the server and enables the delivery of latency-sensitive services in real-time.

[1]We use the term 'fog computing' and 'edge computing' interchangeably throughout the paper, same as 'fog server' and 'edge server'.

After conducting a survey with the farmers to understand the key issues they are facing that could be addressed by smart agriculture, animal intrusion in the field becomes the most concerning one. Farms are usually located in rural areas, close to nature. This makes animal intrusion a major issue for farm owners who must deal with the mess and damage these animals can cause. Compared to some other smart services such as smart irrigation, crop quality monitoring and pest extermination, animal intrusion detection is more difficult because of its uncertainty, uncontrollability, unpredictability. Animals may eat crops and stroll around the field at any time, resulting in a significant production loss. This necessitates more time costs to recover from the damage as well as greater financial security to cover the costs associated with damages.

In this paper, we propose an end-to-end, LoRa (**Lo**ng **Ra**nge)-enabled, fog-based infrastructure for smart agriculture along with a new strategy to detect animal intrusion with PIR sensors and a rotating camera. We are committed to helping farmers detect and locate animal invasions as quickly as possible. To achieve it, a direct solution for field monitoring would be to rely solely on cameras without using PIR sensors. However, this approach necessitates either high-resolution cameras or a greater number of cameras to cover the same area, leading to higher costs and increased energy consumption compared to PIR sensors. Furthermore, if only cameras are employed, they must be fixed in one direction, potentially diminishing the success rate of identification when an animal appears at the boundary of the camera's field of view. Employing a rotating camera would ensure that the animal is captured in the center of the image, significantly enhancing the success rate of animal identification.

The paper is organized as follows: We begin by demonstrating how LoRa's low-power, low-bandwidth, and long-range capabilities transform rural agricultural lands into a smart agriculture system. Next, we detail the design and implementation of a microservice-based edge server, delivering crucial, time-sensitive services to farmers in disconnected Internet environments. For optimized animal intrusion detection, we investigate various sensor placement strategies and devise an algorithm to locate invasive animals and predict future locations. Lastly, we assess our system's performance, comparing it with current state-of-the-art frameworks in terms of cost, latency, and distance.

2690-5965/23/$31.00 ©2023 IEEE
DOI 10.1109/ICPADS60453.2023.00336

This paper makes the following contributions:

- Adoption of LoRa protocol effectively addresses the limitations of intermittent Internet connectivity and high latency of cloud-based infrastructure.
- A microservice-based architecture at the edge to enable latency-sensitive services delivered just in time.
- Propose three sensor layouts and an algorithm that accurately predicts the future locations of animals.
- Comprehensive analysis and comparison of the layouts through experiments.
- Rigorous evaluation and discussion on the accuracy of the algorithm and the practicality of the system.

## II. Related Work

With the rise of smart agriculture, numerous systems have emerged. Yet, most grapple with safety hazards, high costs and resource demands, dependency on internet connectivity, or poor performance. Amid the plethora of systems in literature, our focus lies on the latest intelligent agricultural and animal intrusion detection methods.

Devaraj *et.al.* suggest using traditional electric fence, which shock animals that cross the boundary [2]. While effective and easy to install, it requires a consistent and substantial power supply, along with regular maintenance. In contrast, our system remains unaffected during power outages and, importantly, does not pose risks to animals or people. In [6], authors analyze why traditional methods such as electric fencing are futile in some scenarios and high cost.

Cameras and computer vision are effective at identifying intruding animals. Some researchers [3]–[5] use deep learning algorithms to recognize animals captured by the camera at regular intervals. However, fixed interval detection wastes resources and may miss some animals. Yadahalli *et.al.* [6] instead send images to a TFT display and use a flash light for better night images, which are more expensive and consume more power. Compared to computer vision, it is also harder for humans to accurately identify animals in images where they make up a small percentage. Instead of capturing images, Thomas *et.al.* [29], [30] analyze videos, which is challenging to meet latency requirements and necessitates significantly higher computational power.

Cloud-based infrastructures [10]–[12] are popular in smart agriculture for their powerful computing capabilities. In these systems, data is transmitted over the Internet to the cloud, where the data is stored and processed for decision making. However, these systems rely on Internet connectivity, which may be unavailable in rural areas, and can result in high latency due to data transmission to the cloud.

The systems proposed from 2017 to 2022 [14]–[20] that use infrared sensors lack specifics on sensor placement and algorithms. In [3], it fails to achieve better performance. The works presented in [2], [7]–[9] can not support large service coverage at a low cost.

In comparison, our proposed system excels at accurately detecting and predicting animal locations while minimizing power consumption and transmission latency, and eliminating dependence on internet connectivity by leveraging LoRa communication protocol.

## III. Proposed System

### A. System Architecture

The architecture of the proposed microservice-based fog enabled infrastructure for smart agriculture is shown in Figure 1. It consists of two layers: sensing layer and fog computing layer, which are linked by cross-layer upstream and downstream communication for data and control information flows [28].

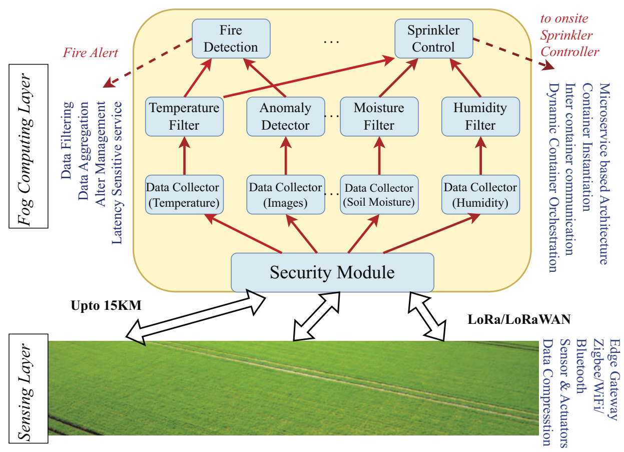


Fig. 1: Proposed system architecture.

The *sensing layer* is comprised of the sensors and actuators deployed across the agricultural field to periodically sense the physical parameters of interest such as air temperature, air humidity, soil temperature and moisture at various depths, wind speed and rainfall. To address the challenge of poor Internet connectivity, we have adopted a LoRa and LoRaWAN enabled communication system due to their support for low power, wide area networking designed to wirelessly connect limited energy operated IoT devices to an edge server at a distance of 1-2 km. The *fog layer* is comprised of one or more servers, and provides an administrative control of the entire IoT infrastructure of the agricultural field. It addresses the limitations of intermittent Internet connectivity, high latency and high network bandwidth consumption of cloud-based infrastructure. The fog nodes execute latency sensitive services, such as animal intrusion detection. To facilitate a flexible architecture that may utilize existing container-based support for various ML/AI services, we have structured the fog layer as a microservice architecture. In this architecture, an application is composed from a collection of loosely-coupled microservices, where each microservice is fine-grained and the associated protocols are lightweight.

*B. Animal Intrusion Detection*

In view of the serious problems caused by animal intrusion to farmers, our goal is to automatically detect animal intrusions, identify animals, repel animals with automatic actuations like beep sounds and laser lights, and inform the farmer(s) in a timely manner about the intrusion. This work is performed using two types of sensors: a PIR sensor for detecting any motion in its field of view and an all-day camera sensor attached to the Raspberry Pi for capturing images that will be processed to identify animals. To meet the low-latency requirement, the scheduling mechanism and the prediction algorithm are implemented in the fog layer, while the object detection is done on the Raspberry Pi. This is because the low bandwidth of LoRa cannot support the transmission of large-sized images. As shown in Figure 2, there are three microservices: 1) The Security module passes the sensor data it receives from authenticated sensors to the appropriate Prediction container; 2) The Prediction container runs localization and prediction algorithms on this data as well as recorded data, and then sends the predicted position at a future time to the camera; 3) The Notification module notifies the farmer via messages once animals are detected.

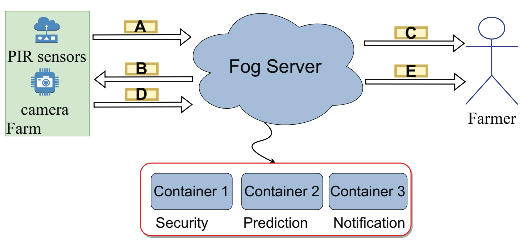


Fig. 2: System architecture for animal intrusion detection.

Corresponding to the process marked with capital letters in Figure 2, the steps are described as follows:

**A**: Animal movement is detected by PIR sensors and the data is transmitted to the server using LoRa.

**B**: A container on the edge server predicts the location of the animal at a future time based on input from multiple sensors and sends this location to the Raspberry Pi that operates a camera on the field.

**C**: The edge server sends a "possible animal invasion" alert to the farmer.

**D**: The Raspberry Pi instructs the camera to rotate in the direction of the predicted position and take a picture. The Raspberry Pi then runs an animal detection algorithm on this image and sends the results to the edge server.

**E**: If an animal was identified, actuations are activated immediately to repel it and the edge server sends a reliable alert to the farmer.

*C. Sensor Layouts*

For animal intrusion detection, a pivotal consideration is the strategic placement of sensors in the field. The objective is to achieve maximum coverage, ensuring thorough data collection, improving prediction accuracy, and accounting for diverse scenarios, all while minimizing the number of sensors. Given the ambiguity surrounding the optimal placement strategy, we put forth and experiment with three plausible layouts tailored for a square-shaped field. The optimal layout certainly depends on the shape of the fields, but the idea remains the same.

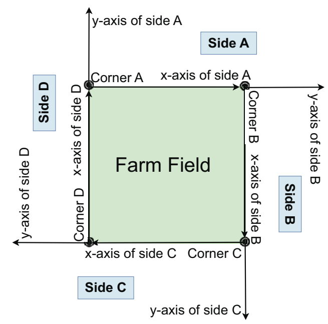


Fig. 3: Virtual coordinate systems built upon the farm.

We establish four virtual coordinate systems based on the four directions on the farmland (as shown in Figure 3), encompassing the x-axis and y-axis, with the corners serving as the origins of these systems. Our localization and prediction algorithms rely on this coordinate framework. This setup also aims to simplify the process of farmers locating invasive animals. In other words, when sensors detect an animal, the animal's location is regarded as a point (marked with $R1$, $R2$, ... in Figures 4 to 6) rather than a range, which is convenient for us to design algorithms to predict animals' position. In order to describe the specific location of the animal to the farmer, we define four corners and four sides. Below we describe the three sensor layouts in detail.

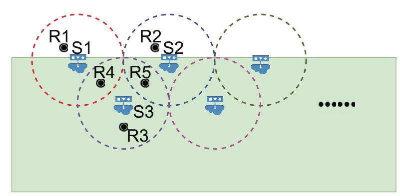


Fig. 4: Layout A: Vertical Placement

*1) Layout A - Vertical:* We place all sensors at a height of $d$ meters above the ground and project them vertically downward, with the coverage area of each sensor being a circle of diameter $h$. This produces a coverage area consisting of many circles. As shown in Figure 4, we put two rows of interlocking and overlapping sensors at the boundary of the field. This not only increases the coverage, but the overlapping areas allow for relatively fine-grained segmentation of the area to improve the accuracy of localization and prediction. The increase in budget associated with an additional row of sensors is well worth it compared to more accurately catching animal intrusions and thus preventing damage to the farm. But the shortcoming of this layout is that the farthest detectable location is too close to the farm boundary, only $h/2$, which leads to a greater chance of animal damage to the farm.

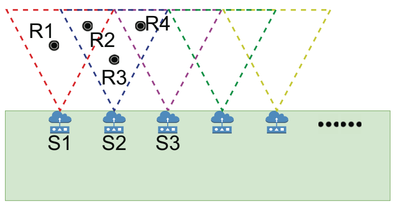


Fig. 5: Layout B: Horizontal Placement

*2) Layout B - Horizontal:* In contrast to Layout A, all sensors are placed on the ground and horizontally projecting towards the farm's exterior, with each sensor covering an isosceles triangle with $h$ as the base and $d$ as the height, resulting in a coverage area of many triangles. As shown in Figure 5, we put one row of interlocking and overlapping sensors at the boundary. This also has the same advantages as Layout A, i.e., increased coverage and fine-grained area segmentation to improve localization and prediction accuracy. Moreover, it overcomes the limitations of Layout A by extending the farthest detectable distance, thus improving protection.

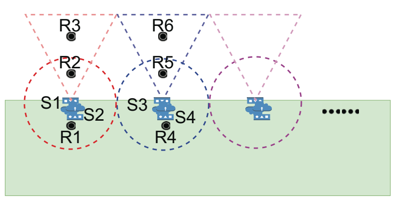


Fig. 6: Layout C: Hybrid Placement

*3) Layout C - Hybrid:* With vertical and horizontal placement, it was natural to explore a hybrid placement. We still place two rows of sensors, one vertically along the boundary and the other projected horizontally outward at the same location, as shown in Figure 6. This layout provides good coverage, fine-grained area segmentation, and a far-reaching detectable location. However, uncovered middle areas can lead to inaccurate or even outrageous predictions. Additionally, this layout has a calculated minimal coverage area.

In addition to these three layouts, we also considered other layouts that required fewer sensors. However, they were ruled out due to their limited coverage, which limits their prediction accuracy, and their short sensing distance to the boundary, which make them unsuitable for fast-moving animals.

### *D. Proposed Algorithm*

We propose and deploy an algorithm (as shown in Algorithm 1) on the fog server to predict the future location of intrusive animals based on the previous readings returned by the sensors. Whenever a sensor detects an animal, it sends the data back to the container running the algorithm on the fog server in the format of "#side-#sensor-timestamp". Each set of data received by the container is combined with the previous record to make a prediction. The "set of data" here may have two scenarios: one is the data read back from a non-overlapping coverage area; the other is the animal is in the overlapping area covered by multiple sensors. In this case there will be multiple sensors with similar timestamps to send back data and the server needs to make the final prediction after receiving data from all these sensors. We use a "tolerance time difference" to define this similar timestamp. In addition, we need to define another threshold as the minimum time interval between animal intrusions, i.e., if the server does not receive new data within the amount of time, the next data received is considered to be a new animal intrusion. It should be determined by the number of animals within the vicinity and their appearance frequency around the field.

We define a mapping of sensor numbers and position coordinates in the algorithm. The server first converts the sensor number in the received data into a coordinate and combines the previous set of coordinates to calculate the distance and direction, and then to calculate the average speed of the animal's movement with the timestamp difference. With the direction and speed, the next position of the animal can be predicted under the assumption that the animal will move in the same direction with the same speed for a short period of time. This period is the sum of the time it takes for the sensor to return data, the time it takes for the algorithm to make the prediction, the time it takes for the instruction to be passed from the server to the Raspberry Pi, and the time it takes for the camera to rotate to point to the predicted position.

The direction and speed of animal movement are not stable, but the constant detection and updating of position information by the sensors, the fast transmission of LoRa, and the high speed calculation of the system can make the predicted deviation be calibrated quickly and continuously. The field of view of the camera can also provide a certain degree of tolerance. Taken together, our proposed algorithm is expected to effectively and accurately locate and predict the location of animals. Next, we evaluate and verify the adequacy of the algorithm through experiments.

## IV. Evaluation

All experiments presented in this paper using the parameters shown in Table I. To evaluate our work, we constructed an end-to-end LoRa communication system, deployed the three sensor layouts proposed in Section III-C, and gathered sensing data by moving along various trajectories. We then implemented our proposed algorithm to analyze the collected data. The tolerance for time difference is set at 0.1 second, and the time threshold is established at 120 seconds.

**Algorithm 1** Algorithm to predict animal locations

**Input:** side number *side*, sensor number *sensor*, and timestamp $t_{cur}$
**Output:** A coordinate of predicted animal position $\{x_{predict}, y_{predict}\}$

$x_{prev}$ ▷ x value of previous location
$y_{prev}$ ▷ y value of previous location
$t_{prev}$ ▷ Timestamp of previous reading
$time\_threshold$ ▷ Interval to refresh the collected data
$time\_tolerance$ ▷ Interval to define similar timestamp
$latency$ ▷ Time required from detection to camera pointing to the predicted position
$pos\_mapping \leftarrow \{sensor : \{x : y\}\}$
**function** PREDICT($side$, $sensor$, $t_{cur}$)
    $x_{cur} \leftarrow pos\_mapping[sensor][x]$
    $y_{cur} \leftarrow pos\_mapping[sensor][y]$
    **if** $t_{cur} - t_{prev} > timing\_threshold$ **then**
        *Do nothing*
    **else if** $t_{cur} - t_{prev} < time\_tolerance$ **then**
        *Wait until all data received*
    **else**
        $dist \leftarrow \sqrt{(x_{cur} - x_{prev})^2 + (y_{cur} - y_{prev})^2}$
        $speed \leftarrow dist/(t_{cur} - t_{prev})$
        $\theta \leftarrow \text{arctangent}(y_{cur} - y_{prev}, x_{cur} - x_{prev})$
        $d \leftarrow latency * speed$
        $x_{predict} \leftarrow x_{cur} + d * math.cos(\theta)$
        $y_{predict} \leftarrow y_{cur} + d * math.sin(\theta)$
        **return** $x_{predict}, y_{predict}$
    **end if**
    $x_{prev} \leftarrow x_{cur}$
    $y_{prev} \leftarrow y_{cur}$
    $t_{prev} \leftarrow t_{cur}$
**end function**
**while** true **do**
    PREDICT($side$, $sensor$, $t_{cur}$)
**end while**

### A. Experiments and Results

*1) Lora Transmission:* We build an end node which consists of PIR sensors, one Arduino Mega microcontroller, LoRa Hat with Antenna. LoRa hat is built using LoRa SX1276 IC. To experiment the scheduling capability of fog node, we connected three end nodes with one LoRa enabled gateway (Raspberry Pi with PG1302 LoRaWAN Concentrator) which is located 3 km away from the field as shown in Figure 7. The end nodes are scheduled in a round robin fashion by fog node to avoid interference of data during simultaneous communication by the three end nodes.

*2) Sensor Placement:* To detect animals in a 25 by 25 meters field, we use PIR sensors. As described in Section III-C, the layouts of the sensors were accurately deployed. The sensors were fixed to a strip and placed around the perimeter of the field to ensure complete coverage. We used an Arduino ATMega2560 along with a LoRa hat using LoRa SX1276 IC powered by a lithium-ion battery to send data to the Gateway for processing and decision-making.

We changed different speeds, directions, and trajectories to simulate 18 different movements (M1-M18 in Figure 8) of animal (cow) to evaluate the accuracy and effectiveness of our algorithm. For each of these movements, our system

TABLE I: System Configuration

| Component Name | Specifications |
|---|---|
| Arduino Mega | 256 KB Flash Memory, 8KB SRAM, 4KB EEPROM, 16 MHz Clock Speed |
| Raspberry Pi | Quad core Cortex-A72 (ARM v8) 64-bit 8GB RAM, 1.5GHz Clock Speed |
| PIR Sensor | Detection range $d$ is 7 meters; Detection distance $h$ is 5 meters |
| Camera | Resolution 2592×1944, Optical Size 6.35mm, Focal Length 2.25mm, FOV 130°(D) 105°(H) |
| Edge Server | 3.1 GHz Dual-Core Intel Core i5, 8 GB RAM, 256GB Disk |

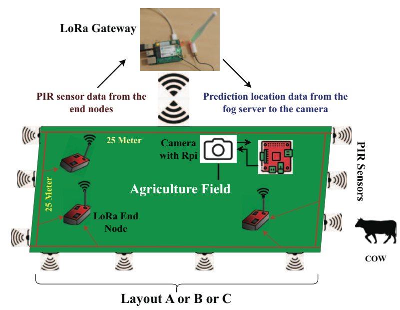


Fig. 7: Experimental setup architecture

sensed and transmitted PIR sensor values to the edge server for multiple locations depending on which sensors detected movements. For example, Table II shows the location values received for Movement M2. This location data collected from 18 different movements forms the ground truth for our evaluation.

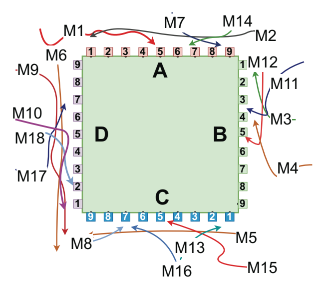


Fig. 8: Movement Trajectories

*3) Position Prediction:* We use a PC as fog server in our experiments. It was placed 10 meters above the ground to increase the transmission speed with end nodes [28]. The prediction algorithm runs continuously waiting for data. To evaluate our algorithm, we make use of our ground truth data, wherein the container extracts three sets of location data from a movement, uses the first two sets of data to predict the location for the time corresponding to the third set of data, and then compares this predicted location with the actual location to assess the accuracy of the prediction.

One measure of accuracy we use is the *distance offset*, which is the distance between the predicted location and the actual location. We measured distance offsets for all movements for which we have at least three location values. For movements such as M2 (Table II) for which we have more than three location values, we measured distance offset for each triplet of location values resulting in 10 distance offsets measured. Figure 9 shows the average distance offset of each movement.

TABLE II: M2 location values: side – coordinate(x, y)

| | Layout A | Layout B | Layout C |
|---|---|---|---|
| 1 | A - (20, 1.5) | A - (20, 3.5) | A - (20, 5.5) |
| 2 | A - (15, 1.5) | A - (15, 3.5) | A - (15, 5.5) |
| 3 | A - (15, 1.5) | A - (15, 3.5) | A - (15, 5.5) |
| 4 | A - (10, 1.5) | A - (10, 3.5) | A - (10, 5.5) |
| 5 | A - (5, 1.5) | A - (5, 3.5) | A - (5, 5.5) |

As we can see, the average distance offset is relatively low (less than 5 meters) for most movements and layouts. We observe that Layout B shows relatively small distance offsets in most of the movement tests, although in M1, it has a higher offset in prediction than the other two layouts. However, in M8, Layout B does not have sufficient readings for the algorithm to make predictions due to the presence of some blind triangles near the boundary where the sensor cannot detect the animal once it moves there. Layout C produces the largest distance offsets in most of the tests due to the presence of many blind areas inside the coverage area, which prevented the animal from being detected quickly and continuously, resulting in more inaccurate predictions. Layout A performs moderately and without data loss, which is due to its continuous and extensive coverage area. Based on our experiments and the analysis in Section 2, we recommend Layout B as the optimal sensor deployment method, which can be readily applied to rectangular fields of varying dimensions.

TABLE III: Animal Detection Experiment Results

| CNN Pre-Trained Model | Latency(s) |
|---|---|
| VGG16 | 2.27 |
| ResNet50 | 3.75 |
| ResNet50V2 | 3.34 |
| InceptionV3 | 4.75 |
| MobileNet | 1.64 |
| MobileNetV2 | 2.74 |
| EfficientNetB0 | 5.07 |

*4) Animal Detection:* To identify the intrusive animals, we connect a camera to a Raspberry Pi to take pictures of the animals (area where the predicted location is) and identify them using computer vision algorithms. The specifications of the Raspberry Pi are shown in Table I. We experimented with several popular CNN pre-trained models to test their speed of processing images. We first train these models on top of a computer and then imported the trained models into the Raspberry Pi. These event-driven models are continuously running on the Raspberry Pi waiting for images to be taken.

The average detection time (based on 20 runs for each image) is summarized in Table III. As we can see, these models take 2 to 5 seconds to identify an animal, with MobileNet having the best performance with an average time of 1.64 seconds.

### *B. Prediction Accuracy*

The goal of predicting location is to be able to rotate the camera in a direction where the animal is expected to be. Using the distance offset statistics, we can determine the accuracy of the algorithm by combining the distance between the predicted animal position and the camera placement. As illustrated in Figure 10, the predicted location is represented by point $P$, and the camera (point $C$) is positioned within the boundary to point towards the predicted position. The camera has a horizontal field of view of 105 degrees, as shown in Table I, and the red shading indicates the current range that the camera can cover. If the actual animal position falls within the red shading, represented by point $Q$, the prediction is considered accurate. Conversely, if the actual animal position falls outside the red shading, represented by point $R$, the prediction is considered incorrect. Since we only have the distance between the predicted location and the actual location, without knowing their relative positions with respect to the camera, $Q$ could be any place on the red circle which is centered at point $P$. We make the assumption that the angle formed by the edge $PQ$ and the edge $CP$ at point $P$ is a right angle, so that angle $\beta$ is the maximum value. In this way, the accuracy of the prediction is the most conservative value.

Based on this validation method, we calculate the prediction accuracy of the three layouts at different distances between the predicted animal position and the camera placement, as shown in the Table IV. The table shows again that layout B is the best layout solution. For layout B, a placement distance of 5 meters can achieve a very accurate prediction. The farther the distance, the wider the coverage, and the higher the accuracy. Nevertheless, we must also consider that increasing the distance results in lower image quality of the animal, which makes animal identification more difficult. We will discuss this further in Section V.

TABLE IV: Camera placement and prediction accuracy.

| Distance between camera and animal(m) | Accuracy(%) | | |
|---|---|---|---|
| | Layout A | Layout B | Layout C |
| 5 | 66.67% | 94.44% | 38.89% |
| 10 | 100% | 94.44% | 94.44% |
| 15 | 100% | 94.44% | 100% |

### *C. System Performance and Cost*

*1) System Performance:* Based on our experiments, we estimate the total time required to achieve animal intrusion detection with the current system configuration (as shown in Table I), which is summarized in Table V. It takes approximately 7.11 to 8.61 seconds for the farmer to receive clear intrusion information, including the predicted location and type

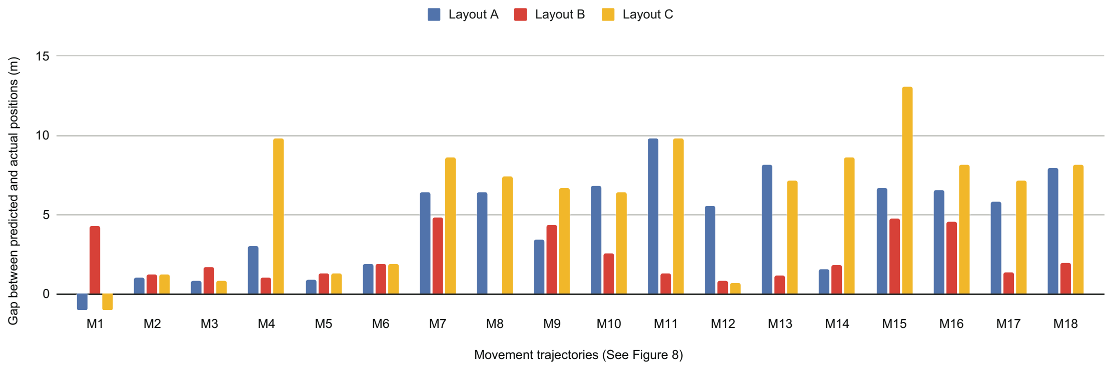


Fig. 9: Average distance offset between predicted and actual positions for different types of movements in the three layouts.

of the animal. Kindly note that this duration encompasses both both location prediction and animal identification. This isn't solely the time it takes to locate the animal when detected by PIR sensors. Furthermore, the farmer will receive consecutive messages to fine-tune the animal's location until the threat is eliminated.

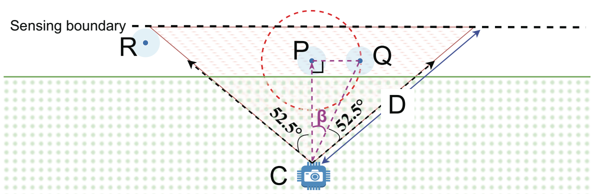


Fig. 10: Camera placement

TABLE V: System Latency

| Step | Latency(s) |
|---|---|
| Transmission of 3 sets of data via Lora | 1 |
| Prediction with proposed algorithm | 0.01 |
| Instruction sent to camera via LoRa | 1 |
| Camera rotation, image capture and processing | $4 \sim 5$ |
| Results sent back to fog server via LoRa | 1 |
| Alert sent to farmer via LTE | $0.1 \sim 0.6$ [27] |
| **In total** | $\mathbf{7.11 \sim 8.61}$ |

*2) System Cost:* To illustrate the expenses incurred during our experiment in the $25 \times 25$ $m^2$ field, we have compiled a detailed cost analysis of all the devices used, which is presented in Table VI. With a total cost of $285.36 (US dollars), our system is a cost-effective solution for monitoring animal intrusions. As the size of the farm increases, the cost will inevitably rise, but the advantage is that additional expensive equipment, such as fog server, is not required.

## V. Discussion

### A. Imaging Quality and Animal Identification

In Section IV-B, we assessed the accuracy of prediction based on the presence or absence of animals in the picture taken. However, to fully evaluate the performance of the system, we must also consider the ability to identify the pictured animals. If the animal image is not clear in the picture, it will be difficult to identify. This depends on two factors: the number of pixels that the animal occupies in the image and the pixel requirements of animal recognition algorithms listed in Table VIII [26]. Assuming the animal size is approximately 2 meters long and 1.5 meters high (cow), we calculate the number of pixels occupied by the animal at different distances between the camera and the animal based on the camera parameters (as shown in Table I) and present the results in Table VII.

TABLE VI: System Cost

| Device Name | Cost(US Dollars) |
|---|---|
| Arduino Shield for LoRa | $6.12/each x 3 = $18.36 |
| Raspberry Pi 4 | $95 |
| GPS Concentrator | $120 |
| PIR Sensor | $0.75/each x 36 = $27 |
| Camera | $25 |
| **In Total** | **$285.36** |

By comparing these two tables, we can confirm that these widely used models listed can successfully identify animals when the distance between the animal and the camera is 40 meters, and we can still use the very effective GoogLeNet and SqueezeNet1_1 when the distance is 80 meters. Therefore, to ensure both a large camera coverage to improve the quality of animal imaging and the tolerance of prediction errors, we need to control the camera placement and maximum rotation angle. Specifically, we must ensure that the maximum distance between the camera and the intersection of the coverage boundary and the sensing boundary (i.e., $D$ in Figure 10) does not exceed 80 meters, with 40 meters being the optimal distance. This allows us to adopt MobileNet which can achieve the best performance shown in Table III.

### B. Other Options for Communication

In addition to the combination of LoRa and cheap camera sensors, alternatives could be to use 4G/5G cellular security cameras or wired solutions. However, 4G/5G cellular security cameras are much more expensive and require a license,

TABLE VII: Pixels an animal occupies at different distances

| Distance(m) | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 |
|---|---|---|---|---|---|---|---|---|
| Pixels | 199x151 | 99x76 | 66x50 | 50x38 | 40x30 | 33x25 | 28x22 | 25x19 |

TABLE VIII: Minimum pixel requirement for CNN models

| Model | GoogLeNet | SqueezeNet1_1 | DenseNet201 | VGG16/19 | MobileNet |
|---|---|---|---|---|---|
| Minimum Pixels | 15x15 | 17x17 | 29x29 | 32x32 | 32x32 |

making the total cost much higher. Wired solutions are also undesirable since cables would be inconvenient for cultivation and prone to corrosion, making maintenance costly.

### C. Instances when our work does not succeed

Based on the results presented in Table IV, it is evident that when an animal is very close to the camera, the success rate decreases. Naturally, fast-moving animals pose a challenge for our system, as they can quickly move out of the camera's range or get too close, making it difficult to predict their next location. This increases the likelihood of the camera capturing the wrong area. Nevertheless, it's worth noting that fast-moving animals are typically not found in close proximity to farm fields. Lastly, small animals that the camera cannot capture clearly are also challenging to detect.

## VI. Conclusion

This paper presents a fog-based smart agriculture system that overcomes high latency and internet connectivity issues by combining fog computing with LoRa communication and Raspberry Pi workload distribution. The system detects and predicts animal intrusion using low-cost PIR sensors and cameras, and proposes various sensor layouts and an algorithm. The paper also experimentally compares the layouts and verifies the effectiveness and accuracy of the algorithm.

## Acknowledgment

This research is supported in part by grants from TIH-IoT (TIH-IoT/2022-07/ IC /NSF/SL/ NIUC-2022-04/005) and NSF (13012641).

## References

[1] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge Computing: Vision and Challenges," in IEEE Internet of Things Journal, Oct. 2016.

[2] M. Aiswarya, E. Banu, JS Jenisha Gifta and S. Allwin Devaraj, "An Intelligent Agricultural Intrusion Detection and Irrigation Control System Using GSM", 2018.

[3] Saieshwar Radhakrishnan and R Ramanathan, "A Support Vector Machine with Gabor Features for Intrusion Detection in Agriculture Fields", 8th International Conference on Advances in Computing and Communication (ICACC-2018) Procedia Computer Science, 2018.

[4] Balakrishna, K., et al. "Application of IOT and machine learning in crop protection against animal intrusion." Global Transitions Proceedings 2.2 (2021).

[5] R. S. Sabeenian, N. Deivanai, and B. Mythili, "Wild animals intrusion detection using deep learning techniques," Int. J. Pharm. Res., 2020.

[6] S. Yadahalli, A. Parmar and A. Deshpande, "Smart Intrusion Detection System for Crop Protection by using Arduino," 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020.

[7] Geetha D, Monisha S P, Oviya J, Sonia G, "Human and Animal Movement Detection in Agricultural Fields," SSRG International Journal of Computer Science and Engineering, 2019.

[8] M. Begum H., D. A. Janeera and A. Kumar. A.G, "Internet of Things based Wild Animal Infringement Identification, Diversion and Alert System," 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 2020.

[9] et. al., S. S. . "Self-Intrusion Detection System for Protection of Agricultural Fields Against Wild Animals". International Journal of Modern Agriculture, Apr. 2021.

[10] W. H. S. Antônio, M. D. Silva, R. S. Miani, J. R. Souza, "A proposal of an animal detection system using machine learning," Appl. Artif. Intell., 2019.

[11] Balakrishna, K., Mohammed, F., Ullas, C. R., Hema, C. M., & Sonakshi, S. K. (2021). Application of IOT and machine learning in crop protection against animal intrusion. Global Transitions Proceedings.

[12] B. K. N and S. K, "Farm Vigilance: Smart IoT System for Farmland Monitoring and Animal Intrusion Detection using Neural Network," 2021 Asian Conference on Innovation in Technology (ASIANCON), PUNE, India, 2021.

[13] Patil, Hardiki and Namrata Ansari. "Automated Wild-Animal Intrusion Detection and Repellent System Using Artificial Intelligence of Things." SSRN Electronic Journal (2021).

[14] R. Nikhil, B. S. Anisha and R. Kumar P., "Real-Time Monitoring of Agricultural Land with Crop Prediction and Animal Intrusion Prevention using Internet of Things and Machine Learning at Edge," 2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), Bangalore, India, 2020.

[15] U. Venkateshkumar, A. V, D. K. V and E. B, "Farm Intrusion Detection System using IoT," 2022 International Conference on Electronics and Renewable Systems (ICEARS), Tuticorin, India, 2022.

[16] S. Jeevitha and S. V. Kumar, "A Study on Sensor Based Animal Intrusion Alert System Using Image Processing Techniques," 2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, India, 2019.

[17] Priya Sharma, Sirisha C K, Soumya Gururaj, & Padmavathi C. (2020). "Neural Network Based Image Classification for Animal Intrusion Detection System," International Journal of Progressive Research in Science and Engineering.

[18] S. Giordano, I. Seitanidis, M. Ojo, D. Adami and F. Vignoli, "IoT solutions for crop protection against wild animal attacks," 2018 IEEE International Conference on Environmental Engineering (EE), Milan, Italy, 2018.

[19] B, Vikhram & B, Revathi & R, Shanmugapriya & S, Sowmiya & G, Pragadeeswaran. (2017). Animal Detection System in Farm Areas. IJARCCE.

[20] Mohandass, S., Sridevi, S., & Sathyabama, R. (2021). "Animal health monitoring and intrusion detection system based on LORAWAN," Turkish Journal of Computer and Mathematics Education.

[21] R. Chen, S. Li and Z. Li, "From Monolith to Microservices: A Dataflow-Driven Approach," 2017 24th Asia-Pacific Software Engineering Conference (APSEC), Nanjing, China, 2017.

[22] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in IEEE Cloud Computing, Sept. 2014.

[23] Lora Alliance. 2017. "LoRaWAN 1.1 Specification," Oct. 2017. https://lora-alliance.org/resource_hub/lorawan-specification-v1-1.

[24] Lora Alliance. https://www.lora-alliance.org.

[25] Alliance, LoRa. "A technical overview of LoRa and LoRaWAN." White Paper, November 20 (2015).

[26] "MODELS AND PRE-TRAINED WEIGHTS," PyTorch. https://pytorch.org/vision/0.12/models.html#models-and-pre-trained-weights (accessed 2017).

[27] Grigorik, Ilya. "High Performance Browser Networking: What every web developer should know about networking and web performance," O'Reilly Media, Inc., 2013.

[28] S. Mishra, S. Nayak and R. Yadav. "An Energy Efficient LoRa-based Multi-Sensor IoT Network for Smart Agriculture System," IEEE Topical Conference on Wireless Sensors and Sensor Networks, January 2023 (WisNet 2023).

[29] A. K. Thomas, P. Poovizhi, M. Saravanan and K. Tharageswari, "Animal Intrusion Detection using Deep Learning for Agricultural Fields," 2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2023.

[30] Kiruthika, S., et al. "Smart Agriculture Land Crop Protection Intrusion Detection Using Artificial Intelligence." E3S Web of Conferences. EDP Sciences, 2023.