Undergraduate Research and Innovation Scheme (URIS)

☑ Progress Report

☐ Completion Report

(*Please tick "√" as appropriate)*

Section A: The Project and Project Team

1. Project Title

AI (Artificial Intelligence) fireguard: an AI-Driven Fire Prevention System with Advanced Monitoring, Data Analytics, and Computer Vision

Project ID P0053535 **Work Programme** TAG3

2. Project Team

Name	Department	Role
Yuan Haoming	DSAI	Student
Tsang Yuen Hong	AP	Chief Supervisor
		Co-supervisor *(if any)*

3. Project Duration

	Original	Revised	Approval Date of Change Request (Mandatory)
Project Start Date	12.Oct.2024		
Project Completion Date			
Duration (*in months*)	7		

Section B: Report on Project Progress

4. Summary of Objectives Achieved

Objectives as per Proposal	Percentage Achieved (Estimated)
1. Collecting images and label them to be used in machine learning process.	80%
2. Deploy yolov5 and train the model successfully detect the obeject.	100%
3. Designing a program which can generate the optimal escape plan considering parameters such as population distribution, possible routes etc.	50%

5. Research Activities

Introduction

Cigarette smoking is a significant public-health issue, which has numerous negative impacts on health consequences such as cancer and heart disease. In addition, the fire risk is associated with smoking: discarded cigarette butts and smoldering ashes are a common ignition for residential fire. Therefore, our project aimed to develop a cigarette smoking detection model using YOLOv5 architecture. In this report, we will analyze our project process and results and provide improvement of the model. This repository contains the code for tracking and detecting fires and smokes in real-time video using YOLOv5. The project uses a pre-trained YOLOv5 model to identify the presence of fire and smoke in a given video frame and track it through subsequent frames.

Features

1. Real-time performance - Based on the efficient YOLOv3 model, the system can complete the analysis of one frame of image within a few milliseconds.
2. Ease of use - Detailed documentation and sample code are provided to facilitate developers to quickly get started and conduct secondary development.
3. Open source and free - The open source license allows users to freely use, modify and share the code.
4. Scalability - The project is designed flexibly and can be easily integrated with other IoT devices to achieve more comprehensive security protection.

Application scenarios

1. Residential security - When no one is at home, the system can monitor in real time and promptly alert, reducing the risk of fire.
2. Commercial premises - For large public areas such as warehouses and shopping malls, smoke detection can detect potential dangers in advance and prevent losses.
3. Industrial environment - In factory workshops where there may be fire hazards, this system can enhance the safety of workers.
4. Outdoor activities - For camping in the wild or other outdoor activities, smoke detection can help prevent forest fires.

a) Progress made during the reporting period

1. Dataset

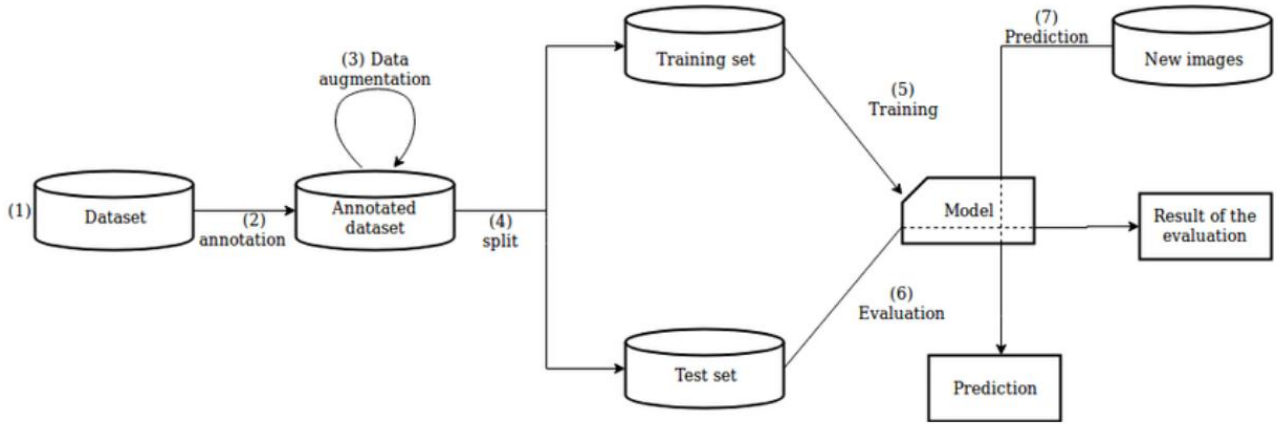
Images source	Numbers
Roboflow	40%
Google images	40%
Kaggle	20%

Collection	Number of images	Target	Annotation
Train	1196	smog and cigarette	different smoking angles and the smoke produced by fires
Valid	2008	smog and cigarette	random sampling ensures diverse scenarios
Test	511	smog and cigarette	no intersection with train images

(data yaml: names: 0 cigarette 1 face-cigarette-smoking nc:3, image size 416x416)

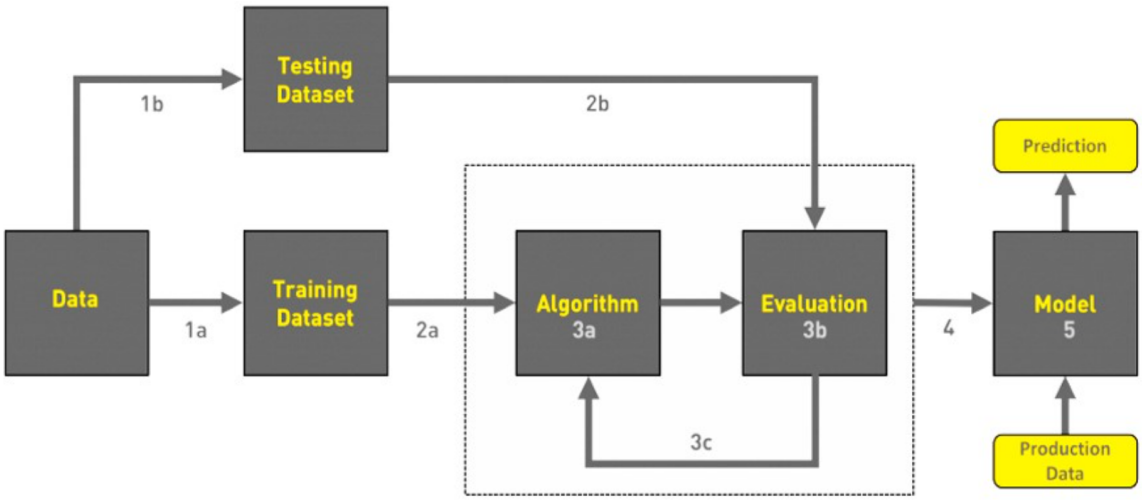
2. Procedure

1. Install YOLOv5
2. CLI Basics
3. Roboflow Universe
4. Preparing a custom dataset
5. Custom Training
6. Validate Custom Model
7. Inference with Custom Model



Workflow of object detection projects

Project Workflow



-> Gathering data
-> Data pre-processing
-> Researching the model that will be best for the type of data
-> Training and testing the model
-> Evaluation

Comparison between CNN and YOLOv5

1 Model Architecture

- **CNN**: Refers to Convolutional Neural Networks, a class of general deep learning models used for image classification, object detection, semantic segmentation, and more.
- **YOLOv5**: A specific object detection model based on CNN architecture, optimized for an end-to-end object detection process.

2 Computational Method

- **CNN (e.g., ResNet, VGG)**: Processes images by progressively applying convolution to extract features, followed by classification.
- **YOLOv5**: Processes the entire image at once, directly predicting multiple object bounding boxes and classes, significantly increasing detection speed.

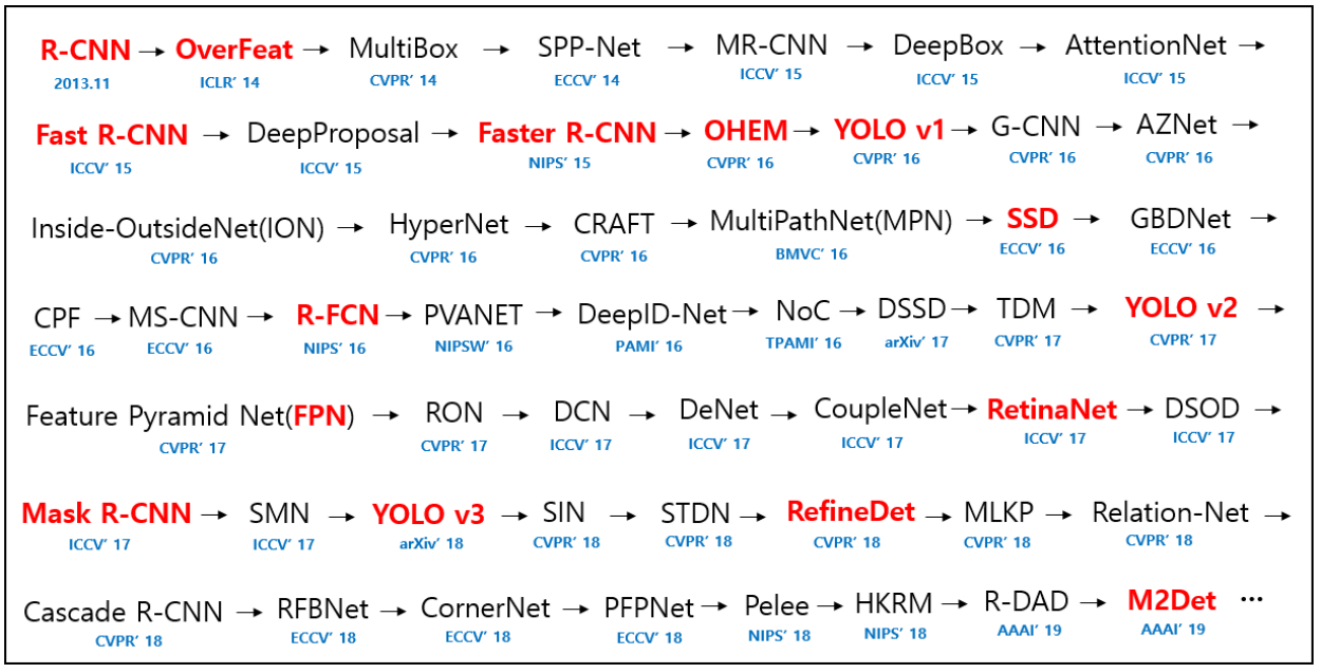
3 Speed vs. Accuracy

Comparison Item	CNN Image Classification	YOLOv5 Object Detection
Computational Method	Layer-wise Feature Extraction	End-to-End Detection
Processing Efficiency	Slower (Multi-step computation)	Faster (Single-pass detection)
Suitable For	Classification tasks (e.g., cat vs. dog)	Object detection (e.g., pedestrian detection)
Applicable Scenarios	High accuracy is needed but can accept slower speeds	Requires fast real-time detection
Computational Resources	Lower (Lightweight models can be optimized)	Higher (Requires GPU acceleration)

YOLOv5 is a specifically optimized object detection model that offers faster processing speeds compared to general CNNs. However, if the task is pure image classification (like recognizing cats and dogs), a standard CNN may be more appropriate. For real-time monitoring and autonomous driving scenarios that require fast detection of multiple objects, YOLOv5 is likely the better choice.

Paper list from 2014 to now(2019)

The part highlighted with red characters means papers that i think "must-read". However, it is **my personal opinion** and other papers are important too, so I recommend to read them if you have time.

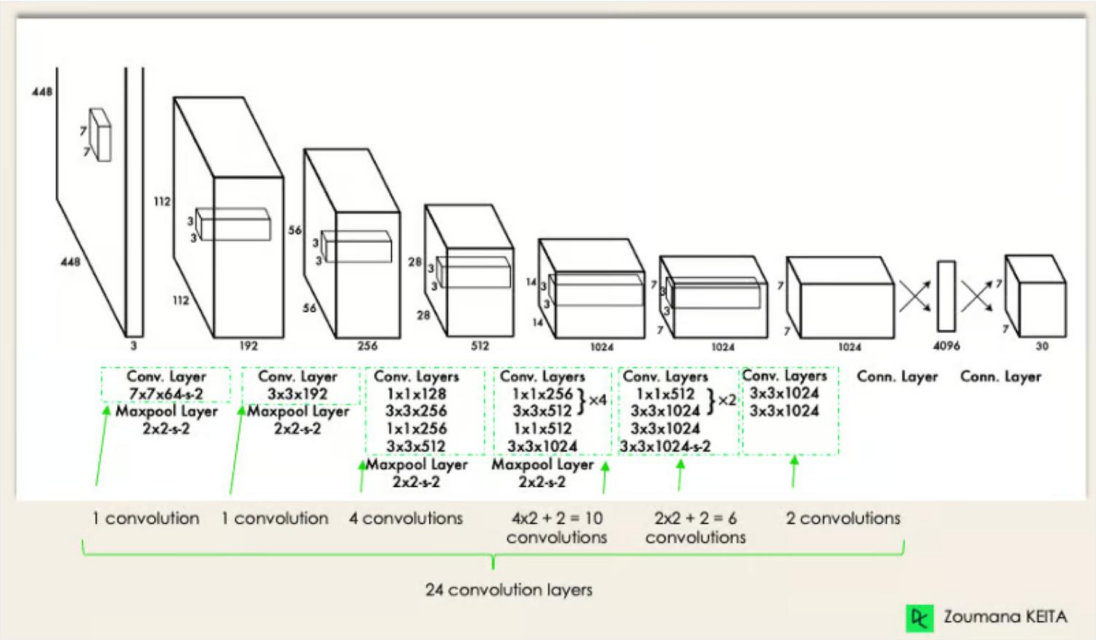


The existing models until 2019, here we use Yolov5 which was released by Ultralytics LLC in June 2020 and is an important version of the YOLO series. It is the first YOLO implemented with PyTorch instead of Darknet, greatly enhancing usability and deployment flexibility. Compared with YOLOv4, YOLOv5 has significant optimizations in inference speed, model size, and training efficiency.

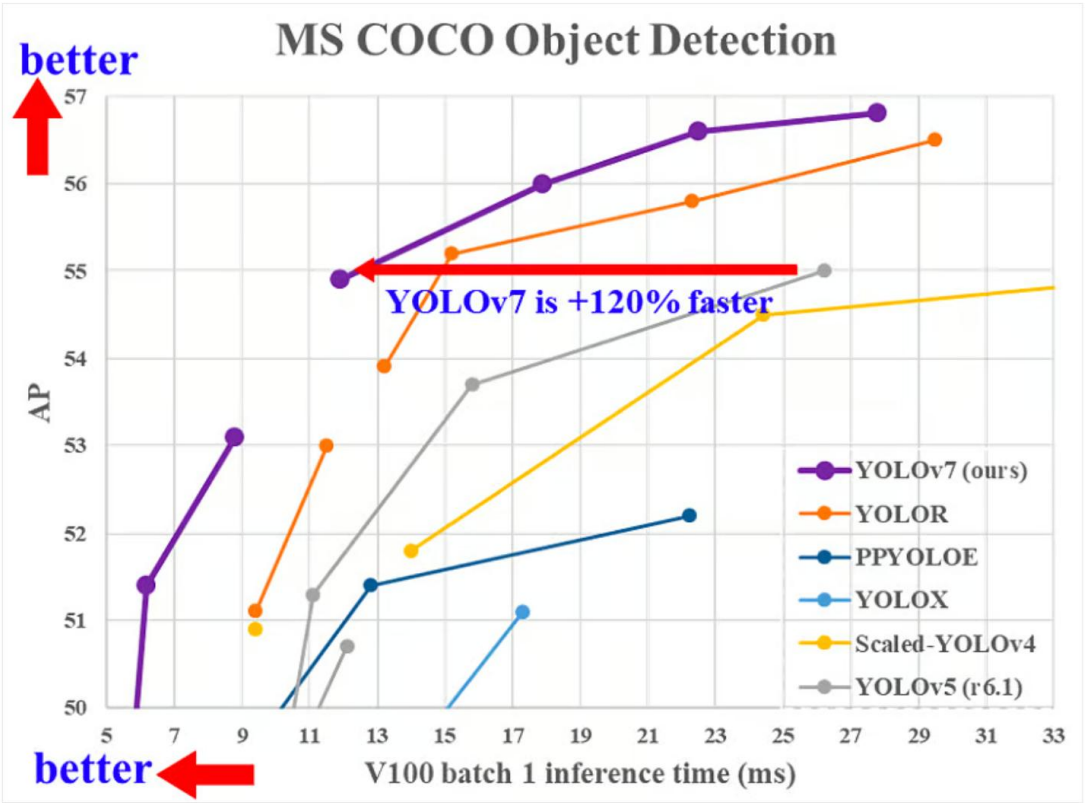
1) create a new file inside the YOLOv5, this is used to store the dataset
2) configure data.yaml
3) train the model (Because we have limited computational power on personal computers, we did other implementations of combinations of batch and epoch as well. The batch and epoch are smaller in other experiments.)

YOLO Architecture

YOLO architecture is similar to GoogleNet. As illustrated below, it has 24 convolutional layers, four max-pooling layers, and two fully connected layers.



YOLO Architecture from the original paper (Modified by Author)



YOLOv5 Implementation Code

```
# Check GPU status
!nvidia-smi

# Clone YOLOv5 repository
!git clone https://github.com/ultralytics/yolov5

# Install dependencies
!pip install -r requirements.txt
!pip uninstall wandb -qy  # Remove deprecated dependency

import torch
from IPython.display import Image, clear_output
print(f'Setup complete. Using torch {torch.__version__}')
```

Listing 1: Environment Setup

```
# Roboflow integration
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="GRuF7VlI4fgSKCtmX0n8")
project = rf.workspace("leo-there").project("smoke-and-cigarette")
dataset = project.version(1).download("yolov5")
```

Listing 2: Dataset Configuration

```
# Training parameters configuration
!python train.py \
  --img 416 \
  --batch 16 \
  --epochs 25 \
  --data /content/smoke-and-cigarette-1/data.yaml \
  --weights yolov5s.pt \
  --name yolov5s_results \
  --cache
```

Listing 3: Model Training

Visualization Code

```
# Install utilities and display results
!pip install utils
from yolov5.utils.plots import plot_results

# Display training metrics
Image(filename='/content/yolov5/runs/train/yolov5s_results2/results.png', width=1000)

# Show validation labels
Image(filename='/content/yolov5/runs/train/yolov5s_results2/val_batch0_labels.jpg',
    width=900)
```

Listing 4: Result Visualization

Compilation Notes

- Requires Python 3.8+ environment with CUDA support
- Requires 10GB+ GPU memory (Tested on NVIDIA RTX 4080)
- Roboflow API key must be replaced for dataset access
- Training time: 45 minutes (25 epochs)

image	416 x 416
Batch	16 images
Epoch	25 rounds
Pre-training weight	yolov5s.pt
equipment	RTX 4080 Laptop GPU

3 Experiment results

3.1 Data analysis

mAP@0.5: measure the degree of overlap between the predicted bounding box of the model and the actual bounding box setting the loU threshold at 0.5.

mAP@0.5:0.95: similar with mAP@0.5, is on IoU from 0.5 to 0.95 calculated mean value.

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}$$

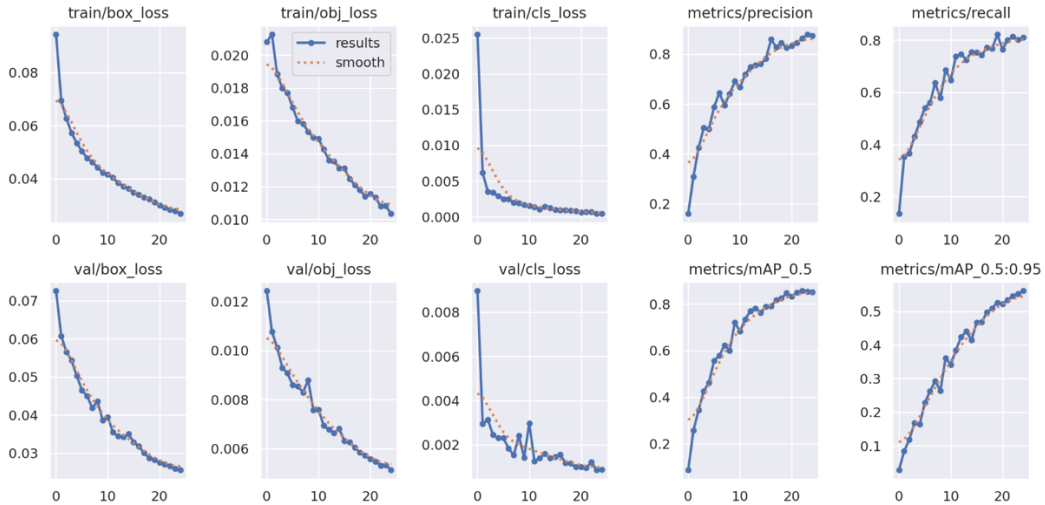


Fig.2. Results

After 25 epochs,

val/cls_loss	0.00088694
val/obj_loss	0.0051253
val/box_loss	0.81015
mAP@0.5	0.8506
mAP@0.5:0.95	0.5617

The model's loss curve shows a consistent, monotonic decline across all components(box/obj/cls). Each drop sharply during the first 5-10 epochs and then gradually decrease their minima by epoch 25, with the validation losses closes tracking the training loss (no significant emerges). In addition, the mAP@0.5 has a steady rise from ~0.00 to ~0.85 and mAP@0.5:0.95 climb smoothly from ~0.00 to ~0.56. This indicates the smooth convergence of both loss and mAP shows that the effective learning without overfitting and the model could accurately detect the object with mAP@0.25 (0.8506).

3.2 Threshold analysis

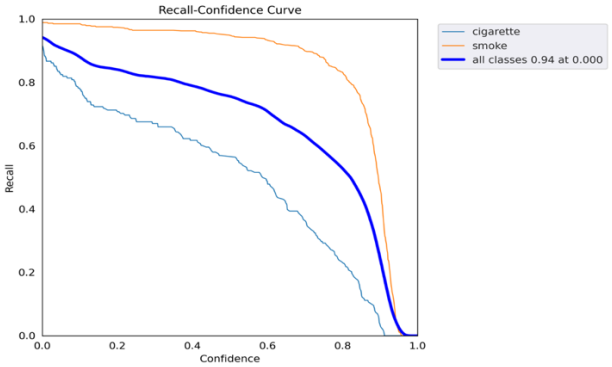


Fig.3.Recall-Confidence Curve

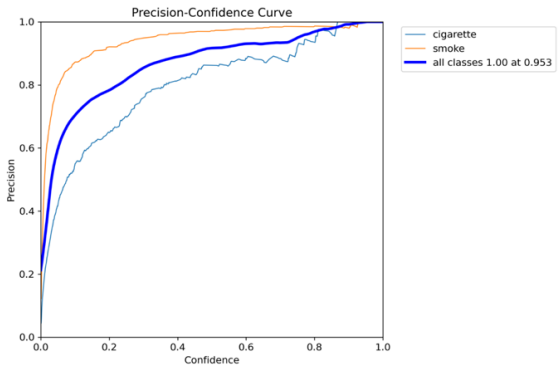


Fig.4.Precision-Confidence Curve

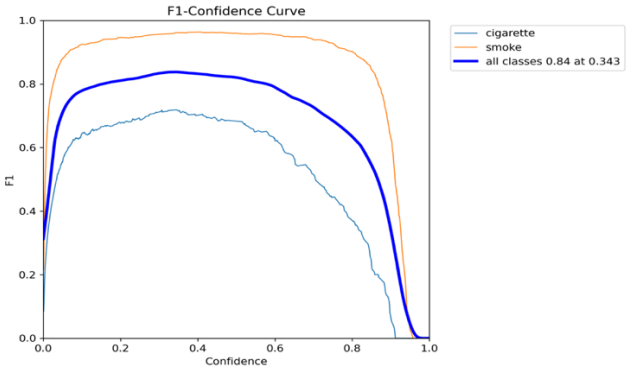


Fig.5.F1-Confidence Curve

Recall: the proportion is that predicted True by the model among all real positive samples
The Recall is highest (0.94) when the confidence threshold set as 0. The overall recall gradually decreased for threshold ranging from 0.00 to 0.85 and dropped sharply for threshold ranging from 0.85 to 0.

Precision: the proportion is that predicted True by the model among all the boxes.

The maximum precision value is 1.00 when the confidence threshold set as 1.00. the predicted probability increased rapidly when the threshold varies between 0.00 and 0.10 and increase gradually from 0.10 to 1.00.

According to F1 formula,

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

the F1-confidecne reach maximum 0.84 at a confidence threshold of 0.343. As shown in Figures 3 and 4, between confidence values of 0.00 and 0.15, precision rises rapidly while recall decreased only slightly, causing F1 score to increase. From 0.15 to 0.343 both Precision and Recall improve slowly which further boost F1; Beyond 0.343 up to 0.80, Recall falls faster than precision increase, so F1 decreases. Finally, the precision increased slowly, and recall decreased rapidly from 0.80 to 1.0. therefore, 0.343 is a balanced point to set the confidence threshold, simultaneously ensuring high precision while also avoiding missed detection targets.

3.3 Precision-Recall

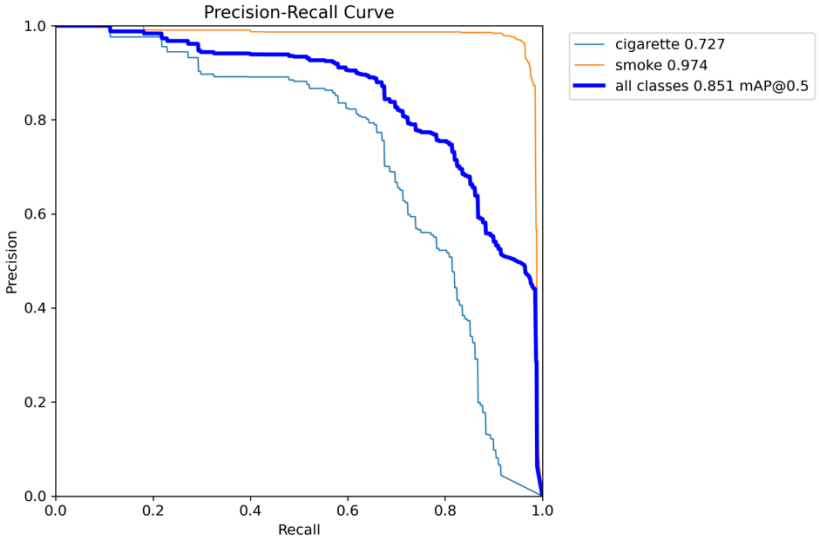


Fig.6.Precision-Recall Curve

The Figure 6 shows the mean Average Precision at IoU threshold on 0.5. The model attain mAP@0.5 of 0.851 including of cigarette AP=0.727 and smoke AP=0.974, showing the good fitting of the model.

3.4 Confusion matrix

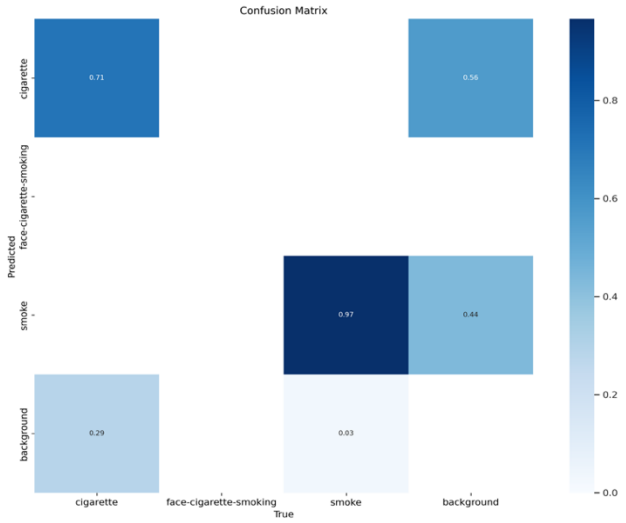


Fig.7 The confusion matrix of object (without face-cigarette-smoking data)

	Predicted True	Predicted False
Actual True	TP	FN
Actual False	FP	TN

According to Recall formula,

$$Recall = \frac{TP}{TP + FN}$$

According to Precision formula,

$$Precision = \frac{TP}{TP + FP}$$

According to Confusion Matrix, 97% of smoke instances are correctly detected with a miss rate of 3%, demonstrating excellent performance on large, diffuse targets. In contrast, only 71% of cigarette target are detected, and among 30% of cigarette could not be detected correctly, likely due to the small size of the object, hand occlusion, and similar color to the cigarette light. conditions. Moreover, 44% of background regions are mistakenly recognized as smoke and 56% of background regions are mistakenly recognized as cigarette. This is because some backgrounds such as fog or white color, are similar to the texture of “smoke” and “cigarette” and recognize as high false positives.

3.5 Case analysis

Listing 1: Implementation.py

```
import sys
import os
import cv2
import torch
import pathlib

# Critical path correction ------------------------------------------------
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
YOLOV5_PATH = os.path.join(BASE_DIR, 'yolov5')
sys.path.insert(0, YOLOV5_PATH)  # Must be placed before other imports

# Temporarily redirect PosixPath to WindowsPath
temp = pathlib.PosixPath
pathlib.PosixPath = pathlib.WindowsPath

from yolov5.models.experimental import attempt_load
from yolov5.utils.general import non_max_suppression, scale_boxes
from yolov5.utils.plots import Annotator, colors
# -------------------------------------------------------------------

def main():
    # Load model
    model = attempt_load(os.path.join(BASE_DIR, 'best.pt'), device='cpu')
    model.eval()

    # Initialize camera
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

    try:
        while True:
            ret, frame = cap.read()
            if not ret:
                break

            # Preprocessing
            img_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            img_tensor = torch.from_numpy(img_rgb).permute(2, 0, 1).float().
                div(255.0).unsqueeze(0)

            # Inference
            with torch.no_grad():
                pred = model(img_tensor)[0]

            # Post-processing
            pred = non_max_suppression(pred, conf_thres=0.5, iou_thres=0.5)

            # Draw results
            annotator = Annotator(frame)
            if pred[0] is not None:
                det = pred[0]
                det[:, :4] = scale_boxes(img_tensor.shape[2:], det[:, :4],
```

```
                    frame.shape).round()
                for *xyxy, conf, cls in reversed(det):
                    label = f'{model.names[int(cls)]} {conf:.2f}'
                    annotator.box_label(xyxy, label, color=colors(int(cls)))

            # Display frame
            cv2.imshow('Fire Detection', annotator.result())
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
    finally:
        cap.release()
        cv2.destroyAllWindows()

    # Restore original PosixPath
    pathlib.PosixPath = temp

if __name__ == "__main__":
    main()
```

Figure 8 on the left side is used to validate the model's accuracy, and Figure 9 on the right shows the detection results produced by the model. Most of images are detected correctly, with recall rate of 0.9375 and average confidence of 0.794. In addition, for the picture in the third column of the fourth row, which do not mark the cigarette, but the model detects it. Therefore, the model has achieved reliability in most detections, but there is still room for improvement in the detection precision and recall rate of "small targets" cigarettes.

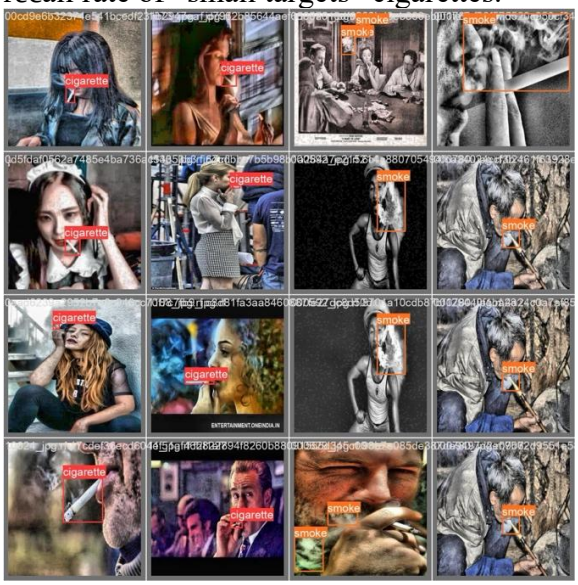


Fig.8. val_batch2_labels

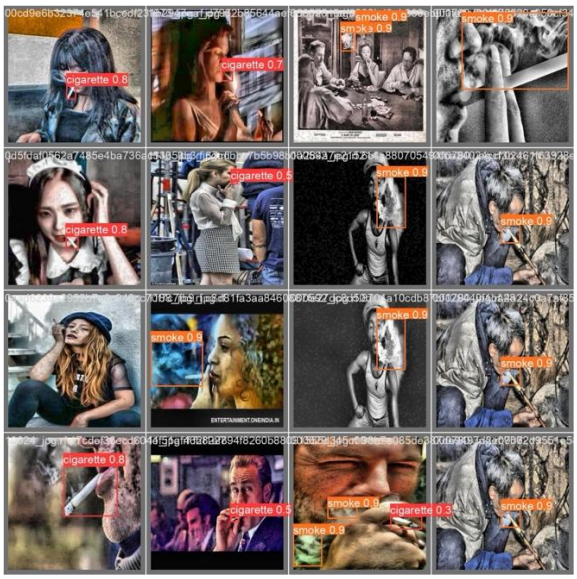


Fig.9.val_batch2_preds

3.6 Difficulties encountered during the reporting period

1, There are 2890 pictures, test 284, train 2007, valid 596. There are numbers of images to label.
2, During the process of training the model, incorrect environment variables were encountered.
3, Improve the accuracy rate of the model

3.7 In conclusion and future work

The model has achieved reliability on the smoke detection ($AP_{smoke} = 0.974$), with precision and recall keeping above 0.9 at the F1 peak of 0.96, indicating most of correct objects. In contrast,

cigarette detection is relatively less robust ($AP_{cigarette} = 0.727$), reaching an F1of 0.70 at precision ≈ 0.7 and recall ≈ 0.65, increasing the confidence threshold to reduce false positives drops cigarette recall below 0.5, but decreasing it to achieve recall >0.9 causing a surge in low precision. The overall mAP@0.5 =0.851 and mAP@0.5:0.95 = 0.561 demonstrates a good general fire smoke detection quality. However, there is still room for improvement in the accuracy of cigarette. In the future work, small-object detection will be further refined by augmenting the training set with cigarette images from diverse viewpoints and under varied lighting conditions. In addition, an alert device will be developed to automatically detect flammable materials and notify security, formulating the escape route in case of fire in the building.

Escape Route Planning Function:

Using C++ language to create an easily designed program and primarily realize the function of generating escape routes according to a series of variables related to building and people. Here is the source code (small part of it, please visit to find the entire source code):

https://github.com/Llleeeo/Smoke-and-Fire-detection-model-and-implementation-

```cpp
#include <iostream>
#include <ctime>
#include <cmath>
#include <algorithm>
using namespace std;

//            min
int minfunction(int timelist[])
{
    int result = timelist[0];
    for (int i = 1; i < 500; i++)
    {
        if (timelist[i] < result)
        {
            result = timelist[i];
        }
    }
    return result;
}

//

struct character
{
    int floor;
    int locationx;
    int locationy;
    int distance;
    int exit;
};

int main()
{
    int floor = 2;

    int people[floor] = {10, 20};

    int exit1[floor] = {0};
    int exit3[floor] = {0};
    int exit5[floor] = {0};

    //
    character charlist2[people[0]];
    character charlist3[people[1]];

    for (int i = 0; i < people[0]; i++)
    {
        charlist2[i].floor = 2;
    }
```

```
50
51      for (int i = 0; i < people[1]; i++)
52      {
53          charlist3[i].floor = 3;
54      }
55      //
56      int location2x[] = {1, 2, 3, 4, 6, 7, 2, 8, 8, 5};
57      int location2y[] = {1, 4, 5, 4, 2, 1, 14, 10, 16, 7};
58
59      int location3x[] = {1, 5, 2, 5, 4, 6, 8, 5, 2, 4, 6, 1,
            3, 3, 3, 5, 4, 8, 7, 1};
60      int location3y[] = {1, 5, 4, 9, 12, 10, 3, 17, 20, 11,
            5, 4, 14, 16, 5, 15, 2, 11, 6, 7};
61      //    cout<<"the location of people on floor 2 are: ";
62      for (int i = 0; i < people[0]; i++)
63      {
64          charlist2[i].locationx = location2x[i];
65          charlist2[i].locationy = location2y[i];
66          //        cout<<"("<<charlist2[i].locationx<<","<<
                charlist2[i].locationy<<")"<<" ";
67      }
68      //    cout<<endl;
69      //    cout<<"the location of people on floor 3 are: ";
70      for (int i = 0; i < people[1]; i++)
71      {
72          charlist3[i].locationx = location3x[i];
73          charlist3[i].locationy = location3y[i];
74          //        cout<<"("<<charlist3[i].locationx<<","<<
                charlist3[i].locationy<<")"<<" ";
75      }
76      //    cout<<endl;
77      int exit12cc;
78      int exit22cc;
79      int exit32cc;
80      int exit13cc;
81      int exit23cc;
82      int exit33cc;
83      int exit12ppeople[1000];
84      int exit13ppeople[1000];
85      int exit22ppeople[1000];
86      int exit23ppeople[1000];
87      int exit32ppeople[1000];
88      int exit33ppeople[1000];
89      int distribution2[1000];
90      int distribution3[1000];
91
92      int exit12ccc;
93      int exit22ccc;
94      int exit32ccc;
95      int exit13ccc;
```

```
96      int exit23ccc;
97      int exit33ccc;
98      int exit12pppeople[1000];
99      int exit13pppeople[1000];
100     int exit22pppeople[1000];
101     int exit23pppeople[1000];
102     int exit32pppeople[1000];
103     int exit33pppeople[1000];
104     int distribbution2[1000];
105     int distribbution3[1000];
106
107     //
108     int timelist[500] = {};
109     int final;
110     for (int i = 0; i < 500; i++)
111     {
112         timelist[i] = 1000;
113     }
114     int min = 10000;
115     int timee;
116     for (int l = 0; l < 500; l++)
117     {
118         for (int m = 0; m < 500000; m++)
119         {
120
121             srand(time(0));
122             //     cout<<"distribution of floor 2 people: ";
123             for (int i = 0; i < people[0]; i++)
124             {
125                 charlist2[i].exit = (rand()) % 3 + 1;
126                 //         cout<<charlist2[i].exit<<" ";
127             }
128
129             //     cout<<endl;
130             //     cout<<"distribution of floor 3 people: ";
131             for (int i = 0; i < people[1]; i++)
132             {
133                 charlist3[i].exit = (rand()) % 3 + 1;
134                 //         cout<<charlist3[i].exit<<" ";
135             }
136             //     cout<<endl;
137             //


138             int exit12c = 0;
139             int exit22c = 0;
140             int exit32c = 0;
141
142             int exit13c = 0;
143             int exit23c = 0;
```

6. **Project Outputs** *(Completion report should include the outputs previously reported in the progress report)*

a) List of publications and other scholarly output

Status (accepted/ published)	Journal	Author List	Publication Title (also provide the hyperlink of the published paper)

b) List of competitions or awards

Date	Competition/Award	Organizer	Result (also provide the hyperlink of competition/ award)

c) List of conferences attended with presentation

Date	Conference	Organizer	Presentation Title (also provide the hyperlink of conference)

d) Innovation and technology development
(e.g. source codes, applications, prototypes)

https://github.com/Llleeeo/Smoke-and-Fire-detection-model-and-implementation-
please visit this website link to find complete implementation code, parameter and weights after training, and yolov5 training code.

Note: Please insert more rows as necessary.

Section C: Declaration

7. Declaration by the Student

☑ I DECLARE the research project is my original work except for source material explicitly acknowledged; and I UNDERTAKE to maintain the highest ethical standard and academic integrity throughout the research. I am aware I will be held responsible for any disciplinary actions.

☑ I DECLARE the research project has not been submitted for more than one purpose, such as an assignment for another course. I understand that the URIS project may be a precursor to a final-year project (FYP). I acknowledge that it should not form part of the FYP or contribute to its grade.

Name: Yuan Haoming Signature: Yuan Haoming Date: 14/05/2025

Section D: Rating and Endorsement by Chief Supervisor

8. Rating and Comments

a) Overall Rating

The progress of the project is recommended as:

*outstanding/ very satisfactory / satisfactory/ unsatisfactory.

b) Comments

_____

_____

_____

_____

_____

_____

_____

9. Research Grant

Information available in Projects and Grants Management System

Balance:	HK$

☐ I DECLARE the grant is only used by student on items which are directly relevant to the project concerned. In addition, the University's prevailing purchasing policy shall be followed.

Name: _____ Signature: _____ Date: _____

Chief Supervisor

* Please delete as appropriate.

Only project rated "Outstanding" is eligible for the Best URIS Research Project Award upon project completion.

Section E: Rating and Endorsement by Departmental / School Research Committee

10. Rating and Comments

Note: If the Chief Supervisor is the D/SRC Chair, the FRC/School Board Chair is requested to complete this Section.

a) Overall Rating

The progress of the project is recommended as:

*outstanding/ very satisfactory / satisfactory/ unsatisfactory.

If the rating differs from the Chief Supervisor's rating as in Section D, please provide the reason for the difference in the comment section below.

b) Comments

_____

_____

_____

_____

_____

_____

_____

Name: _____ Signature: _____ Date: _____

Chair, D/SRC

* Please delete as appropriate.

Only project rated "Outstanding" is eligible for the Best URIS Research Project Award upon project completion.

Section F: Approval by Undergraduate Research and Innovation Committee

11. Rating and Approval

a) Overall Rating

The progress of the project is recommended as:

*outstanding/ very satisfactory / satisfactory/ unsatisfactory.

* Please delete as appropriate.

(For GS use)

GS actions:

- ☐ Eligibility check before URIC Chair decision
- ☐ GS record updated
- ☐ Decision notice sent to student, Chief Supervisor and D/SRC Chair

Updated in Mar 2025