# Fire and Smoke Detection using YOLO through Kafka

Kai-Yu Lien[1], Jung-Chun Liu[1], Yu-Wei Chan[2], and Chao-Tung Yang[1,3]

[1] Department of Computer Science, Tunghai University, No. 1727, Sec. 4, Taiwan Boulevard, Xitun District, Taichung, 407224, Taiwan, ROC
lienkaiyu@gmail.com, jcliu@thu.edu.tw,
[2] Department of Information Management, Providence University, Taiwan
ywchan@gm.pu.edu.tw
[3] Research Center for Smart Sustainable Circular Economy, Tunghai University, No. 1727, Sec. 4, Taiwan Boulevard, Xitun District, Taichung, 407224, Taiwan, ROC
ctyang@thu.edu.tw

**Abstract.** This study is based on deep learning techniques, which compare various detection algorithms and implement the suitable one for firework detection. The considered factors include streaming, speed, accuracy, and portability. Through a detection algorithm, it can simultaneously identify the positions of smoke and fire, providing subsequent control of fire or other applications. After comparison, we plan to perform detection results in a streaming manner, where only real-time detection of the captured scene is carried out. The system can notify people or teams in need of notification via the network.

**Keywords:** YOLO, Deep learning, Machine learning, Fire and Smoke detection, Kafka

## 1 Introduction

Global warming causes climate change, increasing forest fire risks in flammable areas like forests and grasslands. Forests and grasslands are essential to the ecosystem and habitats of animals and plants. Forest fires destroy vegetation and habitats, causing severe ecological impact. Smoke and fire detection systems detect smoke and fire quickly, improving response time to prevent forest fires, and allowing authorities to take measures to protect the environment and public safety. As a result, we plan to build a system based on vision to detect fire and smoke. With the development and advancements in manufacturing technology and the evolution of algorithms of artificial intelligence, components can be made smaller, faster, more accurate, and power-efficient advantages, even making them into portable devices, let image detection has already had a wide range of applications. In this paper, we use the YOLO model as our primary development environment. We will explore the efficiency and accuracy differences by comparing the differences and applicability of various algorithm versions. We will challenge the detection of relatively complex situations - smoke and fire sources

- as a system and combine it with real-time streaming, Kafka data access, and edge devices for further exploration.

## 2 Literature Review

### 2.1 Machine learning

Machine learning is a rapidly evolving field within AI that involves training computers to learn from data inputs and optimize algorithms for accurate predictions on new, unknown data. It has subfields such as supervised learning, deep learning, and reinforcement learning, which involve training models on labeled datasets, training neural networks, and using reward systems to guide optimal decision-making in different environments.

### 2.2 Image recognition (detection)

Image recognition is a type of machine learning technology in artificial intelligence, which mainly enables machines to automatically recognize objects in images, such as objects, people, animals, scenes, texts, and so on. Currently, the common related technologies and methods for image recognition include convolutional neural networks, object detection, YOLO, image segmentation, feature extraction, PyTorch, and TorchVision.

### 2.3 Kafka

Kafka is an open-source distributed event streaming platform developed by Apache, with the ability to process and store data streams reliably and efficiently. It is designed to handle large-scale real-time data streams by providing a unified, high-throughput, low-latency platform for data processing. Kafka operates on a publish-subscribe model, where producers publish messages or events to topics, and consumers subscribe to those topics to receive the messages. The platform is designed to handle massive amounts of data and can scale horizontally by adding more brokers, which are the servers that store and manage the data, also Kafka can be used to manage data from multiple sources, including sensors, web applications, and databases. The platform can process these streams of data quickly and efficiently, making it ideal for real-time analytics and decision-making.

### 2.4 YOLO algorithm

YOLOv5 has diverse applications and support like Pytorch, CUDA acceleration, and Tensor support. This paper focuses on comparing YOLO (You Only Look Once) and extending its functionality for future data applications. This includes comparing algorithmic differences between various versions and their accuracy and efficiency. YOLO is a fast and accurate object detection system that simultaneously performs comprehensive inferences on entire images through regions and bounding boxes. Unlike sliding window detection or region proposal methods, it can see the entire image during training and testing. Therefore, YOLO can conceal globally interrelated class information within its encoding.

In this study, we chose three models from the YOLO series: YOLO v5, v7, and v8. The architecture is based on Region-Based Convolutional Neural Network Layers and is composed of two parts: feature extraction and detection. The feature extractor is a convolutional neural network responsible for extracting features from the image, with down-sampling for feature extraction from local to global regions. The detection head is responsible for mapping these features to object positions and categories and restoring them to their original size.

## 3 Data Collection and Experimental Results

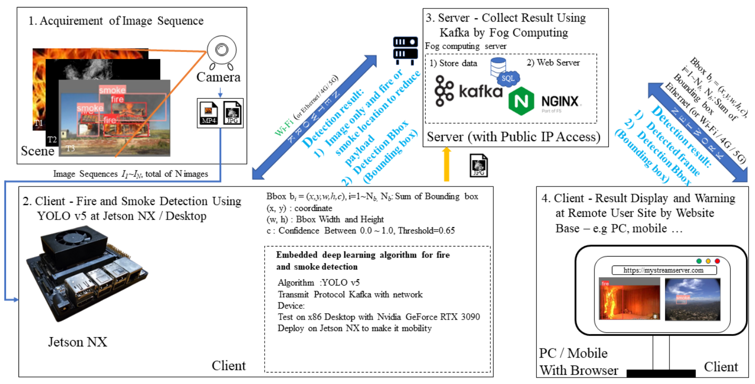
1. Acquirement of Image Sequence
Camera
smoke
fire
T1
T2
T3
Scene
MP4
JPG
Image Sequences $I_1$~$I_N$, total of N images
Wi-Fi (or Ethernet / 4G / 5G)
NETWORK
Detection result:
1) Image only and fire or smoke location to reduce payload
2) Detection Bbox (Bounding box)
3. Server - Collect Result Using Kafka by Fog Computing
Fog computing server
1) Store data
2) Web Server
SQL
kafka
NGINX
Server (with Public IP Access)
JPG
Bbox $b_i = (x,y,w,h,c)$, $i$=1~$N_b$, $N_b$: Sum of Bounding box
Ethernet (or Wi-Fi / 4G / 5G)
NETWORK
Detection result:
1) Detected frame
2) Detection Bbox (Bounding box)
2. Client - Fire and Smoke Detection Using YOLO v5 at Jetson NX / Desktop
Jetson NX
Client
Bbox $b_i = (x,y,w,h,c)$, i=1~$N_b$, $N_b$: Sum of Bounding box
(x, y) : coordinate
(w, h) : Bbox Width and Height
c : Confidence Between 0.0 ~ 1.0, Threshold=0.65
Embedded deep learning algorithm for fire and smoke detection
Algorithm :YOLO v5
Transmit Protocol Kafka with network
Device:
Test on x86 Desktop with Nvidia GeForce RTX 3090
Deploy on Jetson NX to make it mobility
4. Client - Result Display and Warning at Remote User Site by Website Base – e.g PC, mobile …
https://mystreamserver.com
PC / Mobile With Browser
Client

**Fig. 1.** System architecture

This research begins with the development of various algorithms on the x86 computing platform, and training and testing these algorithms in a Windows environment. The trained models are analyzed and compared to evaluate the differences between the algorithms. The advantages and disadvantages of each algorithm are compared quantitatively. The most suitable algorithm is selected for further development and integration with real-time streaming, data storage in Kafka, and the use of portable EDGE devices such as Jetson Xavier NX. The training process includes adjusting model size, training frequency, training content, parameter tuning, etc. to increase the accuracy of the model. The system architecture of this study is shown in Fig. 1.

We want to detect smoke and fire using image detection, as traditional methods are expensive and limited in range.In this study, we used a sample dataset consisting of two categories, fire, and smoke to perform feature extraction and labeling on it. By comparing the model performance using different feature extraction and labeling methods, we found their impact on model accuracy and generalization ability. Additionally, we analyzed the impact of data distribution on model performance, including the impact of different sample quantities and category ratios on model performance. The simplified research process is divided into data collection, processing, model training, and benchmarking the result of models using metrics in machine learning such as Accuracy, Loss, Confusion Matrix, etc.

### 3.1 Training

In this study, the various versions of standard-sized models were first trained using the recommended training frequency in the paper, and the accuracy of the initial training of each version was compared to determine which version would be used as the primary detection method for this study, then performed feature extraction and labeling on the dataset and investigated the impact of feature extraction and labeling on model performance and analyzed the impact of data distribution on model performance.

### 3.2 Server Setup

We have set up four virtual machines running Ubuntu 22.04 Server version on an ESXi server, each with 4 cores, 8GB RAM, and 100G storage. Three of the virtual machines are used as containers for Kafka, and one is used for web hosting. During the setup process, we limited the disk speed of the three Kafka virtual machines to 100 IOPS to simulate the operation of Kafka on multiple machines and test its write performance. As we mainly transmit image data, there is almost no delay in reading and writing, and the data can be quickly transmitted to the host. The images can be displayed on the web page with a low response time (about 3 seconds).

### 3.3 Data Collection

We use Google Search, Kaggle datasets, and crawler to collect approximately 3,100 images of fire and smoke, then use data augmentation techniques to expand the dataset to 15,000 images for training, then split into training and testing sets, typically with 80% of the data for training and 20% for testing. The testing set may be further divided into either the entire validation set or a 10% validation set and the remaining testing set. However, due to the limited number of samples in the dataset, we enhanced the images and split them into training (85%), and validation (15%) sets with a pre-processing model size of 640x640, and keep the quantity of fire and smoke samples were balanced as much as possible, which is approximately 8000 images per class used for training on average. To meet the requirements, In the data augmentation process, we applied rotation enhancement to images taken from different angles to address the focus blur issue and then trained the dataset with default 300 iterations on YOLO v5, v7, and v8, and compared their difference.

### 3.4 Experimental Results

The actual training results were observed to be similar to our expectations, with better accuracy in fire detection and not-so-good results in smoke detection, although still better than expected. The reason may lie in the fact that smoke is more complicated, Moreover, its irregular shape and various factors such as color, lighting conditions during photography, and concentration (translucent), which may easily be confused with the environmental background can lead to uncertainty.

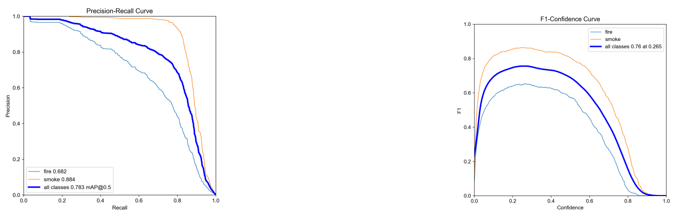

**Fig. 2.** F1 curve and PR curve.

In the YOLO v5 section, we trained the model an image size of 640. Our results showed that the precision (P) and recall rate (R) for smoke and fire were 0.9 and 0.7, respectively. Although the accuracy of smoke detection was 0.2 mAP higher than fire detection, the average accuracy of fire detection during actual testing was around 0.6, while the accuracy of smoke detection was around 0.4 when it was correctly classified. Unfortunately, it was often observed during actual testing that smoke was not detected and ignored by the model. In Fig. 2, we can see that even in the small fire or smoke is still detectable in the long distance of the small pixel.

We will use YOLOv5, an object detection model that is intended for real-time streaming video and can use in a real-time environment, It offers five models that come with differing depth and width multiple to control the network. These differences have a significant impact on the performance and suitability of the YOLOv5 model for different environments. In the following table, we can see the detection speeds of various sizes of YOLO v5 models. We tested these models on an x86 PC platform with an image processing core of RTX3090. As we are running our models on embedded platforms, we tested YOLO v5 on an Nvidia Jetson NX to compare the original processing speed without any accelerators. As a result, the speed of the YOLO v5m model starts to decrease rapidly but accuracy does not significantly improve. since high resolution can lead to bandwidth waste and transmission delays, and a high frame rate is not necessary, we limit our image so the system can handle about one frame per second. This approach is suitable and sufficient for our experimental environment and practical use, where a difference of a second in fire status is not significant. For this reason, we implement the YOLO v5m mode and plan to optimize its detection speed in the future.

**Table 1.** TABLE I. SPEED BETWEEN YOLO V5 MODELS

| Model | Image | Video | Video |
|---|---|---|---|
| | (RTX 3090) | (Stream @ RTX 3090) | (Native Jetson NX) |
| V5n | 16ms | 7.8ms (128FPS) | 590ms (1.7 FPS) |
| V5s | 12ms | 7.0ms (142FPS) | 760ms (1.3 FPS) |
| V5m | 16ms | 8.0ms (125FPS) | 2000ms (0.5 FPS) |
| V5l | 15ms | 10.1ms (99FPS) | 3500ms (0.3 FPS) |
| V5x | 22ms | 13.5ms (74FPS) | 5000ms (0.2FPS) |

**Table 2.** ACCURACY BETWEEN YOLOV5 MODELS

| Scenes | Fire | | | Smoke | | |
|---|---|---|---|---|---|---|
| Model | Accuracy | S | B | Accuracy | S | B |
| YOLO v5m (Enforced) | 0.750̃.98 | V | O | 0.70̃.98 | V | O |
| YOLO v5n | 0.30̃.72 | O | O | 0.30̃.89 | O | V |
| YOLO v5s | 0.20̃.77 | O | O | 0.30̃.92 | O | O |
| YOLO v5m | 0.30̃.85 | O | O | 0.40̃.95 | O | X |
| YOLO v5l | 0.40̃.89 | V | V | 0.40̃.96 | V | V |
| YOLO v5x | 0.40̃.88 | O | V | 0.40̃.95 | O | V |

$$Accuracy \quad = \quad (TP + \ TN)/(TP + FP + FN + TN). \tag{1}$$

We utilize the recall and precision rates as metrics to evaluate the proportion of correctly identified accuracy among all the objects present as the ground truth and estimate (1) and check the lowest and highest result in each test on validation, test, strange and live datasets, Table II shows that increasing the size of the model beyond YOLO v5m does not lead to a significant improvement in accuracy. Instead, it results in higher costs in terms of image processing time and memory resources, while also reducing the detection speed. We chose to optimize the YOLO v5m model to achieve an accuracy of 0.98 while maintaining a speed of around 1 FPS. This approach reduces the payload of transmission and processing and minimizes the risk of excessive data duplication, which would rapidly fill up our storage container and increase the load on the Kafka server. As a result, we can collect useful data continuously without requiring too many resources.

## 4 Conclusion and Future Works

With the continuous development of machine learning detection technology, we should not blindly pursue the latest version but rather conduct experiments and tests to compare and find the development environment and models that are suitable for our application system. In this experiment, the latest version achieved the best results for this task and its algorithm was optimized, but the resulting more closed code and relatively difficult code modification also bring challenges. Although this process may be time-consuming and may use older technology, it is necessary to ensure the practicality and real-time effects of the system. Afterward, we can focus on optimizing and applying the model, as well as studying the new versions.

Currently, this study has successfully implemented firework detection on the YOLO series model. And through Kafka, photos can be transmitted instead of text only. Although more time was spent on encoding and non-optimal encoders resulted in some wasted resources during transmission, this model can accurately recognize different forms of fires, including large flames and small fires, and can also detect the presence of smoke. Even occurs outdoors and the source is hidden or uncleared, the algorithm can provide different prompts through additional functions when it detects thick smoke by adding the smoke detection model function that prevents the system from misjudging some light sources similar to flames to improve the accuracy of detection results. The original images are also kept for subsequent modifications and verification.

## References

[1] . Redmon, S. Divvala, R. Girshick, A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Computer Vision and Pattern Recognition, May 2016.

[2] Online]. Available: https://github.com/ultralytics/yolov5

[3] Online]. Available: https://github.com/ultralytics/ultralytics.

[4] -K Kim and C-S Jeong, ”LARGE SCALE IMAGE PROCESSING IN REAL-TIME ENVIRONMENTS WITH KAFKA”, Proceedings of the 6th AIRCC International, 2017

[5] . Frizzi, R. Kaabi, M. Bouchouicha, J.-M. Ginoux, E. Moreau and F. Fnaiech, ”Convolutional neural network for video fire and smoke detection,” in IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Oct. 2016.

[6] . Namozov, Y. I. Cho, ”An Efficient Deep Learning Algorithm for Fire and Smoke Detection with Limited Data,” Advances in Electrical and Computer Engineering, vol. 18, no. 4, pp. 121-128, 2018.

[7] . Bradski, F. T. von Kleist-Retzow, T. Tiemerding, P. Elfert, O. C. Haenssler ”The OpenCV Library. Dr. Dobb’s Journal of Software Tools,” Journal of Computer and Communications, vol. 4, no. 3, pp. 25-33, March 2016.

[8] . Viswanatha, R. K. Chandana and A. Ramachandra, ”Real Time Object Detection System with YOLO and CNN Models: A Review,” Computer Vision and Pattern Recognition (cs.CV); Image and Video Processing (eess.IV), Jul 2022.

[9] -Y Wang, A.. Bochkovskiy, H-Y .M Liao , ”YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” Computer Vision and Pattern Recognition, Jul 2022.