Third International Conference on "Materials, Computing and Communication Technologies"
International Journal of Scientific Research in Computer Science, Engineering and Information Technology | ISSN : 2456-3307 (www.ijsrcseit.com)

Real Time Fire Detection System Based on Deep Learning Method

Dr. E. Punarselvam[1], Mrs. P. Bhuvaneshwari[2], Mr. B. Deepak[3], Mr. M. Bhuvaneshwaran[3], Mr. K. A. Gowtham[3], Mr. A. Ragul[3]

[1]Professor & Head, [2]Assistant Professor, [3]Student

Department of Information Technology, Muthayammal Engineering College (Autonomous), Rasipuram-637408, Tamil Nadu, India

Article Info
Volume 8, Issue 7
Page Number : 113-120

Publication Issue :
May-June-2022

Article History
Accepted: 01 June 2022
Published: 20 June 2022

ABSTRACT

Due to the numerous shapes, textures, and colorations of fires, fire detection is a tough task. The conventional photo processing technique is predicated closely on artificial capabilities, which isn't always universally relevant to all wooded area scenarios. In order to clear up this problem, the deep learning technology is applied to learn and extract features of fires adaptively. However, the limited learning and perception ability of individual learners is not sufficient to make them perform well in complex tasks. Furthermore, learners tend to focus too much on local information, namely ground truth, but ignore global information, which may lead to false positives. In this paper, a singular ensemble learning method is proposed to detect fires in different scenarios. Firstly, two individual learners Yolov5 and EfficientNet are incorporated to perform fire detection process. Secondly, another individual learner EfficientNet is responsible for learning global information to void false positives. Finally, detection results are made primarily based totally at the selections of three learners. Experiments on our dataset show that the proposed method improves detection performance. After predicting the results, our system sends an alerting message to the concerned authority. So this process of fire detection becomes more effective and digitalized.

Keywords: YOLOV5, OPENCV, FIRE DETECTION, ALERT MESSAGE.

I. INTRODUCTION

DEEP LEARNING

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. Deep learning algorithms seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features.

The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep learning excels on problem domains where the inputs are analog. Meaning, they are not a few quantities in a tabular format but instead are images of pixel data, documents of text data or files of audio data. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

II. PROPOSED SYSTEM

YOLO is an acronym that stands for You Only Look Once. We are employing Version 5, which was launched by Ultralights in June 2020 and is now the most advanced object identification algorithm available. It is a novel convolutional neural network (CNN) that detects objects in real-time with great accuracy. This approach uses a single neural network to process the entire picture, then separates it into parts and predicts bounding boxes and probabilities for each component. These bounding boxes are weighted by the expected probability. The method "just looks once" at the image in the sense that it makes predictions after only one forward propagation run through the neural network.

2.1. ALGORITHM

YOLO v 5 (You Only Look Once version 5)

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real- time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.

This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

YOLO v 5 Model Architecture

An object detector is designed to create features from input images and then to feed these features through a prediction system to draw boxes around objects and predict their classes.

The YOLO model was the first object detector to connect the procedure of predicting bounding boxes with class labels in an end to end differentiable network.

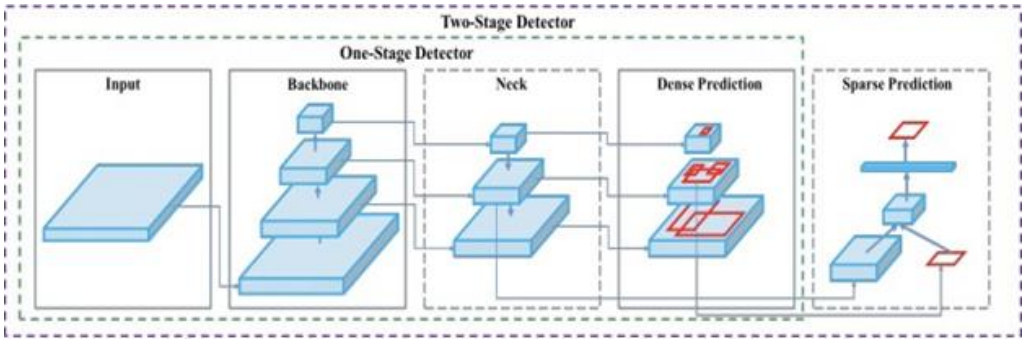


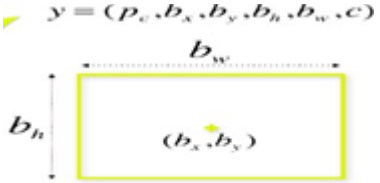
Figure 1 Object Detection Process.

As shown in Figure 1 the YOLO network consists of three main pieces.

Model Backbone is mainly used to extract important features from the given input image. In YOLO v5 the CSP— Cross Stage Partial Networks are used as a backbone to extract rich in informative features from an input image.

Model Neck is mainly used to generate feature pyramids. Feature pyramids help models to generalized well on object scaling. It helps to identify the same object with different sizes and scales.

Model Head is mainly used to perform the final detection part. It applied anchor boxes on features and generates final output vectors with class probabilities, objectness scores, and bounding boxes.



2.2. ADVANTAGES:

It is about 88% smaller than YOLOv4 (27 MB vs 244 MB). It is about 180% faster than YOLOv4 (140 FPS vs 50 FPS). It is roughly as accurate as YOLOv4 on the same task.

III. SYSTEM ARCHITECTURE

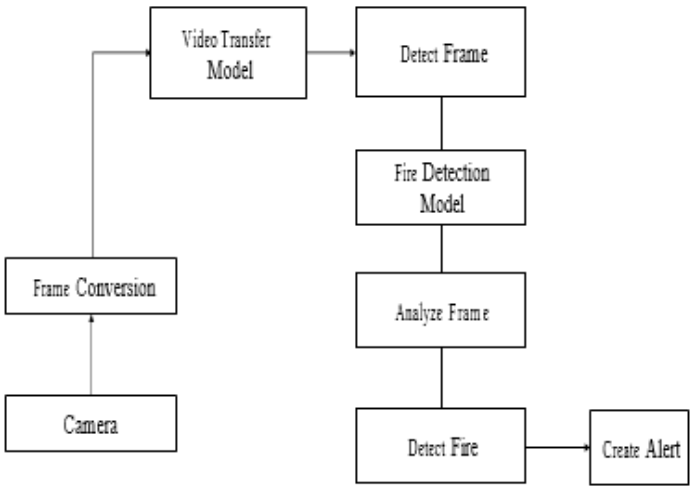


Figure 2 System Architecture

IV. METHODOLOGY

An automated fire detection system will detect the fire in real time. Next, it will identify the fire, and the alert signal is given to the respective authority is processed as shown in Figure 2.

CUSTOM DATASET

Developing a custom model to detect your objects is an iterative process that includes gathering and categorizing photographs, labelling your objects of interest, training a model, deploying it in the world to make predictions, and then collecting examples of edge situations to repeat and improve. We are using 1000 images with label to train our model.

TRAINING YOLOv5 MODEL

A pretrained model is selected to start training. We are using YOLOv5x a large pretrained model available in the YOLOv5. Converting the dataset into YOLO format, create a YOLOv5 YAML configuration file, and host it for importing into your training script (Figure 3 & Figure 4).

DETECTING USING MODEL

The YOLO model stared training by specifying dataset, epochs, batch-size, image size. Once the best result is obtained the model will stop training and the result will get stored in the train folder (Figure 5 to Figure 8).

DETECTING FIRE

Applying detector over live image and video stream with the help of camera to detect the fire. The model will use the best weight obtained from the training to detect the fire accurately (Figure 9).

FIRE ALERT

When the fire is detected the alert message is given to the required concern using alert API.

V. IMPLEMENTATION

System implementation is the important stage of project when the theoretical design is tuned into practical system. The main stages in the implementation are as follows:

- Planning
- Training
- System testing and
- Change over Planning

Planning is the first task in the system implementation. Planning means deciding on the method and the time scale to be adopted. At the time of implementation of any system people from different departments and system analysis involve. They are confirmed to practical problem of controlling various activities of people outside their own data processing departments. The line managers controlled through an implementation

coordinating committee. The committee considers ideas, problems and complaints of user department, it must also consider;

- The implication of system environment
- Self-selection and allocation form implementation tasks
- Consultation with unions and resources available
- Standby facilities and channels of communication

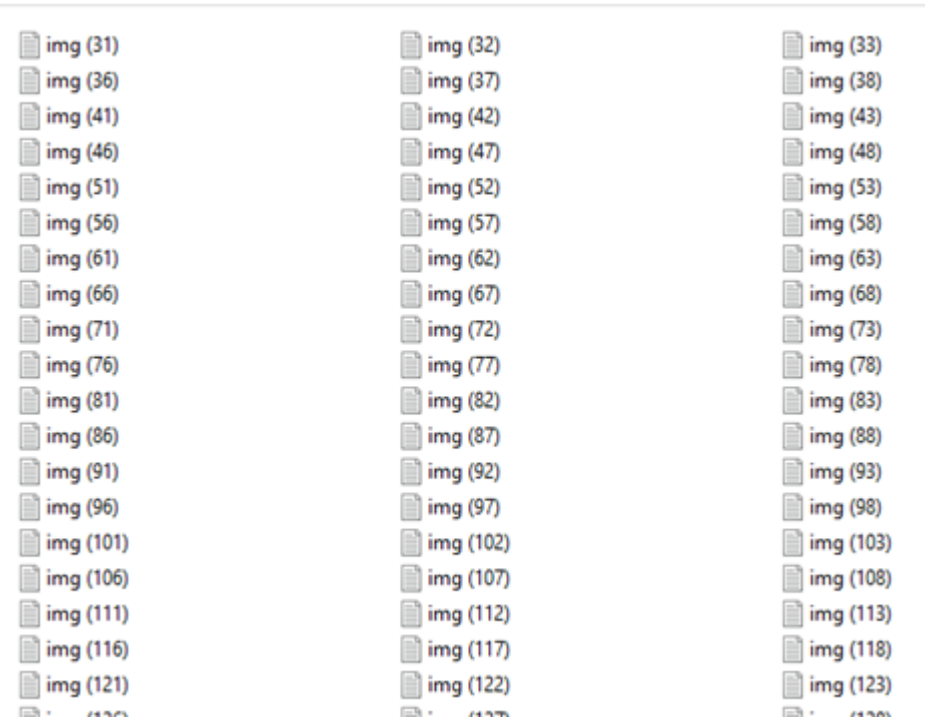


Figure 3 Dataset Label

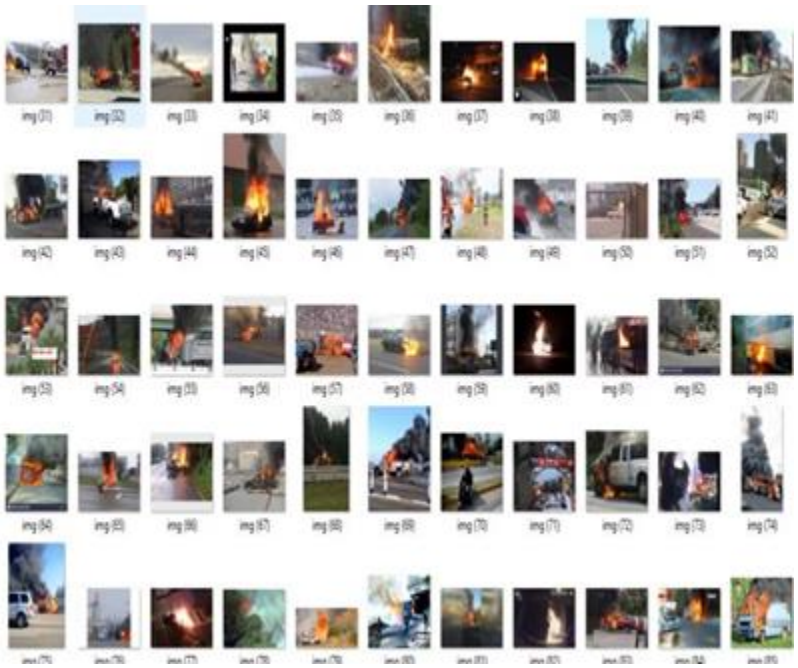


Figure 4 dataset Images

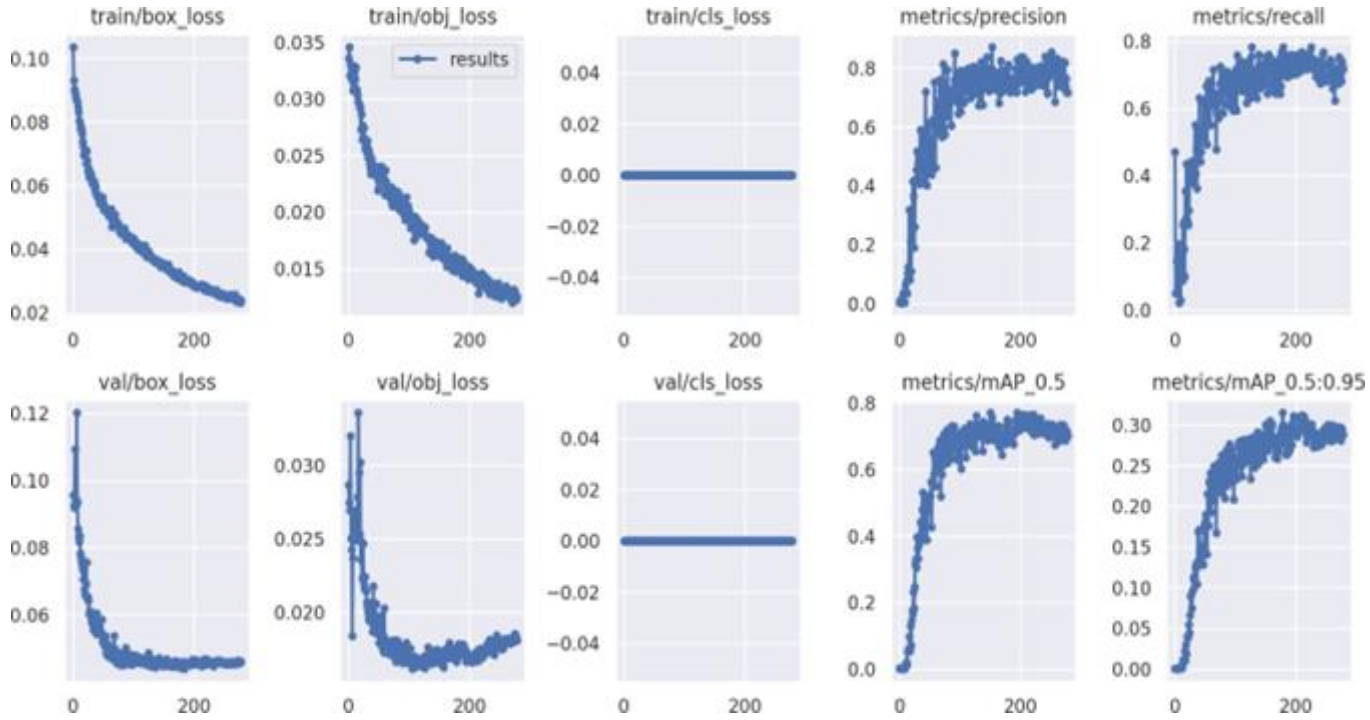
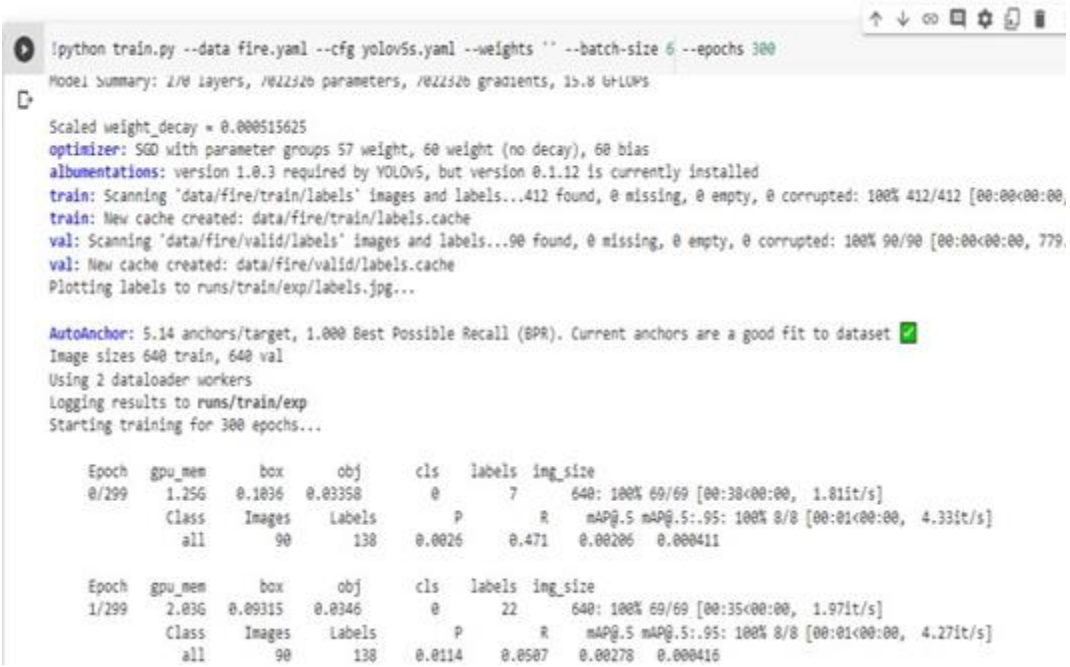
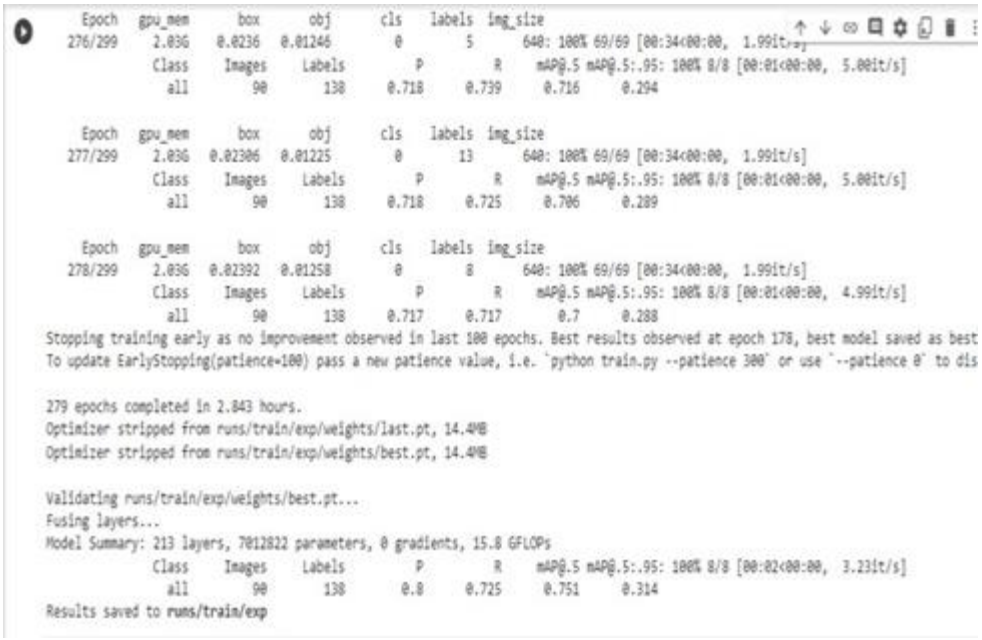


Figure 5 Graphical representation of model



```
!python train.py --data fire.yaml --cfg yolov5s.yaml --weights '' --batch-size 6 --epochs 300
Model Summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.8 GFLOPs

Scaled weight_decay = 0.000515625
optimizer: SGD with parameter groups 57 weight, 60 weight (no decay), 60 bias
albumentations: version 1.0.3 required by YOLOv5, but version 0.1.12 is currently installed
train: Scanning 'data/fire/train/labels' images and labels...412 found, 0 missing, 0 empty, 0 corrupted: 100% 412/412 [00:00<00:00,
train: New cache created: data/fire/train/labels.cache
val: Scanning 'data/fire/valid/labels' images and labels...90 found, 0 missing, 0 empty, 0 corrupted: 100% 90/90 [00:00<00:00, 779.
val: New cache created: data/fire/valid/labels.cache
Plotting labels to runs/train/exp/labels.jpg...

AutoAnchor: 5.14 anchors/target, 1.000 Best Possible Recall (BPR). Current anchors are a good fit to dataset ✅
Image sizes 640 train, 640 val
Using 2 dataloader workers
Logging results to runs/train/exp
Starting training for 300 epochs...

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     0/299     1.25G   0.1036   0.03358         0         7       640: 100% 69/69 [00:38<00:00,  1.81it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:01<00:00,  4.33it/s]
                 all         90        138     0.0026      0.471    0.00206   0.000411

     Epoch   gpu_mem       box       obj       cls    labels  img_size
     1/299     2.03G   0.09315    0.0346         0        22       640: 100% 69/69 [00:35<00:00,  1.97it/s]
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:01<00:00,  4.27it/s]
                 all         90        138     0.0114     0.0507    0.00278   0.000416
```

Figure 6 Initiation of training



```
     Epoch   gpu_mem       box       obj       cls    labels  img_size
   276/299     2.03G   0.0236   0.01246         0         5       640: 100% 69/69 [00:34<00:00,  1.99it/s]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:01<00:00,  5.00it/s]
                   all         90        138      0.718      0.739      0.716      0.294

     Epoch   gpu_mem       box       obj       cls    labels  img_size
   277/299     2.03G   0.02306   0.01225         0        13       640: 100% 69/69 [00:34<00:00,  1.99it/s]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:01<00:00,  5.00it/s]
                   all         90        138      0.718      0.725      0.706      0.289

     Epoch   gpu_mem       box       obj       cls    labels  img_size
   278/299     2.03G   0.02392   0.01258         0         8       640: 100% 69/69 [00:34<00:00,  1.99it/s]
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:01<00:00,  4.99it/s]
                   all         90        138      0.717      0.717        0.7      0.288
Stopping training early as no improvement observed in last 100 epochs. Best results observed at epoch 178, best model saved as best
To update EarlyStopping(patience=100) pass a new patience value, i.e. `python train.py --patience 300` or use `--patience 0` to dis

279 epochs completed in 2.843 hours.
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB

Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model Summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 8/8 [00:02<00:00,  3.23it/s]
                   all         90        138        0.8      0.725      0.751      0.314
Results saved to runs/train/exp
```

Figure 7 Best fit obtained

```
!python detect.py --source 7.jpg --weights ./runs/train/exp/weights/best.pt

detect: weights=['./runs/train/exp/weights/best.pt'], source=7.jpg, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.4
YOLOv5 🚀 6f282a7 torch 1.10.0+cu111 CUDA:0 (Tesla K80, 11441MiB)

Fusing layers...
Model Summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
image 1/1 /content/fire_detection_yolov5/7.jpg: 448x640 1 fire, Done. (0.030s)
Speed: 0.5ms pre-process, 29.7ms inference, 1.6ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp7
```

Figure 8 Result saved

Figure 9 Labeled input

VI. CONCLUSION

With the recent rapid development of computer graphics and hardware, VR technology has entered a relatively mature application state and has been widely applied in fire detection. Image-type fire flame recognition methods are unconventional for early fire flame recognition. This study has investigated image-type fire flame recognition using a support vector machine and rough set theory. In practical applications, the RS method is sensitive to noise and poor in fault tolerance and generalization, while YOLOv5 has strong antinoise capability and generalization performance. This paper has presented an YOLOv5 fire flame recognition algorithm and designed a classifier of fire flame image recognition. By building a model with YOLOv5, the parameters of which have been optimized, this study used more feature variables as criteria, represented the static and dynamic features of flames, selected and extracted the most effective feature subsets, fused the features of fire flame images extracted, and reduced required training to recognize and extract flame regions.

VII. REFERENCES

[1]. X. Huang, L. Du: "Fire Detection and Recognition Optimization Based on VR Video Image" Procedia Eng., vol. 211, pp. 441–446, Jan. 2021..

[2]. Q.-X. Zhang, G.-H. Lin, Y.-M. Zhang, G. Xu, and J.-J. Wang, "Wildland forest fire smoke detection based on faster R-CNN using synthetic

[3]. Mazzeo, R. Paciello, N. Pergola, and V. Tramutoli, "RST-FIRES, an exportable algorithm for early-fire detection and monitoring: Description, implementation, and field validation in the case of the MSG-SEVIRI sensor," Remote Sens. Environ., smoke images," Procedia Eng., vol. 211, pp. 441– 446, Jan. 2021..

[4]. P. Li and W. Zhao, "Image fire detection algorithms based on convolutional neural networks," Case Stud. Thermal Eng., vol. 19, Jun. 2020, Art. no. 100625..

[5]. R. Hou, M. Pan, Y. Zhao, and Y. Yang, "Image anomaly detection for IoT equipment based on deep learning," J. Vis. Commun. Image Represent., vol. 64, Oct. 2019, Art. no. 102599.

[6]. S. S. Esfahlani, "Mixed reality and remote sensing application of unmanned aerial vehicle in fire and smoke detection," J. Ind. Inf. Integr., vol. 15, pp. 42–49, Sep. 2019.

[7]. A. Hackner, H. Oberpriller, A. Ohnesorge, V. Hechtenberg, and G. Müller, "Heterogeneous sensor arrays: Merging cameras and gas sensors into innovative fire detection systems," Sens. Actuators B, Chem., vol. 231, pp. 497–505, Aug. 2018.

[8]. Z. Lin, F. Chen, Z. Niu, B. Li, B. Yu, H. Jia, and M. Zhang, "An active fire detection algorithm based on multi-temporal FengYun-3C VIRR data," Remote Sens. Environ., vol. 211, pp. 376–387, Jun. 2018.

[9]. K. Muhammad, J. Ahmad, and S. W. Baik, "Early fire detection using convolutional neural networks during surveillance for effective disaster management," Neurocomputing, vol. 288, pp. 30– 42, May 2018.Minghua Wan and Zhihui Lai. Generalized discriminant local median preserving projections (gdlmpp) for face recognition. Neural Processing Letters, 49(3):951–963, 2018. vol. 186, pp. 196–216, Dec. 2016.

[10].A. Koltunov, S. L. Ustin, B. Quayle, B. Schwind, V. G. Ambrosia, and W. Li, "The development and first validation of the GOES early fire detection (GOES-EFD) algorithm," Remote Sens. Environ., vol. 184, pp. 436–453, Oct. 2016.

[11].T. Zhang, M. J. Wooster, and W. Xu, "Approaches for synergistically exploiting VIIRS I- and M-Band data in regional active fire detection and FRP assessment: A demonstration with respect to agricultural residue burning in eastern China," Remote Sens. Environ., vol. 198, pp. 407–424, Sep. 2016.

[12].Y. Peng and Y. Wang, "Real-time forest smoke detection using handdesigned features and deep learning," Comput. Electron. Agricult., vol. 167, Dec. 2015, Art. no. 105029.

[13].S. Garcia-Jimenez, A. Jurio, M. Pagola, L. De Miguel, E. Barrenechea, and H. Bustince, "Forest fire detection: A fuzzy system approach based on overlap indices," Appl. Soft Comput., vol. 52, pp. 834–842, Mar. 2015.

[14].V. B. Nemirovskiy, A. K. Stoyanov, and D. S. Goremykina, ". Tomsk Polytech. Univ., vol. 40, no. 5, pp. 740–745, 2015.

[15].G. Marbach, M. Loepfe, and T. Brupbacher, "An image processing technique for fire detection in video images," Fire Saf. J., vol. 41, no. 4, pp. 285–289, Jun. 2015.