**ELECTRONIC, ELECTRICAL & SYSTEM ENGINEERING**

**卡通人物

中度可信度描述已自动生成**

**BEng Final Year Project (FYP)**

**Project title: A smart monitoring system to protect farmed ducks on the farm.**

**Author name: Haiyang Huang**

**Author ID number: 2304193**

**Project supervisor: Peter Jancovic**

**Submitted to The University of Birmingham for the degree of Bachelor of Engineering.**

**Date of submission: April 30, 2024**

**Acknowledgements**

Project Self Assessment

***Checklist: Put a “Y” in the column which corresponds to your assessment of our own ability***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Category** | **I find this very difficult** | **I find this a bit difficult** | **neutral** | **I find this fairly easy** | **I find this very easy** |
| Ability to work independently |  |  |  | Y |  |
| Ability to manage my time |  | Y |  |  |  |
| Ability to learn new skills or concepts in depth |  |  |  | Y |  |
| Ability to learn new concepts or skills quickly | Y |  |  |  |  |
| Ability to focus on targets |  |  |  |  | Y |
| Ability to apply things that I have learned |  |  | Y |  |  |
| Ability to understand the implications of results and findings |  |  | Y |  |  |
| Ability to draw conclusions |  |  |  | Y |  |

|  |  |
| --- | --- |
| What aspects of your project did you enjoy and or went well (up to 50 words) | I enjoy spending time working on this project, even though it is challenging for me to implement. I can understand the full working principles and achieve the final goals I set. Finally forming a deeper understanding of the project targets to make me easily to draw conclusion. |
| What aspects of your project did you find difficult or would you change | I have been learning many theories at a relatively slow pace, which has caused my progress to be lower than others, making me feel nervous and cannot manage time well. I cannot react fast to get a good understanding of the findings. |

**Acknowledgement**

I would like to express my deep and sincere gratitude to Dr. Peter Jancovic, who is not only my project supervisor in the final year of my undergraduate studies, but also act as my personal tutor during my freshman and sophomore years. He has witnessed my progress for the final year project incrementally, He gives me a high degree of freedom to design the entire project based on my interests and a lot of pertinent opinions during the design process and does not spare any effort to teach me to explore the project step by step. Without the support and assistance of my supervisor, I would not have been able to implement the numerous functions of the development.

I also would like to express my appreciation and great thanks to Dr. Linxue Bai, who is my assessor of the first and the second bench inspections, she mentions good ideas about how to make the experimental result more properly to evidence that the designed system is valid and robust, which are helpful to do the further evaluation of project results.

I would like to further appreciate my second bench inspection assessor, he gives reasonable suggestions for me to explain the choice-making of threshold value and confidence score in the final report, which helps me a lot to construct a more rigorous design process.

Contents

Acknowledgement1

Content2

List of figures and tables3

Abstract6

1 Introduction6

1.1 Background6

1.2 Related works7

1.3 Aims7

2 Methodology and design7

2.1 Hardware selection7

2.1.1 Raspberry Pi 4 model B with 4GB RAM and 16GB memory 7

2.1.2 Passive Infrared Motion Sensor 7

2.1.3 Raspberry Pi Camera Module 3 8

2.1.4 18650 lithium-ion rechargeable cell with uninterruptible power supply 9

2.1.5 Nature bytes Wildlife camera casing 10

2.1.6 Overview of hardware architecture 10

2.2 Software development11

2.2.1 Video capturing 11

2.2.2 Video preprocessing 12

2.2.3 Custom weights by yolov5s model 15

2.2.4 Overall system architecture 19

3 Testing20

3.1 Evaluation of preprocessing results21

3.2 Evaluation of trained Yolov5s detection model23

3.3 Optimized results with data augmentation27

4 Result and discussion2**8**

4.1 Deployment results of surveillance system29

5 Conclusion2**9**

5.1 Application scenarios30

5.2 Future work30

Reference**31**

Appendix32

**Abstract**

Amidst the ongoing evolution of embedded devices, the subject of how to autonomously monitor and safeguard farmed ducks has emerged as a significant area of interest in recent years. This paper contributes to developing a smart surveillance system on Raspberry Pi device, which is capable of protecting duck eggs, ducklings and adult ducks from their main aerial and terrestrial predators, including crows, hawks, owls, foxes, rats, weasels and badgers. The implementation of an automatic monitoring system is vital in mitigating the high rate of false detections typically made by humans relying solely on visual observation. This project proposes to train convolutional neural networks (CNN) based detector Yolov5s as the detection model. An innovative approach for implementing a high-accuracy detection system on the Raspberry Pi is presented, which facilitates efficient processing and deployment while maintaining robust detection capabilities with an overall detecting accuracy of 96.9%.

Keywords: Embedded device, CNN-based detector Yolo, Duck protection, Monitoring system.

**1 Introduction**

**1.1 Background**

The duck farming industry has spread worldwide since the nineteenth century. Duck farmers are not only attracted by the profits from selling duck eggs and meat, but also value ducks’ ability to eliminate small pests, eat weeds, and help fertilize the rice plants (Suh, 2014). However, creating a safe environment for duck farming has long been a challenge. Many predators pose a serious threat to the habitat of duck eggs, ducklings and adult ducks.

Nowadays, even though duck keepers have implemented various strategies to ensure the safety of ducks on farms, such as building secure housing, strengthening predator-proof fencing, feeding guard animals and conducting regular patrols (A. Jalaludeen and R. Richard Churchil, 2022), these measures cannot completely prevent ducks from being preyed upon within farms. (Clermont et al., 2021) found that some terrestrial animals like red foxes could dig under or jump over fences that were not adequately secured, cleverly entering duck nesting areas at night, resulting in the loss of both eggs and ducks. However, human monitoring of farming ducks around the clock leads to high levels of fatigue and the potential for errors in manual detection (Pillai et al., 2020). Therefore, implementing an intelligent surveillance system to protect farmed ducks from their various predators automatically performs an important role in the duck farming industry.

**1.2 Related works**

Phiri (2018) proposed a design to use a PIR motion sensor to monitor the temperature and duck intruders in the farm with the aid of a Yolo detection model, then send a notification message by the GSM module to the user. Pillai (2020) proposed to use of a two-stage mask region-based convolutional neural networks (Mask RCNNs) to classify different classes of birds by extracting birds’ segments of their bill shapes, head patterns, tail patterns and wing shapes. Lee (2023) proposed to detect the states of ducks in the duck cage by using deep convolutional neural networks (DCNNs) to train a smart robot, which could be able to recognize whether existed any fell or dead ducks or not. K. S. Balamurugan (2024) proposed a continuous monitoring system based on one-stage convolutional neural networks (CNNs) to train a custom You Only Look Once (Yolo) detector. This module could emit a sound that repels them when a dangerous species is detected, deterring the targeted animals to safeguard crops and farmed animals effectively.

According to the search of existing works, numerous implementable ideas involving motion sensors and CNN-based recognition systems are mentioned, which can be extracted to apply for detecting specific animals and improving the performance and efficiency of this project.

**1.3 Aims**

This project aims to use Raspberry Pi to develop an intelligent monitoring system, which can protect duck eggs, ducklings and adult ducks from their main aerial and terrestrial predators in the duck farm.

The goal of the project is to use the Raspberry Pi system to classify and detect main duck predators automatically with high accuracy, which is programmable with the Python environment. This report is devoted to making contributions to constructing an effective smart farm at a low cost. The targets to be achieved for the project are considered carefully and listed distinctly:

**l** Integrating all hardware components into a portable water-proof casing to support working in the wildlife.

**l** Constructing a detection model in Raspberry Pi system, which can detect and recognize main aerial and terrestrial predators of duck eggs, ducklings and adult ducks, including crow, hawk, eagle, owl, fox, rat, weasel, stoat and badger with high precision.

**l** Developing the function of sending an email from Raspberry Pi to Google Mail, which can send a brief notification message and detection results.

**2 Methodology and design**

Chapter 2 indicates the theoretical analysis and experimental implementation of hardware configuration and Python program development in the Raspberry Pi system.

**2.1 Hardware selection**

The hardware components are selected from the Raspberry Pi website store (<https://thepihut.com/>), facilitating replication of this project by interested researchers.

**2.1.1 Raspberry Pi 4 model B with 4GB RAM and 16GB memory**

The Raspberry Pi board is a core hardware platform to deliver the Raspberry Pi system and support interfaces to connect more compatible components (Ghael et al., 2020). Gamess E (2022) evaluated the performance of the RPi boards under the same loads, the ability of RPi 4 model B to handle complex multi-thread tasks is far superior to others, what is more, the 4-gigabyte RAM is selected to support the CPU operating the task of detection and classification at a moderate speed (Marot et al., 2017). SD card is also a crucial component for booting the RPi system and supporting memory in the RPi system. A 16-gigabyte SD card is selected to provide sufficient memory size to install substantial packages like TensorFlow and OpenCV and store videos with frame-by-frame detection results (Pajankar, 2017).

**2.1.2 Passive Infrared Motion Sensor**

Raspberry Pi is an embedded system with limited computational sources to execute intensively complex work, especially for the tasks of real-time detection and classification (Tarun et al., 2023). Motion sensor is used to construct an event-driven monitoring style, which can only activate the whole system to operate a program by sending the motion signal when the motion is detected successfully (Saranu et al., 2018).

There are mainly three types of motion sensors used for detecting the existence of movements, which are passive infrared (PIR) sensor, ultrasonic sensor and microwave sensor, respectively (Polivka, 2007). However, the ultrasonic sensor and microwave sensor are unsuitable for detecting duck predators within the farm considering both sensors’ sensitivity and penetrability. The generated sound wave by ultrasonic sensor can be easily triggered by non-living objects moved by wind or mechanical forces, which leads to a high false rate of detecting duck predators (Mukhopadhyay et al., 2018). What is more, the emitted microwave pulses from a microwave sensor have the strong capacity to perceive motion through opaque barriers, which pick up unwanted movements from outside of the duck farm.

Among them, The PIR motion sensor meets the design requirements precisely and has been widely used by researchers like Phiri H (2018) in the field of detecting duck and duck intruders. The emitted infrared waves can be received by the PIR sensor when the temperature of a living creature is greater than absolute zero, the Stefan-Boltzmann law is used to measure the infrared radiation emission energy, which is shown in Formula 1 (Mukhopadhyay et al., 2018).

, where (1)

There are two pyroelectric slots installed to detect any heat source movements within the detection area by comparing the amount of infrared radiation received in slots, which is shown in Figure 1. Any differential change between two slots is defined that there is a motion being detected (Chaturvedi et al., 2016).

图示

描述已自动生成图示

描述已自动生成

Figure 1. Working principle of the passive infrared sensor. (Left one)

Figure 2. Schematic drawing of working principle of Fresnel lens. (Right one)

As is shown in Figure 2, the Fresnel lens can focus incoming infrared radiation from the environment onto the pyroelectric elements of the sensor, which increases the sensitivity and effectiveness of the sensor performance.

**2.1.3 Raspberry Pi Camera Module 3**

The compatible camera modules for Raspberry Pi are mainly divided into the Pi camera and the USB Webcam, which is used to capture images and video recordings. Between them, the RPi camera is selected to use with the specific Camera Module 3, which features autofocus capability with a higher frame rate per second, aiding the system in detecting duck predators with a lower false rate (Pagnutti et al., 2017).

**2.1.4 18650 lithium-ion rechargeable cell with uninterruptible power supply UPS HAT B**

There are some issues with charging the RPi system without a constant power supply, such as random reboots of the system, data corruption, failure to boot and the malfunction of connecting peripherals (Borah et al., 2020). So far, lithium polymer USB power bank and lithium-ion battery are the most widely used to charge the RPi system with a constant power source. An uninterruptible power supply (UPS) HAT is required to connect the lithium-ion batteries to the RPi system. Here is a datasheet to compare two power source specifications in Table 1, which is obtained from the Raspberry Pi website store.

Table 1. The datasheet of two types of batteries

|  |  |  |
| --- | --- | --- |
| Specification | USB powered bank | 18650 Lithium-ion battery |
| Nominal capacity | 10000mAh | 3000mAh |
| Charging cycles | 500 | 300 to 500 |
| Max output | 5V / 3A for charging | 4.2V / 5A for charging  3.0V / 2.6A for discharging |
| Dimensions |  | 1 |
| Cost | 25.90£ | 7£ + 25£ (with UPS HAT B) |

The utilizing of lithium-ion batteries is the better choice for this project. Firstly, the UPS HAT features a battery management system that monitors electricity levels and ensures safe charging and discharging of lithium-ion batteries (Miron-Alexe, 2022). Secondly, it connects to the RPi board using pogo pins that engage the underside GPIO pins, freeing up the USB port. This setup allows for an additional power source, such as the USB-powered bank, to provide uninterrupted power, ensuring safe shutdowns or continuous operation in the event of one power source failing (Namdeo et al, 2020). Thirdly, it offers space for two lithium-ion batteries, effectively doubling the capacity from 3000mAh to 6000mAh.

**2.1.5 Naturebytes Wildlife camera casing**

The Naturebytes Wildlife casing is selected to integrate all hardware components for outdoor operation, which can offer excellent waterproof and weather-resistant performance, forming a complete consumer product rather than merely a design project (Marić et al, 2023). This casing not only accommodates two lenses for the PIR motion sensor and RPi camera module but also provides sufficient space to house a lithium-ion battery with a UPS HAT (B). Additionally, it includes extra space to connect a USB power bank, enhancing system capabilities.

**2.1.6 Overview of hardware architecture**

The schematic drawing of the hardware design is depicted in Figure 3, where all components are configured to work with the Raspberry Pi. Each component has been tested to ensure operation correctly. Firstly, the SD card is formatted to delete the existing files, then the SD card is inserted into the card slot of the RPi 4 model B after installing a 64-bit OS Raspbian system by the software ‘Raspberry Pi Imager’. Secondly, the PIR sensor is connected to the RPi 4 Model B, its pins of VCC, OUT and GND are connected to the second pin (+5V), GPIO 11 and the sixth pin on the board respectively. Thirdly, a pair of lithium-ion batteries are installed on the UPS HAT (B), connecting with underside GPIOs on board by the pogo pins on UPS HAT (B). Fourthly, RPi camera module 3 is inserted on the CSI port of the RPi 4 model B board. Finally, all hardware components are integrated into the Naturebytes wildlife casing to form a portable smart device.

During the designing process, peripherals like mouse and keyboard are used to create and modify the Python codes to achieve the targets of the project. An external screen is used to provide a visual display of the 64-bit OS RPi system. For future enhancement, an extra USB power bank can be used to extend operation time by inserting USB ports. An Ethernet port is used to connect the RPi system to the internet, supporting the function of sending detection results to Google Mail.



Figure 3. A schematic drawing of hardware architecture

**2.2 Software development**

This chapter introduces methodologies on how to realize the functional requirements mentioned in Chapter 1, it also explores a creative method to integrate the detection model into the RPi system and the function of sending detection videos to Google Mail. The detection results are also optimized through post-processing. Finally, all functions are integrated to form a comprehensive design process in the Raspberry Pi system.

**2.2.1 Video capturing**

Even though the selected Raspberry Pi 4 model B board with 4GB RAM can perform much better than other boards in handling the multi-threads as mentioned in Chapter 1, it cannot provide sufficient computational sources to maintain the high performance of detecting ducks and duck predators in real-time monitoring and give the correct feedback by sending the detection video to the Google Mail (Tarun et al., 2023). The PIR motion sensor can be used as a switch to control the operation of the whole system, which can create a high-efficiency system and reduce the consumption of limited power sources.

**2.2.1.1 Experimental implementation of video capturing**

When movement is detected, a high-level electric signal is sent from the PIR sensor to the GPIOs on the board. As a result, the RPi Camera Module 3 is activated to capture video upon successfully receiving this signal. However, the system keeps sleeping and receiving low-level signals from the PIR sensor until a motion is detected. The function of video capturing with the PIR sensor is realized in Python 3 of RPi by defining the voltage level of GPIO 11 as high or low in the main loop. The captured video file is stored with the compression codec of ‘H.264’. However, it is not a compatible video file to process or do the detection task directly with the Python packages (Khalifeh et al., 2017), the problem is solved by converting the captured video to ‘MP4’ video format.

**2.2.2 Video preprocessing**

Preprocessing the captured video is aimed to increase detection performance by effectively reducing disturbance and noise (Algazi et al, 1995). There are mainly five algorithms used to process every frame of video, which are video resizing, video normalization, video scaling, histogram equalization and background subtraction separately. A detailed explanation of how to implement and integrate these methodologies in this designed Raspberry Pi system is included in Chapter 3.

**2.2.2.1 Video resizing**

As is well known that each frame of video is recognized as a single image. Video resizing is used as a key step to match the requirements of the detection model, which modifies the image pixels to the certain size pixels for each input image (Saponara et al, 2021). The actual dimension of an image in a frame of video can be calculated by Formula 2 based on the target size:

(2)

Where, wnew and hnew are the width and height of the resized image; M is the target size for the largest parameter of width and height; w and h are the original image dimensions; max (w, h) is the largest parameter of the original width and height.

**2.2.2.2 Video normalization**

图表, 直方图

描述已自动生成

Figure 4. The comparison of the original and the normalized image intensities.

Image normalization is used to rescale the original image’s pixel values within the range of ‘0’ to ‘1’, which is shown in Figure 4 (Depeursinge et al., 2017). This function is required to meet the trained detection model needs, which can enhance detection performance by reducing the noise and increasing the image contrast. Formula 3 shows how the original image converts to the normalized image, where ‘255’ is the maximum pixel value for an 8-bit colorful image.

(3)

**2.2.2.3 Video scaling**

Scaling video (Weissenborn et al., 2019) is a simple method of frame resizing to speed up the detection process by adjusting the resolution of the preprocessed video based on specific requirements. This project is used to reduce the resolution of the preprocessed video, which ensures the video with detection results can be sent to Google Mail below the video size limit of 25 megabytes. Formula 4 presented how it works.

(4)

**2.2.2.4 Histogram equalization**

Histogram equalization is an effective way to increase the global contrast of an image uniformly by redistributing the intensity of pixel values.

图表, 直方图

描述已自动生成

Figure 5. The comparison of the original and equalized histogram distribution

As is shown in Figure 5 (Dorothy et al., 2015), it is realized by computing the histogram of an input image first, which is a plot of the number of pixels in the image against each intensity value. Then the cumulative distribution function (CDF) is drawn by summing up the counts of pixels from the original histogram, starting from the lowest intensity level up to the current intensity level. The equation for calculating the CDF is shown in Formula 5. Finally, the intensity values are redistributed to produce based on the CDF of the image.

(5)

Where, L is the maximum value of a pixel; Pr is the probability density function before applying the histogram equalization.

**2.2.2.5 Background subtraction**

However, implementing histogram equalization also enhances the noise signal of the background, because it increases the contrast of each frame video globally. Therefore, background subtraction is imported to detect the moving objects and remove the noise components in each frame of the captured video recording by the segmentation of the foreground and the background (Garcia-Garcia et al., 2020).

图示

描述已自动生成

Figure 6. The working principle of background histogram

As shown in Figure 6 (Garcia-Garcia et al., 2020), Initially, a background model is constructed from the first frame pixel values, which represent the static part of the scene without any moving objects. The frame difference between the first frame with the subsequent frame is calculated. Then pixel values are compared with a pre-set threshold value ‘N’. When the calculated frame differencing is greater than the value of ‘N’, the background model is recognized as the foreground and updated with new information from incoming frames. However, the input frame of the video is recognized as background when the calculated frame differencing is less than the threshold value. Its working principle is indicated accurately by the Gaussian probability density function in mathematical Formula 6, where and are the mean and variance pixel values of the background model.

(6)

**2.2.2.6 Experimental implementation of video preprocessing**

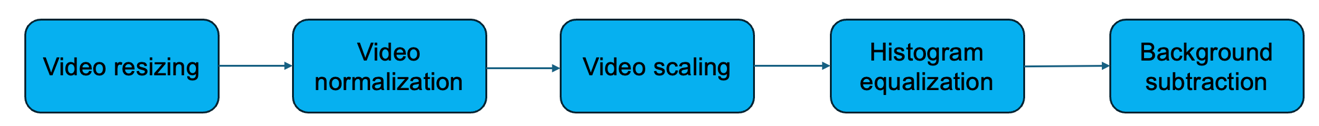


Figure 7. Flow chart for the preprocessing

The ‘OpenCV’ package is required in the environment of Python 3 within the RPi system to process video, calling the package by ‘import cv2’. A Python function is created to integrate all preprocess methods used in Figure 7

Firstly, the image in each frame of the video is resized by 640 (width) 640 (height). Secondly, the resized image is normalized by the mathematical calculation to divide the original resized pixel values by 255.0. Thirdly, video scaling is realized by pre-set a scale percent with a reasonable value of 0.75 without the influence on the detection results (Weissenborn et al., 2019). Fourthly, the color space used by the OpenCV package to capture images from a pi-camera is the RGB (Red, Green, Blue) format. Therefore, the color video format is converted from BGR to YUV (Luminance, Chrominance) to apply the histogram equalization only on each frame of the Y-channel, which effectively enhances the contrast of the image without altering the color properties. However, the YUV format is not compatible with d further preprocessing functions. Therefore, the RGB format is converted back after finishing the histogram equalization process. Finally, when the pixel intensity of the input image is less than the default threshold value of 0.25, it is recognized as the background. In contrast, the input is recognized as the foreground.

**2.2.3 Custom weights by yoloV5s model**

The neural network is used to mimic the operation of the human brain with the aid of a machine learning algorithm, among them, the convolutional neural network can provide a superior performance in classifying and detecting specific objects by its ability to extract features automatically than other traditional neural networks (K. S. Balamurugan, 2024).

The CNN-based detectors are divided into one-stage detectors and multi-stage detectors. The advantage of using a one-stage CNN-based detector is to predict class probabilities with the associated bounding boxes within one pass, which is more suitable to use for speeding up the detection process without sacrificing significant accuracy (Li et al., 2022). So far, there are mainly three one-stage algorithms to form a CNN-based object detector, which are RetinaNet, You Only Look Once (Yolo) and Single Shot detector (SSD) algorithms. Among them, the Yolo algorithm model can offer a faster detection speed with higher accuracy and is easier to implement in the Raspberry Pi system (Gu et al, 2022). The Yolo algorithm was developed by the Ultralytics company’s CEO Glenn Jocher (2022), which provides various versions based on different requirements. This project proposed to use the Yolov5s algorithm to train a detection model with the custom dataset, which occupies a smaller model size and provides a faster detection speed.

A detailed architecture of the yolov5s model is shown in Figure 8 (Santos et al, 2022). The backbone part is used to extract features from the input image to form the different scales of feature maps. The bottleneck part is used to integrate different scales of feature maps from the backbone. Finally, three numbers of two-dimensional convolutional layers are produced and used to detect different sizes of objects in the head part, then output the prediction result of class probabilities and bounding box coordinates in one stage.

图示

描述已自动生成

Figure 8. The theoretical architecture of yolov5s

For the training process, the focus layer uses its artificial neurons to hold the pixel values of labelled images as segments, then the Sigmoid activation function estimates the number of image features as the output to transmit from the focus layer neurons to convolutional layers neurons by applying a non-linear activation function in formula 7 (Jiang et al., 2022). For the mathematics representation of the Silu algorithm:

where, x is the input features. (7)

If the value of x is greater than or equal to zero, the activation function becomes close to ‘1’, and therefore, the convolutional layers are activated to receive the output features from the previous layer. Then features are extracted to form the learnable feature maps based on convolutional layers. During the training process, the feature maps in the backbone, neck and head are learnt by the machine and generate the new weights and biases automatically (Ren et al., 2022).

For the detection process, when the new image is input into the focus layer, the whole image is divided into a few segments, and then the features are extracted and aggregated to form the learnable feature maps in each layer of the backbone, neck and head (Ren and Wang, 2022). Each layer compares the feature maps with the pre-trained weights and biases to predict the class probabilities and the bounding boxes.

**2.2.3.1 Experimental implementation of dataset collection**

Images collection: The high-quality images of duck and its main aerial and terrestrial predators are collected on Google iStock image supporter (<https://www.istockphoto.com/search/2/film>), the collected dataset is composed of eight animal classes, including ‘duck’, ‘crow’, ‘hawk and eagle’, ‘owl’, ‘fox’, ‘rat’, ‘weasel and stoat’ and ‘badger’. An organized file folder is created to divide the collected dataset into the training part and validating part with two subfolders. Each class collects approximately 200 images with multiple angles to focus on different features.

Images labelling: The Yolo model requires the annotated images with the class name and the associated bounding boxes to locate the specific objects before the training process, which helps the training model recognize which feature needs to be learnt with the correct name. ‘LabelImg’ is a graphical image annotation tool, allowing to draw the bounding boxes and class names manually. Figure 11 shows the launched interface, the predefined classes file is modified to follow the classes of dataset.

一群鸟在地上

描述已自动生成

Figure9. Python Tool of ‘LabelImg’

As is shown in Figure 9, all predefined classes can be selected to define the names of the objects in the drawn bounding boxes. Then export the Yolo format annotation file, adding to the collected dataset.

**2.2.3.2 Training the custom Yolov5s detection model**

The training process is done by the NVIDIA RTX 4060 of Windows PC, which can provide far superior speed than the Raspberry Pi. Based on the ability of the Yolo algorithm to continuously learn the input features and be able to change the pre-trained weights and biases, the detection model is trained on the basis of the pre-trained weights, which speeds up the training process and achieves higher accuracy with fewer training cycles.

The batch size is also a key hyperparameter in the training process, which is used to determine how much data samples are processed before updating model weights per training iteration. There are three batch sizes of 16. 32 and 64 are tested during the training process, finding that the smaller batch size of 16 performs better in enhancing the training accuracy and avoiding the risk of over-fitting by training at a lower convergence speed (Kandel and Castelli, 2020). Training with r 100, 200, and 300 epochs, comparing each performance with the model validation accuracy, training loss and validation loss.

Non-maximum suppression (NMS) is a post-processing technique used to improve the accuracy of the detection results by eliminating duplicate detection results (Hosang et al., 2017). This algorithm reduces redundancy by keeping only the most relevant bounding box for the object in each frame video. The probability of overlapping between two bounding boxes can be calculated by Intersection over Union (IoU) in Formula 8, where the denominator is the area encompassed by both the predicted bounding box and the ground-truth bounding box. The bounding box is suppressed when the value of IoU is higher than a pre-set threshold value of 0.5 in this project.

(8)

**2.2.4 Overall system architecture**

**图示

描述已自动生成**

Figure 10. The overall system architecture

The architecture of the overall operation system in Raspberry Pi is shown in Figure 10. Firstly, the pi-camera is booted to capture the video when the PIR sensor detects any movements of living creatures, stopping capturing until no motion is detected after 5 seconds delay, the video file converts from ‘H.264’ to ‘MP4’ format before storing into the SD card. Secondly, matching the detection model specifications by preprocessing to the size of 640640 pixels and normalizing the pixel values in the interval of 0 to 1. Scaling down the video resolution to 75% and compressing the frame rate by a factor of 2, which is used to support the function of sending video to Google Mail below its size limits. Histogram equalization is used to increase the global contrast by redistributing the pixel values in each frame of the video, and background subtraction is implemented to reduce the noise generated from the histogram equalization and remove all unrelated backgrounds. Thirdly, the dataset collected to train the custom weights and biases is required, then the Yolov5s detection model is converted into the TensorFlow Lite model, and it starts to detect the pre-trained duck and its predator in the preprocessed video. Fourthly, when the prediction probability of the duck or its predator is detected above the confidence score of 0.8, the trained detection model adds the bounding box of the image based on the detected coordinates. Then the non-maximum suppression method is used to avoid the overlap of the bounding box. Finally, the system can output the predicted class probabilities and the associated bounding box with the maximum probability for each detected class. Sending the detection video with a notice message to Google Mail only when the duck predator is detected above the confidence score. This system forms a closed loop, after finishing the process of sending Gmail, it will be activated again when the PIR sensor detects motion until the equipped Lithium-ion batteries are out of power.

**3 Testing**

**3.1 Evaluation of preprocessing results**

As is shown in Figure 11, a fox image captured on the duck farm is resized successfully from the original size of 960(width)543(height) to 640(width)640(height) without the changing of image quality. Then the image is normalized and as shown in Figure 7, there is an absence of perceptible disparity when examined visually without the drawing of histogram.

草地上的动物

描述已自动生成草地上的棕色动物

描述已自动生成动物在草地上吃草

描述已自动生成

Figure 11. Original fox image (Left), resized fox image (Middle), normalized fox image (Right)

图表, 折线图

描述已自动生成图表, 折线图

描述已自动生成

Figure 12. The histogram distribution of resized (Left) and normalized fox image (right)

However, the distributions of pixel values are observed quite differently by drawing the histogram distributions of both images in Figure 12, the pixel values of the resized image are widely distributed from 0 to 255. In contrast, the pixel values of the normalized image are accumulated within the interval of 0 to 1. The histogram equalization is applied to the scaled image, and the contrast of pixel values increases for the global region obviously. However, the noises in the background are also amplified. Lastly, the background subtraction method is used successfully to reduce the generated noise and the unrelated components in the background. The result of preprocessing is shown in Figure 13.

动物在草地上吃草

描述已自动生成动物在草地上吃草

描述已自动生成鸟站在草地上

中度可信度描述已自动生成

Figure 13. Scaled fox image (Left), equalized fox image (Middle), background subtracted fox image (Right)

**3.2 Evaluation of trained Yolov5s detection model**

The trained detection model is validated by testing the remaining 20% of labelled images in the validation part of the collected dataset. The Yolov5s detection model is trained by 100 epochs, 200 epochs and 300 epochs, respectively. For training 100 epochs, the detection accuracy is lower than 80%. For training 300 epochs, as is shown in Figure 14, the training loss continues to decrease while the validation loss starts to increase and fluctuates heavily since 200 epochs, indicating that the trained model is overfitting to the training data and not generalizing well to unseen data.

图表, 直方图

描述已自动生成图表, 直方图

描述已自动生成图表, 散点图

描述已自动生成图表, 散点图

描述已自动生成

Figure 14. The training loss and the validation loss of training 300 epochs

Therefore, reducing the training epochs is required to avoid the over-fitting phenomenon, as is shown in Figure 13, 230 epochs provide a better detection performance, indicating that the trained model generates a smooth loss curve without significant fluctuations.

图表, 直方图

描述已自动生成图表, 直方图

描述已自动生成图表, 散点图

描述已自动生成图表, 散点图

描述已自动生成

Figure 15. The training loss and the validation loss of training 230 epochs

The precision and recall rate are two key parameters to determine the performance and make an adjudgment about whether the trained model is valid or not (Santos et al, 2022).

The Precision measures the proportion of positive detection rate in the dataset, for example, there are 49 duck images in the validation dataset of 262 images in Table 2, the precision determines how many images succeed in recognizing ducks with the correct labelled class name ‘duck’, then the precision is calculated by Formula 8.

(8)

The recall rate is defined as a measurement of the proportion of actual positives that are correctly identified by the model. It is defined as the number of true positive results divided by the number of all detected positive samples, which can be calculated by Formula 9.

(9)

There is a sharp increase in overall precision and the recall rate from 0 to 0.8 for the first fifty cycles, then both lines continue upwards smoothly, finally reaching more than 85% accuracy, which evidences that the trained Yolov5s model is highly efficient and valid. Figure 16 demonstrates the calculated mean average precision (mAP) at multiple IoU (Intersection over Union) thresholds, which is a robust way to evaluate the detection model performance. For the threshold of 0.5, the metric plateaus at around 0.8, which illustrates that the model's ability to predict bounding boxes with at least 50% overlap with ground truth stabilized after about 50 epochs. For the threshold intervals of 0.5 to 0.95 by incrementing with 0.05, the lower values of mAP are generated reasonably, because it is difficult to achieve a high degree of overlap between the predicted bounding box and the ground truth one with a limited dataset of 1800 images.

图表, 散点图

描述已自动生成图表, 散点图

描述已自动生成形状

描述已自动生成图表, 形状

描述已自动生成

Figure 16. The overall detection results of the Yolov5s model

The confusion matrix is an effective visual view to define the performance of the detection model by showing the prediction results of all classes in the dataset (Gu et al., 2022). As the result is shown in Figure 17, most of the trained classes can reach a high positive detection rate of around 90% except the eagles and rats, because both classes occupy high rates of misclassifying as backgrounds, which are 0.15 and 0.14 separately.

图表, 瀑布图

描述已自动生成

Figure 17. Confusion matrix of each class in the dataset

Table 2 shows the detailed detection results, which is useful for analyzing the reason why the classes of the dataset have different performances in the confusion matrix. There is a total of 262 images for validating the dataset, however, there are fewer instances used to test for owls and rats, causing the high accuracy in recalling them and less precision in detecting them. There exists a trade-off between the precision and recall rate.

Table 2. Validation results of each class

表格

中度可信度描述已自动生成

However, the benchmark results in Fig16 are not reach the expectations, because the performance of trained model indicates that it has still over-fitting, the training loss kept drop while validation loss line shake severely, the global lowest point is met at around 75th epoch. Even though training with smaller epoch numbers, the validation loss lines still not convergence well, this result is led by collected dataset. The YOLOv5 model has 34 convolution layers to extract features from original input image. If not freeze partial layers or larger input data, the model is to complex so that it can learn similar features fast. Therefore, the data augmentation method is adopted to increase the input data amount to strength dataset to fit model based on keras library, author uses rescaling method firstly to rescale all images to YOLO compatible size with width 640 and height 640, then implementing rotation, horizontal flip, channel shift and horizontal shift methods to augment the amount of images to 4 times compared with former. Then repeat the training step with YOLOv5s pre-trained weights, the early-stopping method is activated with patience 5 to retrain rising of validation loss, the validation results are shown below in Fig 18.

Obviously, the validation results are perfect convergence without the over-fitting trend while the accuracy, recall rate and mAP indicators are all improved above 0.9 at 175th epoch.

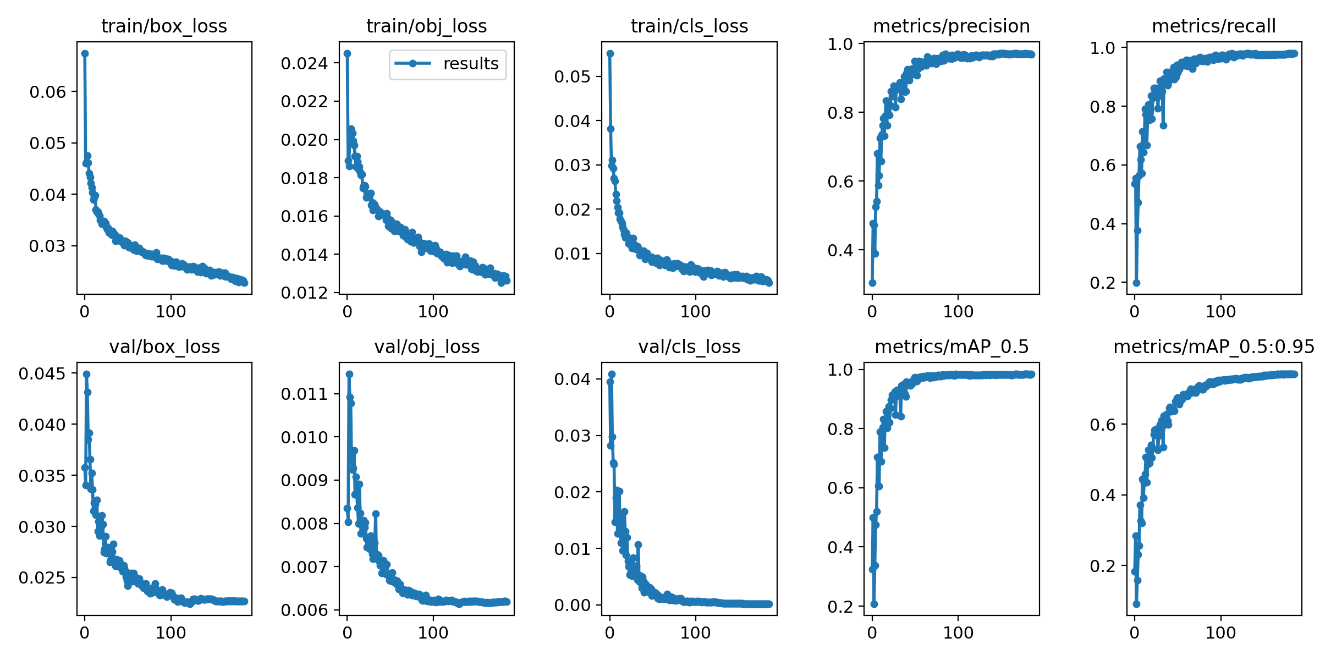


Fig 18: Training results after data augmentation

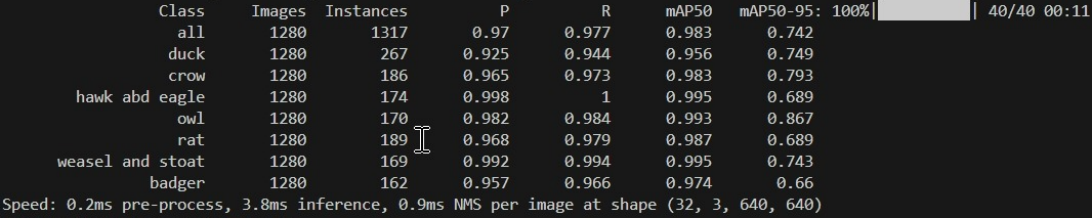


Fig 19: Validation results after data augmentation

The ‘PyTorch’ package is used in the Raspberry Pi system to infer the Yolo model, nevertheless, the RPi is an embedded system with limited computational resources, even though the Yolov5s is a smaller model size compared with others, the detection task still cannot support with the high frame rate. Therefore, optimizing the detection model by converting the format of the detection model from Yolo to TensorFlow Lite is proposed in this project. Compared to the Yolo model, the TensorFlow Lite model occupies less computational power and slightly reduces the precision of the model's parameters to speed up inference time significantly, which shortens the latency time to operate the whole system. The TensorFlow Lite model is designed to be lightweight and easy to integrate into the designed monitoring system (Warden and Situnayake, 2019). TensorFlow and Pytorch framework are spread widely due to the ability of both training and inferring, supporting complex network construction, gradient backpropagation to optimize weights during training process, however, the FPS performance is poor if deploying one of such framework on computational-limited Raspberry Pi that only has CPU. TensorFlow Lite is more lightweight because it not support training model since it has no Nvidia GPU. What is more, optimizing FPS by converting weights format from float 32 to float 16, which decrease model size with 4 times and speed up FPS with 2 times.

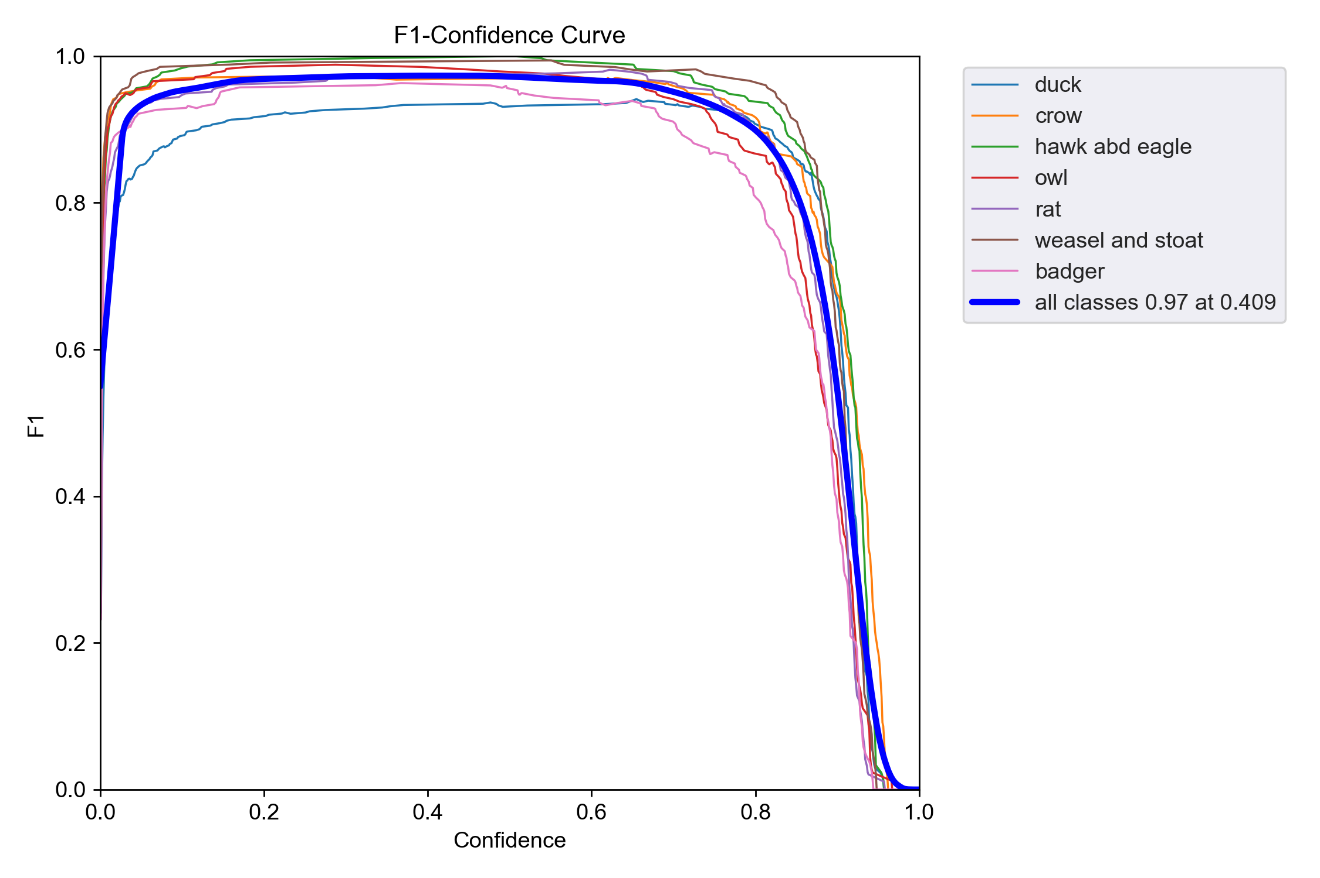


Figure 20. F1 curve-confidence curve

Figure 20 shows that the highest detection accuracy can be obtained when the confidence score is pre-set at 0.309, and the overall detection accuracy of all classes reaches 0.87. Selecting a confidence score of 0.409 means that the detection model only outputs the prediction results of class probability at least 40.9%. However, the confidence value of 0.309 is too low to cause a high false detection rate in backgrounds or unrelated species. This project trains eight different classes of animals, which can easily classify any moving objects as ducks or duck predators at a probability of around 30%. In Figure 18, there does not appear a dramatically decreasing trend of detection accuracy from 0.409 to 0.8. Therefore, the value of 0.8 is selected as the confidence score to reduce the detection false rate and the consumption of operational sources.

**4 Result and discussion**

**4.1 Result of overall system**

Based on the complete design process, the detection results in the Raspberry Pi system are experimentally tested and evaluated. As is shown in Figure 19, the detection results are sent to Google Mail successfully with a notice message and a detection video.

图形用户界面, 文本, 应用程序

描述已自动生成

Figure 19. The function of sending video to Google Mail

Even though the Yolov5s detection model theoretically has better accuracy than the TensorFlow Lite detection model, it can be compensated by preprocessing video before the detection process. As can be seen in Figure 20 and Figure 21, the designed system even performs better in detecting with higher accuracy of labelling the class probabilities and the associated bounding boxes. The disadvantage of YOLO is that the overlapped objects cannot visualize bounding boxes, especially for small objects. Because NMS has been implemented by YOLO, which filters the overlapped bounding boxes with a pre-set IOU threshold.

鸟站在草地上

描述已自动生成一群人在雪地上站着鸟

描述已自动生成

Figure 20. Yolov5s results of duck (Left), converted TensorFlow Lite results on RPi (Right)

草地上的熊

描述已自动生成小孩在草地上

中度可信度描述已自动生成

Figure 21. Yolov5s results of fox (Left), converted TensorFlow Lite results on RPi (Right).

**5 Conclusion**

In this paper, a methodology for implementing an effective detection system based on the Raspberry Pi in video recording is proposed, which is realized by applying a converted pre-trained Yolov5s model in the format of TensorFlow Lite. This method reduces the model size and the inference time of the detection model, compensated but also offers a high accuracy of approximately 86.7% for detecting the classes of duck, crow, hawk, owl, fox, rat, weasel and badger, respectively. A portable device is developed to execute the designed detection system to monitor the environment within the duck farm by installing it on the fencing or at the entrance of the farm door.

In Chapter 1, the importance of implementing an automatic surveillance system on the duck farm is analyzed and discussed with real cases. Various robust techniques for object detection are mentioned in the existing works briefly. The hardware choice is discussed based on the functional requirements in Chapter 2, and the literature review of methodologies to realize the video capturing, video preprocessing and detection model training are introduced. What is more, the implementation of the selected method for each specific target is explained clearly.

Chapter 3 tests the performance of video preprocessing by comparing each processed image in each stage, and the results of the trained Yolov5s model is evaluated by comparing the training parameters, the overall precision and recall rate of each class of pre-trained animals. The results of integrating the detection model with the preprocessing part into RPi system are discussed in Chapter 4.

In conclusion, the designed detection model can raise the high accuracy without occupying the huge computational sources in the RPi system.

**图示, 工程绘图

描述已自动生成5.1 Application scenarios**

The designed device can be placed on a certain height of fencing or at the entrance of the farm without a fully enclosed. By considering the dimensions of duck eggs, ducklings and adult ducks, placing the device on the fencing above 1.5 meters.

**5.2 Future work**

Possible future works will be concentrated on two aspects: 1) The improvement of detection accuracy. 2) Prolong the operating time of hardware. 3) Activating the speaker as an alarm. Based on the discussion in Chapter 3, the instances of the owl and the weasel are much less compared with other classes, which leads to a lower detection precision rate, the accuracy of the detection model can be greatly enhanced by collecting more images. An extra USB power bank acts as an extra power source, creating a hybrid power system with a pair of lithium-ion batteries. The speaker can be used to act as an alarm when the duck predators are detected by the PIR motion sensor, it can not only sound an alarm to remind duck keepers that a duck predator is recognized to attack the farmed duck but also repel dangerous species which are closed ducks.

**Reference**

Suh, J., 2014. Theory and reality of integrated rice–duck farming in Asian developing countries: A systematic review and SWOT analysis. Agricultural Systems, 125, pp.74-81.

Jalaludeen, A. and Churchil, R.R., 2022. Duck Production: An Overview. In: Jalaludeen, A., Churchil, R.R. and Baéza, E., eds. Duck Production and Management Strategies. Singapore: Springer.

Clermont, J., Woodward-Gagné, S. and Berteaux, D., 2021. Digging into the behaviour of an active hunting predator: arctic fox prey caching events revealed by accelerometry. Movement Ecology, 9(58).

Pillai, S.K., Raghuwanshi, M.M. and Borkar, P., 2020. Super resolution Mask RCNN based transfer deep learning approach for identification of bird species. International Journal of Advanced Research in Engineering and Technology, 11(11), pp.864-874.

Phiri, H. and Phiri, J., 2018. Real time sensing and monitoring of environmental conditions in a chicken house. Proceedings of the 2018 ZAPUC International Conference.

Lee, J. and Kang, H., 2023. A Study of Duck Detection using Deep Neural Network based on RetinaNet Model in Smart Farming. Journal of Animal Science and Technology.

B.K, S., C.K, Pradhan, V.A, N., H.G, and G.P., 2024. An internet of things based smart agriculture monitoring system using convolution neural network algorithm. EAI Endorsed Transactions on Internet of Things, 10.

Ghael, H.D., Solanki, L. and Sahu, G., 2020. A review paper on raspberry pi and its applications. International Journal of Advances in Engineering and Management, 2(12), pp.4.

Gamess, E. and Hernandez, S., 2022. Performance evaluation of different Raspberry Pi models for a broad spectrum of interests. International Journal of Advanced Computer Science and Applications, 13(2).

Marot, J. and Bourennane, S., 2017. Raspberry Pi for image processing education. 25th European Signal Processing Conference (EUSIPCO), Kos, Greece, pp.2364-2366.

Pajankar, A., 2017. Introduction to single board computers and raspberry pi. In: Raspberry Pi Supercomputing and Scientific Programming: MPI4PY, NumPy, and SciPy for Enthusiasts, pp. 1-25.

Tarun, I.G.L., Lopez, V.W.M., Serrano, P.A.C., et al., 2023. Performance Evaluation of Face Mask Detection for Real-Time Implementation on an RPi.

Saranu, P.N., Abirami, G., Sivakumar, S., et al., 2018. Theft detection system using PIR sensor. In: 2018 4th International conference on electrical energy systems (ICEES), IEEE, pp. 656-660.

Zhmud, V.A., Kondratiev, N.O., Kuznetsov, K.A., et al., 2018. Application of ultrasonic sensor for measuring distances in robotics. Journal of Physics: Conference Series, 1015(3), 032189.

Polivka, J., 2007. An overview of microwave sensor technology. High Frequency Electronics, 6(4), pp. 32-42.

Mukhopadhyay, B., Srirangarajan, S. and Kar, S., 2018. Modeling the analog response of passive infrared sensor. Sensors and Actuators A: Physical, 279, pp. 65-74.

Chaturvedi, A., Kumar, P. and Rawat, S., 2016. Proposed novel security system based on passive infrared sensor. In: 2016 International Conference on Information Technology (InCITe) - The Next Generation IT Summit on the Theme - Internet of Things: Connect your Worlds, Noida, India, pp. 44-47.

Pagnutti, M., Ryan, R.E., Cazenavette, G., et al., 2017. Laying the foundation to use Raspberry Pi 3 V2 camera module imagery for scientific and engineering purposes. Journal of Electronic Imaging, 26(1), 013014-013014.

Borah, S., Kumar, R. and Mukherjee, S., 2020. Study of RTPPS algorithm in UWB communication medium for a surveillance system to protect agricultural crops from wild animals. In: 2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Chennai, India, pp. 121-126.

Miron-Alexe, V., 2022. PORTABLE LITHIUM-ION BATTERY UPS WITH BMS FUNCTION FOR RASPBERRY PI AND OTHER IOT EMBEDDED SYSTEMS. Journal of Science & Arts, 22(3).

Namdeo, R., Sharma, S., Anand, V., et al., 2020. Smart Automated Surveillance System using Raspberry Pi. International Journal of Recent Technology and Engineering (IJRTE), 9(2), pp.308-311.

Marić, L., et al., 2023. Advancement of an Integrated Technological Platform for Wildfire Management through Edge Computing. 2023 8th International Conference on Smart and Sustainable Technologies (SpliTech), Split/Bol, Croatia, pp.1-6.

Khalifeh, A., Al-Rawi, S., Alabsi, F., 2017. An Automated Testbed for Video Quality Optimization over Lossy Networks. Proceedings of the International Conference on Video and Image Processing, pp.192-196.

Algazi, V.R., Ford, G.E., El-Fallah, A.I., et al., 1995. Preprocessing for improved performance in image and video coding. Applications of Digital Image Processing XVIII, SPIE, 2564, pp.22-31.

Saponara, S., Elhanashi, A., 2021. Impact of image resizing on deep learning detectors for training time and model performance. International Conference on Applications in Electronics Pervading Industry, Environment and Society, Springer International Publishing, Cham, pp.10-17.

Depeursinge, A., Fageot, J., Al-Kadi, O.S., 2017. Fundamentals of Texture Processing for Biomedical Image Analysis: A General Definition and Problem Formulation. ScienceDirect, pp.1-27.

Weissenborn, D., Täckström, O., Uszkoreit, J., 2019. Scaling autoregressive video models. arXiv preprint arXiv:1906.02634.

Dorothy, R., Joany, R.M., Rathish, R.J., et al., 2015. Image enhancement by histogram equalization. International Journal of Nano Corrosion Science and Engineering, 2(4), pp.21-30.

Garcia-Garcia, B., Bouwmans, T., Silva, A.J.R., 2020. Background subtraction in real applications: Challenges, current models and future directions. Computer Science Review, 35, 100204.

Li, Z., Liu, F., Yang, W., Peng, S. and Zhou, J., 2022. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. IEEE Transactions on Neural Networks and Learning Systems, 33(12), pp.6999-7019.

Gu, Y., Wang, S., Yan, Y., et al., 2022. Identification and analysis of emergency behavior of cage-reared laying ducks based on YoloV5. Agriculture, 12(4), p.485.

Jocher, G., Chaurasia, A., Stoken, A., et al., 2022. ultralytics/yolov5: v6.1-tensorrt, tensorflow edge tpu and openvino export and inference. Zenodo.

Santos, C., Aguiar, M., Welfer, D., and Belloni, B., 2022. A New Approach for Detecting Fundus Lesions Using Image Processing and Deep Neural Network Architecture Based on YOLO Model. Sensors, 22, 6441.

Jiang, T., Li, C., Yang, M., and Wang, Z., 2022. An Improved YOLOv5s Algorithm for Object Detection with an Attention Mechanism. Electronics, 11, 2494.

Ren, J. and Wang, Y. (2022). Overview of Object Detection Algorithms Using Convolutional Neural Networks. *Journal of Computer and Communications*, [online] 10(1), pp.115–132.

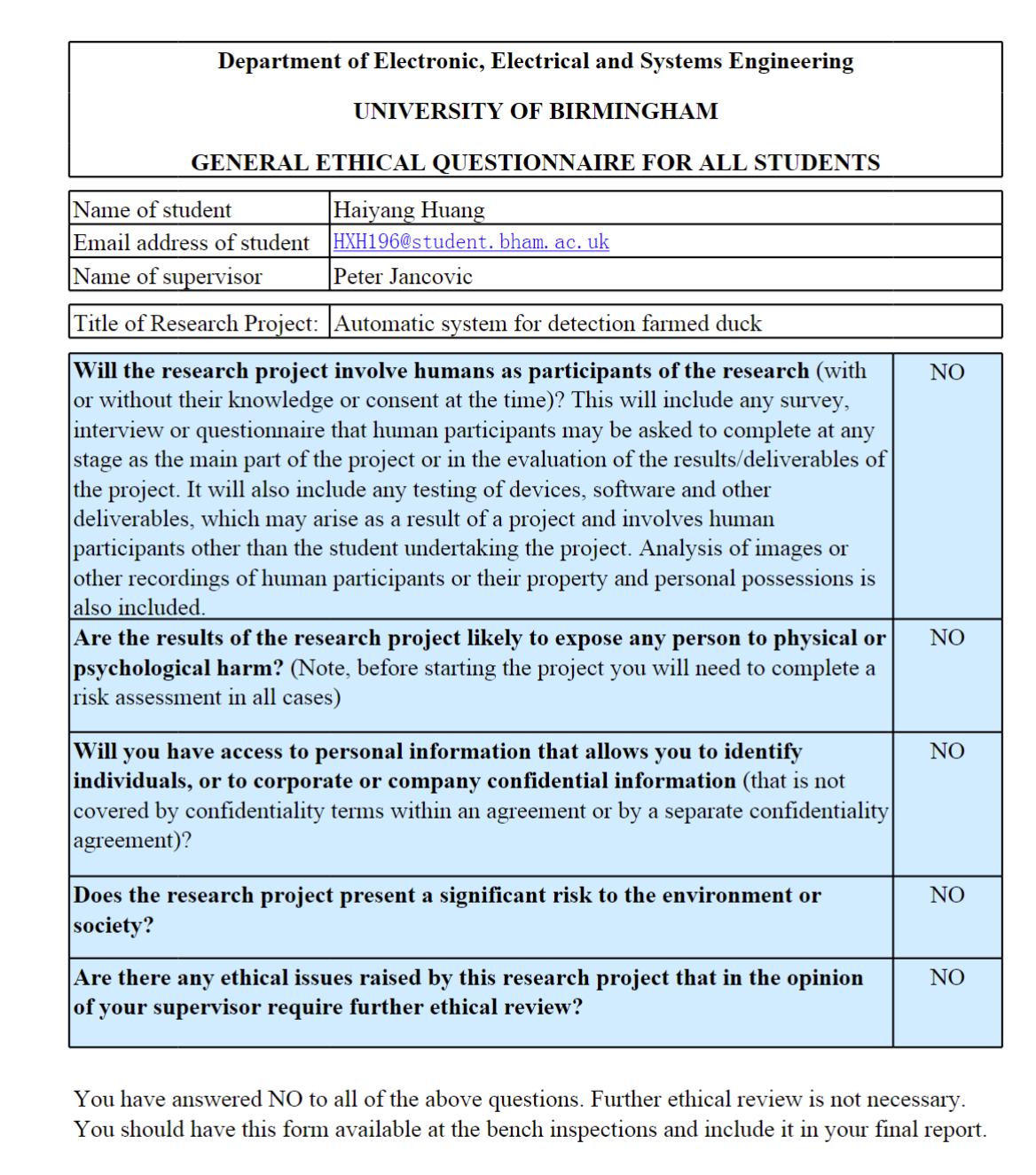
Kandel, I. and Castelli, M., 2020. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. ICT Express, 6(4), pp.312-315.

Hosang, J., Benenson, R., and Schiele, B., 2017. Learning non-maximum suppression. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.4507-4515.

Warden, P. and Situnayake, D., 2019. Tinyml: Machine learning with tensorflow lite on Arduino and ultra-low-power microcontrollers. O'Reilly Media, pp.355-392.

**Appendix A**

Ethical form:



**Appendix B**

The time management:

Gantt chart:

**图表, 瀑布图

描述已自动生成**

**7.3 Appendix C**

The experimental implementation of software structure of yolov5s:

电脑屏幕的照片上有文字

描述已自动生成

Data augmentation

# import all necessary library

from tensorflow.keras.preprocessing import image

import os

import glob

import numpy as np

img\_path = 'C:/Users/29924/Desktop/prepared\_aug/\*'

in\_path = 'C:/Users/29924/Desktop/prepared\_aug/'

out\_path = 'C:/Users/29924/Desktop/output/'

name\_list = glob.glob(img\_path)

datagen = image.ImageDataGenerator(rotation\_range=30)

gen = image.ImageDataGenerator()

data = gen.flow\_from\_directory(in\_path, batch\_size=1, class\_mode=None, shuffle=True, target\_size=(640,640))

np\_data = np.concatenate([next(data) for i in range(data.n)])

datagen.fit(np\_data)

gen\_data = datagen.flow\_from\_directory(in\_path, batch\_size=1, shuffle=False,

                                       save\_to\_dir=out\_path+'rotation\_range\_owl',

                                       save\_prefix='rotate\_owl', target\_size=(640,640))

for i in range(data.n):

    next(gen\_data)

The implementation Python code of detection in Raspberry Pi system:

import smtplib

from email.mime.base import MIMEBase

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

from email import encoders

import subprocess

from time import sleep

from datetime import datetime

import RPi.GPIO as GPIO

import os

import cv2

import random

import numpy as np

import tensorflow as tf

from tensorflow.lite.python.interpreter import Interpreter

from typing import List

# GPIO setup

GPIO.setwarnings(False)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.IN)

# Email parameters

subject = 'Security Alert: A motion has been detected'

bodyText = """\

Hi,

A dangerous species has been detected in your duck farm.

Please check your Gmail sent from raspberry pi security system.

Regards,

AS Tech-Workshop

"""

SMTP\_SERVER = 'smtp.gmail.com'

SMTP\_PORT = 587

USERNAME = 'haiyangh61@gmail.com'

PASSWORD = 'fjuqthmjaieqhouq' # 16-digit App Password

RECEIVER\_EMAIL = 'huanghaiyang720@gmail.com'

# Video filename and path

filename\_part1 = "surveillance"

file\_ext = ".mp4"

filepath = "/home/pi/tflite-custom-object-bookworm-main/"

tflite\_path = "/home/pi/tflite-custom-object-bookworm-main/best.tflite"

threshold\_prob = 0.8

def inference(image: np.ndarray,

model: Interpreter,

threshold: float = 0.8) -> (np.ndarray, List[str], List[float]):

"""

Yolo inference is performed on input image

The result of the detection box is written back to image and returned.

:param model: tensorflow lite model

:param image: image to be detected

:param threshold: minimum threshold of category

:return: detection result includes image, class list and probability list

"""

# Image resize and normalization

input\_data = cv2.resize(image, (640, 640))

input\_data = input\_data / 255.0

input\_data = np.expand\_dims(input\_data, axis=0).astype(np.float16)

# Yolov5s model inference

input\_details = model.get\_input\_details()

output\_details = model.get\_output\_details()

model.set\_tensor(input\_details[0]['index'], input\_data)

model.invoke()

output\_data = model.get\_tensor(output\_details[0]['index'])

# Output data analyzation

num\_boxes = output\_data.shape[1]

boxes, scores, classes = [], [], []

for i in range(num\_boxes):

if output\_data[0, i, 4] > threshold: # Confidence threshold

box = output\_data[0, i, :4]

x, y, w, h = box

# Convert boundingbox coordinates back to the original image's scale

x \*= image.shape[1]

y \*= image.shape[0]

w \*= image.shape[1]

h \*= image.shape[0]

# Convert the bbox coordinates from center coordinates to top-left and bottom-right corner format

boxes.append([x - w / 2, y - h / 2, x + w / 2, y + h / 2])

scores.append(output\_data[0, i, 4])

class\_id = np.argmax(output\_data[0, i, 5:])

classes.append(class\_id)

# Initialization of return values

valid\_scores = []

valid\_class\_names = []

# Non-maximum supression is enabled only when a bounding box exists

if len(boxes) > 0:

indices = tf.image.non\_max\_suppression(boxes,

scores,

max\_output\_size=num\_boxes,

iou\_threshold=0.5,

score\_threshold=threshold)

# Adding boundingbox

for i in indices:

box = boxes[i]

score = scores[i]

class\_id = classes[i]

class\_name = class\_idx2name[class\_id] # Class name

cv2.rectangle(image, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (255, 0, 0), 2)

cv2.putText(image, f'{class\_name} {score:.2f}', (int(box[0]), int(box[1] - 10)),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (255, 255, 255), 2)

valid\_scores.append(float(score))

valid\_class\_names.append(class\_name)

return image, valid\_class\_names, valid\_scores

def preprocess\_video(input\_video\_path: str, output\_video\_path: str, scale\_percent: int = 75) -> (List[str], List[float]):

"""

preprocess video

:param input\_video\_path: input video address

:param output\_video\_path: processed video address ， return empty string if not detected

:param scale\_percent: video scaling

:return: detection result includes image, class list and probability list

"""

# Open the input video

cap = cv2.VideoCapture(input\_video\_path)

if not cap.isOpened():

print("Error: Could not open input video.")

return

# Define the histogram subtraction

backSub = cv2.createBackgroundSubtractorMOG2()

# Get video properties

original\_width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

original\_height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

fps = cap.get(cv2.CAP\_PROP\_FPS)

# Calculate the target size of video

target\_width = int(original\_width \* scale\_percent / 100)

target\_height = int(original\_height \* scale\_percent / 100)

# Define the codec and create VideoWriter object with the target size

# Compress the video frame

fourcc = cv2.VideoWriter\_fourcc(\*'XVID')

out = cv2.VideoWriter(output\_video\_path, fourcc, fps // 2, (target\_width, target\_height), True)

# Process each frame

total\_names = [] # Result of all detection categroies in preprocessed video

total\_scores = [] # The result scores of all detection types in preprocessed video

while True:

ret, frame = cap.read()

if not ret:

break

# Resize the input video frame

resized\_frame = cv2.resize(frame, (target\_width, target\_height),

interpolation=cv2.INTER\_AREA)

# Apply histogram subtraction

fgMask = backSub.apply(resized\_frame)

foreground = cv2.bitwise\_and(resized\_frame, resized\_frame, mask=fgMask)

# Apply histogram equalization on each channel separately if it's a color video

if len(frame.shape) == 3:

# Convert to YUV

yuv\_frame = cv2.cvtColor(foreground, cv2.COLOR\_BGR2YUV)

# Equalize the histogram of the Y channel

yuv\_frame[:, :, 0] = cv2.equalizeHist(yuv\_frame[:, :, 0])

# Convert back to BGR

equalized\_frame = cv2.cvtColor(yuv\_frame, cv2.COLOR\_YUV2BGR)

else:

# If it's a grayscale video, apply histogram equalization directly

equalized\_frame = cv2.equalizeHist(foreground)

equalized\_frame, names, scores = inference(equalized\_frame, interpreter, threshold\_prob)

total\_names.extend(names)

total\_scores.extend(scores)

# Write the preprocessed frame

out.write(equalized\_frame)

# Release everything when done

cap.release()

out.release()

print(f"Processed video saved to {output\_video\_path}")

return total\_names, total\_scores

def send\_email(file\_to\_send: str):

"""

send email

:param file\_to\_send: send file path

"""

message = MIMEMultipart()

message["From"] = USERNAME

message["To"] = RECEIVER\_EMAIL

message["Subject"] = subject

message.attach(MIMEText(bodyText, 'plain'))

with open(file\_to\_send, "rb") as attachment:

mimeBase = MIMEBase('application', 'octet-stream')

mimeBase.set\_payload(attachment.read())

encoders.encode\_base64(mimeBase)

mimeBase.add\_header('Content-Disposition',

f"attachment;filename={os.path.basename(file\_to\_send)}")

message.attach(mimeBase)

text = message.as\_string()

session = smtplib.SMTP(SMTP\_SERVER, SMTP\_PORT)

session.starttls()

session.login(USERNAME, PASSWORD)

session.sendmail(USERNAME, RECEIVER\_EMAIL, text)

session.quit()

print("Email sent")

def capture\_video():

now = datetime.now()

current\_datetime = now.strftime("%d-%m-%Y\_%H:%M:%S")

filename = filename\_part1 + "\_" + current\_datetime + file\_ext

video\_path = filepath + 'newvideo.h264'

converted\_video\_path = filepath + filename

# Use libcamera-vid to record the video

process = subprocess.Popen([

'libcamera-vid',

'-t', '0', # Indefinite recording

'-o', video\_path

])

# Start a loop that checks for motion and stops recording after 5 seconds of no motion

print("Started recording")

try:

while True:

if GPIO.input(11) == GPIO.LOW: # If no motion detected

print("Waiting for no motion")

sleep(5) # Wait for 5 seconds to confirm no motion

if GPIO.input(11) == GPIO.LOW: # If still no motion detected

print("No motion detected for 5 seconds, stopping recording")

process.terminate() # Send SIGTERM to stop libcamera-vid

process.wait() # Wait for the process to terminate

break

sleep(1) # Check for motion every second

except subprocess.SubprocessError as e:

print(f"Error stopping recording: {e}")

# Convert h264 to mp4

result = subprocess.run(['MP4Box','-add', video\_path,converted\_video\_path], stderr=subprocess.PIPE)

if result.returncode != 0:

print("Failed to convert video to mp4")

print(result.stderr.decode())

return None

processed\_video\_path = filepath + "processed\_" + current\_datetime + file\_ext

scale\_percent = 75 # Scale down to 75% of the original size

detected\_names, \_ = preprocess\_video(converted\_video\_path, processed\_video\_path, scale\_percent)

# Clean up the original h264 and mp4 files

remove\_file(video\_path)

remove\_file(converted\_video\_path)

# An empty string is returned if the result is an empty list

return processed\_video\_path if len(detected\_names) > 0 else "", detected\_names

def remove\_file(path):

if os.path.exists(path):

os.remove(path)

print(f"Removed file {path}")

else:

print(f"File {path} does not exist")

# Main code for method call

if \_\_name\_\_ == "\_\_main\_\_":

# Load tflite model

interpreter = tf.lite.Interpreter(model\_path=tflite\_path)

interpreter.allocate\_tensors()

# Class\_name to index

class\_name2idx = {

"duck": 0,

"crow": 1,

"hawk and eagle": 2,

"owl": 3,

"fox": 4,

"rat": 5,

"weasel and stoat": 6,

"badger": 7

}

# Index to class\_name

class\_idx2name = {item[1]: item[0] for item in class\_name2idx.items()}

# Class\_name to bbox color

class\_name2color = {}

for idx in range(len(class\_name2idx)):

r = random.randint(0, 255) # Random value of the red channel

g = random.randint(0, 255) # Random value of the green channel

b = random.randint(0, 255) # Random value of the blue channel

class\_name2color[idx] = (r, g, b)

# Method call

while True:

if GPIO.input(11) == GPIO.HIGH: # If motion detected

print("Motion Detected")

video\_file, detected\_names = capture\_video()

detected\_ids = [class\_name2idx[i] for i in detected\_names]

if video\_file: # If video captured and processed successfully

send\_email(video\_file)

remove\_file(video\_file) # Remove the processed mp4 file

sleep(1) # Sleep to prevent rapid re-checking.