Journal of Real-Time Image Processing (2021) 18:889–900
https://doi.org/10.1007/s11554-020-01044-0

ORIGINAL RESEARCH PAPER

Real-time video fire/smoke detection based on CNN in antifire surveillance systems

Sergio Saponara[1] · Abdussalam Elhanashi[1] · Alessio Gagliardi[1]

Received: 18 May 2020 / Accepted: 23 October 2020 / Published online: 10 November 2020

Abstract

This work presents a real-time video-based fire and smoke detection using YOLOv2 Convolutional Neural Network (CNN) in antifire surveillance systems. YOLOv2 is designed with light-weight neural network architecture to account the requirements of embedded platforms. The training stage is processed off-line with indoor and outdoor fire and smoke image sets in different indoor and outdoor scenarios. Ground truth labeler app is used to generate the ground truth data from the training set. The trained model was tested and compared to the other state-of-the-art methods. We used a large scale of fire/smoke and negative videos in different environments, both indoor (e.g., a railway carriage, container, bus wagon, or home/office) or outdoor (e.g., storage or parking area). YOLOv2 is a better option compared to the other approaches for real-time fire/smoke detection. This work has been deployed in a low-cost embedded device (*Jetson Nano*), which is composed of a single, fixed camera per scene, working in the visible spectral range. There are not specific requirements for the video camera. Hence, when the proposed solution is applied for safety on-board vehicles, or in transport infrastructures, or smart cities, the camera installed in closed-circuit television surveillance systems can be reused. The achieved experimental results show that the proposed solution is suitable for creating a smart and real-time video-surveillance system for fire/smoke detection.

Keywords Video fire/smoke detection · Ground truth labeler · YOLOv2 · Embedded video systems · Real-time · CNN

1 Introduction

Fire is one of the leading hazards endangering human life, the economy, and the environment [1]. Due to the rapid increase in fire accidents, every building or passenger vehicle for public transportation is equipped with fire protection and fire prevention systems. These systems consist mainly of point-type thermal and smoke detectors that need to be installed in proximity of the fire; otherwise, they may easily fail without detecting the fire. In addition, these devices must be properly installed and positioned as they can be damaged during the fire itself. Video-based fire detection is currently a standard technology due to image processing, computer vision, and Artificial Intelligence. These systems have remarkable potential advantages over traditional methods, such as a fast response and wide detection areas.

Traditional smoke/fire sensors based on photometry, thermal, or chemical detection can react within several minutes, requiring a large amount of fire/smoke to trigger an alarm. Moreover, they cannot provide information about fire location and fire size, and they cannot work for outdoor scenes. The development of new camera-based solutions improves the robustness and reliability of smoke and fire detection by filling the gap of previous systems. Cameras and closed-circuit television (CCTV) systems are already installed for surveillance purposes in most human environments, such as city streets, industry, public transportation. The existing infrastructure includes hundreds of video cameras, a communication network, possible processing units, and monitor screens in a control room. Their utilization would allow a reduction in purchase and installation cost as there is no need for additional products. The fire detection algorithm can be easily integrated into this infrastructure with the installation of additional software. A possible low-cost alternative could be the installation of an ad-hoc architecture based on a distributed video camera IoT nodes system. Such IoT distribution system could provide a web platform for video streaming and could be able to trigger a fire alarm by itself [2].

✉ Sergio Saponara
sergio.saponara@iet.unipi.it

1 Dip. Ingegneria Dell'Informazione, University of Pisa, Via G. Caruso 16, 56122 Pisa, Italy

With the rapid development of computer vision technology and digital camera technology, intelligent video fire detection methods have been proposed and applied in industry. At the beginning, color and shape characteristics of smoke were used to extract hand-designed features for detecting smoke and fire in a video stream. Recently we have witnessed the rapid dissemination of hardware accelerations, graphics processing units (GPUs), and high-performance processors capable of handling a massive amount of computation, and hence to the development of artificial intelligence techniques. Deep learning models have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection, and many other domains, outperforming human-level performance especially in a custom computer vision application, such as image classification [3]. Deep learning model has the ability to perform feature extraction and classification within one network. Consequently, the hand-crafted visual detection approaches have been progressively replaced by deep learning approaches due to their ability to extract features from raw images automatically.

In this paper, we present an intelligent fire/smoke detection approach based on YOLOv2 network, which aims to achieve high detection rate, low false alarm rate, and high speed. Such an algorithm has been tested with a state-of-the-art dataset plus a set of other not public videos. Finally, the detector has been deployed on a single board embedded system, NVIDIA Jetson Nano, as a standalone application for real-time video processing.

Hereafter, the paper is organized as follows: Sects. 1 and 2 deal with introduction and state-of-the-art video-based fire/smoke detectors. Section 3 presents the algorithm description. Section 4 discusses the global architecture and then each of the layers used in the video processing steps. Section 5 shows the experiment results and discussion. Section 6 presents the algorithm implementation in the embedded system. Conclusions are drawn in Section 7.

2 State-of-art video-based fire/smoke detectors

Conventional video smoke detection methods addressed the problem by extracting a multi-dimensional feature vector from the input smoke image [4], which may be the color, texture, shapes, irregularity, flutter, or frequency, and classifying the feature vector into “smoke” or “non-fire” class. Celik et al. [5] proposed a method based on different color models for both fire and smoke, obtained by statistical analysis fuzzy-logic to achieve discrimination between fires and fire-like colored objects. Rafiee et al. [6] used static characteristics (two-dimensional wavelet analysis) and dynamic characteristics like smoke disorder. The first for detecting the color and the motion while the second implements background subtraction using frame differentiation. However, the false-negative rate remains an issue here also due to the presence of other objects in the background with similar color properties as the fire pixels. A similar technique is used in work in [7] that generates the background subtraction using visual background estimation (ViBe). Recent work in [8] proposes a smoke detector based on Kalman estimator, color analysis, image segmentation, blob labeling, geometrical features analysis, and M of N decisor, to extract an alarm signal within a strict real-time deadline. Such presented methods can be deployed on embedded systems achieving good performance in terms of power consumption and frame rate. The drawback of these techniques lies in having to extract by hand the features from the video streams.

With the rapid development of artificial intelligence and deep learning, computer vision has achieved significant attention from academia and industry. On the other hand, deep learning techniques have the advantage of extracting the features automatically, making this process more effective and dramatically improving the state-of-the-art in Image Classification and object detection methods [9]. Various deep learning methods have been proposed for fire and smoke detection. In [10], Wu et al. used popular object detection methods like R-CNN, YOLO, and SSD for real-time forest fire detection. Sharma et al. [11] instead propose a CNN-based fire detection based on a pre-trained VGG16 and Resnet50 as baseline architecture. In [12] and in [13], both authors used YOLO method for fire detection and flame detection respectively. In all cases, albeit they achieve good results in terms of accuracy, they do not provide a low-cost implementation on an embedded platform. This is due to the large disk size and the total number of parameters that make these models not suitable for that purpose.

In this paper, we used YOLOv2 algorithm to identify and locate fire and smoke objects using a video camera. Our target in this work is to create a light-weight deep learning model for embedded application, able to fit into low-cost, low-performance hardware such as Jetson nano and can achieve good performance for real-time fire and smoke detection [14]. A Ground Truth Labeler application has been used for labeling and creating the training set of collected images for our benchmark. We used such dataset to identify and label the features of fire and smoke in which to be trained for YOLOv2 detector.

3 Algorithm description

3.1 Principle of YOLO

You look only once (YOLO) is a deep learning model for object detection. It was first explored by Redmon et al.

[15]. YOLO can detect the location of multiple classes at one time. It is accurate and fast, which meets the requirements for real-time processing, thus outperforming in speed object detection models based on region techniques, such as Regional convolutional neural network (R-CNN) models. R-CNN detectors create bounding boxes in the image and then classify these proposed boxes. After the classification, R-CNN refines these boxes and initiate the scores for the objects to be detected. These processes are hard and slow to optimize because each stage needs to be trained separately [16]. Instead, YOLO network uses a single-stage architecture with less neural network layers and fewer filters to these layers. For the proposed YOLOv2 technique, we used full images to train and test the network: the algorithm takes the input image and splits it into $S \times S$ grids. It extracts the features from each grid and predicts the bounding boxes and the confidence score for the detected objects in these boxes, see Fig. 1.

There are five values for predictions in the algorithm: the rectangle box is represented with (x, y, w, h), plus there is the confidence score that defines the probability of the class presence in the box. Only one class will be predicted in each grid. A specific confidence score is obtained for each box. The confidence score is defined as seen in Eq. (1). If there is an object in the cell, the confidence score will be equal to the Intersection over Union (IoU) value between the ground truth and the bounding boxes. Otherwise, the confidence score will be zero if no object exists in the cell.

$$\Pr(\text{Classi}|\text{Object}) \times \Pr(\text{Object}) \times \text{IOU} = \Pr(\text{Classi}) \times \text{IoU} \tag{1}$$

where Pr(Object) is the probability that the box contains the object. IoU is the (Intersection over Union) between the ground truth and predicted boxes.

IoU is an evaluation algorithm used in object detection benchmarks. It determines the overlap between two areas and measures how these two areas are equal in terms of location and size. It is an indicator that judges the distance between Ground truth bounding boxes and the predicted bounding box from the module in object detection. The formula of IoU is defined as the following Eq. (2):

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|}, \tag{2}$$

where A is the prediction boxes. B is the ground truth boxes.

3.2 YOLOv2

YOLOv2 is the second version, which has several enhancements to YOLO, as presented in YOLO9000 [17]. Indeed, YOLO has two drawbacks for object detection. The first weakness is that it is inaccurate to locate and position the classes to be detected in the images. The second problem is a low recall rate when it compares to the regional based detectors. YOLOv2 resolved these issues, thus increasing the accuracy and the speed of the architecture. This algorithm is suitable for GPU-based embedded computing modules for real-time processing [18]. YOLOv2 used batch normalization layers in all convolutional layers to normalize the value distribution from one layer to another. Batch normalization improved the regularization for YOLOv2 neural network model. It reduced the requirement for a dropout layer to overcome the overfitting problems. It normalizes its input by calculating the mean and variance values over the mini-batch, and it calculates the activation, as seen in Eq. (3).

$$\hat{x}_i = \frac{xi - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{3}$$

where ϵ is the property Epsilon and improves the mini-batch when the variance is small.

Fully connected layers were removed in YOLOv2, and instead, anchor boxes were used to predict bounding boxes. This idea was drawn from the state of art Faster R-CNN detector. YOLOv2 uses these anchors to detect the objects in the images. YOLOv2 anchor boxes are a set of rectangle boxes predefined with a specific height and width. These boxes are used to capture the specific scale for the object to be detected. The size of anchor boxes is chosen based on the scale of bounding boxes in the training dataset. YOLOv2 with anchor boxes increases the output resolution for the networks' convolutional layers. Anchor boxes can evaluate

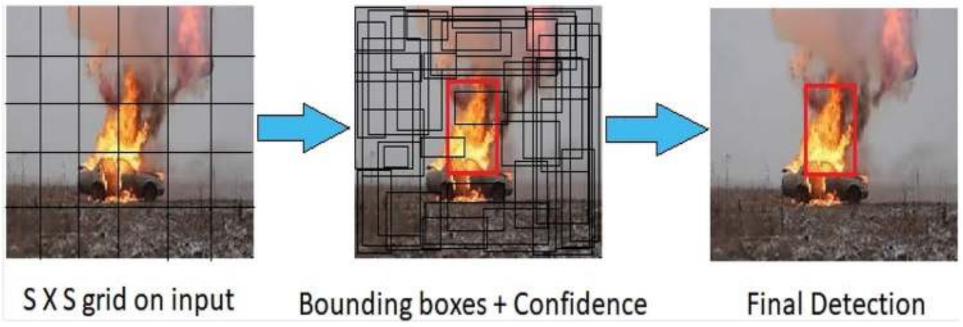


Fig. 1 YOLO model: input images split into $S \times S$ grid cell, each grid predicts the bounding boxes and the confidence scores and finally, the score encodes the probability with enclosed bounding box on the detected object

the prediction of all classes at once. It can also eliminate the requirement of scanning the image with a sliding window such as the detectors based on regional R-CNN and Fast R-CNN Algorithms**.**

4 Fire and smoke detection methodology

Many researches were focused on the traditional method of feature extraction for fire and smoke detection. The main problem for such techniques was time consumption for computing these feature extractions. This resulted in low performance and slow real-time for fire and smoke detection. These methods also generated a number of false positives and mistakes in the detection of background.

Motivated to the new development on the deep learning models, we proposed Fire and smoke detection based on a video camera using YOLOv2 model. Fire and smoke detection have a higher speed in imaging processing. In such a case, YOLOv2 is the best technique to encounter the detection of these objects. This detection is essential for firefighters because it will give an early alerting sign for fire and smoke accidents to take remedial actions accordingly.

Figure 2 shows the workflow for building up our architecture for fire and smoke detection. It started with creating the ground truth data by labeling the training images. Then we designed the model with neural network and YOLOv2 layers. Furthermore, we trained, validated, and tested YOLOv2 technique to assess its performance and accuracy. Finally, the proposed technique is deployed to the target node (Jetson Nano) to run as a stand-alone application in this embedded device.

4.1 The proposed YOLOv2 neural network design

In this section, we will discuss the design for our YOLOv2 model. We used Deep Neural Designer tool in MATLAB to build YOLOv2 neural network layers. To establish a light-weight deep learning model to fit the embedded system, we constructed CNN with 21 layers, see Fig. 3. This light-weight model is suitable for real-time performance, and it is worthy enough to be deployable on low-cost IoT devices. The proposed approach includes the input layer, middle layers, and subnetwork of YOLOv2 layers.

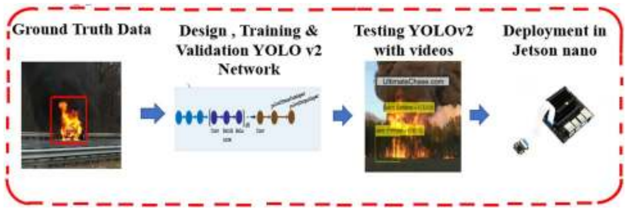

Ground Truth Data
Design , Training & Validation YOLO v2 Network
Testing YOLOv2 with videos
Deployment in Jetson nano

Fig. 2 Workflow for building up our architecture for fire and smoke detection

Our proposed model starts with the input image layer. We set the image input layer with a minimum image size of $128 \times 128 \times 3$ for the proposed model. Then we used middle layers, which contain a cascade of convolutional, batch normalization, ReLU (rectified linear unit), and max-pooling layers. Convolutional layers were used to map the features for the input images. The filter size on the convolutional layers is set to $[3 \times 3]$. This size is commonly used for the convolutional neural network architecture. Filter size defines the width and height of the regions in which the neurons connect in the input. Batch normalization layers are used to regularize the model, normalize the training, and speed up the convergence. Batch normalization layers reduce the sensitivity of the initialization of the network. Then we used ReLU activation function to introduce the non-linearity to the neural network. We utilized max-pooling layers to downsample the images into pooling regions. We applied 2×2 for size of pooling with a stride of 2×2 for all pooling layers in our network.

We used 'Batch normalization_4′ as the feature extraction layer. The features extracted from this layer are given as input to YOLOv2 object detection subnetwork. YOLO2Layer subnetwork is used in this model, which creates YOLOv2 detection network. YOLOv2 Transform layer is used in our model to enhance the network stability for object localization. Finally, YOLOv2 output layer is used to refine the location of bounding boxes to the targeted objects. The architecture was examined with deep network analyzer in MATLAB, which reported zero errors.

4.2 Data pre-processing

We prepared a dataset of 400 images for fire and smoke to train YOLOv2 detector in our experiments. The images were collected from Kaggle website [19]. These images were selected from different realistic situations for fire and smoke accidents. Ground truth application was used to label the images in MATLAB. The following steps were carried to pre-process the proposed model:

- Load the collection of images of fire and smoke into ground truth application.
- Label the objects of interest for fire and smoke with (rectangular boxes) in the selected images using a custom algorithm tool in the ground truth labeler [20].
- Export the ground truth data to the MATLAB workspace to obtain the arrays of labeled features.

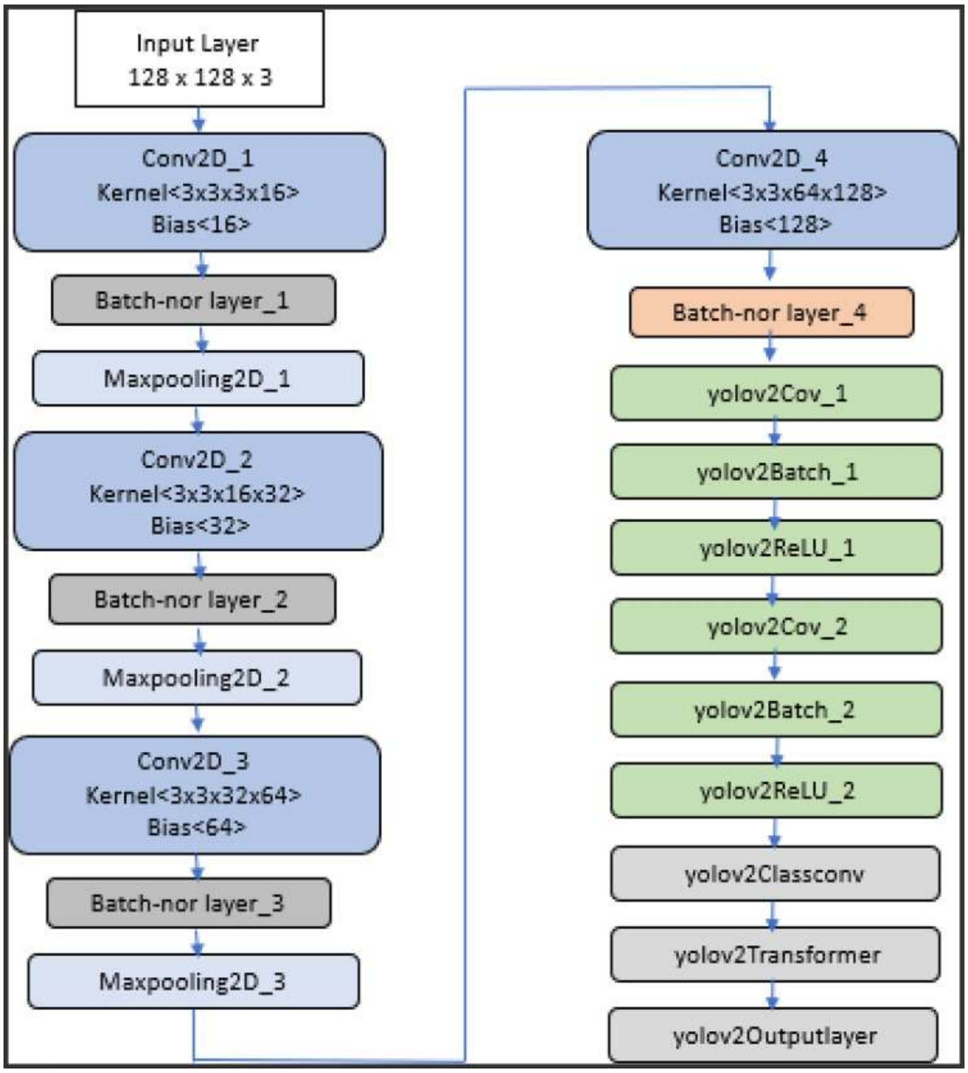


Fig. 3 Architecture of the proposed YOLOv2 Neural Network

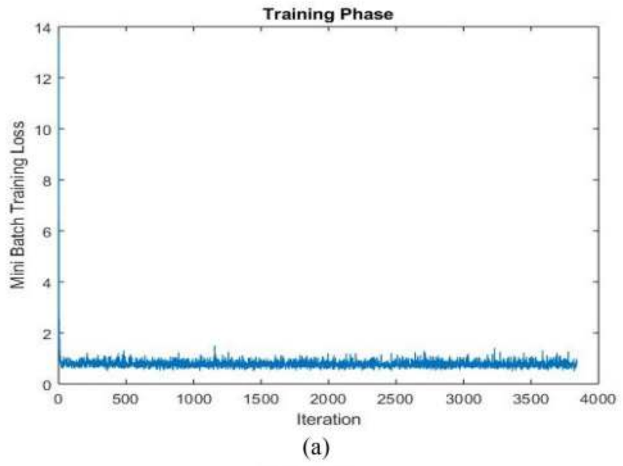
Table 1 Training hyper-parameters for YOLOv2

Parameter	Method
Training options	sdgm
L2 regularization	0.06
Number of epochs	160
Verbose frequency	50
Mini-batch size	16
Learning rate	0.001

4.3 Training

We trained the proposed detector on MATLAB script and saved the model at the end of the training process. The image size of training dataset images has been resized from $416 \times 416 \times 3$to $128 \times 128 \times 3$ to account the requirement of YOLOv2 model. We trained the network with stochastic gradient descent (sdgm) [21] see Table 1. The number of epochs was set to 160. These epochs define the number of times that the learning algorithm will work through the training dataset. The model was trained with an optimal number of epochs. This is to avoid overtraining, which could lead to overfitting and overconfident the model in its predictions. We used a learning rate parameter in the training option to control the model change in response to the error [22]. We started the learning rate with 10^{-2}. However, we observed that the network was unstable during the training process. We fine-tuned the learning rate at 10^{-3}, and we obtained the best result of 0.2. The mini-batch loss curve was stable with small fluctuation, see Fig. 4.

For anchor boxes, we run *k*-mean clustering in MATLAB to select a good set of labeled boxes in the training dataset. It is important to have the correct sizes of these boxes (width, height) for YOLOv2 to detect the objects accurately. We measured the IoU score of *k*-means to define the required number of these boxes for the model. This is to avoid using more anchor boxes, which could result in overfitting and poor performance of the detector.



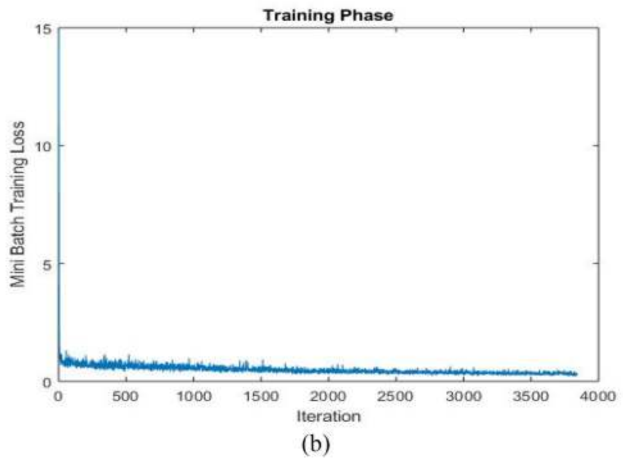


Fig. 4 **a** Mini-batch loss curve before fine-tuning **b** mini-batch loss curve after fine-tuning

4.4 Validation

We validated the proposed model with 200 images (100 images with fire/smoke, and 100 images with NO fire / smoke). This is an independent test bench, which is different from the training dataset. As per the results from receiver operating characteristic (ROC) analysis, the accuracy for this validation was 93%. See Table 2 and Fig. 5.

Table 2 Summary for validation results by ROC tool analysis

Matrices	Validation values
Number of images	200
Accuracy	93%
Sensitivity	94%
Specificity	80%

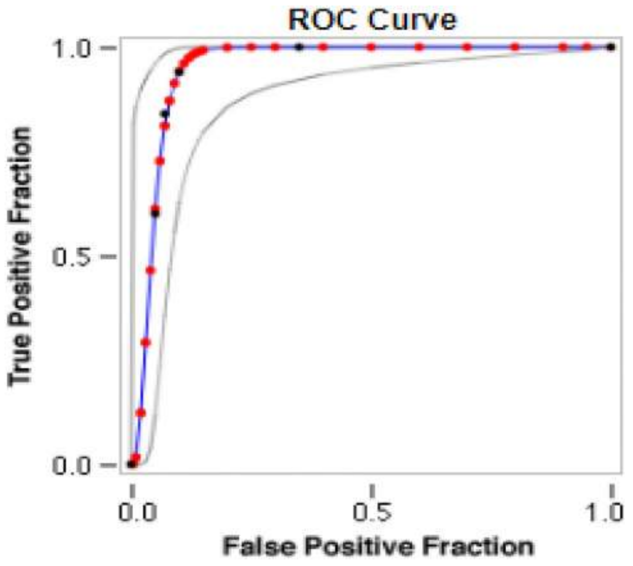


Fig. 5 Results obtained by the proposed system in terms of ROC curve for validation Dataset

5 Experiment results and discussion

In literature, methods used for testing data are still images instead of videos. In method [23], there is a lack of diversity for using videos to encounter various realistic situations for fire/smoke and normal conditions. In method [24], the work refers to monitor the fire only in forest areas. However, our model will be exploited in different indoor and outdoor conditions. The experiments were carried out on a large scale of videos as testing dataset. In our exploration, we used two testing datasets from different sources in our proposed approach. Dataset_v1 is our test bench; it consists of 287 videos from different environments (indoor, outdoor, forest, railways, parking, and public area). 117 videos contained a non-smoke/fire condition, and 170 videos contained smoke and fire. This dataset has been made challenging for motion-based and color-based objects. This has been obtained by capturing videos which include objects like smoke, such as clouds. This is one of the motivations for selecting this dataset for our experiments. Dataset_v2 is used from method [26] to evaluate our model, which will be illustrated in the following sub-section.

As a further experiment, we designed and trained R-CNN object detector for fire and smoke detection based on a regional method. We compared this R-CNN to our proposed model using the same Dataset_v1.

We used confusion matrix criteria to analyze different matrices in terms of (false positive rate, false-negative rate, and accuracy) see Eq. (4) and Table 3. This is to evaluate the performance of the proposed approach in comparison to the other methodologies,

Table 3 Confusion matrix criteria

	Fire/smoke	No fire/smoke
Alarm	TP	FP
No alarm	FN	TN

$$\begin{aligned}&\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP}\\&\text{False - positive rate} = \frac{FP}{FP + TN}\\&\text{False - negative rate} = \frac{FN}{FN + TP}\end{aligned} \quad (4)$$

where we mark the result as: TP if the model detects fire/smoke objects in positive videos. FP if the model detects fire/smoke objects in negative videos. TN if the model does not detect fire/smoke in negative videos. FN if the model does not detect fire/smoke in positive videos.

Based to the results from these experiments, the proposed method achieves good classification performance for fire and smoke detection, see Fig. 6a and b and overcomes all other methodologies [12, 23, 25]. The fact that in Fig. 6a the alarm confidence is above threshold, but it is far from 1 (i.e. 100% confidence) is due to fact that, as explained in Sect. 4.C (training), the model was trained with a proper number of epochs to avoid overtraining. Indeed, overtraining would lead to overfitting and would make the model overconfident in its predictions. If the model gets overconfident and with high confidence scores, this will result in false positive detection in non-fire/smoke videos.

Table 4 compares the classification accuracy of the proposed method, which is 96.82%, with respect to state-of-the-art methods whose accuracy ranges from 90 to 92.86%, using Dataset_v1. In addition, the proposed approach shows better accuracy when compared to R-CNN object detector. YOLOv2 sees the whole image at once as opposed to looking only at generated region proposals in R-CNN method. YOLOv2 algorithm helps to reduce false positives problems for fire and smoke detection. In addition to that, YOLOv2 is by 25 times faster than R-CNN for real-time object detection (see Sect. 5.b). Therefore, the proposed approach made an improvement in the background mistakes for the videos with no—fire/smoke objects.

5.1 Performance of our method with Dataset_v2

To evaluate the efficiency of the proposed approach, we tested it also with Dataset_v2 [26]. The test bench consisted of 160 non-fire images, 46 fire videos, and 16 non-fire videos. The dataset is limited, but it is challenging, e.g., it consists of videos for no-fire/smoke, which contains sunset light, see Fig. 7. Such a dataset was used for [26] method, also based on deep learning approach. It is important to note that no videos from Dataset_v2 were utilized in training the proposed architecture for fire and smoke detection. The results for our method is compared with four different methods reported in [26, 27], and [28]. We used metrics (Accuracy, Recall, F-Score, and Precision) to evaluate the effectiveness of our approach. According to the results from Dataset_v2, our approach showed the best result for Accuracy compared to the other state-of-the-art methods, see Table 5.

All experiments were carried on a personal computer using MATLAB 2019b, a built-in application Neural Network Designer, Ground Truth Labeler, and Intel® Core TM I3-6006U CPU @ 2 GHz were utilized as support tools. Jetson Nano, see details in Sect. 6, then was used as a Hardware test platform.

5.2 Our method vs. other object detectors for real-time fire and smoke detection

Fire and smoke detection have a vital role in protecting people, the environment, and properties. The key aspect of fire and smoke detection is to identify the accident occurrence in a timely manner. Early fire/smoke detection is a major element of disaster risk reduction [29].

To understand further, we carried experiments to compare our proposed method to the other object detectors such as R-CNN and Fast R-CNN. We used MATLAB with our bench-test dataset of fire and smoke videos. We run the three detectors simultaneously while calculating frames per second for each detector. The experiment results demonstrate that the proposed YOLOv2 approach is by 25 times faster than R-CNN and by 23 times than the Fast R-CNN object detectors, see Fig. 8.

6 Embedded system implementation

6.1 Jetson Nano embedded platform

Jetson Nano is a powerful but compact embedded computer with low cost of approximately $100 [30]. Jetson Nano runs multiple neural networks in parallel for object detection. It is suitable device for applications that are based on distributed networks. It consists of GPU 128-core Maxwell. CPU is Quad-core ARM A57 at 1.43 GHz. The memory for Jetson Nano is 4 GB, 64-bit, LPDDR4 25.6 GB/s. It has 4 × USB 3.0, USB 2.0 Micro-B. We used a Raspberry Pi Camera Module V2 with 8 M pixel resolution as a testing camera. This camera can work with an image resolution of 1080p @30 frames per second. We connected the camera to the CSI (camera serial interface) port in Jetson Nano. The trained YOLOv2 detector has been deployed in NVIDIA Jetson Nano and run it as a

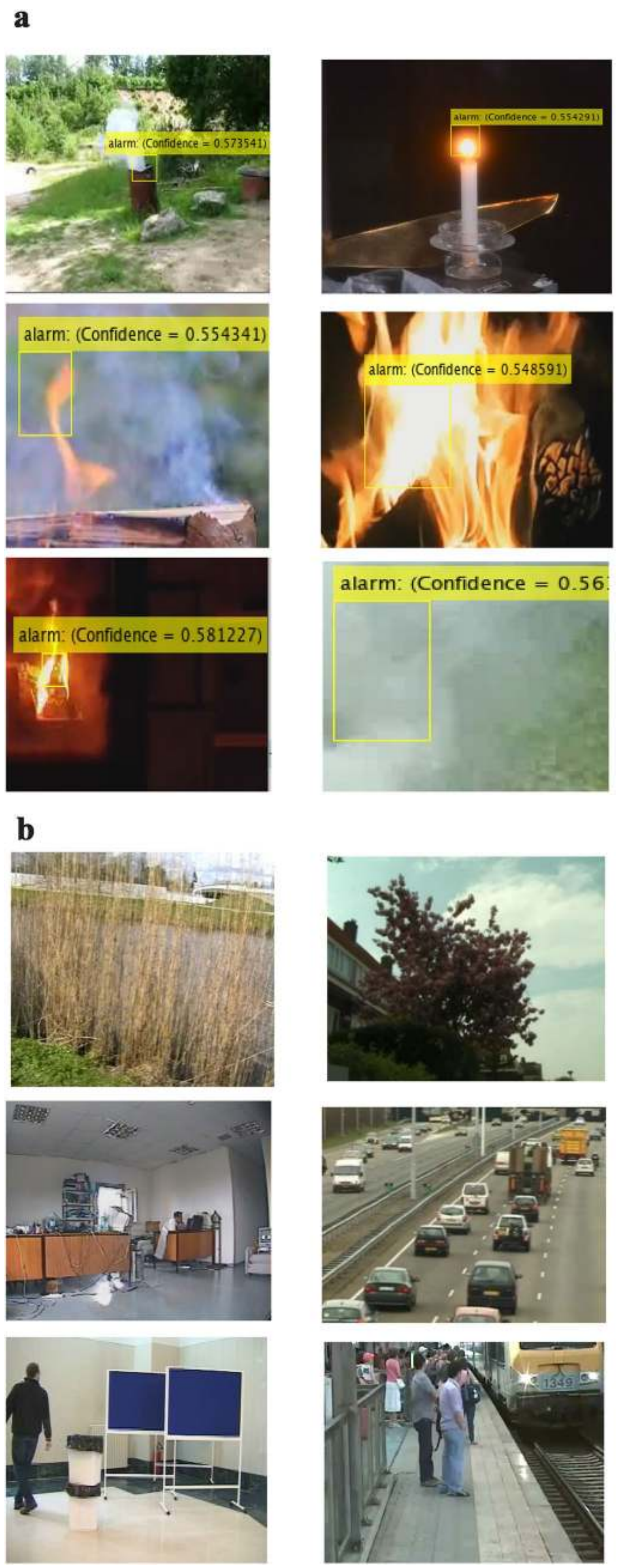

a
alarm: (Confidence = 0.573541)
alarm: (Confidence = 0.554291)
alarm: (Confidence = 0.554341)
alarm: (Confidence = 0.548591)
alarm: (Confidence = 0.581227)
alarm: (Confidence = 0.56
b
1349

Fig. 6 **a** Sample Images from videos with fire and smoke. **b** Sample images from videos with no fire and smoke

Table 4 Performance of the proposed approach vs. state-of-the-art

Method	False-positive (%)	False-negative (%)	Accuracy (%)
This work	3.4	2.9	96.82
R-CNN	8.5	0.0	96.5
De Lascio et al. [23]	13.33	0	92.86
Fu et al. [25]	14	8	91
YOLO [12]	5	5	90

Fig. 7 Sample examples of Dataset_v2, which shows sunset light in non-fire videos

Table 5 The Performance of the proposed approach using Daraset_v2 vs. other methodologies (the *F*-score is the harmonic mean of the precision and recall)

Method	Accuracy %	Recall %	*F*-Score %	Precision %
Proposed method	96.58	**97**	**95.4**	**97**
Jadon et al. [26]	93.9	94	95	96.9
Filonenko et al. [27]	85	96	90	85
Yuan et al. [28]	86	53	65.5	86

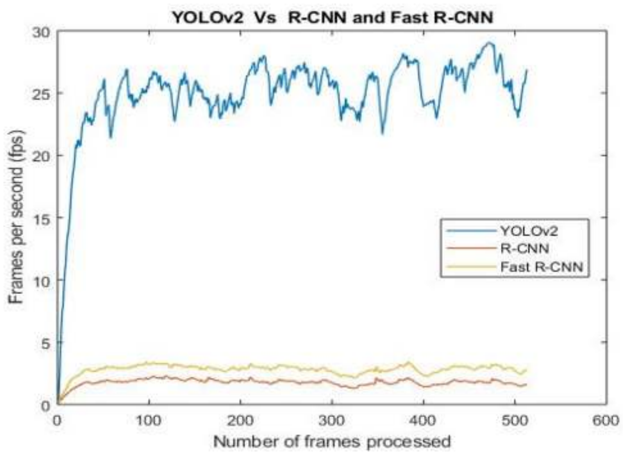


Fig. 8 The comparison of YOLOv2 detector vs. R-CNN and Fast R-CNN models for Real-time fire/smoke detection

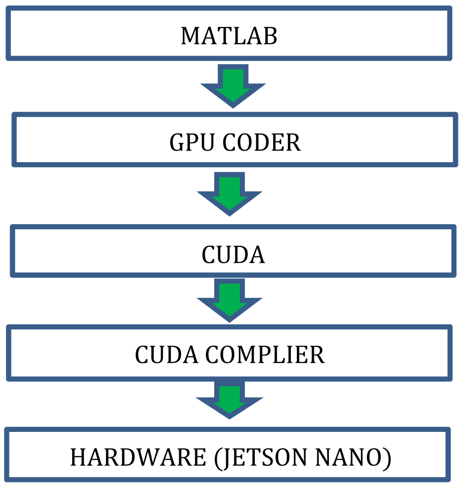


Fig. 9 Diagram for the deployment process of the CNN code from MATLAB to targeted hardware Jetson Nano

stand-alone application. We used MATLAB environments and third-party packages to generate the C code. We used GPU coder to convert MATLAB code into an optimized CUDA code. CUDA stands for (Compute Unified Device Architecture). It is an extension of C programming language which is designed for NVIDIA framework. We connected Jetson Nano to a host computer using an ethernet cable. Figure 9 shows the simplified diagram for the deployment process of the proposed CNN model from MATLAB to Jetson Nano.

MATLAB Coder is used to generate the C Code from MATLAB to Jetson Nano. Parallel Computing Toolbox is utilized to solve the computational and data problems using a multicore processor and GPU. We also used Deep Learning Toolbox, which provides a framework for implementing the neural network with the algorithm. GPU Coder Interface for Deep Learning Libraries customizes the generated code by utilizing a specific library in Jetson Nano. GPU Coder Support Package for NVIDIA GPU's is used to generate and deploy the CUDA code. It enabled the communication remotely between MATLAB and NVIDIA on the targeted hardware. We used Embedded Coder for code generation to Jetson Nano. It is an optimized tool that improves the code generation to the hardware precisely. We installed the required environment variables and applications such as JetPack Developer AI tool in Jetson Nano. This is to be applicable for code generation of our CNN detector from MATLAB. We also installed Microsoft visual studio 2019 as complier support for generating GPU code to Jetson Nano. We used CUDA Deep Neural Network libraries to accelerate primitives for deep neural networks.

6.2 Test of the proposed method on embedded system

We tested the deployed detector as a stand-alone application in Jetson Nano to evaluate its real-time performance. Raspberry webcam model V2 was connected to Jetson Nano. The camera was exposed to another computer, which was simulating a number of videos of fire/smoke and negative videos, see Fig. 10. While the proposed detector is running in Jetson Nano, we recorded various parameters. The real-time measured was 21 fps, which showed better real-time detection in comparison to [31].

We also measured the time delay before our method started to detect fire and smoke. When the camera is in video mode, the time delay between the start of fire/smoke in videos and YOLOv2 detection is 1–2 s. It means that the algorithm requires 1–2 s to trigger a smoke or fire alarm. Note that our approach can produce better time decision for fire/smoke detection in comparison to the method [32], which proposed Faster R-CNN model. This is the advantage of using our method for early fire and smoke detection. During the test, we also measured the power consumption for Jetson Nano. We removed all accessories such as keyboard, mouse, and monitor from Jetson Nano. The power consumption of Jetson Nano was 1.24 W when the detector is off. While our method was executed, the power consumption measured was 4.19 W. Table 6 shows the power consumption measurement for Jetson Nano in different scenarios.

We performed the measurement of CPU (Central Processing Unit) and the GPU (Graphics Processing Unit) % resource utilization in Jetson Nano. The CPU runs the operating system and applications. The GPU is designed to handle graphic operations. These characteristics are important to evaluate its computation processing. The table shows the recorded values for the CPU and GPU processors while our proposed detector is executed in Jetson Nano. In addition to that, the temperature of Jetson Nano was measured while our method was in processing mode. The temperature was recorded for CPU at 53.1 °C, and for GPU at 54 °C see Table 7.

As a further advantage of our approach, we measured its final memory size when deployed on Jetson Nano, which is 7.1 MB. In literature, other methodologies used large CNN layers such as Alexnet, VGG16, and Resnet50 for approaches in methods [33, 34]. These massive CNNs layers require a large disk size for deployment on the hardware. This could influence the real-time detection and low performance while running on low-cost embedded devices. Therefore, this is an advantage of our architecture in comparison to the other state-of-the-art methods and it can be superior for real-time fire and smoke detection.

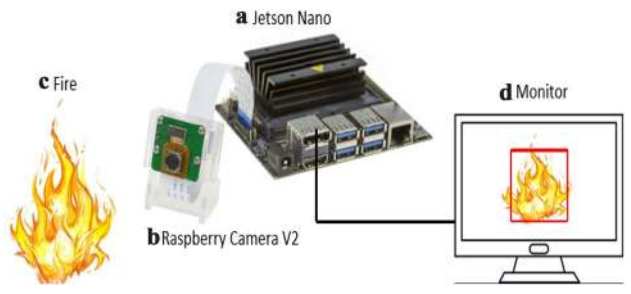


Fig. 10 Testing the proposed Detector with real state of fire and smoke. Hardware setup consists of: **a** proposed detector running as a stand-alone application in Jetson Nano **b** Raspberry Camera V2 **c** fire **d** monitor shows the detected objects which are enclosed with bounding box

7 Conclusions

This paper proposed a real-time and embedded implementation of a fire and smoke detection technique that can reuse standard video cameras of surveillance systems. The target of this work is to develop smart IoT devices for fire/smoke detection in indoor and outdoor environments. The

Table 7 The % resource utilization and temperature measurement for the GPU and CPU in Jetson Nano while our method is running on it

	Performance (%)	Temperature (°C)
Jetson Nano (GPU)	99	54
Jetson Nano (CPU)	53.1	53.1

Table 6 Power consumption measurement in different scenarios

	YOLOv2 status	Power measurement (W)
Jetson Nano without monitor, keyboard, mouse	Off	1.24
Jetson Nano without monitor, keyboard, mouse	Running	4.19
Jetson Nano with monitor, keyboard, mouse	Off	2.24
Jetson Nano with monitor, keyboard, mouse	Running	5.19

algorithm is simple to design and can be trained fast. The proposed solution achieved promising results for accuracy in comparison to the state-of-the-art. In terms of false positive, it reduced the background mistakes in non-fire/smoke videos. Indeed, the proposed method performed very well, even with videos containing challenging features such as sun and clouds. The proposed detection solution stands for its low-latency and real-time performance when compared to the other regional-based detectors. Indeed, it can detect fire and smoke in 1 or 2 s as an early alerting alarm for the occurrence of fire and smoke accidents. In the future, we will extend our research to connect the proposed system to iCloud facilities for providing visual status and feedback of fire/smoke remotely. Moreover, recently released YOLOv4 models [35] will also be considered.

Acknowledgements Work partially supported by Dipartimento di Eccellenza Crosslab Project by MIUR. We thank the Islamic Development Bank for their support to the Ph.D. work of A. Elhanashi.

Funding Open access funding provided by Università di Pisa within the CRUI-CARE Agreement.

References

1. Hall, J.R.: The total cost of fire in the United States. National Fire Protection Association, Quincy (2014)
2. Gagliardi, A., Saponara, S.: Distributed video antifire surveillance system based on IoT embedded computing nodes. Springer LNEE **627**, 405–411 (2020a)
3. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
4. Saponara, S., Pilato, L., Fanucci, L.: Early video smoke detection system to improve fire protection in rolling stocks. SPIE Real Time Image Video Process **9139**, 913903 (2014)
5. Celik, T., ¨Ozkaramanlı, H., Demirel, H.: Fire and smoke detection without sensors: image processing based approach. In: 2007 15th European signal processing conference, IEEE, pp. 1794–1798 (2007).
6. Rafiee, A., Dianat, R., Jamshidi, M., Tavakoli, R., Abbaspour, S.: Fire and smoke detection using wavelet analysis and disorder characteristics. IEEE 3rd international conferance on computer research and development, vol. 3, pp. 262–265 (2011)
7. Vijayalakshmi, S.R., Muruganand, S.: Smoke detection in video images using background subtraction method for early fire alarm system. In: IEEE 2nd international conference on communication and electronics system (ICCES), pp. 167–171 (2017)
8. Gagliardi, A., Saponara, S.: AdViSED: advanced video SmokE detection for real-time measurements in antifire indoor and outdoor systems. Energies **13**(8), 2098 (2020b)
9. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp. 1097–1105. MIT Press, Cambridge (2012)
10. Wu, S., Zhang, L.: Using popular object detection methods for real time forest fire detection. In: IEEE 11th ISCID, vol. 1, pp. 280–284 (2018)
11. Sharma, J., Granmo, O.C., Goodwin, M., Fidje, J.T.: Deep convolutional neural networks for fire detection in images. In: International conference on engineering applications of neural networks, pp. 183–193, Springer, Cham.
12. Lestari, D., et al.: Fire hotspots detection system on CCTV videos using you only look once (YOLO) method and Tiny YOLO model for high buildings evacuation. In: 2nd international conference of computer and informatics engineering (IC2IE2019), Banyuwangi, Indonesia, pp. 87–92 (2019)
13. Shen, D., Chen, X., Nguyen, M., Yan, W.Q.: Flame detection using deep learning. In: IEEE 4th ICCAR, pp. 416–420 (2018)
14. Barmpoutis, P., Dimitropoulos, K., Grammalidis, N.: Real time video fire detection using spatio-temporal consistency energy. In: 10th IEEE international conference on advanced video and signal based surveillance, pp. 365–370 (2013).
15. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: IEEE conference on computer vision and pattern recognition (CVPR), Las Vegas, NV: CVPR, pp. 779–788 (2016)
16. Girshick, R., et al.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: IEEE CVPR, pp. 580–587 (2014)
17. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In: IEEE CVPR, pp. 6517–6525 (2017)
18. Hossain, S., Lee, D.J.: Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices. Sensors **19**(15), 3371 (2019)
19. Fire and Smoke dataset.: Kaggle.Com. https://www.kaggle.com/ashutosh69/fire-and-smoke-dataset?. Accessed 20 Aug 2020.
20. MathWorks Team.: Using ground truth for object detection. MATLAB central file exchange. https://www.mathworks.com/matlabcentral/fileexchange/69180-using-ground-truth-for-object-detection (2019). Accessed 24 Oct 2019.
21. Glorot, X., et al.: Understanding the difficulty of training deep feedforward neural networks. IEEE Trans Instrum Meas **54**(4), 249–256 (2005)
22. Brownlee, J.: Deep learning with python. Machine Learning Mastery, Vermont (2017)
23. Di Lascio, R., Greco, A., Saggese, A., Vento, M.: Improving fire detection reliability by a combination of video analytics. In: International conference image analysis and recognition, Vilamoura, Portugal, Springer: Cham, CH (2014)
24. Wang, G., Zhang, Y., Qu, Y., Chen, Y., Maqsood, H.: Early forest fire region segmentation based on deep learning. In: Chinese control and decision conference (CCDC2019), Nanchang, China, pp. 6237-6241 (2019)
25. Fu, T.J., Zheng, C.E., Tian, Y., Qiu, Q.M., Lin, S.J.: Forest fire recognition based on deep convolutional neural network under complex background. Comput. Mod. **3**, 52–57 (2016)
26. Jadon, A. et al.: FireNet: a specialized lightweight fire and smoke detection model for real-time IoT applications (2019). https://arxiv.org/abs/1905.11922. Accessed 7 Nov 2020

27. Filonenko, A., et al.: Fast smoke detection for video surveillance using CUDA. IEEE Trans. Ind. Inform. **14**(2), 725–733 (2018)
28. Yuan, F., Fang, Z., Wu, S., Yang, Y., Fang, Y.: Real-time image smoke detection using staircase searching-based dual threshold AdaBoost and dynamic analysis. IET Image Process. **9**(10), 849–856 (2015)
29. Fire Safety Search.: Very early warning fire detection—fire safety search. https://www.firesafetysearch.com/very-early-warning-fire-detection/ (2020). Accessed 26 April 2020.
30. Jetson Nano Developer Kit. https://developer.nvidia.com/embedded/jetson-nano-developer-kit (2020). Accessed 25 Feb
31. Habiboğlu, Y., et al.: Covariance matrix-based fire and flame detection method in video. Mach. Vis. Appl. **23**, 1103–1113 (2012)
32. Kim, B., Lee, J.: A video-based fire detection using deep learning models. Appl. Sci. **9**, 2862 (2019). https://doi.org/10.3390/app9142862
33. Barmpoutis, P., Dimitropoulos, K., Kaza, K., Grammalidis, N.: Fire detection from images using faster R-CNN and multidimensional texture analysis. In: IEEE ICASSP2019, Brighton, UK, pp. 8301–8305 (2019)
34. Li, Pu., Zhao, W.: Image fire detection algorithms based on convolutional neural networks. Case Stud. Thermal Eng. **19**, 100625 (2020). https://doi.org/10.1016/j.csite.2020.100625
35. Bochkovskiy, A. et al.: YOLOv4: optimal speed and accuracy of object detection (2020). https://arxiv.org/pdf/2004.10934. Accessed 7 Nov 2020

Sergio Saponara is a Full Professor of Electronics and leader of the I-CAS (Integrated and embedded Circuits and Systems) lab, at Dipartimento di Ingegneria della Informazione, Università di Pisa, via G. Caruso 16, 56122, Pisa, Italia.

Abdussalam Elhanashi is a PhD student at the I-CAS (Integrated and embedded Circuits and Systems) lab, Dipartimento di Ingegneria della Informazione, Università di Pisa, via G. Caruso 16, 56122, Pisa, Italia.

Alessio Gagliardi is a PhD student at at the I-CAS (Integrated and embedded Circuits and Systems) lab, Dipartimento di Ingegneria della Informazione, Università di Pisa, via G. Caruso 16, 56122, Pisa, Italia.