Journal of Information Systems Engineering and Management
2025, 10(3)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The Development of an AI-Based Fire Detection System for Hospital Buildings

[1]Dalal Alanezi, [2]Ibrahim A. Motawa, [3]Labib M. Labib,
[1]PhD researcher-Mansoura University Communications engineer, Head of Facilities and Buildings Department, Ministry of Health, Kuwait (Corresponding author)
Email: d_h_alanezi@std.mans.edu.eg
[2]Belfast School of Architecture and the Built Environment, Ulster University, Belfast, UK, And Department of Structural Engineering, Mansoura University, Egypt.
Email: i_a_motawa@mans.edu.eg
[3]Deprtment of Computer Engineer and Control Systems, Mansoura university, Egypt.
Email: labib_essa@mans.edu.eg

ARTICLE INFO

Received: 26 Dec 2024

Revised: 14 Feb 2025

Accepted: 22 Feb 2025

ABSTRACT

The study presented is aimed at implementing the fire detection technique by using the object detection approach at the hospitals in Kuwait. The objective is to overcome the shortcomings associated with the existing fire detection systems involving poor response time, poor accuracy, and frequency of maintenance required. The proposed approach involves a deep learning-based approach involving image processing to optimize the overall fire detection strategy, and improve the response time and accuracy. A parallel pooling approach has been proposed to enable the system to handle multiple data streams simultaneously. Such an approach offers improved efficiency in hospital settings where the detection of fire in a rapid manner is inevitable to avoid any disaster. By utilizing such a parallel pooling approach with deep learning object detection algorithms. The study offers a novel fire detection system. Following the Yolo architecture version 3 has been used for tracking the instances of fire and capturing the areas where any such instances exist.

Keywords: Fire Detection, Object Detection, Image Processing, AI Technologies.

1 INTRODUCTION

Fire is an indispensable part of everyday life that serves as a fuel to drive many appliances. Yet upon getting uncontrolled, it can be highly fatal. At the global level, the deaths and injuries caused by the fire are considerably high. As per the statistics issued by the US Fire Department, an average of 5,750 instances of fire outbreaks took place in healthcare facilities alone during the period 2011-2015. The intensity of such instances generally remained small for 96 percent of the time. In the remaining ratios, the fire breakout spread outside the rooms and resulted in several deaths. A total of 157 injuries were recorded with a total damage to the property mounted to 50.4 USD. The ratio of injuries caused in nursing homes due to fire maintains a large proportion. This indicates that the distribution of fire and potential sources can vary across healthcare facilities. (NFPA,2017).

According to the General Fire Service's annual report in launched in Kuwait, financial losses from fires in the first half of 2023 were estimated to be at around 17 million dinars, or roughly 52.5 million dollars (Murad, 2023). Specifically, the damage in residential areas was valued at 531 thousand dinars which is high as compared to previous years, while non-residential areas, including factories and stores, suffered losses exceeding 14 million dinars or 45 million dollars. Additionally, land transport-related fire losses totaled 1.6 million dinars. These figures represent substantial losses relative to the regions assessed.

The hospitals sites are sensitive and need intensive care for caring patients and medical staff. Fire accidents have become very dangerous, and attention must be paid to reducing or preventing such disasters. Not far away was the tragic accident that claimed the lives of 49 people according to (Reuters,2024) in the burning of a residential building containing workers in the Al-Mangaf area in Kuwait, which contributed to enriching the search for solutions to contain the dangers that threaten people’s lives as a result.

Numerous factors contribute to the ignition of fire in hospital settings. The ageing infrastructure and inadequate measures taken regarding the prevention and control of fire are the primary reasons. As the hospital buildings and equipment get older (without undergoing routine maintenance and checks), it can increase the probability of fire outbreaks. Once the fire gets ignited due to such factors, it can spread rapidly and can lead to extensive injuries to already admitted patients who then become difficult to relocate (Yousefli, 2017).

Once a fire outbreak, it needs to be controlled by firefighters. Such a task is challenging in the circumstances of the outbreak and the spread is severe. It then requires the services of firefighters and technology to curb the issue (Healey, 1993). Therefore, the idea is to prevent the ignition and spread of fire at a very early stage where the internal staff can use basic firefighting equipment to prevent any major injuries and challenges. In making such rapid detections, the role of artificial intelligence (AI) is seen as essential to device accurate and state-of-the-art fire detection systems. This work aims to analyze the use of AI systems in making fire detection systems. The aim is to make use of object detection techniques for capturing the instances of fire and overall improve the fire detection outcomes.

1.1 Workplace overview

The study is to be applied on Aladan Hospital which is a general medical facility located in Kuwait, specifically in the Ahmadi Governorate - Hadiya area - King Fahd bin Abdulaziz Road. It was established in 1981, and is one of the largest government hospitals in the country. It specializes in providing medical care services to all patients in the Ahmadi area, in addition to the Mubarak area (Gesa,2019). The hospital includes 1037 beds totally equipped to receive patients divided between all departments as ICU, pediatrics, internal medicine depts... etc as illustrated in table 1 which is summarizing the hospital capacity and facility descriptions for Al-Adan Hospital (Alanbaa, 2024).

Table 1: description of different depts included in Al-Adan hospital

Facility	**Description**	**Area (m^2)**	**Beds Count**
Main Hospital (Old Building)	Primary hospital building (old section): Basement, Ground, 3 Floors.	25,770	602
Internal Medicine Building	Specializes in internal medicine: Basement, Ground, 4 Floors.	1,950	245
Heart Center for Adults (Cardiology)	Center focused on heart treatments for adults: Basement, Ground, 4 Floors.	950	73
Heart Center for Children	Pediatric cardiology center: Ground, 3 floors.	855	6
Al-Kharraz Kidney Center	Center for kidney dialysis: Ground, 2 Floors.	11,634	71
New Ambulance Center	Specialized center providing additional healthcare services: Ground, 1 Floor.	1,014	
Multi-level Parking	Parking facility for visitors and staff: Ground, 3 Floors.	11,288	
Warehouse	Storage facility for hospital operations: Ground.	1,491	
Nuclear medicine, Radiology and Laboratories Building	Diagnostic center for imaging and radiology services: Basement, Ground, 2 Floors.	1,903	
Comprehensive Care Unit (ICU)	Centralized medical care for specific needs: Ground, 1 Floor	1,595	40

Administrative Building	Office space for administrative staff: Ground.	1,044	
Nurses' Residential Building	Residential facility for nurses and hospital staff: Basement, Ground, 2 Floors.	4,875	
Medical Oversight & Health Admin. Center	Administration for managing hospital services: Basement, Ground, 2 Floors.	1,300	
Dental Center	Specializes in dental care and treatments: Ground.	4,300	
Blood Donation Transport Center	Facilitates the transport and handling of blood donations: Basement, Ground, 3 Floors.	14,69 8	

It is clear from the table that the hospital has a wide range of facilities, including specialized centers for cardiology (adults and children), kidney dialysis, blood donation, and dental care, as well as support facilities like parking, administrative offices, and residential buildings for hospital staff so it serves a very large residential area with a population of about one million people all these buildings are connected currently to main fire alarming panel and efficiency of work will be deeply affected by updating the currently installed fire alarm techniques (Sami, 2023).

1.2 AI and Object Detection Techniques for Fire Detection

The use of AI offers better capabilities and opportunities to detect the fire at a much earlier stage and deploy fire prevention strategies. The AI system can further be leveraged to predict the potential spread of the fire and the behaviour that may take place. Thus by leveraging their power, advanced fire detection systems can be developed and deployed in fire-prone areas. In recent times, some AI-based fire alarm systems have gained significant traction. Such systems have been trained on large datasets having fire and non-fire instances. A study published by Khan et al. (2022) evaluated the efficiency of the neural network in detecting fire by surveying fire alarm owners. This led to making an empirical comparison between the AI and non-AI alarm systems and determining the accuracy of true alarm rates. It was found that AI-based alarm systems offer improved efficiency compared to conventional systems. The adoption of neural networks resulted in lower false alarms and higher accuracy leading to a reliable detection of fires. It was also found that the AI-based systems offer improved detection time compared to the traditional systems. Such findings help us understand the efficacy of the use of AI in the detection of fire (Ge. L, 2020).

While analyzing the AI-based fire detection methods, they fundamentally reside on the object detection approach. Falling under a broader domain of computer vision, object detection has remained an active area of research across numerous applications including robotics, medical imaging, etc. Object detection methods focus on localizing the distinct properties associated with the targets, presented in the form of static moving frames. Such techniques by relating these properties with labels capture location of those targets. Many types of deep learning methods can be used in object detection. Some of these methods are presented in Figure 1.

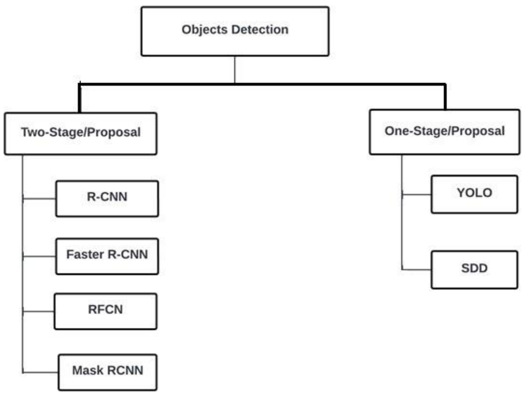


Figure 1: Deep Learning-based Object Detection Methods

1.3 Object Detection Methods

As presented above, object detection can be carried out by adopting several deep learning-based approaches. Amongst those methods, You Only Look Once (YOLO) is one of the leading methods that is being developed and adopted in recent times. There are numerous versions of the YOLO method available ranging from YOLOv1 to YOLOv8. Amongst these methods, YOLOv3 is one of the prevailing methods that has been adopted as part of this study. Known for its ability to detect distinctive features it is suitable for capturing instances of fire. One of the primary benefits that the method offers is its real-time performance capabilities that allow instinctive detection of the fire outbreak. Therefore, the approach offers precise object localization by using the anchor boxes and multi-scale detections to locate the objects (fire and smoke etc.). Such precision helps improve the efficiency of the system and leads to the detection of fire-related objects in images or video frames (Jin, 2020).

The YOLOv3 becomes particularly suitable for fire detection due to its ability to detect fire from live video streams (such as CCTV cameras). Such immediate detection by processing the visual data leads to swift identification of the fire instances in the form of smoke and flames. This can allow the ground staff to promptly react to the fire hazards (Yanjia, 2019). By residing on such improved response time, YOLOv3 helps improve the responsiveness of the system and its effectiveness (Ge. L, 2020).

The YOLOv3 system fundamentally resides on the underlying dataset used for training and the baseline models. The model has been trained on the COCO dataset having many categories of objects and a Darknet-53 architecture as a base model. The transfer learning capabilities of the model further make it suitable for detecting fires under ad-hoc settings. Through a simultaneous control of classification and localization, this model allows fast and efficient detection of fire and makes it an ideal candidate for developing advanced fire detection systems (J. Redmon, 2016).

The MS COCO dataset, as alluded above to, comprises several object categories. Under this dataset, a comparison between several YOLO architectures in the form of YOLOv3, YOLOv4 and SSD is available under the metrics of Average Precision, etc. The average precision measures the accuracy of the detection method and analyzes its capability in detecting and localizing the object in the given frames. Another metric used for analyzing the effectiveness of such models is Frames Per Second which indicates the speed of the algorithm in processing the object detection tasks in real-time (Woo et al, 2022). The relationship between the average precision and frames per second is depicted in Figure 2. The trade-offs between the accuracy and speed can be analyzed. For example, YOLOv3 and YOLOv4 offer a balance between high precision and real-time performance. SSD on the other side offers trade-offs in terms of prioritizing speed (Woo, 2022). Such comparative analysis helps determine a suitable model based on the specific requirements of the system (Pal, et al. 2021).

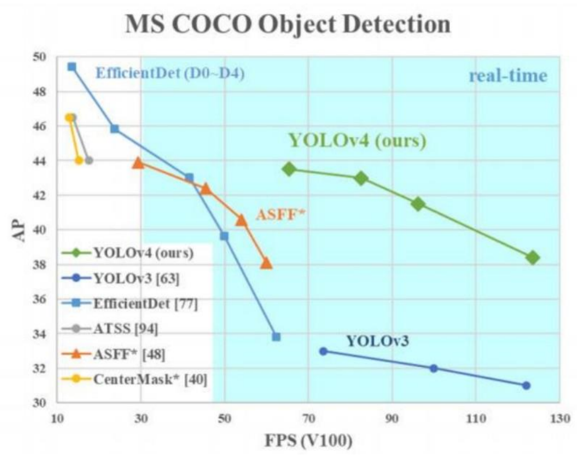


Figure 2: Comparison between YOLO architectures and SSD (Rakhimov ,2021)

The selection between the Average Precision (AP) and Frames Per Second or FPS is depending upon several factors and applications. Below is an illustration of how these parameters can be chosen to analyze the performance of the models.

1.3.1 Higher AP

The higher AP measure the accuracy that the object detection model can attain in localizing and detecting an object contained in the image. The measure is computed as the area under a precision-recall curve. Such analysis offers a trade-off between the correctness of detections and the completeness of detections. When AP is high, it indicates that the algorithm is capable of detecting and localizing the object. This aspect is particularly important for critical settings like medical facilities. Therefore, the application of accurate object detection is critical for accurately determining the object time present inside a frame.

1.3.2 3.2 Higher FPS

FPS determines the number of images that the object detection algorithm can process in a second. When FPS is high, it indicates a rapid detection capability of the algorithm and vice versa. Such a metric is critical for applications that require real-time performance (Pal, et al, 2021). For example, in those areas where video surveillance is important, the FPS becomes of paramount importance. Similarly, in robotic applications and the use of Augmented Reality, the use of FPS as a performance metric is critical (Woo et al., 2022).

Thus, if the goal is to detect objects with high accuracy, then the developers can prioritize AP. These include medical applications and others like autonomous driving. Even minor misdetections in these instances can lead to higher costs (Pal, et al. 2021). In other cases where speed of detection is critical such as live streaming and surveillance, the use of FPS can serve as a better metric. The models are enabled to process and respond to the data in real time and prevent security lapses (Rakhimov, 2021).

While prioritizing one metric over another, another consideration is to maintain a suitable or acceptable balance between both variables AP and FPS to attain the optimal performance of the system in objects detection (Pal, et al. 2021). While trading-off one metric for another can offer a balanced approach to attain high accuracy and efficiency. A system having a higher FPS rates can offer real-time monitoring of fires and logical rapid alerts. Therefore, in the detection of fire, maintaining a balance between the both is of paramount importance.

1.4 Processing Techniques

These are several steps that are involved in making object detections. Multiple types of layers are employed that starting from a broader frame led to capturing the object itself. Some of such techniques have been presented as part of this section.

1.4.1 Image Dimensionality Reduction

Pooling layers are amongst one of the prevailing approaches that are coupled along the convolutional layers to narrow down the object detection area. Such an approach aids in reducing the spatial resolution and dimensions of the input images by discarding the information associated with the background or irrelevant areas. This leads to improving the computational load and reducing the effect of overfitting (Mekhriddin, 2016). Pooling algorithms are generally performed on the feature maps which can be derived from the contained convolutional layers in the images. Each of such feature maps is present in the form of a selected grid of values at several spatial positions inside the frame. Pooling thus helps in aggregating the values contained in the local regions of the features map. Many types of pooling operations exist that include the average pooling, and LP pooling (Absalom et al, 2020).

Max pooling is a widely used method that splits up the input feature map into non-overlapping rectangular objects depending on the kernel size. The maximum values are taken from within these regions to down-sample the feature maps. While the size is reduced, the values that can attain the strongest activation are retained (Mekhriddin, 2016). The pooling operation involves the sliding kernels (windows) across the feature maps.

1.4.2 Parallel Pooling

While the pooling operations involve the involvement of kernels to reduce the size of images, another related idea is the use of parallel pooling. While the YOLOv3 architecture follows a cascaded deep-learning network, it suffers from problem related to increased time of inference. The network would involve numerous smaller networks used for processing the input image in a sequential format and thus render slower detection times (Absalom, 2020). To overcome these issues, a parallel pooling approach can be adopted.

Parallel pooling is based on transforming the input image into several smaller images. This allows the model to simultaneously process these images by using multiple threats and cores available at the hardware levels. Thus, such parallelism, the inference time of the model gets reduced at a significant level (Absalom, 2020).

Another limitation of the YOLOv3 cascaded networks is that they suffer from loss of information across different stages of the network. Each of the stages aims to detect different types of objects having different scales.

This can lead to a loss of contextual information related to the studied objects. For instance, small objects may get filtered out at the first stage of object detection which can result in an overall poor accuracy (Lin et al, 2014).

Parallel pooling aids in overcoming such limitations as well by using a fusion approach of intermediate features. It is better to combine the stages rather than separating them, so the features from different stages are combined to allow better retention of the contextual information which can lead to improve the selected object detection (Yanjia,2019).

2 THE SELECTED TECHNIQUE ILLUSTRATION

In the selected approach, parallel pooling in YOLOv3 has been adopted to improve upon the existing detection methods. The approach works like spatial pyramid pooling by employing multiple pooling layers of varying sizes to capture the features having varying scales.

In YOLOv3 the Darknet-53 serves as a backbone where convolutional layers are stacked together to reduce the spatial dimensions of the input image, at the same time increasing the number of channels. By adopting a series of convolutional layers combined with a pooling layer, fine-grained spatial information is achieved (Elgendy et al, 2020). Parallel pooling comprises several kernel sizes which are applied to the feature maps.

Each of these features this captured at a particular scale. The output from these feature maps is combined along the dimensions of the channel. This helps the model to capture features at various scales and retain the spatial information which is critical for object detection. The concatenated feature maps from the parallel pooling are then processed further through several convolutional layers to predict the bounding boxes and class probabilities associated with the objects. This leads to attaining improved confidence for the detection.

2.1 Improvements achieved by incorporating Parallel Pooling

There are several benefits of incorporating parallel pooling presented as follows:

- Parallel pooling helps in the simultaneous processing of several regions. This allows the model to attain the desired results in less inference time. This helps in processing a large set of input images in a limited timeframe.
- Parallel pooling involves various processing capabilities inside its structure such as multi-threading procedures and the utilization of parallel hardware accelerators aspects. This helps in maximizing the computational power optimization. The hardware resource can be used in a highly effective manner.
- In parallel pooling when the images are divided into smaller regions, more information can be captured to retrieve contextual features instead of overly relying on the localization details. This can help in improving the accuracy of the object detection. This aspect particularly helps in the instances when the objects are of varying aspect ratios and sizes.
- Parallel pooling permits improved scalability and enables the detection of objects through a wider range of selected images sizes and its specified dimensions (Jin, 2020).

The implementation of YOLOv3 requires making several changes at the architectural levels such as defining the training parameters. This enables the model to support distributed processing and efficient parallelization strategies such as constrained associated with hardware and memory.

2.2 Deep Learning Integration

The YOLOv3 architecture fundamentally resides in the use of deep learning algorithms. Such algorithms are essential in making the detection of fire that can be associated with complex patterns in the image (Pal, et al, 2021). The models are trained on a huge amount of labelled data that is structured on multi-layered neural network architectures. Some of the implementation stages involved in the model development are illustrated below:

Feature Extraction with Convolutional Neural Networks (CNNs): CNNs layers are the most critical and fundamental layers involved in the deep learning models. The deep learning architectures of this type involve several CNN layers that capture features including edges, textures, and shapes of the objects. Such features are imperative for the detection of the object (Ge, 2021).

Utilizing the YOLOv3 Architecture: The YOLOv3 architecture employs several CNN layers in combination with max-pooling layers to analyze and detect the images. The CNN layers extract relevant features while the max-pooling layers down-sample the images to infer the bounding boxes of the target objects. This leads to efficient and precise detection of the objects.

Prediction of Bounding Boxes and Object Classes: The YOLOv3 model outputs the returns of both the bounding box and background information during the object detection. This allows the system to capture multiple objects such as fire and smoke.

Fine-Tuning and Model Training: Deep learning models like YOLOv3 are trained on labelled datasets having bonding box information of the known objects. A backpropagation approach helps in the optimization of determining the stochastic gradient descent and iteratively improves the accuracy levels. The model fine-tuning further improved the ability to detect fire-related features (Elgendy, 2020).

Real-Time Inference and Detection: During the detection phase, the trained model is subjected to new images to detect any instances of the fire. The images are passed through the model and this results in the identification of fires and trigger any alarms if a bounding box is detected.

YOLOv3 model detects the bounding boxes along the class probabilities by adopting 53 convolutional layers and 23 residual layers. This makes it highly effective for the real-time detection tasks (Redmon & Farhadi, 2018).

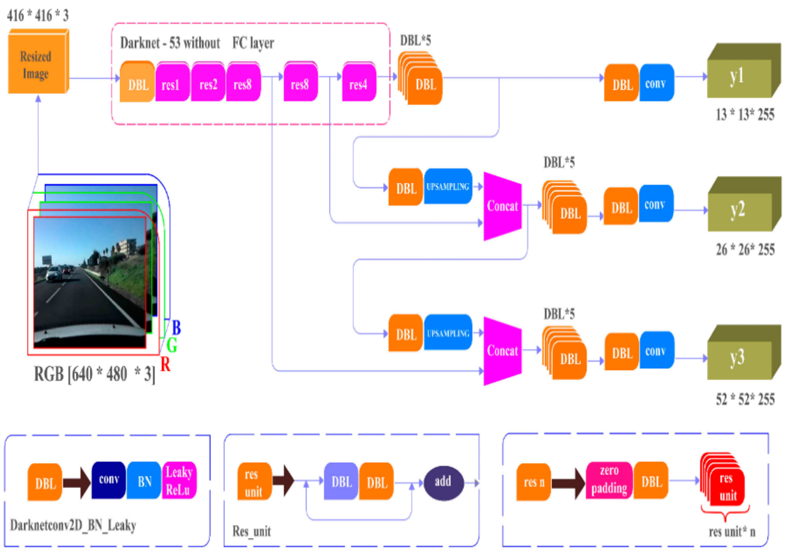


Figure 3: System's deep learning integration (Guerrieri, 2021)

The kernel sizes of 1, 2 and 3, are to be applied in the convolutional layers of the system in order to extract the image features. This allows the model to attain improved detection and classification of the fire. Such a combination of layers leads to the guaranteed detection and convergence of the model.

The kernel size of 1 can help render complex features by combining information from numerous channels.This improved the model's ability to capture the intricate patterns. Such kernels help adjust the number of channels associated with the feature maps without discarding the spatial dimensions. Thus, they serve as channel-wise pooling and a linear combination of the input channels. The kernel size of 3 follows more standard convolutional operations to extract broader and more detailed spatial features from the input images.

It can capture the information from the neighbouring pixels that are essential for robust extraction of the information. Thus, it can capture the local dependencies by sliding over the images and capturing the features associated with fire including edges, and irregular shapes. The 3x3 convolution operation potentially reduces the feature map while capturing the spatial features. This helps in capturing the most appropriate features contained in the images (Pal, et al, 2021).

Through the application of these kernels, the model extracts a wide range of features at multiple abstraction levels.

This allows the model to efficiently capture the fine-grained and coarse features. This improved the classification and detection capabilities.

Following the convolutional layers, there are several layers designed to capture the remaining relationships and converge the model to the desired output classes. These can be in the form of activation functions, normalization layers, and fully connected layers. The convergence of the model is critical for it to perform on unseen datasets (Ren et al., 2020). For the detection of several areas included in same object simultaneously, the parallel pooling employed fuses three kernel sizes (52, 26 and 13). Through a fusion of these three scales, the model integrated various resolutions and improved the ability of detection (at numerous locations and scales). The fusion process helps in enhancing the robustness of detection and allows the model to capture the objects accurately regardless of the size (Pal, et al, 2021).

3 SYSTEM IMPLEMENTATION

The proposed YOLOv3 architecture has been developed using SqeezeNet which serves as a lightweight model at the base. SqeezeNet offers efficient operations in terms of optimizing the computational resources, controlling the model size and making it suitable for the detection of fire (Ren et al., 2020).

Two detection heads are in use. The first head starts capturing the large objects which are contained in the images of the fire dataset. Then second head is 2 times larger than the first and focuses on the detection of smaller objects. This hierarchical approach allows the model to capture objects of various sizes and improve the overall detection performance.

Anchor boxes are used for improved localization and making predictions. The predefined bounding boxes are used during the training as they help in the identification of the known objects. These include various sizes and aspect ratios passed on during the training phase. Such variations help the model to capture numerous characteristics of the object classes and this attains a high detection accuracy (Ren et al,2020).

The bounding box mechanism starts through the offset predictions. The model directly predicts the dimensions of the bounding boxes which can result in non-convergent training outcomes. To cater to this, the model predicts the offsets relative to the predefined anchor boxes. This allows the model to apply such predicted offsets to the anchor boxes to determine the final dimensions of the bounding boxes (Pande et al 2018).

The model employs three anchor boxes corresponding to each cell in the features map. This results in making predictions on the three bounding boxes per cell. Such bounding boxes are passed into a transfer learning model taking anchors derived from the COCO dataset using k-means clustering. This ensures that the boxes are of the exact shape and sizes which are found in the dataset. The second anchor is the dimensions of the offset through which predictions are made using these offset values. These employ cluster centroids that stably allow improved prediction convergence (Pande et al 2018). Resultantly, the center coordinates are predicted to correspond to the location of the filter application. These make use of the sigmoid function which allows for predicting the centers in a prescribed range. These have been determined using the following formula:

$$b_x = \sigma(t_x) + c_x \qquad (1)$$

$$b_y = \sigma(t_y) + c_y \qquad (2)$$

The dimensions of the predicted bounding box are calculated as follows:

$$b_h = p_w e^{t_h} \qquad (3)$$

$$b_w = p_w e^{t_w} \qquad (4)$$

Here b_x, b_y, b_w, and b_h are the x, and y coordinates of the center which represents, width, and height of objected prediction. t_x, t_y, t_w, t_h (xywh) can represents the network output. The top-left coordinates of the grid are abbreviated as c_x and c_y. p_w and p_h are the anchor dimension of the bounding predicted by using a log-space transformation applied to the network. Such transformation allows for catering to the requirements of a wide range of network sizes (Pande et al 2018). Such a log-space transformation further would involve taking the log of the network output. By taking such transformation of log-space then results are multiplied by the anchor having pre-defined anchor boxes of various sizes or shapes. Each of these anchor boxes is integrated with aspect ratios and a specific scale (Guerrieri, 2021).

The outputs of the network are changes to get the final bounding boxes that are predicted by applying the log-space transforms and the offsets to the predicted anchor boxes. This allows a stable convergence of the model and determines the accurate bounding boxes. The proposed YOLOv3 architecture has been designed for such an efficient object detection approach when the sizes of bounding boxes can vary. Through the incorporation of multiple detection heads and using log-space transform predictions, the model attains robust and accurate detection performance (Ren et al., 2020).

Through the application of a log-space transformation applied to the corresponding anchors, the output dimensions of the bounding box can be determined. This gets aligned with the anchor properties and produces accurate dimensions of the bounding boxes.

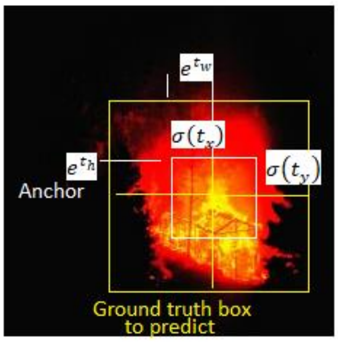


Figure 4: The dimensions of the bounding box

Predictions, b_w, and b_h are transformed into normalized versions by using the corresponding dimensions of the image. The training labels are determined using this approach such that prediction b_x and b_y for the box containing the desired object are based on the actual dimensions of the features map. During the training, the ground truth labels are further incorporated and normalized in a manner to match the predictions made by the network (Pande et al 2018).

The following equations can be used for determining the ground truth boxes.

Considerting# $g_x, g_y, g_h and\ g_w$ corresponding to the real values of predicted parameters $b_x, b_y, t_h and\ t_w$ respectively. The true quantities of $\hat{b}_x, \hat{b}_y, \hat{t}_h and\ \hat{t}_w$ can be determined as follows:

$$\sigma(\hat{t}_x) = g_x - c_x \qquad (5)$$

$$\sigma(\hat{t}_y) = g_y - c_y \qquad (6)$$

$$\hat{t}_w = \log(\left(\frac{g_w}{p_w}\right) \qquad (7)$$

$$\hat{t}_h = \log(\left(\frac{g_h}{p_h}\right) \qquad (8)$$

The coordinates prediction errors can be computed using the loss function provided below:

$$E_2 = \sum_{t=0}^{S^2} \sum_{f=0}^{B} \mathrm{W}_{\mathrm{tf}}^{obj} [(\sigma(t_x)_i^j - \sigma(\hat{t}_x)_i^j)^2 + \left[(\sigma(t_y)_i^j - \sigma(\hat{t}_x)_i^j\right)^2] + \sum_{t=0}^{S^2} \sum_{f=0}^{B} \mathrm{W}_{\mathrm{tf}}^{obj} [((t_w)_i^j - (\hat{t}_w)_i^j)^2 + [((t_h)_i^j - (\hat{t}_h)_i^j)^2] \qquad (9)$$

The model's layered architecture comprises numerous components that have been designed to improve the feature extraction capabilities. The convolutional layers have been employed to determine the features contained in the input images including edges, textures and shapes. The ReLU activation function has been used that introduce non-linearity in the model and allow it to learn complex patterns in the data. The resizing layers have been added to reduce the spatial dimensions of the feature maps which allows the layers to be correctly scaled and suitable for the next processing steps. A depth concatenation layer has been added that combines the features that have been extracted at different scales. Integrating this multi-scale information allows the model to detect objects of numerous sizes with precision.

The overall architecture of the model has been presented below in Figure 5. The model focuses on extracting the contextual information from the images to allow a better understanding of the inherent details. The model employs SqueezeNet's efficiency and a YOLOv3 architecture that helps improve the effectiveness of fire detection. Such optimization allows an efficient and accurate detection of fire.

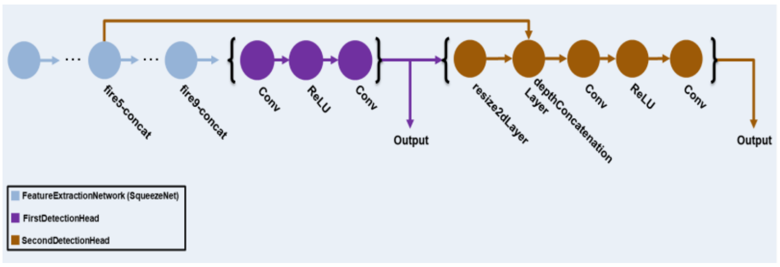


Figure 5: YOLOv3 Architecture (Ishfaq, 2022)

The YOLOv3 architecture has been modified to add parallel pooling to enhance the speed and performance. This allows the model to distribute the tasks among multiple workers and thus speed up the training process. This has been incorporated by using MATLAB's parallel pooling toolbox. After confirming the availability of the parallel pools available in the processing system, the parfor loops are used to distribute the iteration across multiple workers. This allows tasks to be executed in parallel. This approach results in improvements in the processing speed and has been captured using the following equation:

$$S = \frac{T_{serial}}{T_{parallel}} \tag{10}$$

Where T_{serial} is the time for non-parallel executions, and $T_{parallel}$ is the execution time for the parallel version. Through adopting this approach, the scalability of the model has been enhanced through optimal utilization of the available resources. By increasing the processors to handle the tasks, the execution time gets enhanced. The approach is depicted in Figure 6.

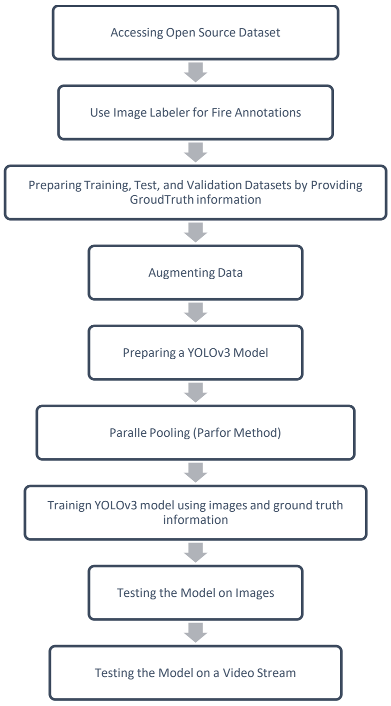

Accessing Open Source Dataset
Use Image Labeler for Fire Annotations
Preparing Training, Test, and Validation Datasets by Providing GroudTruth information
Augmenting Data
Preparing a YOLOv3 Model
Paralle Pooling (Parfor Method)
Trainign YOLOv3 model using images and ground truth information
Testing the Model on Images
Testing the Model on a Video Stream

Figure 6: YOLOv3 System Level Modelling

3.1 The Open-Source Datasets Accessing

The initial part carried out in the model was to obtain the image dataset. These images were taken from a datastore where numerous scenarios were available. The dataset consists of FiSmo-Images that offer numerous examples of fire occurrences. In this context, the images were taken from the FlickrFire having 984 files with varying dimensions. To all uniformity and model optimization, the images were resized to 227x227x3. This allows adding a seamless integration of the images to the model and also allows optimized use of memory resources.

3.2 Annotations by Image Labeler

To prepare images suitable for the training, an image labeler was used. Although numerous methods exist in the form of colouring thresholds for the identification of the fire regions, this approach allows the training of the model in an object detection manner by placing the bounding boxes. The image labeller aided in adding annotations to the images by placing such bounding boxes. A uni-label case has thus been introduced by selecting 'Fire' as a label. Such annotations including the image paths and fire coordinated were stored in the form of a matrix file. The entire dataset was passed through such annotations for training and testing. The outcomes of the sample image annotations have been depicted in Figure 7.

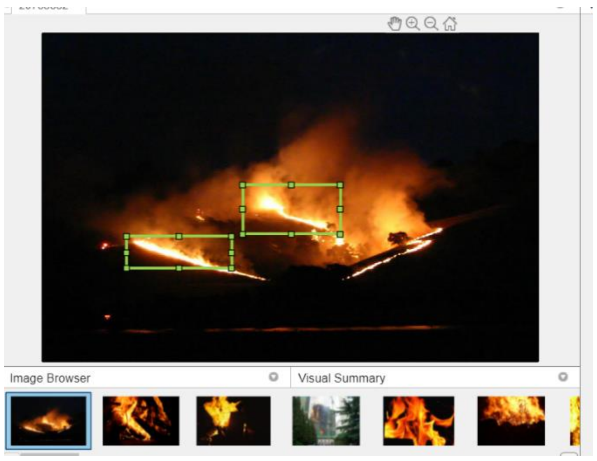


Figure 7: Image Labeler Layout

3.3 Pre-processing and Split (Preparing Data)

In the initial phase, it's important to ensure the accuracy of the ground truth data provided to the model, especially concerning the 'Fire' label category, to avoid potential errors downstream. To achieve this, a validation check is conducted to ensure that every image in the dataset contains label information. Any images lacking such annotations are removed from further consideration. Subsequently, the dataset is split into training and test subsets, with 60 percent of the data allocated to training. The training subset is used to train the model, while the test subset is reserved for assessing its performance. Following this, both subsets undergo data augmentation to enrich the dataset with variations, thereby enhancing the model's accuracy and resilience.

3.4 Data Augmentation

During this phase, data augmentation methods are applied to diversify and strengthen the dataset, ensuring the model's adaptability to different angles and scenarios. This process is essential for accommodating real-world variations, such as slight image tilts, ensuring the model's effectiveness in practical applications.

To achieve this, a data augmentation function is utilized, introducing a range of transformations to the images. These include random horizontal flips to simulate varied viewing angles and scaling with overlap to capture object size variations. Additionally, adjustments to colour attributes, such as contrast, hue, saturation, and brightness, are applied to introduce variability in image appearance.

A demonstration of these data augmentation techniques applied to sample images is presented below in Fig. 8, showcasing the diverse transformations implemented to enrich the dataset and enhance the model's robustness.

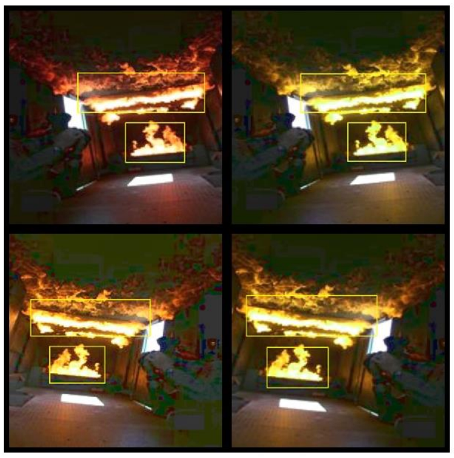


Figure 8: Images Data Augmentation Example

3.5 Preparing YOLOv3 Model

The data pre-processing involves setting up a YOLOv3 model and passing it through the pre-processed data for training. The YOLOv3 existing object in MATLAB has been utilized for the said purpose. The overall architecture developed in the model thus involves numerous parameters such as the allocation of the base network, class names corresponding to various categories, detection box dimensions, and the input image size range. These have further been illustrated as follows:

- Base Network: Defines the fundamental architecture of the YOLOv3 model.
- Class Names: Specifies the names of the object classes that the model is trained to recognize and detect. This has kept to be Fire in this case.
- Detector Boxes: Defines the dimensions of the fire regions corresponding to the background images.
- Image Size: The input image size is in the form of a range.

Such parameters allow the model to be used for training and subsequent detection of the objects provided in the dataset. The arguments employed for the model training and testing have been provided in Table 1.

Table 2: Object Detector of YOLOv3

Variables	Values
Basements	Squeezenet
classNames	'Fires'
Anchor_boxes	Step 2 Taken
Image_Sizes	"227x227x3"

Once the detector is in place, it is trained to pre-process the data that has been obtained using the data augmentation to enhance the training process (Çakar et al, 2021). The next step involved is the allocation of training parameters that are selected through an iterative approach and the dataset type. The training parameters adopted have been presented in Table 2.

Table 3: Training Parameters of YOLOv3

Variables	Values
Epoches	80
Batch_Size	8
Learning_Rate	$1*10^{-3}$
L2 Regularization	$5*10^{-4}$
Penalty_Threshold	0.5

3.6 The Model Training

When the selected model is subjected to training, the MATLAB parallel pooling helps in expediting the process. The loss function helps determine the disparity between the predicted and ground truth labels in the form of bounding boxes. This helps in optimizing the training parameters that the model can learn. Some of the loss functions employed during the training process have been presented below:

Localization Loss (bbox_loss): The loss function helps evaluate the difference among box parameters of the predicted bounding and the ground-truth parameters. The loss is measured in the form of mean squared error (MSE) and a smooth L1 loss (Hu et al, 2022). The calculations involve an assessment of the difference between each of these bounding boxes and summing up the overall outcomes. Such disparity is quantified by taking the sum of squared differences between the actual and predicted dimensions of the bounding box and taking an average over all the instances. This is configured using the MSE equation presented below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \quad (11)$$

Where:

yi represents the ground truth bounding box coordinates.

$\hat{y_i}$ represents the predicted bounding box coordinates.

n is the total number of bounding boxes.

Which can be modified to be:

$$bbox_{loss} = \frac{1}{n}\sum_{i=1}^{n}((b_{xi} - t_{xi})^2 + \left(b_{yi} - t_{yi}\right)^2 + \left(\sqrt{b_{wi}} - \sqrt{t_{wi}}\right)^2 + \left(\sqrt{b_{hi}} - \sqrt{t_{hi}}\right)^2) \quad (12)$$

The confidence loss is another loss function that helps in the quantification of the difference between the predicted object detection scores. This helps in identifying if an object is present or not. The comparison is assessed by using a binary cross-entropy function. This helps in identifying the model's ability to detect the objects accurately (Hu et al, 2022).

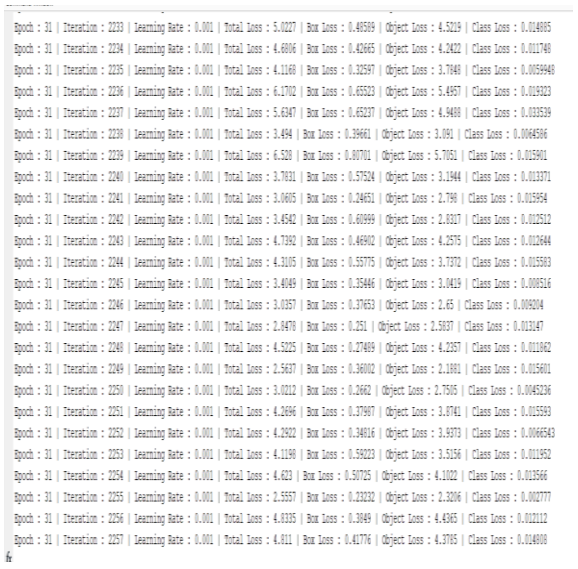
$$Obj_{loss} = -\frac{1}{m}\sum_{i=1}^{m}(y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)) \quad (13)$$

The next loss function employed is the class loss that quantifies the difference between the probabilities assigned to a class and the true class labels (as specified in the actual data). The loss is computed by using the categorical cross-entropy that helps in optimizing the model's ability to classify the objects (Çakar et al, 2021).

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i . \log(\hat{y}_i) \quad (14)$$

When the training is conducted, the total loss is computed through a combination of the individual loss components. Each of these components assigns specific weights to allow their contributions and balanced outcomes. The weights of the components can vary depending on the specific implementation and task requirements. The fundamental idea during the training is to minimize the overall loss values to ensure that the model is being trained to make accurate predictions. The optimization process involves an adjustment of the model's parameters by using techniques such as backpropagation and gradient descent. These techniques help in reducing the loss and improve the model's performance. There are several steps involved in the training procedure

- Gradient, State, and Loss Determination: This step involves determining the gradient values, state and loss information of the model. This is accomplished by utilizing the function that calculates the gradients of the model and passes the dataset and the YOLOv3 detector as the parameters.
- L2 Regularization: After determining the gradients, the L2 regularization is carried out that help prevention from the overfitting of the model.
- Updating Learning Parameters: The learning parameters of the model are updated by using the calculated gradients. The corresponding gloss and information are displayed in the MATLAB workspace for monitoring and analysis purposes. These outcomes have been depicted in Figure 9.



```
Epoch : 31 | Iteration : 2233 | Learning Rate : 0.001 | Total Loss : 5.0227 | Box Loss : 0.48589 | Object Loss : 4.5219 | Class Loss : 0.014885
Epoch : 31 | Iteration : 2234 | Learning Rate : 0.001 | Total Loss : 4.6806 | Box Loss : 0.42665 | Object Loss : 4.2422 | Class Loss : 0.011748
Epoch : 31 | Iteration : 2235 | Learning Rate : 0.001 | Total Loss : 4.1168 | Box Loss : 0.32597 | Object Loss : 3.7848 | Class Loss : 0.0059948
Epoch : 31 | Iteration : 2236 | Learning Rate : 0.001 | Total Loss : 6.1702 | Box Loss : 0.65523 | Object Loss : 5.4957 | Class Loss : 0.019323
Epoch : 31 | Iteration : 2237 | Learning Rate : 0.001 | Total Loss : 5.6347 | Box Loss : 0.65237 | Object Loss : 4.9488 | Class Loss : 0.033539
Epoch : 31 | Iteration : 2238 | Learning Rate : 0.001 | Total Loss : 3.494 | Box Loss : 0.39661 | Object Loss : 3.091 | Class Loss : 0.0064586
Epoch : 31 | Iteration : 2239 | Learning Rate : 0.001 | Total Loss : 6.528 | Box Loss : 0.80701 | Object Loss : 5.7051 | Class Loss : 0.015901
Epoch : 31 | Iteration : 2240 | Learning Rate : 0.001 | Total Loss : 3.7831 | Box Loss : 0.57524 | Object Loss : 3.1944 | Class Loss : 0.013371
Epoch : 31 | Iteration : 2241 | Learning Rate : 0.001 | Total Loss : 3.0605 | Box Loss : 0.24651 | Object Loss : 2.798 | Class Loss : 0.015954
Epoch : 31 | Iteration : 2242 | Learning Rate : 0.001 | Total Loss : 3.4542 | Box Loss : 0.60999 | Object Loss : 2.8317 | Class Loss : 0.012512
Epoch : 31 | Iteration : 2243 | Learning Rate : 0.001 | Total Loss : 4.7392 | Box Loss : 0.46902 | Object Loss : 4.2575 | Class Loss : 0.012644
Epoch : 31 | Iteration : 2244 | Learning Rate : 0.001 | Total Loss : 4.3105 | Box Loss : 0.55775 | Object Loss : 3.7372 | Class Loss : 0.015583
Epoch : 31 | Iteration : 2245 | Learning Rate : 0.001 | Total Loss : 3.4049 | Box Loss : 0.35446 | Object Loss : 3.0419 | Class Loss : 0.008516
Epoch : 31 | Iteration : 2246 | Learning Rate : 0.001 | Total Loss : 3.0357 | Box Loss : 0.37653 | Object Loss : 2.65 | Class Loss : 0.009204
Epoch : 31 | Iteration : 2247 | Learning Rate : 0.001 | Total Loss : 2.8478 | Box Loss : 0.251 | Object Loss : 2.5837 | Class Loss : 0.013147
Epoch : 31 | Iteration : 2248 | Learning Rate : 0.001 | Total Loss : 4.5225 | Box Loss : 0.27489 | Object Loss : 4.2357 | Class Loss : 0.011862
Epoch : 31 | Iteration : 2249 | Learning Rate : 0.001 | Total Loss : 2.5637 | Box Loss : 0.36002 | Object Loss : 2.1881 | Class Loss : 0.015601
Epoch : 31 | Iteration : 2250 | Learning Rate : 0.001 | Total Loss : 3.0212 | Box Loss : 0.2662 | Object Loss : 2.7505 | Class Loss : 0.0045236
Epoch : 31 | Iteration : 2251 | Learning Rate : 0.001 | Total Loss : 4.2696 | Box Loss : 0.37987 | Object Loss : 3.8741 | Class Loss : 0.015593
Epoch : 31 | Iteration : 2252 | Learning Rate : 0.001 | Total Loss : 4.2922 | Box Loss : 0.34816 | Object Loss : 3.9373 | Class Loss : 0.0066543
Epoch : 31 | Iteration : 2253 | Learning Rate : 0.001 | Total Loss : 4.1198 | Box Loss : 0.59223 | Object Loss : 3.5156 | Class Loss : 0.011952
Epoch : 31 | Iteration : 2254 | Learning Rate : 0.001 | Total Loss : 4.623 | Box Loss : 0.50725 | Object Loss : 4.1022 | Class Loss : 0.013566
Epoch : 31 | Iteration : 2255 | Learning Rate : 0.001 | Total Loss : 2.5557 | Box Loss : 0.23232 | Object Loss : 2.3206 | Class Loss : 0.002777
Epoch : 31 | Iteration : 2256 | Learning Rate : 0.001 | Total Loss : 4.8335 | Box Loss : 0.3849 | Object Loss : 4.4365 | Class Loss : 0.012112
Epoch : 31 | Iteration : 2257 | Learning Rate : 0.001 | Total Loss : 4.811 | Box Loss : 0.41776 | Object Loss : 4.3785 | Class Loss : 0.014808
```

Figure 9: Training Stages sample outputs

4 TESTING AND RESULTS PHASE

The trained model has been subjected to two discrete testing procedures to evaluate the performance. During first case of operation, images are passed through the detector section and the outcomes are analyzed in the form of annotation boxes. During this process, the images are passed through the trained model for the identification of the objects that are of interest. After this stage, the objects detected are outlined using the bounding boxes to analyze the location within the images. In Figure 10, samples of such detections have been presented.

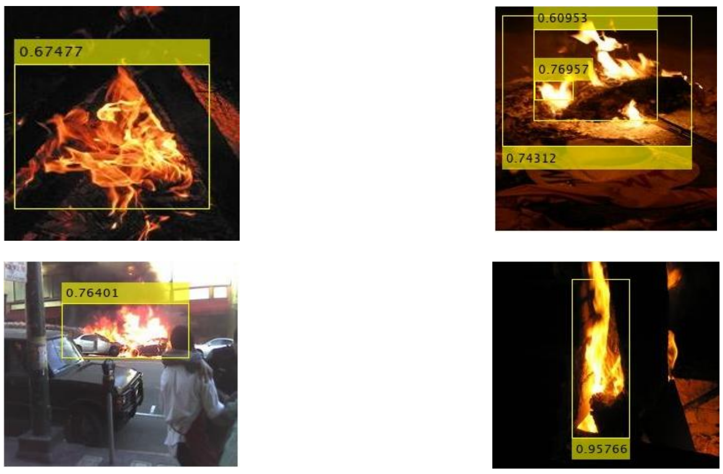


Figure 10: The Images fire detections

During second phase of operation, the detector was supplied by a video stream or via monitoring camera. In this case, the video is analyzed on a per-frame basis mimicking real-time detection. The annotated boxes were generated on each frame and represented in the form of a video as well. The model was able to capture the identified regions that contained the fire and the videos continued. A sample outcome recorded from the video stream has been depicted in Figure 11.

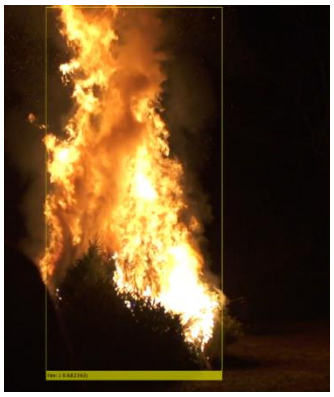


Figure 11: Fire Detection in Frames

In the experimentation setup, a camera having a frame rate of 18 frames per second has been used to capture the images across numerous resolutions. These including sizes of 540x300, 720x400 and 360x200 pixels. These images were supplied to the detector, and their outcomes were successfully analyzed.

5 CONCLUSION

The YOLOv3 algorithm proposed in this work was trained based on the parallel pooling approach for the detection of fire specifically targeting hospital settings of Kuwaiti hospitals. By integrating AI methodologies and object detection techniques based on YOLO architecture, the system offered good performance with improved response times. This resulted in the utilization of the model in real time and reduced false alarm rates. By adopting the parallel pooling strategies, the overall systems were optimized and the efficiency and scalability of the fire detecting mechanism improved. The approach involves the use of open-source data and the real-world image training of the image dataset. The YOLOv3 model depicted real-time fire detection capabilities by passing it through a live video stream and showcased its ability to rapidly capture the regions associated with fire in Kuwaiti hospitals.

The model thus offered a successful fusion of advanced object detection algorithms and parallel pooling techniques. The design was coupled to the existing challenges identified in the detection of the objects using manual interventions and an aim to attain overall precision in the fire detection processes. The finding holds significant implications in numerous fields including fire safety and disaster management. The developed fire detection system offered a reliable and efficient solution towards capturing any instance of fire at a much earlier stage of ignition.

In the future, the data from multiple sensors including the smoke and thermal cameras can be fed to the model for training and testing purposes. This will help improve the robustness of the model during instances where line of sight may not be available at all camera locations. By complementing the ability of the model through such sensor data, the system will provide further insights into fire incidents and improve detection capabilities. Future developments can further focus on the adaptive learning strategies and feedback methods that can allow the system to be improved gradually by adopting the proposed changes and updates. The analysis of the historical data and user feedback can help refine the detection algorithms and optimization of performance over time. Exploring the developments associated with edge computing within the hospitals can further enhance the scalability and efficiency of the fire detection systems. By the adoption of edge computing methods, the system will be able to handle the intensive processing tasks at the local levels and the reliance on the centralized servers will be reduced. Thus, a distributed computation can take place at nodes instead of transmitting raw video data to the servers increasing the detection time and latency.

REFERENCES

[1] Abdusalomov, A., Mukhiddinov, M., Djuraev, O., Khamdamov, U., & Whangbo, T. (2020). Automatic salient object extraction using locally adaptive thresholding to create tactile graphics. Applied Sciences, 10, 3350. https://doi.org/10.3390/app10103350

[2] Adel Sami. ""Al-Adan"...The Highest Government Authority in the Bed Occupancy Rate at 67.5%." *Kuwaiti Jarida*, 13 Nov. 2023, www.aljarida.com/article/44296. Accessed 23 Oct. 2024.

[3] Alanbaa. "KUNA : Al-Sayer: The capacity of Al-Adan Hospital reached 865 beds after the expansion - Health - 29/04/2010." *Kuna.net.kw*, 13 Nov. 2016, www.kuna.net.kw/ArticleDetails.aspx?id=2079589&language=en.

Accessed 26 Oct. 2024.

[4] Adhav, P., Naik, M., Tonge, S., Choudhari, M., & Pawar, P. (2022). Object detection based on convolutional neural networks. 4, 3922-3927.

[5] Ahmad, I., Xu, S., Khatoon, A., Tariq, U., Khan, I., Rizvi, S., & Ullah, A. (2022). Analytical study of deep learning-based preventive measures of COVID-19 using the RISTECB model. Scientific Programming, 2022, 1-17. https://doi.org/10.1155/2022/6142981

[6] Çakar, M., Yildiz, K., & Demir, Ö. (2021). Thumbnail selection using convolutional neural networks based on emotion detection. International Journal of Advances in Engineering and Pure Sciences, 33. https://doi.org/10.7240/jeps.900561

[7] Chauhan, R., Ghanshala, K., & Joshi, R. (2018). Convolutional neural network (CNN) for image detection and recognition. 278-282. https://doi.org/10.1109/ICSCCC.2018.8703316

[8] Elgendy, M. (2020). Deep Learning for Vision Systems. Simon and Schuster, New York, NY, USA. [Google Scholar]

[9] Ge, L., Dan, D., & Li, H. (2020). Accurate and robust monitoring method of full-bridge traffic load distribution based on YOLO-v3 machine vision. Structural Control and Health Monitoring, 27, e2636. [Google Scholar] [CrossRef]

[10] Gesa. "AL ADAN HOSPITAL - KUWAIT - GHESA Ingeniería Y Tecnología, S.A." *GHESA Ingeniería Y Tecnología, S.A.*, 5 June 2019, www.ghesa.es/en/portfolio_page/al-adan-hospital-kuwait-2/. Accessed 23 Oct. 2024.

[11] Guerrieri, M., & Parla, G. (2021). Deep learning and YOLOv3 systems for automatic traffic data measurement using the moving car observer technique. Infrastructures, 6, 134. https://doi.org/10.3390/infrastructures6090134

[12] Hu, M., Wei, Y., Li, M., Yao, H., Deng, W., Tong, M., & Liu, Q. (2022). Bimodal learning engagement recognition from classroom videos. Sensors, 22, 5932. https://doi.org/10.3390/s22165932

[13] Jin, Z., & Zheng, Y. (2020). Application of improved YOLO V3 algorithm in road target detection. Journal of Physics: Conference Series, 1654, 012060. [Google Scholar]

[14] Karne, M., Karne, R., Vaigandla, K., & Arunkumar, A. (2023). Convolutional neural networks for object detection and recognition. 3, 1-13. https://doi.org/10.55529/jaimlnn.32.1.13

[15] Li, B., Jiang, W., Gu, J., Liu, K., & Wu, Y. (2020). Research on convolutional neural networks in object detection. 820-827. https://doi.org/10.1109/ICPICS50287.2020.9202194

[16] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. (2014). Microsoft COCO: Common objects in context. 8693. https://doi.org/10.1007/978-3-319-10602-1_48

[17] Rakhimov, M. R., & Tolcha, Y. K. (2016). Parallel processing of ray tracing on GPU with dynamic pipelining. International Journal of Signal Processing Systems, 4(3), 209-213. https://doi.org/10.18178/ijsps.4.3.209-213

[18] Ahmed, M. (2023). Losses of fires in Kuwait. Youm7. Retrieved from https://t.ly/BrJMQ, last accessed June 18, 2024.

[19] Pal, S., Pramanik, A., Maiti, J., & Mitra, P. (2021). State of the art in multi-object detection and tracking using deep learning. Applied Intelligence, 51. https://doi.org/10.1007/s10489-021-02293-7

[20] Pan, Q., Guo, Y., & Wang, Z. (2019). Scene classification algorithm for visual robots based on Tiny YOLO v2. Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019, pp. 8544–8549. [Google Scholar]

[21] Pandey, A., Puri, M., & Varde, A. (2018). Object detection with neural models, deep learning, and common sense for smart mobility. 859-863. https://doi.org/10.1109/ICTAI.2018.00134

[22] Pinapatruni, R., Rao, L., & Lakshmi, P. (2020). CNN-based object detection system: A real-time application. 48-50. https://doi.org/10.26480/cic.01.2020.48.50

[23] Rakhimov, M., Elov, J., Khamdamov, U., Aminov, S., & Javliev, S. (2021). Parallel implementation of real-time object detection using OpenMP. 1-4. https://doi.org/10.1109/ICISCT52966.2021.9670146

[24] Ren, P., Wang, L., Fang, W., Song, S., & Djahel, S. (2020). A novel squeeze YOLO-based real-time people counting approach. International Journal of Bio-Inspired Computation, 16, 94. https://doi.org/10.1504/IJBIC.2020.109674

[25] Tanios, C. (2024). Fire in Kuwait building kills 49 foreign workers. Reuters. Retrieved from https://www.reuters.com/world/middle-east/least-35-people-killed-fire-southern-kuwait-state-media-2024-06-12/, last accessed June 18, 2024.

[26] Wang, L., Yang, S., & Yang, S. (2019). Automatic thyroid nodule recognition and diagnosis in ultrasound imaging with YOLOv2 neural network. World Journal of Surgical Oncology, 17, 12. [Google Scholar] [CrossRef]

[27] Woo, J., Baek, J.-H., Jo, S.-H., Kim, S. Y., & Jeong, J.-H. (2022). Study on YOLOv4's object detection performance for autonomous driving of trams. Sensors, 22, 9026. https://doi.org/10.3390/s22229026

[28] Xu, J., Zhao, J., Wang, W., & Liu, M. (2013). Predicting tubular truss temperature under fire using artificial neural networks. Fire Safety Journal, 56, 74-80.

[29] Yolo v3 of Yolo Series. Available online: https://blog.csdn.net/leviopku/article/details/82660381 (accessed on 1 August 2021).

[30] Zhao, L., & Li, S. (2020). Object detection algorithm based on improved YOLOv3. Electronics, 9(3), 537. https://doi.org/10.3390/electronics9030537