Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

Accelerating Fire Detection for Improving Maintenance Opeartions in Hospitals

Dalal Al Enzi
PhD researcher- Mansoura University
Maintenance Engineer, Ministry of Health
Adan Hospital, Kuwait

Dr./ Ibraheem Mutawee
Prof of Project Management

***Abstract*— The article delves into the enhancement of fire detection systems in Kuwaiti general hospitals through the integration of advanced object detection techniques. Particularly, the study focuses on the implementation of YOLOv3, an AI-driven object detection algorithm, supported by parallel pooling for optimized speed and accuracy. The research builds upon valuable insights obtained from a survey on existing fire alarming systems in hospitals, outlining the deficiencies in response time, accuracy, and maintenance procedures. Through the utilization of deep learning and image processing, the optimization of fire detection to reduce response times significantly and minimize false alarms has been achieved. By leveraging the parallel architecture of computing systems and the efficient processing capabilities of the YOLOv3 algorithm, the study presents a systematic approach to addressing critical challenges in fire detection within hospital environments.**

***Keywords*— Fire Detection, YOLOv3, Parallel Computing, Object Detection, Image Processing, AI Technologies, Hospital Safety.**

I. INTRODUCTION

The potential of fire alarming using artificial intelligence based on neural networks is revolutionary. This technology can be used to accurately detect and respond to fires, allow for more precise predictions about the behaviour of fire, and ultimately save lives and property. This essay will explore the opportunities provided by fire alarming using artificial intelligence based on neural networks and explain how it can be implemented to effectively identify and respond to fires. (Jin,2020)

In recent years, Artificial Intelligence (AI) based neural networks have become increasingly popular for fire alarm accuracy. This is due to their ability to effectively detect and recognize objects and patterns in large datasets. The use of AI technology in fire alarm systems has enabled these systems to become more accurate and reliable. A Khan et al. (2022) conducted a study to evaluate the impact of AI-based neural networks on the accuracy of fire alarms. The study involved a survey of fire alarm owners, an analysis of fire alarms' false positive and false negative rates, and a comparison of AI-based neural networks to traditional fire alarm systems. The results of the study showed that AI-based neural networks had a significantly higher accuracy than traditional fire alarm systems. The false positive rate was lower and the false negative rate was higher for neural networks than for traditional systems. Furthermore, the study found that AI-based neural networks were more reliable in detecting actual fires and were able to detect fires more quickly. These results demonstrate that AI-based neural networks can provide an effective and reliable solution for improving fire alarm accuracy. (Ge, L,2020)

II. OBJECT DETECTION TECHNIQUES

Object Detection is one of the most famous and vastly researched topics in the field of Computer Vision and Machine Learning. It has attracted many researchers working in different areas such as computer vision, robotics, medical imaging, mechanical engineering, and telecommunications. Object Detection is a methodology in Machine Learning focused on localizing and recognizing distinct objects in images and videos. The methodologies using those features in determining objects, identifying, and labelling them has the following tree of divisions as illustrated in fig 1.

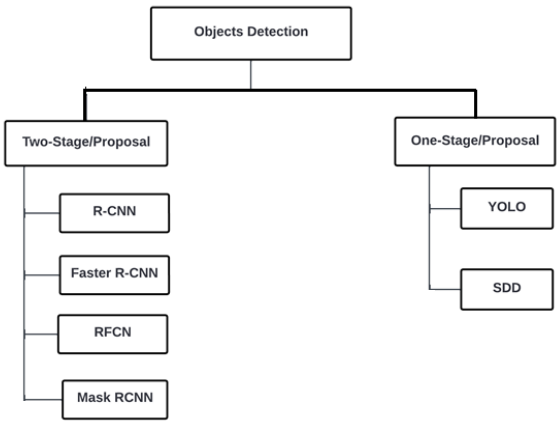

Fig 1: Object detection techniques tree

III. SELECTING TECHNIQUES

The selection of the utilized algorithms in implementing the system will be based on selecting one YOLO version and then selecting processing technique.

A. Selecting YOLOv3 version

Selecting YOLOv3 algorithm for fire detection is based on its unique characteristics and functionalities that offer exceptional benefits for implementing an effective fire detection system. The use of YOLOv3 is particularly advantageous due to its real-time performance capabilities, allowing for swift responses to potential fire incidents, which is crucial for timely intervention. YOLOv3's precise object localization features, facilitated by anchor boxes and multi-scale detections, enable accurate identification of specific objects like flames and

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

smoke, key indicators of fire outbreaks. This precision further enhances the system's efficiency in recognizing and analysing fire-related objects within images promptly. (Jin,2020)

The real-time performance capabilities of the YOLOv3 algorithm refer to its ability to process and analyse visual data promptly and efficiently. In the context of fire detection systems, real-time performance is vital as it enables immediate responses to potential fire incidents. YOLOv3 can rapidly identify objects of interest, such as flames or smoke, in images or video feeds, ensuring that fire-related hazards are detected and addressed without delay. This quick processing speed is crucial in emergency situations where timely detection of fires can mitigate risks and facilitate rapid intervention to minimize damage or prevent further escalation of the fire. By operating in real-time, YOLOv3 enhances the responsiveness and effectiveness of fire detection systems, providing actionable insights and alerts as soon as potential fire hazards are detected. (Ge, L,2020)

The algorithm's robustness to object categories, thanks to its integration of an extensive set of classes from datasets like COCO, makes it adaptable to diverse visual characteristics associated with fire detection. YOLOv3's efficient Darknet-53 architecture effectively extracts high-level features and intricate spatial details. By handling localization and classification simultaneously in a single pass, YOLOv3 ensures fast and efficient fire detection.

Its openness as an open-source algorithm promotes flexibility, customization, and integration into fire detection systems. Through leveraging YOLOv3's real-time performance, accurate object localization, robustness to object categories, efficient architecture, and flexibility, researchers can develop effective fire detection systems that analyse images in real-time, aiding in the early detection and prevention of potential fire incidents. (J. Redmon, 2016)

YOLOv3 model originally has been developed for making various objects detection and will create a suitable fire detections capabilities as will be illustrated in the following sections due to the highly complex modelling of these systems, some of the models have been taken as baselines to make them fit the fire detection objective. To do so, one of the YOLOv3 models available in MATLAB has been treated as a baseline and has been tailored to fit out the intended application. In this research, YOLOv3 is adopted because the other YOLO family is a series of end-to-end deep learning models planned and designed for fast real-time object detection and the quick fires detection is essential for the proposed technique. (Yanjia,2019)

In the MS COCO dataset, the comparison of Average Precision (AP) and Frames Per Second (FPS) values for different object detection algorithms like YOLOv3, YOLOv4, and SSD can vary based on various factors. The shown relations in fig 1 bellow illustrates that relation for some detection algorithms:

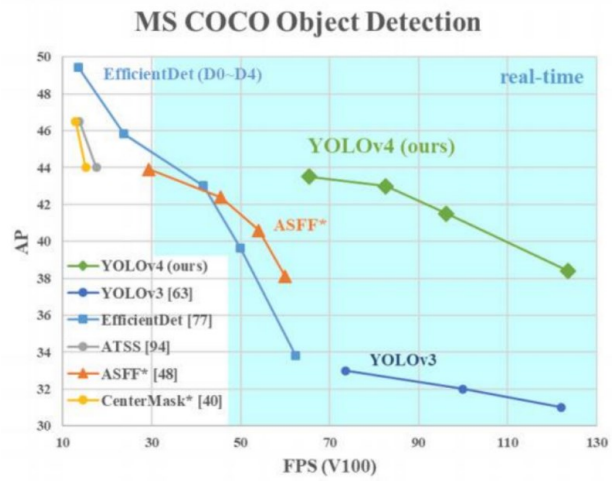

Fig 1: Comparison from some YOLO versions (Rakhimov,2021)

The higher value between AP and FPS depends on the specific use case and requirements of the detection system

- Higher AP: A higher Average Precision indicates better accuracy in object detection, which is crucial for precision-centric applications where detection correctness is a top priority. Algorithms with higher AP values offer superior object localization and class-specific detection accuracy, ensuring reliable results even in challenging scenarios.
- Higher FPS: On the other hand, a higher Frames Per Second (FPS) reflects superior processing speed, enabling faster detection and real-time performance for applications where speed and efficiency are critical. Algorithms with higher FPS values can process a larger number of frames in a given time, making them ideal for scenarios demanding rapid action and quick object identification. (Rakhimov,2021)

The choice between a higher AP and FPS ultimately depends on the specific requirements of the application. A balance between accuracy (AP) and speed (FPS) is often sought to achieve optimal performance in object detection systems. In some cases, trading-off between these metrics might be necessary based on the specific use case to achieve the desired balance between accuracy and efficiency and according to our object A system with higher FPS can provide real-time monitoring and quick alerts in case of potential fire incidents. The faster processing speed allows for immediate responses, which can be crucial in detecting fires early.

B. Selecting Pooling Techniques

pooling layers, refers to a technique used to down-sample the feature maps generated by convolutional layers. The purpose of pooling is to reduce the spatial dimensions of the input while retaining important information.

Pooling is typically performed on feature maps derived from convolutional layers, where each feature map consists of a grid of values representing different features at different spatial positions. The pooling operation aggregates the values within local regions of the feature map to produce a lower-resolution output.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

There are different pooling operations commonly used, such as max pooling, average pooling, and LP pooling. Max pooling as example divides the input feature map into non-overlapping rectangular regions (usually 2x2 or 3x3) and takes the maximum value within each region as the output for that region. This down-samples the feature map by reducing its size while retaining the strongest activation values. (Mekhriddin,2016)

The pooling operation involves sliding the pooling window (e.g., 3x3) across the feature map with a fixed stride (e.g., 3), and at each position, taking the maximum value within the window as the output. The pooling window moves across the feature map, reducing its size by a factor determined by the stride. One main disadvantage of the cascaded deep-learning network of YOLOv3 is the increased inference time. The network involves multiple smaller networks that process the input image sequentially, resulting in slower detection times compared to other object detection algorithms. (Absalom, 2020)

To avoid this disadvantage, parallel pooling can be used. Parallel pooling involves splitting the input image into multiple smaller regions and processing them simultaneously using separate threads or parallel hardware accelerators. By parallelizing the processing, the overall inference time can be reduced significantly.

Another disadvantage of the cascaded network is the potential loss of information between different stages of the network. Each stage focuses on detecting different types of objects at different scales, but this can result in the loss of contextual information about the objects. For example, in the first stage, small objects may be filtered out before the second stage, which may affect the accuracy of detection. To mitigate this disadvantage, parallel pooling can be used with intermediate feature fusion. Instead of completely separating the stages, features from different stages can be combined and fused, allowing for better retention of context and more accurate object detection. (Yanjia,2019)

C. Illustrating Selected Technique

Parallel pooling in YOLOv3 is the proposed improvement for the existing detection techniques. It works as spatial pyramid pooling as its uses of multiple pooling layers of different sizes in parallel to capture features at various scales.

In YOLOv3, the feature extraction backbone (Darknet-53) is a deep convolutional neural network that gradually reduces the spatial dimensions of the input image while increasing the number of channels. This process is achieved through a series of convolutional and pooling layers so there is a single pooling layer is used to reduce the spatial dimensions of the feature map, resulting in a loss of fine-grained spatial information. (Elgendy, M.2020)

The parallel pooling consists of several pooling layers with different kernel sizes applied to the feature map. Each pooling layer captures features at a particular scale. The output feature maps of these pooling layers are then concatenated along the channel dimension. By doing this, the model can capture features at different scales and preserve fine-grained spatial information. This is crucial for detecting objects of various sizes in an image.

The concatenated feature map from the parallel pooling is then further processed by subsequent convolutional layers and finally passed through the detection layers to predict the bounding boxes, class probabilities, and confidences for object detection.

By incorporating parallel pooling into YOLOv3, we can achieve the following improvements:

1. Faster Inference Time: Parallel pooling allows for concurrent processing of different regions, resulting in a significant reduction in inference time. This is particularly beneficial when dealing with large input images or datasets.
2. Improved Utilization of Resources: Parallel pooling effectively utilizes parallel processing capabilities, such as multi-threading or parallel hardware accelerators like GPUs. This maximizes the computational power available, making the best use of hardware resources.
3. Enhanced Accuracy: By dividing the image into smaller regions and processing them independently, parallel pooling can help capture more localized and contextually relevant information. This can improve the accuracy of object detection, especially when dealing with objects of varying sizes and aspect ratios.
4. Scalability: Parallel pooling allows for efficient scalability, enabling the detection of objects across a wide range of image sizes and resolutions. (Jin,2020)

It is worth noting that the implementation of parallel pooling in YOLOv3 may require modifications in the network architecture and training pipeline to support distributed processing. Additionally, an efficient parallelization strategy considering hardware and memory constraints should be employed to ensure optimal performance which will be illustrated after implementing and simulating the system in the end of this chapter.

D. Deep Learning Integration

Deep learning plays a significant role in proposed fire detection systems by utilizing advanced algorithms capable of learning intricate patterns and features from image data. The models implemented in deep learning are formed using a large number of tagged data and neural network architectures that contain various layers. In the context of proposed techniques, the implementation process involves several key steps: (Ge, L, 2021)

1. Extracting Features Using Convolutional Neural Networks (CNNs):

CNNs are integral in deep learning for image processing tasks. Layers of the CNN extract features like edges, textures, and shapes from the input images. This process forms the foundation for identifying objects.

2. Implementing the YOLOv3 Architecture:

YOLOv3 consists of a series of convolutional layers, max-pooling layers, and detection layers. Each layer serves a specific function in feature extraction, down-sampling, and bounding box prediction. Implementing this architecture in the system enables accurate object detection.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

3. Predicting Object Classes and Bounding Boxes:

The YOLOv3 model predicts the class probabilities of objects present in an image and estimates bounding boxes around these objects. By assigning class labels and drawing bounding boxes, the system can identify and localize instances of interest, like fires and smoke.

4. Training and Fine-Tuning the Model:

The deep learning model is trained on a labeled dataset containing images of fires and non-fire scenarios. Through backpropagation and optimization techniques like stochastic gradient descent, the model learns to classify objects accurately. Additionally, fine-tuning the model on domain-specific data enhances its ability to detect fire-related features effectively. (Elgendy, M.2020)

5. Inference and Real-Time Object Detection:

During inference, the trained model processes new images in real-time to detect fire-related objects based on learned features. By feeding input images through the model, the system can detect fires and issue timely alerts when potential hazards are identified. It does not require a region proposal network (RPN) and directly performs regression to detect targets in the image as it includes 53 convolutional layers and 23 residual layers as shown in below which makes it as a significant advancement in real-time object detection, especially in the detection of critical objects. Consequently, YOLOv3 can be used for detection system. (Redmon, 2018)

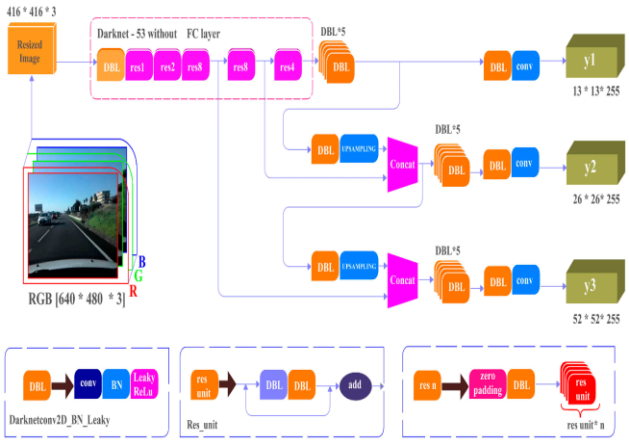

Fig 2: System's deep learning integration (Guerrieri,2021)

1 × 1, 3 × 3/2 and 3 × 3 convolution kernels of three sizes are applied in the convolutional layers to sequentially extract image features, ensuring the model has remarkable classification and detection performances. The remaining layers guarantee the convergence of the detection model.

In order to detect several areas of the same object at the same time, parallel pooling fuses three feature maps of different scales (52 × 52, 26 × 26 and 13 × 13) by three-time down sampling.

IV. SYSTEM IMPLEMENTATION

The YOLOv3 architecture developed as part of this study has been developed on top of SqueezeNet architecture. This architecture is a lightweight CNN architecture that can effectively handle the computational requirements and the size of the model. The modified YOLOv3 architecture, apart from being driven by SqueezeNet, has further been augmented by two detection heads. These have been appended to the architecture of the model.

The initial detection head captures the larger objects contained in the fire dataset images. The second head has been developed to capture small objects. The second head is twice as big as the first head. Such a hierarchical model allows capturing the features in the object space that allow effective detection of the objects having different sizes and input images. To further enable the localization of the object and predictions, the model makes use of anchor boxes. Such boxes are predetermined bounding boxes, which allow the network to analyse the specific classes of the objects contained in the training part of the data. Such boxes help train the model and analyse the characteristics of the datasets.

Predicting the bounding box's width and height might make sense, but that may lead to unstable gradients during training. Instead, the proposed system for object detectors predicts log-space transforms or offsets to pre-defined default bounding boxes anchors.

Then, these transforms are applied to the anchor boxes to obtain the prediction. The proposed system has three anchors, which result in the prediction of three bounding boxes per cell which are bounding box priors that were calculated on the COCO dataset using k-means clustering. We are going to predict the width and height of the box as offsets from cluster centroids. The box's center coordinates relative to the location of the filter application are predicted using a sigmoid function. The following formula describes how the network output transformed to obtain bounding box predictions:

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

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

The width and height of the predicted bounding box are calculated thus

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

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

Here b_x, b_y, b_w, b_h are the x, y center coordinates, width, and height of our prediction. t_x, t_y, t_w, t_h (xywh) is what the network outputs. c_x and c_y are the top-left coordinates of the grid. p_w and p_h are anchors dimensions for the box which can be illustrated bellow in Fig. where the dimensions of the bounding box predictions are obtained through a log-space transformation applied to the network output. This transformation helps in handling a wide range of object sizes effectively. The log-space transformation involves taking the logarithm of the network output. After the log-space transformation, the result is multiplied by an anchor which are pre-defined boxes or priors of different shapes and sizes as previously explained that act as reference points for predicting bounding box dimensions. Each anchor is associated with specific scale and aspect ratio characteristics. (Guerrieri,2021)

By multiplying the log-space transformed output by the appropriate anchor, the final dimensions of the bounding box predictions are obtained. This step helps adjust the predictions based on the anchor properties and results in more accurate bounding box sizes and shapes.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

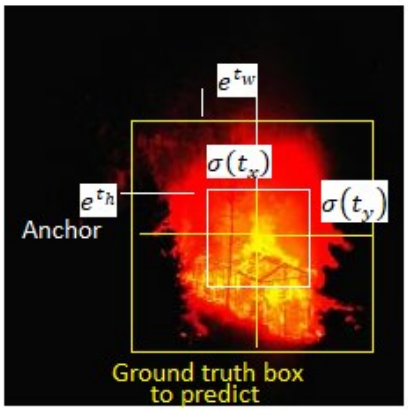

Fig 3: The dimensions of the bounding box

Predictions, b_w, and b_h, are normalized by the height and width of the image. (Training labels are chosen this way). So, if the predictions b_x and b_y for the box containing the dog are (0.3, 0.8), then the actual width and height on the 13 x 13 feature map are (13∗0.3, 13∗0.8). During training, the ground truth labels are also chosen in a normalized manner to match the network's predictions.

For example, if the predictions for the bounding box containing a dog are (0.3, 0.8), these values are normalized by the height and width of the image. If the feature map size is 13x13, then the actual width and height of the bounding box on this feature map can be calculated by multiplying the predictions by the feature map size. In this case, the actual width would be 13 * 0.3 = 3.9 and the actual height would be 13 * 0.8 = 10.4.

The parameters of ground truth box can be calculated according to the following equations which evaluates $g_x, g_y, g_h and\ g_w$corresponding to the real values of predicted parameters $b_x, b_y, t_h and\ t_w$ respectively then the truth quantities of $\hat{b}_x, \hat{b}_y, \hat{t}_h and\ \hat{t}_w$will be calculated according to the following set of equations:

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

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

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

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

Then to calculate square error of coordinate prediction as one part of loss function E_2 as shown in the following equation:

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

The layered architecture of the model comprises many layers. These include Convolutional layers for the extraction of features and rectified Linear Unit (ReLU) activation functions to capture the non-linearity in the data. Resizing layers have been used to make adjustments to the spatial dimensions of the feature maps. A depth concatenation layer has been added to combine features extracted at different scales and stages of the network. This overall architecture allows for capturing the contextual information across the images.

The overall architecture adopted allows an optimized detection of fire. By leveraging the strength of SqueezeNet, the extraction capabilities of the modified YOLOv3 architecture have been made highly efficient. The architecture of the YOLOv3 adopted in this work has been depicted in fig 4 below:

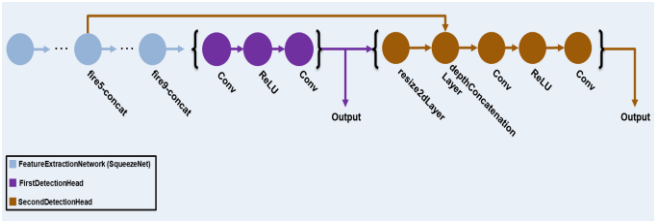

Fig 4: YOLOv3 Architecture

The YOLOv3 architecture has been modified to incorporate a sense of parallel pooling for improved performance and speed. Parallel pooling helps accelerate the training process by making a distribution of computations across multiple workers. To incorporate such functionality, MATLAB's parallel computing toolbox has been utilized. Initially, the model starts by making a check on the availability of parallel pools for processing. Once the availability is flagged high, the parfor loops have been employed to distribute the iterations across multiple workers creating an overall scenario of concurrent executions. The parallel pooling-based customization has helped improve the speed of the process by the following relation:

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

Where T_{serial} is the time for non-parallel executions, and $T_{parallel}$ is the execution time for the parallel version. Further by adopting this approach, the scalability of the model has been improved by effectively utilizing the additional available resources. The model measures the degree of execution time scales with an increased number of processors. The steps involved in completing the model to work have been presented in Fig.

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

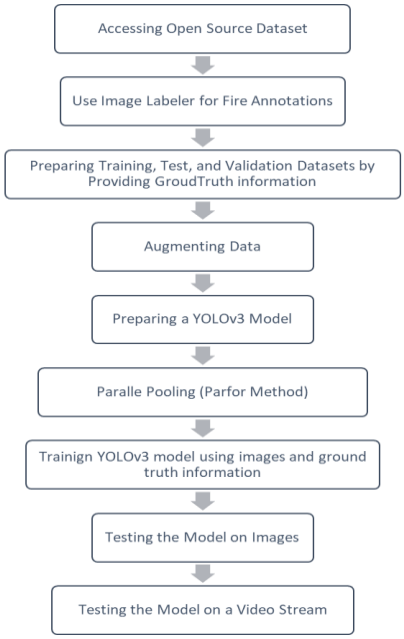

Fig 5: YOLOv3 System Level Modelling

A. Accessing Open-Source Datasets

The dataset used for this study has been taken from image datastore which contains a large repository of scenarios and images. FiSmo-Images have been downloaded which contains different instances of fire by employing various scenarios. Specifically, Flickr-Fire data has been used that contains 984 images of various sizes and dimensions.

The images have been resized to 227x227x3 to retain a standard size making them fit for the model and to have reduced samples to avoid memory resources to runout.

B. Image Labeller for Annotations

To make the YOLO model learn object detection features, it is imperative to make use of an image labeller to generate ground truth information against each image. There are numerous ways in which this can be achieved. For example, a commonly used way is to employ colour thresholding, where the function takes close to the fire regions. However, those are only estimates and far from precise annotations. Therefore, in the present case, the images have been equipped with bounding boxes in the Image Labeller manually. The Image Labellers have been introduced with a new label class called 'Fire', and the rectangular boxes have been placed manually at the fire regions. The complete set of information has been stored in a .mat file which contains the path to the images, and the coordination of one or multiple boxes where instances of fire are existing. The activity has been performed on the complete set of 984 images to be used as part of training and testing. The layout of the imageLabeler has been depicted as below for this particular implementation:

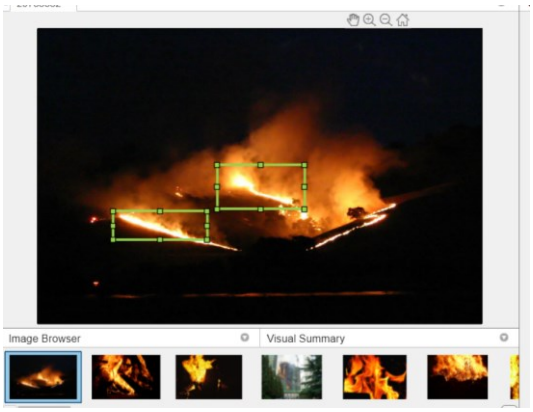

Fig 6: Image Labeler Layout

C. Preparing Data (Pre-processing and Split)

In step, the first part is to ensure that the ground truth information provided to the model is correct. Since the missing annotations in the Fire label can lead to errors later. To ensure this, a check has been introduced that ensures that no image is having an empty label. If this holds for any image, it has been discarded. The second part is to divide images into training and test datasets. The training dataset has been used for fitting the model, and the test dataset has been used for testing the model. To do so, 60 percent of the dataset has been used for training. These data sets are later subjected to data augmentation as the next step to improve the accuracy of the model.

D. Data Augmentation

In this step, the data has been augmented to introduce different colours and angles to keep the model tolerant toward captured angles. For instance, in real-world instances, an image may be recorded with a slight tilt and shall be detectable by the model. A data augmentation function has been employed that transforms images by application of random flips at horizontal levels, and to make scaling in the images by introducing overlap. Additionally, various colour variations have been introduced in the images, by making a variation in the contrast, hue, saturation, and brightness. An example of the data augmentation performed on the sample images has been presented below:

Fig 7: Images Data Augmentation Example

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

E. Preparing YOLOv3 Model

This step is to prepare a YOLOv3 model and pass on the data prepared above for the modeling. In this case, the first part is to create a YOLOv3 object detector by calling MATLAB's inbuilt function. Preparing the detector involves passing various arguments including the base network, class name, detector boxes, and the size of images that the model shall expect. The arguments presented in this case include the following:

Table -1: YOLOv3 Object Detector

Parameters	Values
basement	Squeezenet
className	'Fire'
anchor boxes	Taken from Step 2
Image Size	227x227x3

The detector has later been subject to the pre-processed data as returned from the data augmentation for the training.
The next part after creating a detector is to determine the training parameters. These parameters can be selected based on the hit-and-trial method or prior experience in similar datasets. These parameters selected in the present case have been depicted below:

Table -2: YOLOv3 Training Parameters

Parameters	Values
Epochs	80
Batch Size	8
Learning Rate	1e-3
L2 Regularization	5e-4
Penalty Threshold	0.5

F. Training the Model

The training of the model involves the utilization of parallel pooling in MATLAB to make quick processing. The loss functions are used to measure the discrepancy between the predicted bounding box outputs and the ground truth labels during training. The loss functions help in optimizing the model parameters by penalizing the differences between predictions and ground truth, encouraging the model to make more accurate predictions. There are several loss functions used to train the model. The primary loss function is the sum of the following components:

1. Localization Loss (bbox_loss): This loss measures the difference between the predicted bounding box coordinates (b_x, b_y, b_w, b_h) and the ground truth bounding box coordinates. The Mean Squared Error (MSE) or a smooth L1 loss is used for this purpose. To calculate this function, we calculate the difference between the predicted bounding box coordinates (b_x, b_y, b_w, b_h) and the ground truth bounding box coordinates. Applying Loss to measure the discrepancy between the predicted and ground truth values and finally sum up the losses for each bounding box prediction. The equation of MSE is:

$$bbox_{loss} = \frac{1}{n}\sum_{i=1}^{n}((b_{xi} - t_{xi})^2 + (b_{yi} - t_{yi})^2 + (\sqrt{b_{wi}} - \sqrt{t_{wi}})^2 + (\sqrt{b_{hi}} - \sqrt{t_{hi}})^2) \quad (11)$$

2. Confidence Loss (obj_loss): This loss measures the difference between the predicted objectness score (indicating the presence or absence of an object) and the ground truth objectness score. Binary cross-entropy loss is commonly used for this component.

$$Obj_{loss} = -\frac{1}{m}\sum_{i=1}^{m}(y_i * log(\hat{y}_i) + (1 - y_i) * log(1 - \hat{y}_i)) \quad (12)$$

3. Class Loss (class_loss): This loss measures the difference between the predicted class probabilities and the ground truth class labels. It is typically calculated using categorical cross-entropy loss.

$$L = -\frac{1}{m}\sum_{i=1}^{m} y_i.\log(\hat{y}_i) \quad (13)$$

The overall loss is the sum of these individual loss components, weighted by certain coefficients to balance their contributions. The specific values for these coefficients can vary depending on the implementation and specific requirements of the task.
During training, the goal is to minimize the overall loss value, indicating that the model is making accurate predictions. The optimization process adjusts the model's parameters through backpropagation and gradient descent, aiming to reduce the loss and improve the model's performance. The pools available are determined as the first step, followed by performing multiple operations as follows:

a. Determining the Gradient, State, and Loss Information. This is determined by using a model gradients function, and passing on the dataset and the YOLOv3 detector as the parameters

b. The gradients determined are subjected to the L2 Regularization as established above

c. The proposed system learning parameters are updated, and the corresponding loss and information are displayed on the MATLAB workspace.

A sample of the training stage depicting the corresponding outcomes has been presented in fig 8 bellow.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

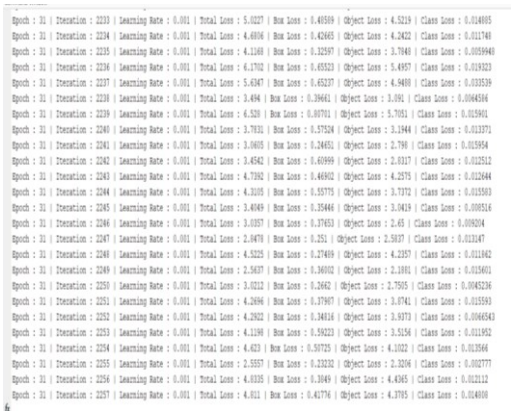

Fig 8: Samples from the Training Stage

V. RESULTS AND TESTING PHASE

Two types of tests have been conducted on the trained model. In the first case, the images are passed on to the detector and outcomes are analysed in the form of annotated boxes. These images have been depicted below:

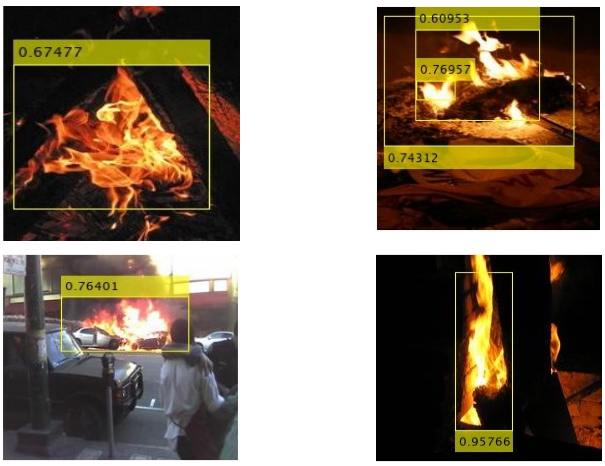

Fig 9: Fire Detections on the Images

In the second phase of testing, a video stream of data has been passed through and the detector is made to analyse frames and make annotated boxes in real time. The model successfully detected the fire region as the video streams is driven. The outcomes have been provided below:

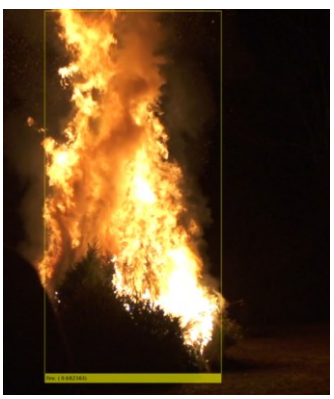

Fig 10: Fire Detection in Frames.

In the experimental setup, a camera capturing 18 frames per second provides images with varying resolutions of 720x400, 540x300, and 360x200 pixels. These images are fed into the input layer of a neural network for processing. An important focus of the study is the comparison of computational tasks using parallel computing architectures. Leveraging parallel computational methods is key to enhancing the efficiency of computing systems, as it enables concurrent processing of tasks for accelerated performance.

In image processing, a common approach involves sequentially applying a series of standard processing operations to the input data. While multi-core processors demonstrate parallel processing by design, their performance excels in streaming applications. To exploit parallelization efficiently, the activation function and image filtering components of the YOLOv3 parallel operation algorithm were optimized for parallel processing. This optimization involved utilizing both serial and parallel processing techniques on a computer equipped with a quad-core processor, enabling tasks to be executed simultaneously for faster and more efficient image processing results.

VI. CONCLUSION

The successful implementation of the YOLOv3 algorithm, augmented by parallel pooling for fire detection in Kuwaiti hospitals, represents a significant advancement in enhancing safety measures. By leveraging cutting-edge AI technologies and integrating deep learning methodologies, the system has been able to drastically reduce response times, enhance accuracy, and minimize false alarms in fire detection processes. The adoption of parallel computing techniques has not only optimized the speed and performance of the system but has also improved the scalability and efficiency of fire detection mechanisms in hospital settings. Through a meticulous process of data collection, model preparation, and training with real-world image datasets, the YOLOv3-based system demonstrates a remarkable capability to analyze and detect fire incidents in real-time, fostering quick and effective responses to potential hazards.

The successful combination of advanced object detection algorithms, parallel pooling, and innovative AI technologies underscores the critical role that modern computational approaches play in ensuring the safety and security of hospital environments. By addressing key challenges identified through surveys and data analytics, the implemented system offers a robust solution that automates fire alarming processes, minimizes manual intervention, and enhances the overall precision and reliability of fire detection mechanisms. Moving forward, further refinements and optimizations based on feedback and real-world testing will continue to elevate the capabilities of the system, contributing to the ongoing evolution of fire safety measures in Kuwaiti hospitals.

REFERENCES

[1] Chauhan, Rahul & Ghanshala, Kamal & Joshi, R.. (2018). Convolutional Neural Network (CNN) for Image Detection and Recognition. 278-282. 10.1109/ICSCCC.2018.8703316.+P24N2:W24M2:W24L2:W24K2:W24J2:W24I2:W24H2:W25G2:W25F2:W25E2:W25H2:W25K2:W25M2:W24O2:W24P2:W24

[2] Abdusalomov, Akmalbek & Mukhiddinov, Mukhriddin & Djuraev, Oybek & Khamdamov, Utkir & Whangbo, Taeg. (2020). Automatic Salient Object Extraction Based on Locally Adaptive Thresholding to Generate Tactile Graphics. Applied Sciences. 10. 3350. 10.3390/app10103350.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

[3] Adhav, Prashant & Naik, Madhura & Tonge, Shravani & Choudhari, Manali & Pawar, Prajakta. (2022). OBJECT DETECTION BASED ON CONVOLUTIONAL NEURAL NETWORK. 4. 3922-3927. Mekhriddin Rakhimov Fazliddinovich and Yalew Kidane Tolcha, "Parallel Processing of Ray Tracing on GPU with Dynamic Pipelining," International Journal of Signal Processing Systems, Vol. 4, No. 3, pp. 209-213, June 2016. doi: 10.18178/ijsps.4.3.209-213+P2:W24

[4] Elgendy, M. Deep Learning for Vision Systems; Simon and Schuster: New York, NY, USA, 2020. [Google Scholar]

[5] Elgendy, M. Deep Learning for Vision Systems; Simon and Schuster: New York, NY, USA, 2020. [Google Scholar]

[6] Ge, L.; Dan, D.; Li, H. An accurate and robust monitoring method of full-bridge traffic load distribution based on YOLO-v3 machine vision. Struct. Control. Health Monit. 2020, 27, e2636. [Google Scholar] [CrossRef]

[7] Ge, L.; Dan, D.; Li, H. An accurate and robust monitoring method of full-bridge traffic load distribution based on YOLO-v3 machine vision. Struct. Control. Health Monit. 2020, 27, e2636. [Google Scholar] [CrossRef]

[8] Guerrieri, Marco & Parla, Giuseppe. (2021). Deep Learning and YOLOv3 Systems for Automatic Traffic Data Measurement by Moving Car Observer Technique. Infrastructures. 6. 134. 10.3390/infrastructures6090134.

[9] Guerrieri, Marco & Parla, Giuseppe. (2021). Deep Learning and YOLOv3 Systems for Automatic Traffic Data Measurement by Moving Car Observer Technique. Infrastructures. 6. 134. 10.3390/infrastructures6090134.

[10] Jin, Z.; Zheng, Y. Research on application of improved YOLO V3 algorithm in road target detection. J. Phys. Conf. Ser. 2020, 1654, 012060. [Google Scholar]

[11] Jin, Z.; Zheng, Y. Research on application of improved YOLO V3 algorithm in road target detection. J. Phys. Conf. Ser. 2020, 1654, 012060. [Google Scholar]

[12] Karne, Ms & Karne, Radhakrishna & Vaigandla, Karthik & Arunkumar, A. (2023). Convolutional Neural Networks for Object Detection and Recognition. 3. 1-13. 10.55529/jaimlnn.32.1.13.

[13] Li, Bingzhen & Jiang, Wenzhi & Gu, Jiaojiao & Liu, Ke & Wu, Yangyong. (2020). Research on Convolutional Neural Network in the Field of Object Detection. 820-827. 10.1109/ICPICS50287.2020.9202194.

[14] Mekhriddin Rakhimov Fazliddinovich and Yalew Kidane Tolcha, "Parallel Processing of Ray Tracing on GPU with Dynamic Pipelining," International Journal of Signal Processing Systems, Vol. 4, No. 3, pp. 209-213, June 2016. doi: 10.18178/ijsps.4.3.209-213

[15] Pan, Q.; Guo, Y.; Wang, Z. A scene classification algorithm of visual robot based on Tiny Yolo v2. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 8544–8549. [Google Scholar]

[16] Pan, Q.; Guo, Y.; Wang, Z. A scene classification algorithm of visual robot based on Tiny Yolo v2. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 8544–8549. [Google Scholar]

[17] Pinapatruni, Rohini & Rao, L & Lakshmi, P. (2020). CNN BASED OBJECT DETECTION SYSTEM: A REAL TIME APPLICATION. 48-50. 10.26480/cic.01.2020.48.50.

[18] Rakhimov, Mekhriddin & Elov, Jamshid & Khamdamov, Utkir & Aminov, Shavkatjon & Javliev, Shakhzod. (2021). Parallel Implementation of Real-Time Object Detection using OpenMP. 1-4. 10.1109/ICISCT52966.2021.9670146.

[19] Wang, L.; Yang, S.; Yang, S. Automatic thyroid nodule recognition and diagnosis in ultrasound imaging with the YOLOv2 neural network. World J. Surg. Oncol. 2019, 17, 12. [Google Scholar] [CrossRef] [Green Version]

[20] Wang, L.; Yang, S.; Yang, S. Automatic thyroid nodule recognition and diagnosis in ultrasound imaging with the YOLOv2 neural network. World J. Surg. Oncol. 2019, 17, 12. [Google Scholar] [CrossRef] [Green Version]

[21] Xu, J., Zhao, J., Wang, W., & Liu, M. (2013). Prediction of temperature of tubular truss under fire using artificial neural networks. Fire Saf. J., 56, 74–80

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

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

[24] Zhao, Liquan, and Shuaiyang Li. 2020. "Object Detection Algorithm Based on Improved YOLOv3" Electronics 9, no. 3: 537. https://doi.org/10.3390/electronics9030537

[25] Zhao, Liquan, and Shuaiyang Li. 2020. "Object Detection Algorithm Based on Improved YOLOv3" Electronics 9, no. 3: 537. https://doi.org/10.3390/electronics9030537.