HIGHER SCHOOL OF COMPUTER SCIENCE

4TH-YEAR MULTIDISCIPLINARY PROJECT REPORT

Fire and smoke detection on edge devices

Submitted By :
BOURAHLA Amayas
BAHA Meriem
DOU Baha Houcine
SLIMANI Medjda Rihab

Submitted To :
B.KHALDI

Contents

List of Figures

1 Introduction

Fire accidents are the most commonly occurring disasters nowadays caused significant damage to human lives and social properties.Since fire is one of the most severe types of accidents, there is always a need to improve fire detection capabilities.The detection of smoke will help for the early detection of fire and helps to reduce the loss that caused from this kinds of accidents.
Precise, fast, and portable solutions to detect fire have received increased attention among the masses. Previous fire detection approaches mainly use physical sensors, such as smoke, thermal, and flame sensors, to detect fires. However, these sensors might cause false alarms as the thresholds to trigger the fire alarms are difficult to set. Besides, the response time of these sensors greatly depends on the sampling rate of the discrete signals. Thus a low sampling rate will cause considerable delays in the fire alarm.
The technologies underlying fire and smoke detection systems play a crucial role to address these issues by ensuring and delivering optimal performance in modern surveillance environments.
The objective of this 4Th-year project is to create a system that employs deep learning models to detect fire/smoke in real-time .The system can be used with a variety of hardware, including the Jetson Nano and Raspberry Pi with the Intel Movidius DL Stick.
The technology accurately detects fires and smoke using video-based deep learning algorithms providing valuable information for emergency responders.
Real-time fire and smoke detection provided by the proposed system will enable prompt and effective actions, minimizing fire damage and saving lives.Potential uses for the system include industrial settings, building fire protection, and forest fire detection.
Overall, this project demonstrates the potential of deep learning models for real-time fire detection. The system's use on devices like the Jetson Nano or Raspberry Pi underlines the adaptability of deep learning models and the potential for these systems to be deployed on many platforms.

2 State of the art

CNNs have proven to be highly effective in image classification and other computer vision tasks, including fire detection. Several researchers have explored the use of CNN architectures for fire detection, each proposing unique approaches and achieving notable results.

CNN architecture based on SqueezeNet was developed which successfully detected fires at an early stage with a high accuracy rate and low false positive rate. also a adaptive prioritization mechanism was introduced in the surveillance system to enhance fire detection efficiency [6]. faster region-based CNN combined with Long short-term memory (LSTM) was utilized to detect suspected fire regions and classify the presence of fire. decision fusion from successive video sequences was incorporated to improve the final decision accuracy [4].
Other researchers explored additional factors such as motion intensity rate, sensor data, image data, color, shape variation, and motion analysis to enhance fire detection performance. They utilized various models, including AdaBoost, neural networks, fine-grained patch classifiers, and dynamic/static feature analysis, to detect and locate fires accurately [7].
To address challenges in fire detection under extreme weather conditions. An efficient CNN-based system was proposed that considered uncertain surveillance scenarios with smoke, fog, and snow and focused on early smoke detection in typical and foggy IoT environments using deep convolutional neural networks [1].
A multiscale feature extraction technique and a channel attention mechanism were added to achieve a balance between accuracy, model size, and speed. When compared to less efficient ways, this methodology was faster and smaller and yet achieved good accuracy [5].
Transfer learning techniques were also employed by researchers who fine-tuned pre-trained InceptionV3 and MobileNetV2 models using curated datasets. Transfer learning proved beneficial, especially when working with limited datasets [7] .

3 Experiment Setup

3.1 Data set Description

The "Wildfire and smoke" data set, sourced from Roboflow contains 3,255 images and 6,825 annotations . With all annotations falling under the "Fire" and "smoke" classes , the data set focuses mainly on fire-related study. On average, the images have a size of 0.05 megapixels, ranging from 0.02 to 2.07 megapixels. The median image ratio indicates a predominantly wide composition, with dimensions of 299x250. It is significant that the "Wildfire and smoke" data set includes images taken in both bright and low-light conditions, illustrating the range of illumination conditions. This fluctuation in lighting conditions enhances the realism of the data set by reflecting actual fire events. The data set improves the robustness of fire and smoke detection models, enabling them to function well in a variety of lighting settings by considering various lighting scenarios. In Figure 1, we show a few sample images from the dataset.

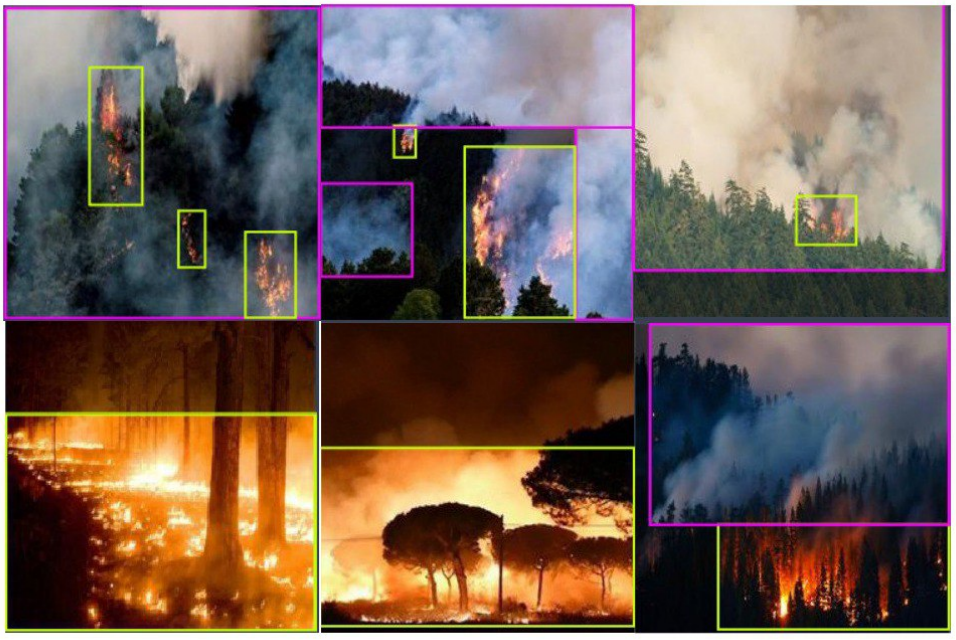

Figure 1: Few images from dataset.

The dataset is composed of a train, test and validation set, all contained in a folder, where there's a file 'data.yaml' that descries the dataset and its different classes.each folder (train, test, validation) contains images and their corresponding bounding boxes and labels which are contained in text files.

3.2 Hardware Setup

The Jetson Nano is a powerful embedded computer designed for AI and machine learning applications. It features a Quad-core ARM Cortex-A57 CPU running at 1.43 GHz and a 128-core NVIDIA Maxwell GPU. The Jetson Nano is equipped with 2 GB of LPDDR4 RAM and supports various deep learning frameworks. It has multiple I/O interfaces, including USB, HDMI, and Ethernet, enabling easy connectivity with peripherals. The Jetson Nano also provides hardware acceleration for AI tasks, making it capable of real-time inference and processing of high-definition video streams.

Figure 2: jetson nano

The Raspberry Pi 3 is an affordable and versatile single-board computer with a quad-core 1.2 GHz Broadcom processor, 1 GB of RAM, built-in Wi-Fi, and Bluetooth connectivity. It supports various operating systems and features a 40-pin GPIO header for easy external component integration. The Raspberry Pi Camera Module, compatible

with the Raspberry Pi 3, offers high-quality image and video capture, connects through the CSI port, and can be controlled using software libraries.

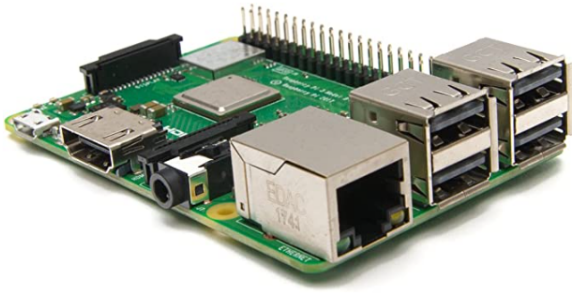

Figure 3: raspberry

We use the Raspberry Pi to capture the camera feed using the Raspberry Pi Camera module and streams it to the Jetson Nano. Then we train the model with a local machine, deploy the trained model to the Jetson Nano. The Jetson Nano processes the frames in real time and outputs the fire and smoke coordinates

4 Methodology

4.1 Architecture

The architecture involved in the fire and smoke detection system combines the capabilities of the Raspberry Pi, Jetson Nano, and the YOLO (You Only Look Once) algorithm. The Raspberry Pi serves as the initial component, equipped with the Raspberry Pi Camera module for capturing the camera feed. The Jetson Nano, on the other hand, acts as the processing unit with its powerful GPU and AI capabilities. The YOLO algorithm, renowned for its real-time object detection performance, is employed for accurate fire and smoke detection. By dividing the input image into a grid, YOLO predicts bounding boxes and class probabilities for objects within each grid cell, enabling precise localization and classification of fire and smoke instances. The combined architecture allows for the streaming of camera feed from the Rasp-

berry Pi to the Jetson Nano, where the YOLO model performs inference on the preprocessed frames, swiftly identifying fire and smoke occurrences.

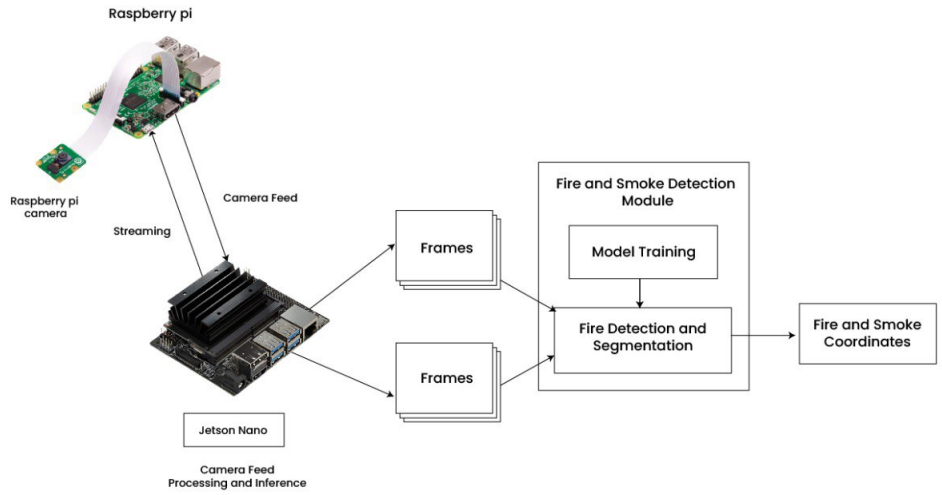

Figure 4: The hardware architecture.

This setup provides an efficient and reliable solution for real-time fire and smoke detection, making it applicable in various safety and monitoring applications.

To enhance the streaming and inference efficiency of the fire and smoke detection system, a modification was implemented to optimize the processing of frames. Instead of performing inference on every frame within a second, the system now selectively chooses a representative frame for inference while maintaining consistent bounding box predictions across subsequent frames.

This improvement was motivated by the understanding that fire instances typically do not change significantly within a short time interval, such as one second. By selecting a single representative frame for inference, the system reduces the computational load and achieves faster processing without compromising the accuracy of fire and smoke detection.

The chosen representative frame serves as a reference point, capturing the initial appearance and location of the fire or smoke. The bounding box predicted for this frame is then applied to all subsequent frames

within the one-second interval. This approach eliminates the need for redundant inference on similar frames and allows for a more streamlined processing pipeline.
By leveraging this enhancement, the system significantly improves the real-time streaming capabilities, ensuring efficient utilization of computational resources. Moreover, since the bounding box remains consistent across frames, the system maintains accurate fire and smoke localization throughout the continuous video stream. This reliability in bounding box prediction enables prompt and reliable detection, minimizing the risk of false negatives and enhancing the system's effectiveness in fire and smoke monitoring applications.

4.2 Models

We tested various deep learning models for fire and smoke detection including CNN,YOLOv4, YOLOv5, YOLOv7, and YOLOv8. Each model was evaluated based on its performance, accuracy, and computational efficiency. We also explored the possibility of training a dist-YOLO model for calculating distances. However, a significant challenge we encountered was the unavailability of a suitable dataset tailored for this specific task [9].

4.2.1 CNN

The CNN architecture of the model includes convolutional layers, pooling layers, and dense layers. These layers allow the model to extract meaningful features from the input images and make accurate predictions

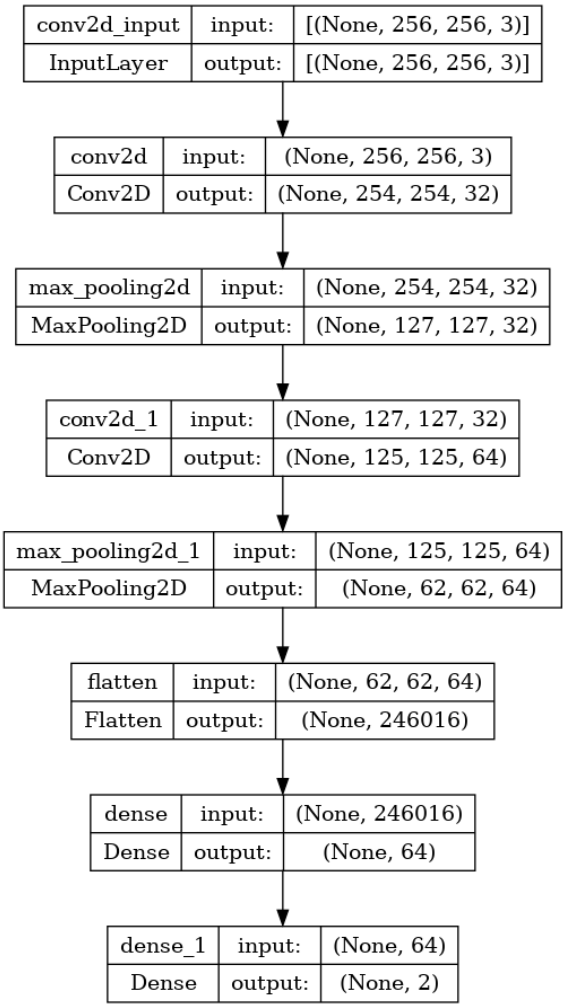

Figure 5: CNN architecture

The model is compiled using the Adam optimizer and cross-entropy loss, which are common choices for classification tasks. The model is trained on the augmented dataset, iteratively adjusting its parameters to minimize the loss and improve accuracy. The training progress is monitored and evaluated using both training and validation data. By visualizing the training and validation accuracy over epochs, the effectiveness of the model can be assessed.

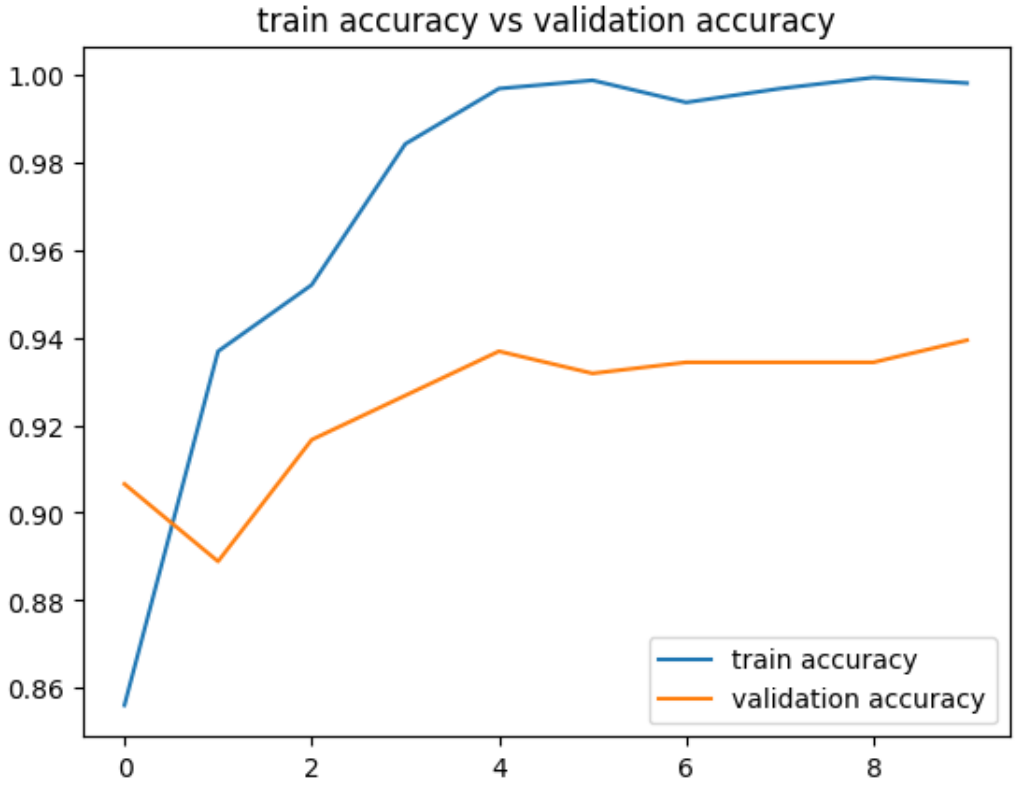

Figure 6: CNN Train Accuracy vs Validation Accuracy

The model achieved an accuracy of 96% on the test set

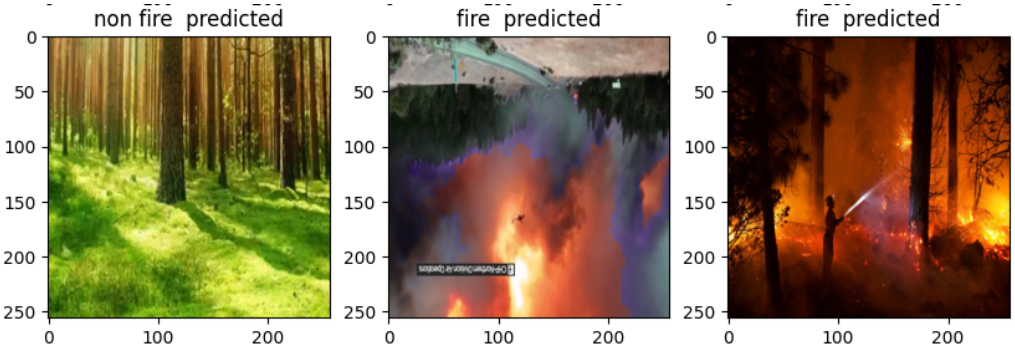

Figure 7: prediction using CNN

In order to adapt the YOLO (You Only Look Once) versions (YOLOv4, YOLOv5, YOLOv7, and YOLOv8) to our custom dataset for fire and smoke detection, a process of fine-tuning was employed. Fine-tuning involves taking a pre-trained YOLO model on a large-scale dataset, such as COCO, and retraining it on our fire and smoke dataset. This process allows the model to learn features specific to fire and smoke, improving its detection for our targeted objects. During the fine-tuning process, several hyperparameters were carefully selected. These hyperparameters include the learning rate, batch size and the number of training epochs. By tuning these hyperparameters, we aimed to optimize the model's performance and achieve the best possible detection results. Additionally, to set a threshold for confidence in detection, a grid search approach was conducted to determine the optimal confidence threshold that balances precision and recall. This threshold determines the minimum confidence level required for a predicted bounding box to be considered valid. By carefully selecting the threshold, we can control the trade-off between detecting as many instances of fire and smoke as possible while minimizing false positives.

4.2.2 Yolov4

The primary improvement in YOLO v4[2] is the use of a new CNN architecture called CSPNet (shown below). CSPNet stands for "Cross Stage Partial Network" and is a variant of the ResNet architecture designed specifically for object detection tasks. It has a relatively shallow structure, with only 54 convolutional layers. However, it can achieve state-of-the-art results on various object detection benchmarks.

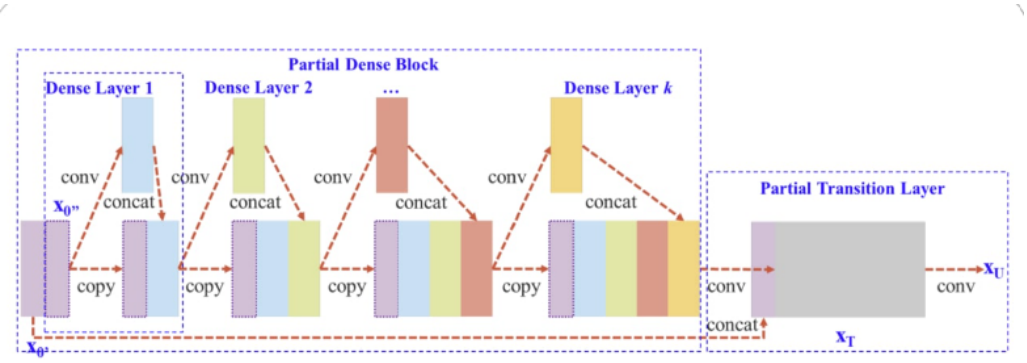

Figure 8: Architecture of CSPNet

Firstly, in the input layer, we adjusted the image resolution to 640x640 to match the characteristics of fire and smoke instances in our dataset.It enhances the model's ability to detect smaller or larger instances.
In the backbone layers, we the CSPDarknet53 backbone,which has been proven effective for various object detection tasks. The backbone extracts high-level features from the input image, which are crucial for accurate detection.
Next, in the neck layers, we have the FPN (Feature Pyramid Network), These layers fuse features from different scales to improve the model's ability to detect fire and smoke instances at various sizes.

The following table showcases the hyperparameters used for YOLOv4 during the fine-tuning process and confidence thresholds.

YOLO Version	Learning Rate	Batch Size	Training Epochs	Confidence Threshold
Yolov4	0.001	32	50	0.5

The YOLOv4 model achieved a mean Average Precision (mAP) of 0.54

Here are the results of inference of yolov4 model:

Figure 9: Yolov4 inference

4.2.3 Yolov5

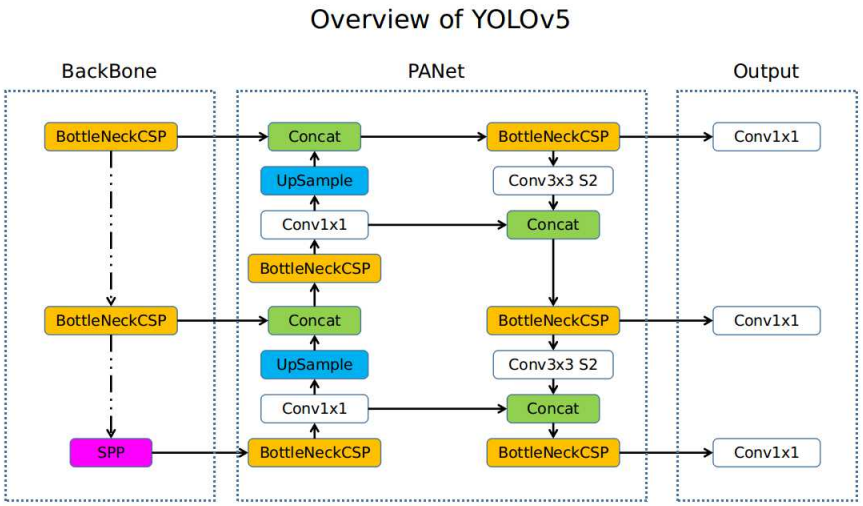

Figure 10: Yolov5 architecture

The backbone is responsible for extracting features from the input image. In YOLOv5[3], the backbone is typically a variant of the EfficientNet architecture. To enhance fire and smoke detection, we fine-tuned the backbone by training it on a large dataset of fire and smoke images. This process enables the backbone to learn specific features related to fire and smoke, making it more sensitive to these characteristics during inference.

The neck module fuses features from different levels of the backbone to create a multi-scale representation of the image. This enables the model to detect objects of various sizes and scales. For fire and smoke detection, we can maintain the neck module as is, as it is generally effective in capturing contextual information for different object categories.

To adapt the head for fire and smoke detection, we modified the output layer to accommodate the specific classes we are interested in, such as "fire" and "smoke." This requires adjusting the number of output channels.

The following table showcases the hyperparameters used for YOLOv during the fine-tuning process and confidence thresholds.

YOLO Version	Learning Rate	Batch Size	Training Epochs	Confidence Threshold
Yolov5	0.01	32	30	0.48

By fine-tuning the backbone, modifying the head, and adjusting anchor boxes, we can effectively adapt the YOLOv5 architecture for fire and smoke detection, enabling the model to detect and localize these critical elements in images or video frames.

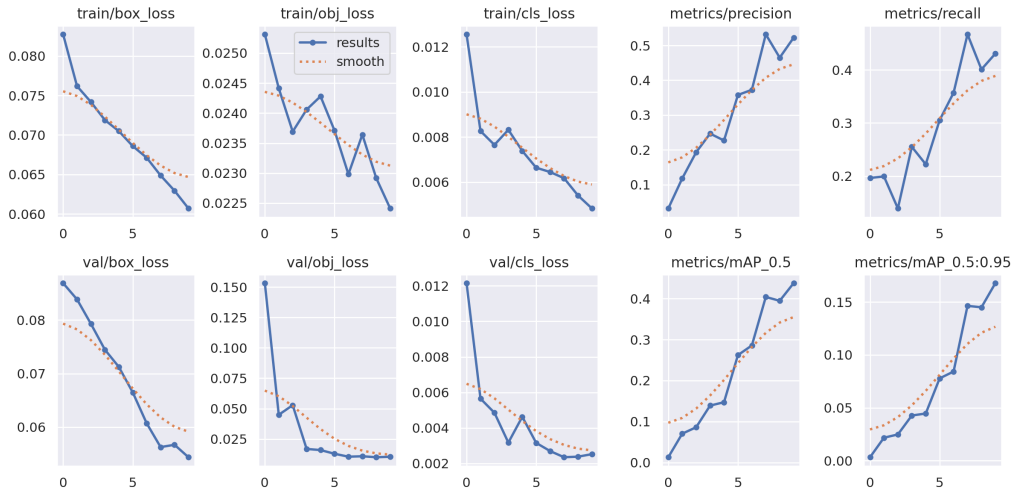

Figure 11: Yolov5 results

```
YOLOv5s summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 7/7 [00:08<00:00,  1.22s/it]
                   all        411        621      0.512      0.435      0.436      0.168
                  Fire        411        303      0.473      0.462      0.435       0.18
                 Smoke        411        318       0.55      0.407      0.437      0.156
```

Figure 12: Yolov5 Performance

The overall mAP of 0.43 suggests that the YOLOv5 model achieved a moderate level of accuracy in object detection across all classes. It implies that, on average, the model correctly localized and classified objects with an acceptable level of precision and recall.

The higher mAP of 0.46 specifically for the "fire" class indicates that the model performed slightly better in detecting and localizing fire instances. This is a positive outcome, sine fire detection is a crucial task in our application.

On the other hand, the mAP of 0.403 for the "smoke" class suggests that the model had a slightly lower performance in detecting and localizing smoke instances compared to fire. It could indicate that the model had some difficulty accurately identifying and distinguishing smoke instances from other objects or backgrounds.
Here are the results of inference of yolov5 model:

Figure 13: Yolov5 inference

4.2.4 Yolov7

YOLOv7 has several improvements over the previous versions. One of the main improvements is the use of anchor boxes.
Anchor boxes are a set of predefined boxes with different aspect ratios that are used to detect objects of different shapes. YOLO v7 uses nine anchor boxes, which allows it to detect a wider range of object shapes and sizes compared to previous versions, thus helping to reduce the number of false positives.
[10] YOLOv7 use also a new loss function called "focal loss." Previous versions of YOLO used a standard cross-entropy loss function, which is known to be less effective at detecting small objects.

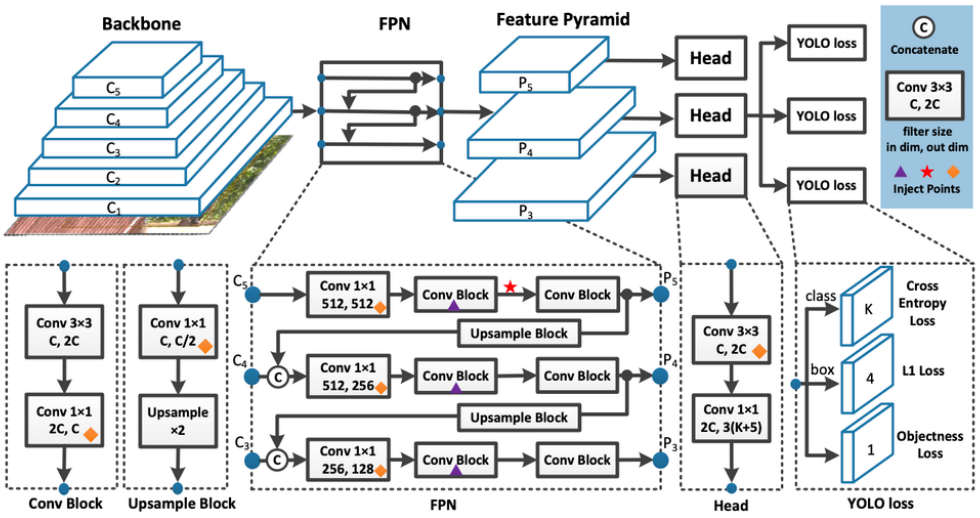

Figure 14: Yolov7 architecture

The fine-tuning process for YOLOv7 would involve freezing the initial layers and fine-tuning the subsequent layers on our fire and smoke dataset, similar to the other versions.

Our YOLOv7 model takes an input image of size 640x640 and preprocesses it by resizing it to a fixed size and normalizing the pixel values. This ensures consistency in the input data across different images.

The backbone layers in YOLOv7 are responsible for feature extraction from the input image. They typically consist of multiple convolutional layers followed by activation functions such as ReLU. These layers learn hierarchical features at different levels of abstraction, capturing both low-level and high-level visual information.

The neck layers in YOLOv7 help to further enhance the feature representation by incorporating spatial information and context. Commonly used neck architectures include feature pyramid networks (FPN) and spatial pyramid pooling (SPP) modules. These layers allow the model to capture objects of various sizes and effectively handle scale variations.

The detection layers in YOLOv7 are responsible for predicting bounding boxes, class probabilities, and confidence scores for potential fire and smoke instances. These layers employ a set of anchor boxes of different sizes and aspect ratios. For each grid cell in the output feature map, multiple anchor boxes are used to predict the coordinates of the

bounding boxes and the associated class probabilities.
For the loss function,YOLOv7 uses a combination of regression loss and classification loss to train the model. The regression loss measures the discrepancy between the predicted bounding box coordinates and the ground truth coordinates, while the classification loss measures the discrepancy between the predicted class probabilities and the ground truth class labels. The loss function is designed to penalize inaccurate predictions and encourage accurate localization and classification.
The following table showcases the hyperparameters used for YOLOv7 during the fine-tuning process and confidence thresholds.

YOLO Version	Learning Rate	Batch Size	Training Epochs	Confidence Threshold
Yolov7	0.001	32	50	0.52

The YOLOv7 model achieved a mean Average Precision (mAP) of 0.59.
Here are the results of inference of yolov7 model:

Figure 15: Yolov7 architecture

4.2.5 Yolov8

YOLOv8 [8] is an anchor-free model. This means it predicts directly the center of an object instead of the offset from a known anchor box. Anchor boxes were a notoriously tricky part of earlier YOLO models,

since they may represent the distribution of the target benchmark's boxes but not the distribution of the custom dataset.
Anchor free detection reduces the number of box predictions, which speeds up Non-Maximum Suppression (NMS), a complicated post processing step that sifts through candidate detections after inference.

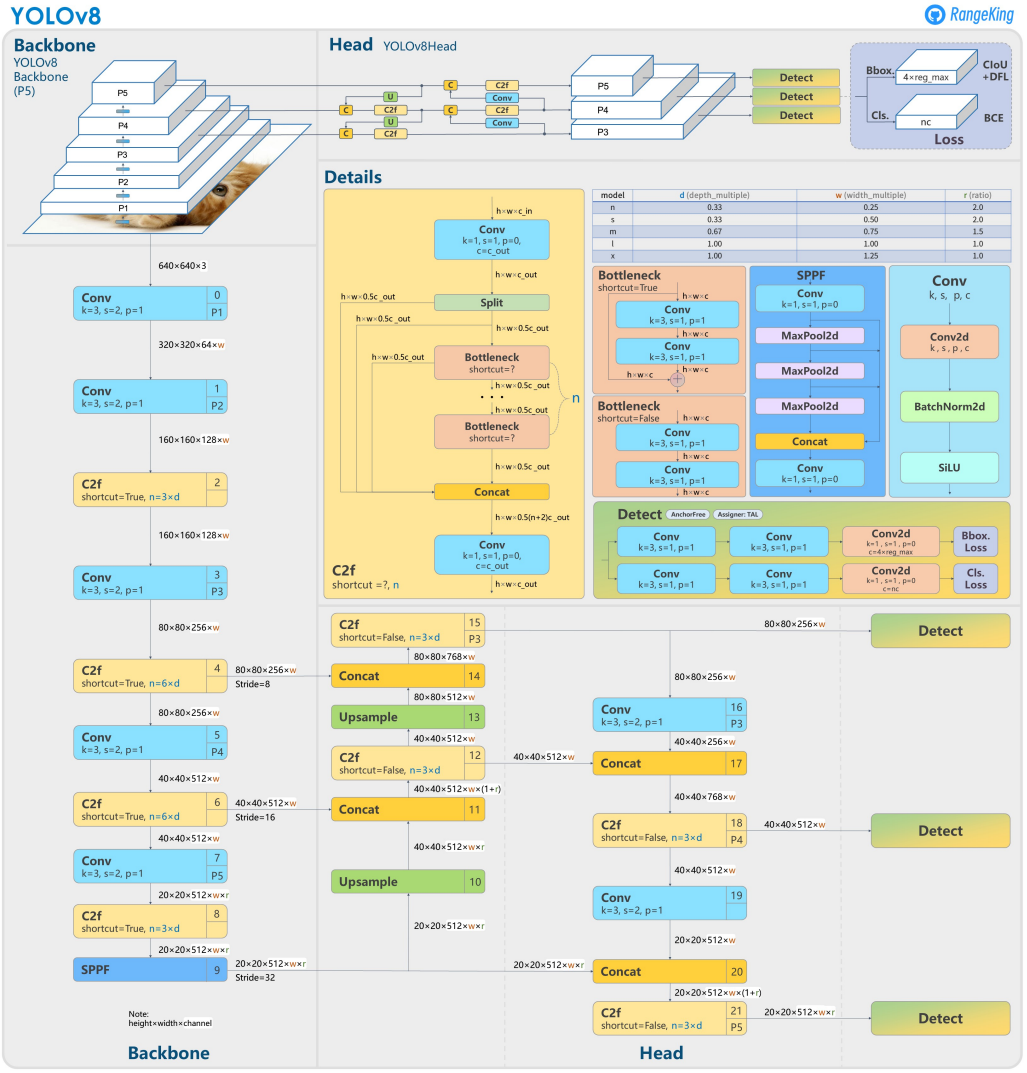

Figure 16: Yolov8 architecture

YOLOv8 augments images during training online. At each epoch, the model sees a slightly different variation of the images it has been provided.

One of those augmentations is called mosaic augmentation. This in-

volves stitching four images together, forcing the model to learn objects in new locations, in partial occlusion, and against different surrounding pixels.
On a fire and smoke dataset, we trained the Yolov8 model using transfer learning that involve freezing the initial layers responsible for extracting general features, such as low-level visual patterns to adapt the model to the specific characteristics of fire and smoke detection.
The following table showcases the hyperparameters used for YOLOv8 during the fine-tuning process and confidence thresholds.

YOLO Version	**Learning Rate**	**Batch Size**	**Training Epochs**	**Confidence Threshold**
Yolov8	0.001	32	25	0.6

each image is processed by YOLOv8 in the following way. The input image is first resized to a size of 640x640 pixels,640x640 strikes a balance between capturing sufficient details for accurate detection and maintaining reasonable inference speed also larger input image sizes tend to provide better object detection accuracy as they capture more detailed information. The resized image is then passed through the Darknet-53 backbone network, which extracts high-level features from the image.
The resized image is then passed through the Darknet-53 backbone network, which extracts high-level features from the image. These features are then fed into the Feature Pyramid Network (FPN) to capture multi-scale information. The FPN combines features from different levels of the backbone network to create a feature pyramid that is capable of detecting objects at various scales.
Next, the processed features are passed through the detection head, which consists of detection layers. These layers predict bounding boxes and class probabilities for the detected objects. In our case, there are two classes: fire and smoke. The model assigns class probabilities to each bounding box, indicating the likelihood of it containing fire or smoke. The model uses a combination of classification loss, localization loss, and confidence loss to optimize the model's parameters during training, ensuring accurate detection of fire and smoke in the

images.
Overall, YOLOv8 processes each image by extracting features through the backbone network, fusing multi-scale information through the FPN, and predicting bounding boxes and class probabilities through the detection head. The training process and specified settings help the model learn to accurately detect and classify fire and smoke in the input images.
We used the nano version of yolov8 which is the lighetr version of it,so it can be easy to deploy on the jetson nano.

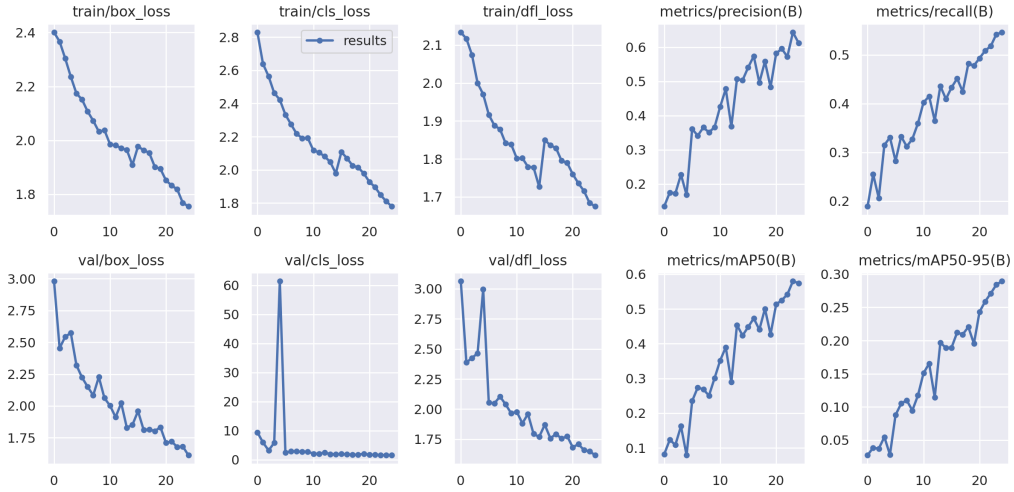

Figure 17: Yolov8 results

```
Validating runs/detect/train5/weights/best.pt...
Ultralytics YOLOv8.0.20 🚀 Python-3.9.16 torch-1.13.1+cu116 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11126358 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 7/7 [00:11<00:00,  1.65s/it]
                   all        411        621      0.742      0.567      0.631      0.356
                  Fire        411        303      0.674      0.488      0.529        0.3
                 Smoke        411        318      0.811      0.646      0.732      0.412
Speed: 2.2ms pre-process, 5.0ms inference, 0.0ms loss, 2.5ms post-process per image
Results saved to runs/detect/train5
```

Figure 18: Yolov8 performance

The results obtained from YOLOv8 object detection model indicate a significant improvement in performance compared to YOLOv5. With an overall mAP of 0.63, the model demonstrates a higher level of accuracy in detecting and localizing objects across all classes. The mAP of 0.52 for the ”fire” class suggests that the model performs reasonably well in identifying fire instances, although there is room for further enhancement. On the other hand, the impressive mAP of 0.73 for the ”smoke” class indicates that the YOLOv8 model excels in detecting

and localizing smoke instances with high precision and recall. These results highlight the effectiveness of the YOLOv8 architecture in addressing the specific challenges posed by smoke detection, which is particularly valuable in scenarios where detecting smoke is of critical importance, such as fire detection and prevention systems. Overall, the YOLOv8 model demonstrates promising performance, showing substantial improvements in object detection accuracy, especially for the smoke class.
Here are the results of inference of yolov8 model:

Figure 19: Yolov8 inference

4.3 Discussion

The table provides a comparison of the Mean Average Precision (mAP) scores achieved by different models in the fire and smoke detection system. The mAP metric is commonly used to evaluate the accuracy and performance of object detection models.
The mAP metric represents the average precision across different classes, where precision and recall which are calculated by comparing predicted bounding boxes to ground truth bounding boxes using the IoU metric(IoU is the ratio of the area of overlap between the predicted bounding box and the ground truth bounding box to the area of their union. If the IoU exceeds a certain threshold (usually 0.5 or 0.75), the detection is considered a true positive).

Model	mAP
Yolov4	0.54
Yolov5	0.436
Yolov7	0.59
Yolov8	0.63

According to the findings, YOLOv4 achieved a mean Average Precision (mAP) of 0.54, indicating a moderate level in detecting fire and smoke .YOLOv5 obtained an mAP of 0.43, showcasing slightly lower performance than YOLOv4 . With a mAP of 0.59, YOLOv7 showed considerable improvement in detecting fire and smoke incidents. The YOLOv8 model, which achieved a mAP of 0.63 and demonstrated the best performance among the tested models in detecting fire and smoke. These results suggest that YOLOv8 outperforms the other models , makinNg it the most suitable choice for precise fire and smoke detection. However, it is important to consider other factors such as computational efficiency and deployment requirements when selecting the appropriate model for a specific application.

In conclusion, the chosen methodology and architecture, along with the YOLOv4 , YOLOv5, YOLOv7and YOLOv8 models, have demonstrated promising performance in fire and smoke detection. The models exhibit a balance between performance and computational efficiency, with YOLOv8 showing best performance .However,the computational requirements of the chosen models must be considered. While the Jetson Nano provides powerful AI capabilities, it may still have limitations when it comes to handling large-scale models or processing multiple streams simultaneously.

5 Experimental Results

We developed a user-friendly web interface using Flask which is a popular web framework for developing web applications in Python and it is lightweight, flexible, and easy to use that allows users to choose the desired algorithm for fire and smoke detection. The interface provides options to select input types, including images, videos, and real-time

streaming using a camera. This flexibility in input selection enhances the usability and applicability of the system across different scenarios.

The web interface streamlines the process of running the detection system and provides a seamless user experience. Users can easily switch between different algorithms and input types, enabling efficient and customizable fire and smoke detection.

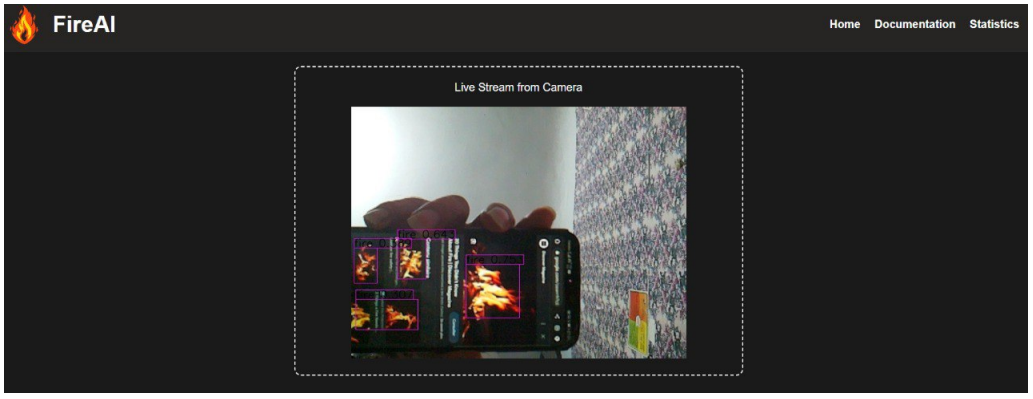

Figure 20: fire and smoke detection from camera live stream

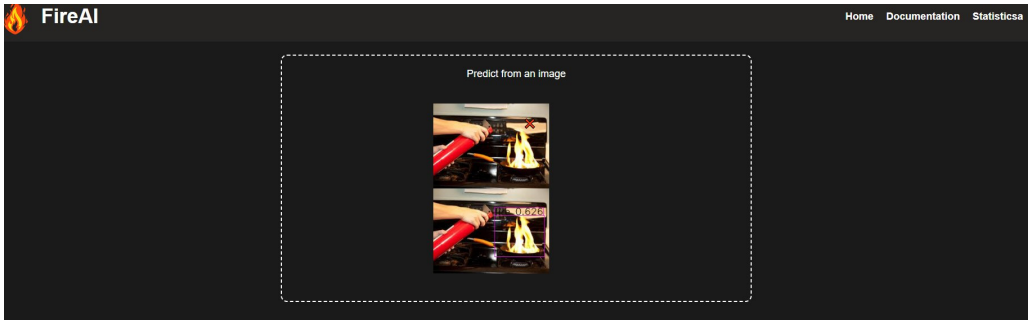

Figure 21: fire and smoke detection from image

the architecture of the fire and smoke detection system combines the Raspberry Pi, Jetson Nano,YOLO models and the Flask web interface enable real-time analysis and detection of fire and smoke instances. The integration of hardware components, deep learning models, and the web interface creates an efficient and reliable solution for fire and smoke detection applications.

6 Future Work

In terms of future work, there are several areas of improvement and expansion that can be explored in our fire and smoke detection project. One crucial aspect is the incorporation of distance estimate into the detection system . We can improve and enhance this capability by building on the successful integration of distance estimation in the YOLO architecture. We can enable the model to predict the absolute distance of objects from the camera while also simultaneously identifying their class and bounding box coordinates by increasing the output prediction vector, altering the loss function, and adding distance information to the training data.
Another important aspect to consider is the size estimation of fire instances. By developing algorithms that analyze the fire region and infer its dimensions or intensity, we can provide valuable insights into the scale and severity of the fire. Factors such as pixel intensity, area, and color gradients can be utilized to estimate the size or extent of the detected fire.
In our project, the focus has been primarily on the development and implementation of the fire and smoke detection system. However, we recognize the importance of incorporating robust security measures to ensure the integrity and confidentiality of the system's data and operations. Moving forward, one of the key areas of future work will involve enhancing the security aspects of the system. This includes implementing measures to protect the network communication between the devices, securing the storage and transmission of captured video data, and implementing authentication and authorization mechanisms to control access to the system.

7 Conclusions

In conclusion, this project has created a real-time fire and smoke detection system that can be used with embedded devices like the Jetson Nano and Raspberry Pi. It has been demonstrated that the system's architecture, which combines the strength of these platforms with the YOLOv4, YOLOv7, and YOLOv8 models, is efficient in detecting fire and smoke cases.

The project's results provided important light on the advantages and disadvantages of the approach, design, and models that were selected.The evaluation of different models, including CNN,YOLOv4, YOLOv5, YOLOv7, and YOLOv8, revealed varying levels of accuracy.
Furthermore, the system's usability and accessibility have been improved with the addition of a user-friendly web interface built with Flask. With this interface, users can easily select the preferred algorithm and input format, whether it be for real-time camera streaming, videos, or photos. The identification process is streamlined and the user experience is improved because to Flask's seamless connection with the hardware parts and deep learning models.
The developed fire and smoke detection system holds great potential for advancing fire safety and monitoring applications. It can be deployed in various settings, including residential, commercial, and industrial environments, to ensure timely and accurate detection of fire and smoke incidents. The system's real-time capabilities enable prompt action and response, potentially saving lives and mitigating property damage.

References

[1] Abdul Bari, Tapas Saini, and Anoop Kumar. Fire detection using deep transfer learning on surveillance videos. In *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*, pages 1061–1067. IEEE, 2021.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[3] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, Yonghye Kwon, Kalen Michael, Jiacong Fang, Zeng Yifu, Colin Wong, Diego Montes, et al. ultralytics/yolov5: V7. 0-yolov5 sota realtime instance segmentation. *Zenodo*, 2022.

[4] Byoungjun Kim and Joonwhoan Lee. A video-based fire detection using deep learning models. *Applied Sciences*, 9(14):2862, 2019.

[5] Songbin Li, Qiandong Yan, and Peng Liu. An efficient fire detection method based on multiscale feature extraction, implicit deep supervision and channel attention mechanism. *IEEE Transactions on Image Processing*, 29:8467–8475, 2020.

[6] Khan Muhammad, Jamil Ahmad, Zhihan Lv, Paolo Bellavista, Po Yang, and Sung Wook Baik. Efficient deep cnn-based fire detection and localization in video surveillance applications. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(7):1419–1434, 2018.

[7] Khan Muhammad, Salman Khan, Mohamed Elhoseny, Syed Hassan Ahmed, and Sung Wook Baik. Efficient fire detection for uncertain surveillance environment. *IEEE Transactions on Industrial Informatics*, 15(5):3113–3122, 2019.

[8] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *arXiv preprint arXiv:2304.00501*, 2023.

[9] Marek Vajgl, Petr Hurtik, and Tomáš Nejezchleba. Dist-yolo:

Fast object detection with distance estimation. *Applied Sciences*, 12(3):1354, 2022.

[10] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023.