ADVANCED FIRE DETECTION SYSTEM: INTEGRATING SENSOR AND ARTIFICIAL INTELLIGENCE

1st Huynh Phuc Thinh
ID: 21139054
HCMC University of Technology and Education

2nd Phan Duy Hoang
ID: 21139019
HCMC University of Technology and Education

3rd Le Truong Thinh
ID: 21139055
HCMC University of Technology and Education

May 14, 2024

Abstract

This project delves into the field of object detection by starting with a fundamental understanding and progressing to application in a specific project: a fire detection system. This project not only utilizes sensors to detect smoke but also employs YOLOv5 technology to accurately determine the presence of fire, addressing the issue of high energy consumption associated with continuous imaging and monitoring. Specifically, when sensors detect signs of fire, the camera is activated to take a photo, and YOLOv5 evaluates the image to determine if there is an actual fire. To succeed, the project needs to focus on two main challenges: 1. Developing the appropriate hardware specifications for the project. 2. Carefully fine-tuning the parameters in the fire detection mechanism to ensure the desired outcomes.

1. INTRODUCTION

Accidental fire outbreak threatens people's life and property safety, and it is of great significance to study fire detection and alarm early. The detection range of traditional fire detectors is limited, and the conventional detection algorithm has the problems of low precision and long detection time. Aiming at these problems, a fire detection method based on YOLOv5 is proposed in this project. The experimental results show that the algorithm has fast detection speed and high detection accuracy. It can accurately detect not only large-scale flame but also small-scale flame in the early stage of fire.

1.1. Aim

The goal of this project are two fold:

- Successful design of the hardware for the system has been achieved: this involves the choice of circuit boards, sensors, servo motors, etc. The ultimate goal is to create a complete and robust system
- The main goal of the project is to design a two-stage fire alarm system. While fire detection using sensors can be effective, it still has limitations. Improving and developing the fire alarm system by integrating YOLOv5 (a fire detection algorithm) to enhance accuracy and reduce energy consumption is one of the most important goals that the project aims to achieve.

1.2. Motivation

- Hardware design for fire alarm system. The hardware component of the system plays an important role in detecting the initial signs of a fire. It requires a combination of different components to ensure quick, timely detection.
- Improve fire alarm system by detecting fire through YoloV5. We can prevent the sensor from mistaking a fake fire for only smoke. We aim to address the limitation by developing a novel fire detection model that is computationally efficient and capable of accurately detecting fires, including small and large fire region, in real-world settings. Considering the limitations of scalar and traditional sensors based fire detection methods, we explore an efficient object detection model for real-time fire scene analysis in surveillance networks. The proposed method shows a significant improvement in early fire detection with higher performance and reduced false alarming rate.

Overall, hardware design for detecting signs of fire involves selecting appropriate sensors, combined with fire detection using YoloV5 to achieve an optimal system.

1.3. Problem Statement

The specific challenges that need to be addressed in creating the fire detection system will be presented in detail below.First, the image quality obtained from the ESP32-CAM is often unsatisfactory, especially in conditions of low light or dense smoke, leading to compromised image analysis and decreased accuracy in fire detection. This is a major limitation because image quality is a crucial factor for YOLOv5 to analyze accurately and effectively.The second challenge is accessing suitable hardware resources to train the YOLOv5 model. Although YOLOv5 is one of the most effective object recognition models available today, it requires substantial computational power and extensive training data to develop a highly accurate model. The shortage of computers with powerful configurations and appropriate graphics cards has become a significant barrier, slowing down the model development and improvement process

2. THEORETICAL

2.1. Deep Neuron Network

DNNs are generally feedforward networks, in which data flows continuously from the input layer to the output layer, and the connections between the layers are one-way in a forward direction. A fundamental component of a DNN is an ANN inspired by biological neurons found in the human brain. A DNN calculates and transmits the sum of the data received on its input side. Each hidden layer applies an activation function before producing an output, which is essential to facilitate learning and approximation due to the non-linearity of real-world issues. The softmax activation function, given in equation 4, is used in the output layer. The binary crossentropy loss function is utilized to measure the loss of a sample by calculating it using equation 1.

$$\begin{aligned} \text{Loss} = -\frac{1}{\text{outputsize}} \sum_{i=1}^{\text{outputsize}} y_i * \log \hat{y}_i \\ + (1 - y_i) * \log(1 - \hat{y}_i) \end{aligned} \tag{1}$$

where $\hat{y}_i$ is the i^{th} scalar value in the model output, y_i is the corresponding target value, and *outputsize* is the number of scalar values in the model output.

In a DNN model, there are no feedback links. The three major components of the Feed-forward Neural Network are the input and output layers, as well as the hidden layer, which may include many hidden units. The layers are each weighed separately. These units are responsible for initiating the activation operations of the units from the previous layer [40].

A simple DNN can be depicted as Figure 2. Each hidden layer with a relu activation function can be as the following mathematical equation:

$$g(x) = f(x^T w + b) \tag{2}$$

$$g(x) = g_i(g_{i-1}(\dots(g_1(x)))) \tag{3}$$

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{4}$$

$$\text{ReLu} = \max(0, x) \tag{5}$$

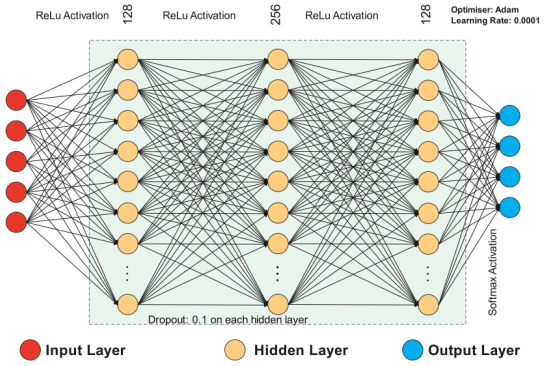

Fig. 1: DNN model architecture.

2.2. YoLoV5

YOLOv5 is an object-detection architecture known for its high performance and efficiency, demonstrating rapid, precise, and adaptable features. It has broad applications in various computer vision tasks and provides powerful tools and technical support for real-time object detection. Its unique design and optimization make YOLOv5 perform exceptionally well in handling large-scale data with less computational resources. This advanced architecture enables efficient processing of vast amounts of information without compromising performance. YOLOv5's strength lies in its ability to adapt to various target detection tasks. This versatility and flexibility empowers researchers and developers to explore new possibilities and drive innovation in the field.

YOLOv5 utilizes a single-stage detection technique in which the whole image is fed into the network, allowing it to directly determine the locations and classifications of objects within the images. This end-to-end detection approach allows YOLOv5 to quickly and precisely identify multiple objects, making it well-suited for real-time tasks. The complete network architecture of YOLOv5 is depicted in the Figure 3. YOLOv5 mainly consists of two parts: the backbone and the neck. In the backbone, the cross-stage partial network (CSP, or C3 as shown in the figure) structure uses two 1×1 convolutions for the transformation of input features. The first CSP is

used for feature extraction in the backbone section. The second CSP is used for feature fusion in the neck section. In addition to the CSP module, the backbone also has a spatial pyramid pooling fast (SPPF) module, whose function is to extract the global information of the detection target. The neck of YOLOv5 uses the path aggregation network (PANet) structure to aggregate the features. The neck is primarily used for generating feature pyramids in order to enhance the model's detection of objects at different scales, thereby enabling the model to recognize the same object at different sizes and scales.

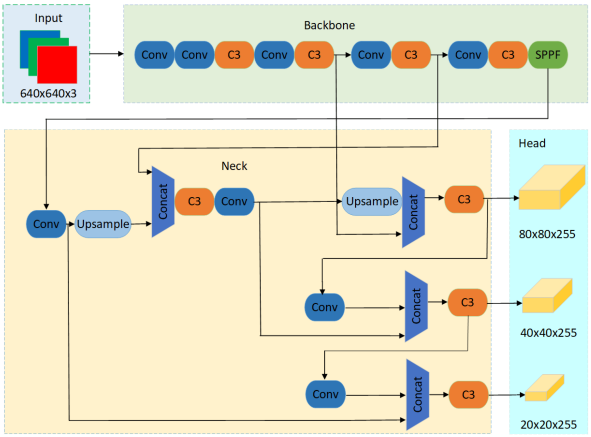

Fig. 2: YOLOv5 Model Architecture.

3. SYSTEM DESIGN

3.1. Hardware Components

In this project, we utilize a variety of hardware components to construct a comprehensive system capable of performing specific tasks. The selection of each component is crucial for the overall functionality and efficiency of the system. Below is a detailed description of the main hardware used:

3.1.1. Servo Motors

We use two SG90 180-degree servo motors, which are compact yet powerful, ideal for precision control tasks. These servos enable the ESP32-CAM module to rotate and capture images from various angles, enhancing the system's ability to detect smoke and fire by providing comprehensive visual coverage. This capability allows for dynamic adjustment of the camera's view in response to detected hazards, improving the effectiveness of our emergency response system.

3.1.2. Smoke Sensor

The smoke sensor in our project is used to detect the presence of smoke particles in the environment, which can be indicative of a fire. This sensor is crucial for safety in environments where there is a potential fire risk.

3.1.3. Buzzer

A buzzer is incorporated into the system for auditory signals. It is used to alert users in case the smoke sensor detects smoke, providing a critical warning that can help prevent accidents and ensure safety. In our project, the buzzer is activated by signals from the server, which processes the fire-related data captured by the ESP32-CAM. Upon confirming a fire hazard, the server instructs the buzzer to sound an alarm, effectively alerting occupants to potential danger and facilitating timely responses.

3.1.4. ESP32-CAM

The ESP32-CAM module combines an ESP32-S chip and a camera module, making it highly suitable for applications involving wireless communication and image capturing. This component is essential for tasks that require remote monitoring and image processing.

Each component was selected based on its performance characteristics and compatibility with the other elements of the system. Integrating these components allows us to build a robust and effective system tailored to the project's requirements.

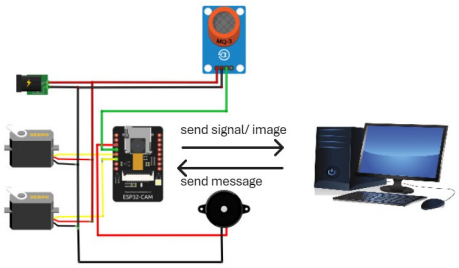

Fig. 3: System Overview.

3.2. Code

In this section, we provide an overview of our system's architecture and functionality. Our system is designed to operate between a client and server, utilizing an ESP32-CAM as the client device and HTTP as the communication protocol.

Our system operates through a client-server architecture, with an ESP32-CAM serving as the client device and HTTP as the communication protocol. The client's primary function, involves smoke detection

using an MQ2 sensor. Upon detecting smoke, the client sends a signal to the server, indicating the need for fire detection. Simultaneously, the client captures images of the room, which are used in fire detection on server, detailed in Section 3.1.1.

On the server side, the captured images are processed to detect the presence of fire within the room. To detect fire, model YOLOv5 is used for detecting because its precision. Once the detection is complete, the server communicates the outcome back to the client, signaling either the presence or absence of fire, detailed in Section 3.1.2.

The communication between the client and server is facilitated entirely through HTTP. This choice of protocol because wide availability of WiFi and it easy to use.

3.2.1. Server

This operation consists of several steps:

(i) Creating a Web Server to Listen for Signals from ESP Cam to Initiate Fire Detection ,

(ii) Implementing Communication Protocol for Receiving Responses from ESP Cam,

(iii) Create 2 thread

- Thread 1: To send a response to the ESP Cam module indicating successful receipt of data and allowing it to continue listening to other HTTP endpoints.

- Thread 2: To resume the fire detection process after receiving acknowledgment from the ESP Cam module.

(iv) In Thread 2, we'll fetch images from the ESP Cam's designated endpoint for capture and employ the YOLOv5 model to detect fire within those images,

(v) After detection, we'll assess whether the images contain fire by checking if the ratio of result exceeds 0.5. If fire is present, we'll send the message "Chay"; otherwise, we'll send "Nothing" to the ESP Cam's endpoint for message transmission,

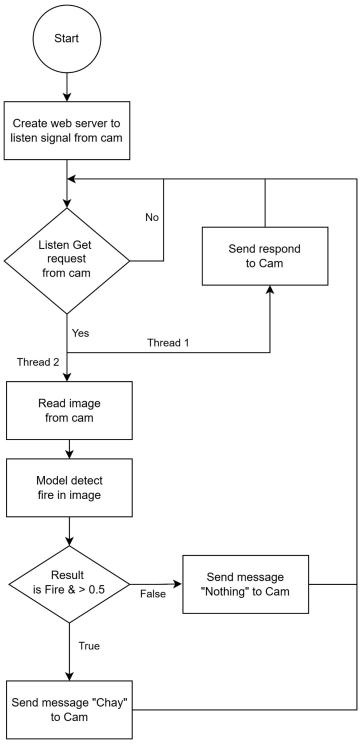

Fig. 4: Block diagram of server operations.

3.2.2. ESP Cam

(i) The setup function initializes the essential variables and values required for the system. It also configures the parameters needed for connecting to the web server. The web server is created on port 80 and starts in a "free" state. Upon receiving a signal, it changes state, captures an image, and sends it to the server,

(ii) The 'server.handleClient();' function is responsible for handling incoming client requests. It processes any HTTP requests received by the web server, allowing the server to respond appropriately. This function continuously listens for and manages client communications, ensuring that the server can react to requests, such as triggering the fire detection process or sending responses back to the ESP Cam. ,

(iii) Afterward, the system checks the signal from the gas sensor. If smoke is detected, it sends a signal to the server and waits for the server's response. If no smoke is detected, the system continues to monitor for smoke or fire signals, ensuring continuous vigilance.

(iv) If the server's response is "200", it confirms that the server is still connected. The system then waits for the initial detection message from the server. If the message is "Chay" (fire), it activates the buzzer to alert people. If the message is not "Chay", the system adjusts the angles of

the camera and captures another image of the room, continuing the detection process.

(v) After detection, we'll assess whether the images contain fire by checking if the ratio of result exceeds 0.5. If fire is present, we'll send the message "Chay"; otherwise, we'll send "Nothing" to the ESP Cam's endpoint for message transmission,

(vi) Finally, if the servo has rotated through three angles and repeated this cycle three times, capturing a total of nine pictures, but no fire is detected, the system will turn on the blinking LED to indicate an error. If the rotation process is not yet complete, the servo continues the detection process.

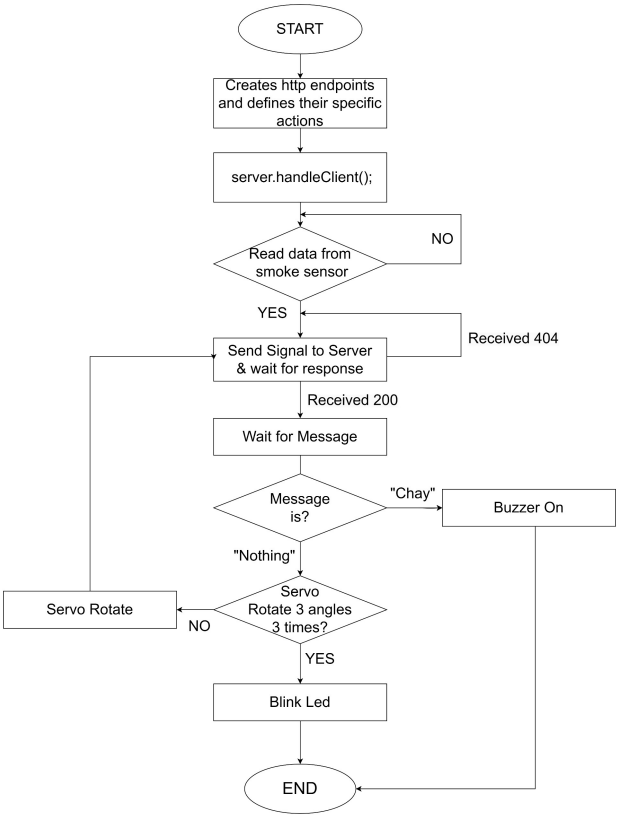

Fig. 5: Block diagram of hardware operations.

4. IMPLEMENTATION

Now that each system component has been described, the "big picture" of how the entire system is controlled will be illustrated. Illustrated in Figure 3, the system comprises two main components:

On the client side, we utilize the ESP Cam along with the I/O devices described in section 3.1. For programming the ESP Cam, we use C++ within the Arduino IDE application. The code for the ESP Cam follows the structure outlined in Fig. 5, encompassing smoke detection, image capture, and fire alarm functions.

For our server-side setup, we selected the 'IndoorFire - v3 2023-11-29' dataset from Roboflow to train the YOLOv5m model. We configured the model with an image size of 640x640, a batch size of 32, and set it to run for 30 epochs. The data used for training is from the chosen dataset, and we initialized the model with the YOLOv5m weights. Upon completing the training, we will save the best weights checkpoint to be used for fire detection in images captured by the ESP Cam. Subsequently, we will develop a server-side application using Python, adhering to the structure illustrated in Fig. 4. This application will listen for signals from the ESP Cam to initiate fire detection, fetch images from the client, analyze these images for fire presence, and send back a message indicating whether fire is detected or not.

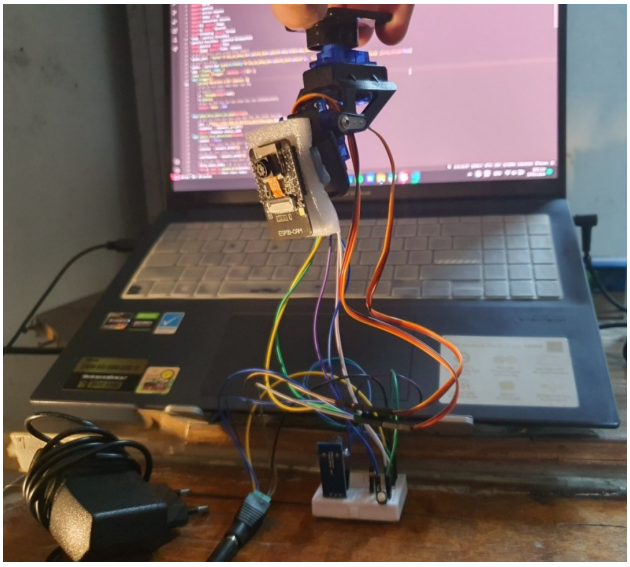

Fig. 6: Our completed system.

5. TESTING AND REVIEWS

5.1. MODEL EVALUATION

In this report, three statistical metrics were evaluated: recall, precision and mAP.

Recall is the ratio of the correct values among the values that were correctly identified, given by:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{6}$$

where TP: True Positive, FN: False Negative.

Precision is the ratio of correct values among values defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7}$$

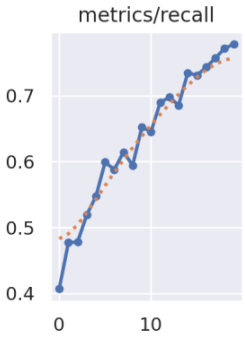

Fig. 7: Recall during training.

where TP: True Positive, FN: False Positive.

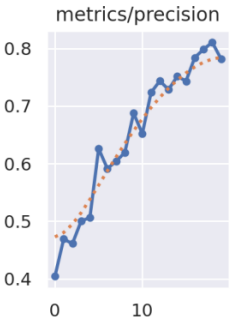

Fig. 8: Precision during training.

MAP (Mean Average Precision) is the mean of AP. The AP value of the different classes is calculated as follows:

$$AP = \sum_{1:N} (r_m - r_{m-1}) p_m \tag{8}$$

where r and p are the recall and the precision at the m threshold.

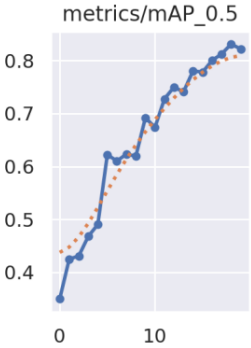

Fig. 9: MAP during training.

5.2. TESTING SYSTEM

Despite the relatively low resolution of images captured by the ESP Cam, the server is still capable of accurately detecting fire in these images, as demonstrated in Figure 10.

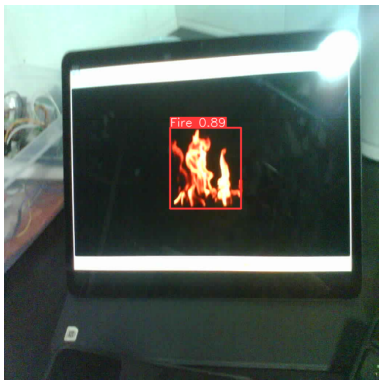

Fig. 10: Result after running system.

6. CONCLUSION

In this project, we successfully integrated convolutional neural network (CNN) technology with the power of the ESP32-Cam and the flexibility of Servo motors to create an exceptional system. Utilizing the YOLOv5 model, known for its rapid and accurate object detection and identification, our system achieves high performance in real-time tasks.

This combination provides a compact yet powerful solution, ideal for applications requiring mobility and versatility. The precise control from the Servo motors enhances adaptability to various scenarios. By leveraging these advanced technologies, our project can quickly and reliably detect objects and notify people in the vicinity, improving safety and situational awareness.

Overall, our project demonstrates the potential of combining cutting-edge neural network models with versatile hardware components to create innovative and practical solutions.

REFERENCES

[1] "ESP32 CAM Object Detection Identification with OpenCV",(`https://how2electronics.com/esp32-cam-based-object-\detection-identification-with-opencv/`), asseced: 2024-05-03

[2] Random Nerd Tutorials, "ESP32-CAM Pan and Tilt Video Streaming Web Server (2 Axis)", (`https://randomnerdtutorials.com/esp32-cam-pan-and-tilt-2-axis/`), asseced: 2024-05-06

[3] Wided Souidene Mseddi, Rafik Ghali, Marwa Jmal. Fire Detection and Segmentation using YOLOv5 and U-NET, 742-745, (https://ieeexplore.ieee.org/document/9616026), asseced: 2024-05-06.

[4] Jefferson Silva Almeida, Chenxi Huang , Fabrício Gonzalez Nogueira , Surbhi Bhatia , and Victor Hugo C. de Albuquerque. EdgeFireSmoke: A Novel Lightweight CNN Model for Real-Time Video Fire–Smoke Detection, VOL. 18, NO. 11, 7892-7893,(https://ieeexplore.ieee.org/document/9684982), asseced: 2024-04-20.

[5] Ultralytics. (2022). YOLOv5 Custom Training. Asseced from (https://docs.ultralytics.com/yolov5/tutorials/train_custom_data/), asseced: 2024-04-18.

[6] MDPI. (2022). Challenges in AI and Machine Learning: A Comprehensive Review. Computers, 12(3), 60. Asseced from (https://www.mdpi.com/2073-431X/12/3/60), asseced: 2024-10-05.