Road Accident Detection & Alert

Submitted by

Name	Reg No:
NIDIN V NANDAN	**223039**
ADARSH PS	**223004**
ADHISH S SUJAN	**223005**

In partial fulfillment of the requirements for the award of Master of Science in Computer Science with Specialization in Data Analytics

Of

School of Digital Sciences

Kerala University of Digital Sciences, Innovation, and Technology

(Digital University Kerala)

Technocity Campus, Thiruvananthapuram, Kerala – 695317

July 2023

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **Road Accident Detection & Alert**
submitted by

Name	Reg No:
NIDIN V NANDAN	**223039**
ADHISH S SUJAN	**223005**
ADARSH PS	**223004**

in partial fulfillment of the requirements for the award of Master of Science in Computer Science with Specialization in Data Analytics is a Bonafide record of the work carried out at KERALA UNIVERSITY OF DIGITAL SCIENCES, INNOVATION AND TECHNOLOGY under our supervision.

Supervisor

Prof. MANOJ KUMAR TK
School Of Digital Sciences
DUK

Course Coordinator

Dr. ANOOP V.S
School of Digital Sciences
DUK

Head of Institution
Prof. SAJI GOPINATH
Vice Chancellor
DUK

DECLARATION

We,**NIDIN V NANDAN, ADHISH S SUJAN, ADARSH PS** ,students of Master of Science in Computer Science with Specialization in Data Analytics, hereby declare that this report is substantially the result of our own work, and has been carried out during the period September 2023-January 2024

Place: TRIVANDRUM

Date: 31/01/2024

NIDIN V NANDAN

ADHISH S SUJAN

ADARSH PS

ACKNOWLEDGEMENT

We would like to express our sincere and deepest gratitude to my guide Dr. T.K. Manoj Kumar, Associate Professor, Digital University Kerala, Trivandrum, for his valuable guidance, advice, and support, which enabled me to complete this project successfully.

Also, we would like to thank Prof. Saji Gopinath from the bottom of our heart for giving us the kind of environment, valuable guidance, and educational resources that made it easier for me to take on a project this size.

Using this opportunity, I would also like to express my gratitude to my family and friends for their invaluable help, inspiration, and support throughout the project's completion.

ABSTRACT

This project addresses the pressing issue of road safety by employing advanced computer vision techniques for real-time accident detection. Utilizing the YOLOv5 model, a cutting edge object detection algorithm, the system efficiently identifies potential accidents in video footage. Pre-trained weights enhance the model's accuracy, and GPU deployment ensures swift processing.

The algorithms effectiveness is visible through meticulous preprocessing of video frames, optimizing them for YOLOv5. While detecting a potential accident with high confidence, the system triggers an alert and sends a notification to a designated Telegram channel. This immediate response enhances the emergency services and facilitates rapid intervention.

The project put forward a novel approach to accident detection, merging cutting-edge deep learning with real-time video analysis. YOLOv5's accuracy and speed prove reliable in identifying road accidents. Integration this with a Telegram bot ensures seamless communication and enables swift information dissemination to relevant authorities. This system contributes to intelligent transportation systems while promoting road safety and reducing accident response times.

Future improvements can be done on top of it to explore additional features like vehicle tracking, license plate recognition, and enhanced anomaly detection for further advancements in road safety and emergency response. This innovative system offers a promising solution for the critical challenge of timely intervention in road accidents, enhancing public safety through the integration of advanced technology and efficient communication channels.

Contents

INTRODUCTION

In modern days the road accidents remain a significant global concern, posing threats to public safety and causing considerable economic and social consequences. This problem raises the need for innovative methods to reduce the effects of accidents becoming more and more evident as the number of vehicles on the road rises day by day. This project mainly aims at the creation and deployment of a road accident detection system, which aims to solve this serious issue by detecting road accidents and alerting . This project intends to change how we view and react to traffic accidents by integrating and using the cutting-edge technology that are available today in the modern tech world, which includes real-time data analysis using the technology of artificial intelligence and computer vision technology.

The increase in the number of automobile crashes is a serious problem that has to be addressed right away in order to save a lot of lives. Global traffic safety reports state that traffic accidents cause millions of deaths and countless injuries each year. The human cost is immense and affects people and families worldwide. In addition, the financial strain brought on by lost productivity, medical costs, and property damage emphasizes the necessity of taking preventative action to lessen the frequency and severity of traffic accidents. The goal of the Road Accident Detection project is to bring about positive change by providing a technology solution that may greatly enhance the safety and security of the driving environment.

At the core of the project is the integration of cutting-edge technologies. Artificial intelligence and particularly the machine learning algorithms, plays a pivotal role in the system's capability to process vast amounts of data in real-time in an efficient manner. By leveraging computer vision, the system can accurately identify and classify potential road accidents based on visual facts. The incorporation of real-time data analysis further enhances the system's responsiveness, allowing for immediate notification and alert

deployment to emergency services. This amalgamation of technologies positions the Road Accident Detection project as a sophisticated and effective tool in preventing accidents and minimizing their impact.

The proposed model is capable of adapting to various surroundings and road conditions is one of its primary strengths. The Road Accident Detection system is made to work flawlessly in both distant and heavily trafficked urban regions. Because of its adaptability, the technology can be successfully implemented into both industrialized and developing countries' infrastructure. This inclusion guarantees that the project's advantages are not restricted by geographical boundaries and is essential in tackling the global nature of the road accident problem.

The Road Accident Detection project is an essential step towards a future where driving is safer and more secure. This innovative solution could revolutionize how we approach and respond to traffic accidents by combining the artificial intelligence and computer vision along with the real-time data processing. The project is positioned as a ray of hope in the continuous fight to lessen the effects of traffic accidents globally because of its dedication to inclusivity, adaptability, and technological sophistication. Subsequent portions of this extensive research will go deeper into the complexities of this project, examining its components, problems, and potential societal impact.

Objective of this project:

- Develop a real-time object detection system using YOLOv5.
- Detecting and responding to accidents in video footage.
- Send notifications to a Telegram channel when an accident is detected.

DATASET CREATION

For the purpose of detection of accidents through a camera we need a model which will be capable of analyzing and detecting when there is an accident occurring in the frame of the camera. So making the model capable of detecting such an accident occurring on the frame we need to train the model with a lot of images containing accidents so that the model will become capable of detecting the presence of an accident in the frame of the camera and it can send the alert

So here we are utilizing a model called YoloV5,a model for real-time object detection,which is renowned for their quickness and accuracy in identifying and categorizing items in pictures and videos. But it is trained only for the detection of the objects that are present in the frames. So here we aim to create a model which will have the capability to detect 5 objects in the video frame

They are:

- Road Accident
- Car
- Truck
- Bike
- Person

By utilizing the YOLOV5 training architecture we will create a .pt file out of it using our Dataset containing the images of the mentioned objects like Accident,Car,Truck,Bike and Person

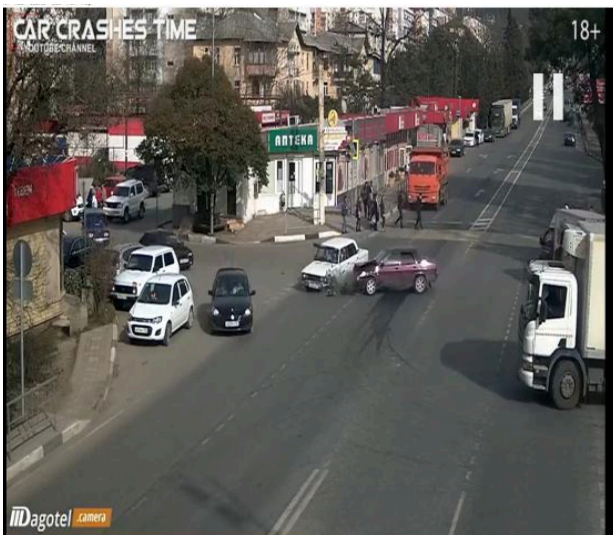
Dataset for creation of the .pt file:

While training a machine learning model in a PyTorch model we typically need a labeled dataset,for object detection tasks. So for this we need a dataset consisting of images with corresponding annotations indicating the locations of objects in the images,by using a bounding box and their associated class labels for identifying the object.

Here in our dataset we have a lot of images that contains road accidents,cars,truck,bike and person

Some of the sample images are given below:

Images used for detecting objects like cars and trucks



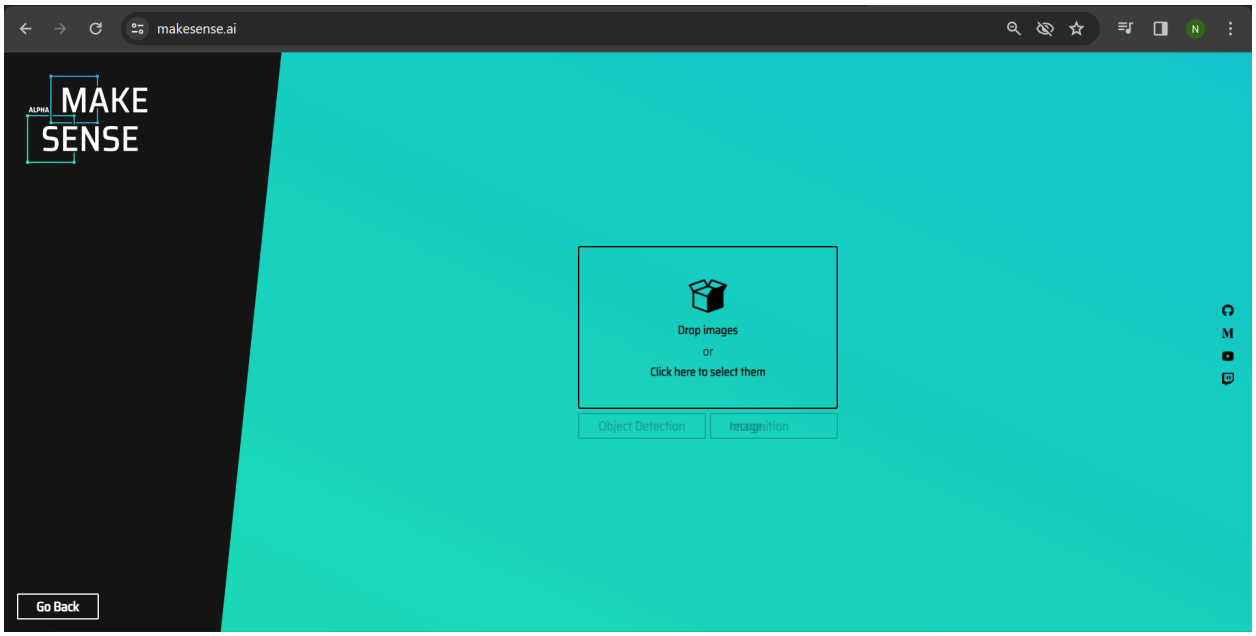
Images usedfor detecting accident

Similarly we have collected a number of pictures containing accidents and a number of pictures that contains our target object for detection.Now the images in the dataset is split it to two folders one for validating and another for training

So in our dataset we have 2 subfolders one for validating and one for training.

Labeling the images for object detection:

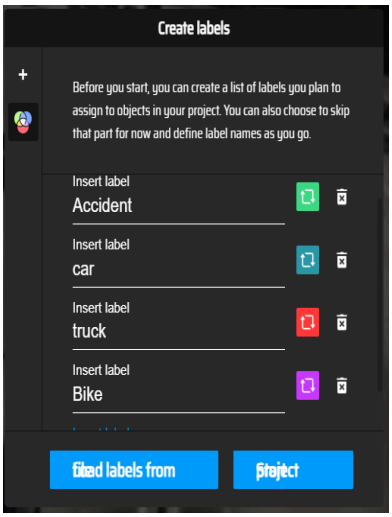
Now we have a lot of images with us. Next our aim is to annotate the images with bounding boxes and class labels for the objects we want the model to detect using any of the annotation tools available. Here we are using one of the popular annotating tool called Make Sense which is a free annotating tool available online



Tool used for annotating

Here we will draw the bounding boxes for the object detection manually by plotting the bounding boxes for different objects in the images in our dataset and create a corresponding label file,a .txt file for each of the images.

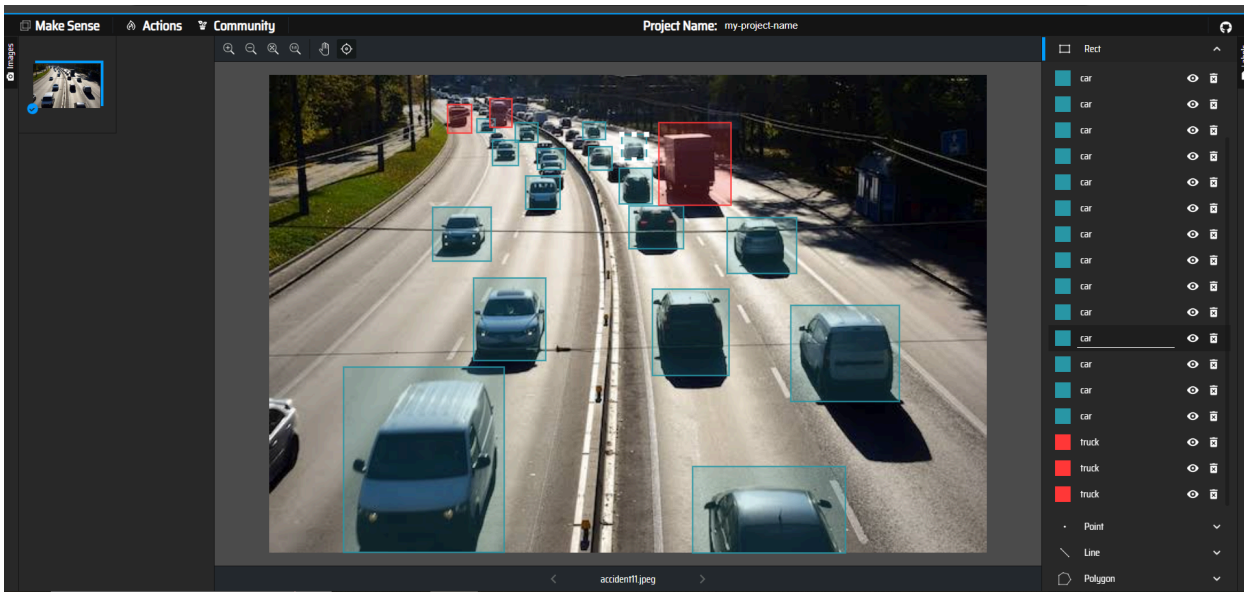
Before start label we need to define the classes for the label



Here we have created 5 different labels for drawing the bounding boxes for 5 different classes like Accident,car,truck,bike and person

So Accident have class label-0,car-1,truck-2,bike-3,person-4

Example of drawing the bounding boxes:



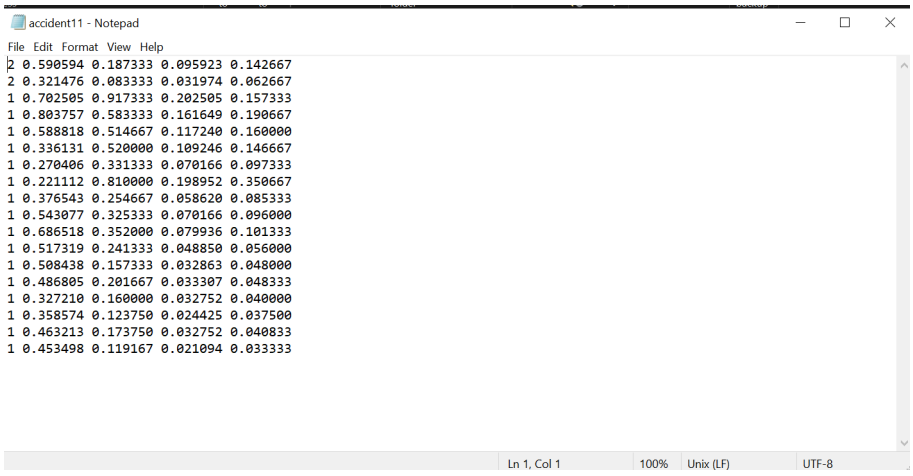
Here we manually annotated the cars and trucks using the Make sense tool which will automatically provide different colors for different labels

In the above figure the Accident part was annotated with green bounding boxes and the remaining objects like car truck etc was also annotated

And we will repeat the same step for the rest of the images and finally we will use the make sense ai tool to export the annotations into a zip file containing files in the YOLO format for doing the task of object detection.

Now we will get a folder containing the labels of the images that we have annotated in a .txt format, which is the files that is in the label folder

Eg:



In the .txt file each line in the data corresponds to a bounding box annotation for an object in an image. The format typically includes the information about the object's class, as well as the coordinates and dimensions of the bounding box.

Consider the first line in the above figure ie,

"2 0.590594 0.187333 0.095923 0.142667"

The first number 2 indicates the class label of the object which indicate Truck

The next four numbers (0.590594 0.187333 0.095923 0.142667) represent the bounding box coordinates and dimensions. These values are typically in the format (x_center, y_center, width, height)

1. x_center: The normalized bounding box center's x-coordinate to the image width.
2. y_center: The bounding box center's y-coordinate, normalised to the image height.
3. width: The width of the bounding box normalized to the image width.
4. height:The bounding box's height was adjusted to match the height of the image.

So finally the dataset will be having two folders train and val for the purpose of training and validating and inside each of the folders we will be having two separate folders called "**images"** and **"labels"** containing the images and the corresponding labels.

METHODOLOGY

Transfer Learning:

It is a machine learning method that involves adapting an already existing model that has been trained on one task to a similar task. Transfer learning uses the information from a source task to improve performance on a target task, as opposed to starting from scratch when training a model for a new task. When there is a shortage of labeled data for the intended purpose, this is especially helpful.It is a technique commonly used in deep learning, including object detection tasks with YOLOv5, where a model that has already been pre-trained on a huge dataset is refined on a smaller dataset relevant to our task.The idea is to leverage the knowledge gained by a pre-trained model on the general dataset to improve the performance of the model on a task with limited labeled data.

Here for our purpose we use this pre-trained model available in the YOLOv5 repository to train our own dataset that includes object classes like accident, car, truck, bus, and person.

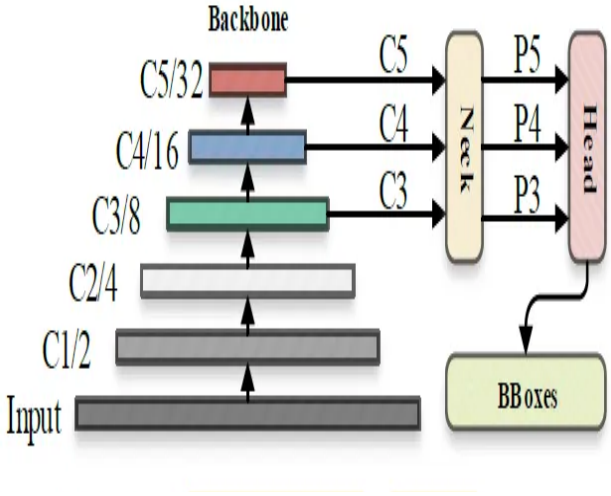
YOLOv5 Model Selection:

You Only Look Once version 5, or in short YOLOv5 is a cutting edge real-time object identification model renowned for its accuracy and speed. YOLOv5, which was created as an advancement of the YOLO series, brings about notable architectural enhancements, rendering it a sturdy option for an extensive array of computer vision uses.YOLOv5 uses a neural network to interpret an entire image in a single forward pass, following to the YOLO paradigm. Because of its distinct architecture, which allows for real-time object identification, it is ideally suited for applications where minimal latency

is essential.When it comes to identifying and categorizing objects in pictures and video frames, YOLOv5 excels. For several items, it gives real-time bounding box coordinates, class labels, and confidence scores.Because of YOLOv5 exceptional adaptability to unique datasets, researchers and developers can train the model to do certain object recognition tasks. YOLOv5 may be used for a numerous range of applications, such as vehicle recognition and pedestrian detection.

YOLOv5 Architecture:

The inputted image will be taken through the input layer and then processed and then send for feature extraction in the Backbone



The neck, or feature fusion network, is used by the backbone to fuse feature maps of different sizes in order to produce three feature maps, P3, P4, and P5, which are used to detect tiny, medium, and big objects in the image, respectively. (The sizes of 80x80, 40x40, and 20x20 are used to indicate the dimensions in the YOLOv5).

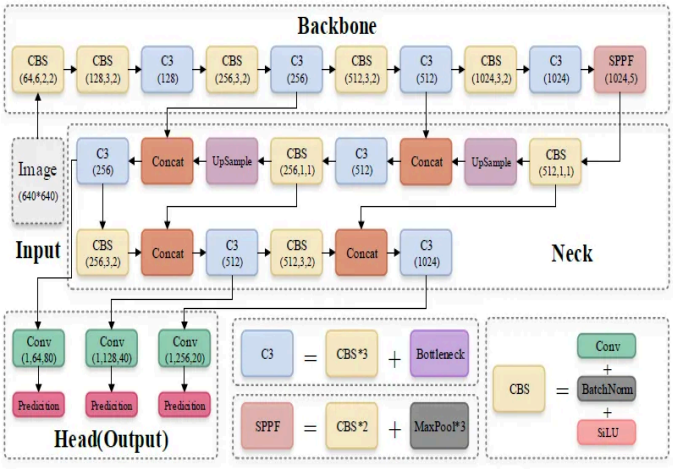
After the delivery of the three feature maps in to the prediction head (head) where the bounding-box regression and the confidence calculation were carried out for Using each pixel in the feature map the pre-established prior anchor. This resulted in the creation of a multidimensional array (BBoxes) that included data on box coordinates, object class, class confidence, width, and height.

Applying a non-maximum suppression (NMS) process and setting the corresponding thresholds (confthreshold, objthreshold) to remove the unnecessary data from the array can provide the detection information's final result.

YOLOv5: Backbone

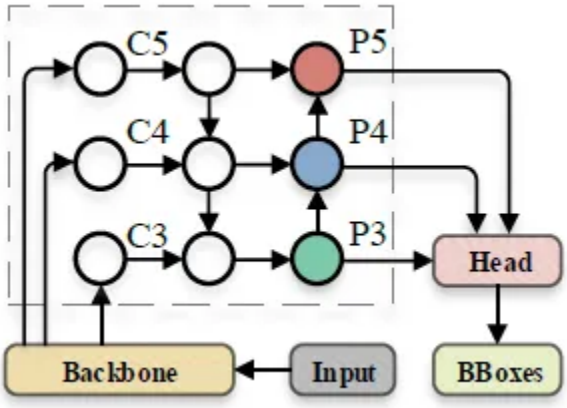
It uses CSPDarknet53 as the backbone.

C3 and Multiple CBS (Conv + BatchNorm + SiLU) modules are arranged to form the main structure, and one SPPF module is attached at the end.



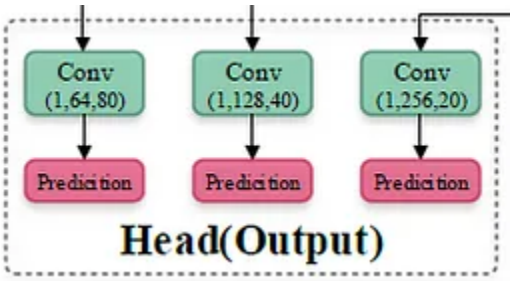
YOLOv5: Neck

YOLOv5 Neck utilizes PAN(Path Aggregation Network) and FPN(Feature Pyramid Network) as a primary component in its architecture.



The fundamental idea behind FPN is to increase the output feature map (C3, C4, and C5) obtained from the feature extraction network by sequential convolution down sampling operations in order to construct multiple new feature maps (P3, P4, and P5) for the purpose of detecting objects at different scales.

YOLOv5: Head



The model head is where the processes' last stages are completed. Anchor boxes are applied to feature maps, and the final output which includes classes, objectness scores, and bounding boxes is produced. It is in charge of the process of adjusting the centre coordinate and size of the preset previous anchor to match those of the final prediction box.

OpenCV (cv2) - Computer Vision and Image Processing Library:

A robust and open-source library for computer vision and image processing applications is called an Open Source Computer Vision Library(OpenCV). Numerous disciplines, such as robotics, machine learning, and computer vision research, heavily rely on OpenCV. The Python wrapper for the OpenCV library, the cv2 module it was utilised in our model for displaying the result video and was utilised to capture the image of the accident inorder to send along with the alert message.

The cv2.imshow and cv2.imwrite methods that we have in the OpenCV library make it easier to visualise and display images. Visual inspection of input data, intermediate outcomes, and final outputs is crucial for computer vision programme development and evaluation.OpenCV supports various video processing tasks like video capture, video writing, and frame extraction from video streams. For applications like object tracking, real-time processing, and video analysis, this functionality is quite helpful.

PyTorch - Deep Learning Library:

PyTorch is an open-source deep learning library popular for its flexibility, dynamic computational graph, and extensive support for deep learning task development and research.PyTorch is a popular platform for creating and training neural networks, proposed by Facebook's AI Research unit (FAIR).

One of the important aspects of PyTorch is its dynamic computational graph, which enables dynamic graph building in runtime. Its dynamic nature makes it particularly well-suited for research and development because it makes experimentation, debugging, and model customisation simple. The building of neural networks is made simple and intuitive with PyTorch. The torch.nn module simplifies the process of creating and training complex models by providing a range of pre-defined layers, loss functions, and optimisation techniques.During the neural networks backward pass, gradients are automatically computed by PyTorch Autograd automated differentiation module. This feature makes gradient-based optimisation easier to understand and train deep learning models more effectively.

Pytorch has a library called TorchVision which was created especially for computer vision applications. It offers complete resources for creating computer vision applications, including pre-trained models, datasets, and an assortment of image processing methods.

Telegram Bot and Channel for Notifications

Telegram channels provide a platform for broadcasting messages to a large audience. In our project we are employing a telegram channel to which the alert message and the image of the accident will be sent using the API key that is offered by the Telegram.

Model Training

We have assembled an extensive collection of photos that show both accident and non-accident scenes. We then used this carefully annotated dataset to train the YOLOv5 architecture by labeling the objects of interest as "accident" using the bounding box annotation method. Next we specify the configuration parameters required to train the YOLOv5 model. This entails defining variables like the number of training epochs, batch size, and learning rate. Depending on the features of our dataset and the available processing power, we can change those parameters. Use transfer learning to get the training process started. To enable the YOLOv5 model to inherit information about generic characteristics, use pre-trained weights on a large-scale dataset. Utilising the target dataset, adjust the YOLOv5 model. The model gains the ability to adjust to the unique properties of the objects in our dataset as training goes on. Here we have to keep an eye on the training measures, such as accuracy and loss inorder to make sure the model is effectively converging. We validate the model's performance on the validation during training.

With the dataset prepared and annotated our next aim is to train the YOLOv5 model for detecting accidents. For that we need to create a .pt file out of the training inorder to make the model able to detect the accidents in the videos in real time.Now we initializes the YOLOv5 repository by cloning it from the official GitHub repository.The inclusion of pre-trained weights from a smaller YOLOv5 model (yolov5s.pt) facilitates transfer learning. This method uses information inherited from a model built on a larger dataset to begin the training process. Then the model adjusts to the unique characteristics of the annotated accident dataset as training goes on.

data.yaml file:

In order to facilitate the training of our dataset on top of the YOLOv5 we need to provide a unique dataset that was exclusively created for the purpose of our task i.e., accident and some of the object detection like car,truck,bike,person. So for this we need to employ a separate file called data.yaml.

The data.yaml file is a configuration file which is used to specify information about the dataset, classes, and file paths.

The below shown is the data.yaml file used for this project

C: > Users > Admin > Desktop > dataset code > yolov5 > data > ! data.yaml

```yaml
train: /dataset/train/images
val: /dataset/val/images


nc: 5
names: ['accident','car','truck','bike','person']


```

- train and val: Paths to the directories containing training and validation images, respectively.
- nc (Number of classes): Specifies the total number of classes in our dataset, including the background class. In our project , there are five classes in total.
- names: A list of class names corresponding to the classes in our dataset.here it is accident,car,truck,bike,person. We have to ensure that the order of class names matches the order of class indices in the YOLOv5 model's output.

Training parameters used for the project:

- Image Size (--img): which specifies the input image size during training. Larger sizes can capture finer details but demand more computational resources. Here we used 640
- Batch Size (--batch): which gives the number of images processed in each iteration. Adjusting this parameter is crucial for the availability of the GPU memory.Here we used a batch size of 16
- Epochs (--epochs): It is for setting the number of training epochs. An epoch denotes a complete pass through the entire dataset, and the chosen value influences the model's learning process. Here we used a epoch of 150
- Data Configuration (--data): It is the data.yaml configuration file. This file holds essential information about the dataset, including classes and file paths.
- Pre-trained Weights (--weights): It is the pre-trained weights from yolov5s.pt used as an initial starting point for the model. Transfer learning with pre-trained weights enables the model to inherit generic characteristics from a large-scale dataset.

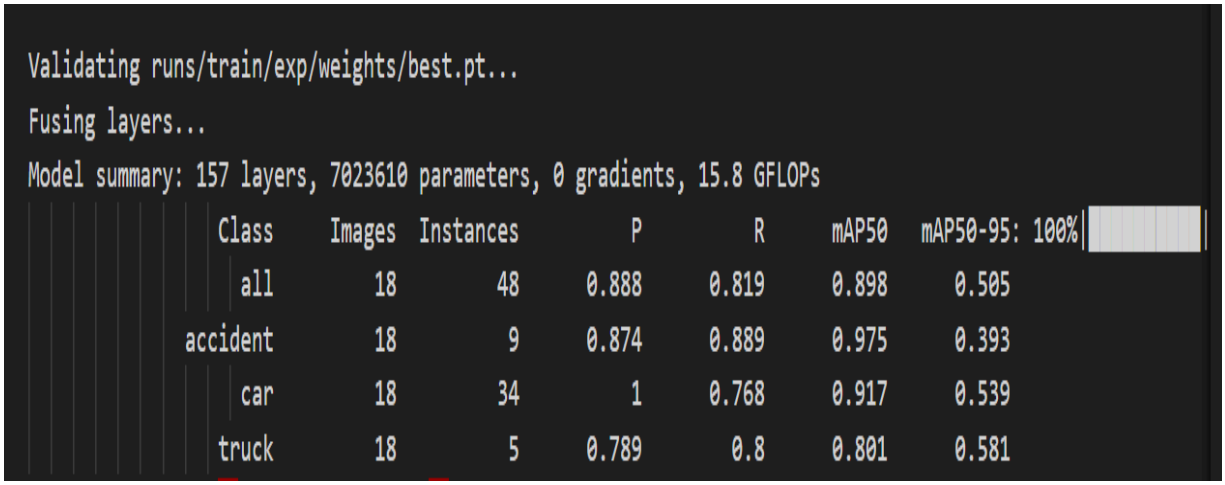
Important parameters like accuracy and loss are analyzed during the training phase. The model's convergence and performance on the training dataset are shown by these indicators. To make sure that the model is learning efficiently, these indicators are updated on a regular basis.

A crucial component of training is validation, which evaluates how well the model generalizes to new sets of data. The possibility of the model overfitting to the training data is reduced by frequently evaluating it on an independent validation dataset. For this purpose we have already created a separate folder for the validation dataset.

During the training process, the script saves the model’s checkpoints at specified intervals. Upon completion of training, a .pt file named "best.pt"(this is the same file which was renamed as with o&a.pt) which encapsulates the trained YOLOv5 model. This file can be used for subsequent inference tasks, such as detecting accidents in new images or videos.

Model Evaluation

Final summary of the model training is shown below:



```
Validating runs/train/exp/weights/best.pt...
Fusing layers...
Model summary: 157 layers, 7023610 parameters, 0 gradients, 15.8 GFLOPs
               Class     Images  Instances          P          R      mAP50   mAP50-95: 100%
                 all         18         48      0.888      0.819      0.898      0.505
            accident         18          9      0.874      0.889      0.975      0.393
                 car         18         34          1      0.768      0.917      0.539
               truck         18          5      0.789        0.8      0.801      0.581
```

The model summary provides a comprehensive assessment of the model's performance on the dataset used.

- Precision (P)is the ratio of all expected positive cases to correctly forecasted positive cases.
- Recall (R): The ratio of correctly forecasted positive instances to the total actual positives.
- mAP50 (mean Average Precision at IoU 0.50): The average precision across different IoU thresholds, with IoU 0.50 being a common threshold for object detection tasks.
- mAP50-95: The mean Average Precision across different IoU thresholds from 0.50 to 0.95.

The high precision and recall values indicate that the model is accurate in its predictions, minimizing false positives. The balance between recall and precision depends on the specific requirements of the task.The mean Average Precision at IoU 0.50 provides an overall measure of the model's accuracy. A higher mAPvalue indicates better object localization.

For our model the precision value for the accident class is 0.874 and the recall value is 0.889 and the mAP50 value is 0.975 indicating a high performance and the capability of detecting the accident correctly in the video frame. And for the remaining classes also the performance is very good which implies that the model is capable of detecting the accident and the objects that we have mentioned

So overall the performance of the model is satisfying and the good value of precision and recall and mAP50 indicates that the model is capable of detecting the presence of an accident in the video. The model’s accuracy can be further improved by adding more and more images and retraining.

Now we have the wights file .pt which is capable of detecting the accident in a similar way of detecting the objects in the video. We can employ this weight file along with the opencv library to do the job of detecting the accident in real time videos.

Results and Insights

In our code we have created two parts

1. Code for detecting the accident
2. Code for detecting the accident and sending alert to a channel in telegram

In the first we just created the model using the created file to detect the accident

Part 1- Detecting the accident

The code snippet is:

“

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
weights = 'o&a.pt'
model = attempt_loads(weight)
model.to(device)  # Move the model to the specified GPU (GPU 0 in this case)
stride = int(model.stride.max())
imgsz = 640
cap = cv2.VideoCapture("accidentvideo.mp4")
accident_frames_threshold = 80  # Set the number of consecutive frames for accident detection
accident_frames = 0
alert_displayed = False

while True:
    frame,ret = cap.read()
    if not ret:
        break

    img0 = letterbox(frame, new_shapes=imgsz)[0]
```

```
    imgs = img0[:, :, ::-1].transpose(2, 0, 1)
    imgs = np.ascontiguousarray(img)

    imgs = torch.from_numpy(imgs).to(device)
    imgs = imgs.float()  # uint8 to float32
    imgs /= 255.0  # 0 - 255 to 0.0 - 1.0
    if imgs.ndimension() == 3:
        img = img.unsqueeze(0)

    predict = model(imgs)[0]
    predict = non_max_suppression(predict, 0.4, 0.5, classes=None, agnostic=False)

    accident_detected = False

    for i in predict:
        if i is not None and len(det):
            i[:, :4] = scale_boxes(imgs.shape[2:], i[:, :4], frame.shape).round()
            for *xyxy, conf, cls in i:
                label = f'{model.names[int(cls)]} {conf:.2f}'
                plot_one_box(xyxy, frame, label=label, color=(0, 255, 0), line_thickness=2)
                if model.names[int(cls)] == 'accident':
                    accident_detected = True
                    break

    if accident_detected:
        accident_frames += 1
        if accident_frames >= accident_frames_threshold and not alert_displayed:
            cv2.putText(frame, 'Accident Detected!', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 0, 255), 2)
            alert_displayed = True
    else:
        accident_frames = 0
        alert_displayed = False
```

```
    cv2.imshow('accident detector', frame)
    if cv2.waitKey(50) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()”
```

Functions Used:

scale_boxes:

This function is used to resize the bounding boxes (boxes) from one image shape (img1_shape) to another image shape (img0_shape). It considers a rescaling factor (gain) and optional padding (pad) for doing this. The final rescaled boxes are clipped to fit within the specified image shape.

clip_boxes Function:

This function clips bounding boxes (boxes) to be within the specified image shape (shape). This function ensures that the coordinates of the boxes are within the valid range.

xywh2xyxy Function:

This function converts bounding boxes from the format [x, y, w, h] to [x1, y1, x2, y2], where (x1, y1) represents the top-left corner and the (x2, y2) represents the bottom-right corner of the box.

box_iou Function:

This computes the Intersection over Union (IoU) between two sets of bounding boxes (box1 and box2). IoU is a measure of the overlap between two bounding boxes.

non_max_suppression Function:

This will perform non-maximum suppression (NMS) on a set of bounding boxes (prediction). NMS is a post-processing step commonly used in object detection to remove redundant bounding boxes and keep only the most confident ones.

send_telegram_message_to_channel Function:

This function is used to send the alert message to the created telegram channel. This employs the use of telegram API inorder to send the alert message.

Here the code uses the multiple functions that we have already defined to detect the presence of accident in the frame

The result of the code is:

The model is detecting the presence of the accident in the video frame here the model identifies the accident in the frame with a probability close to 85%

Demo detection of accident:
https://github.com/nidinvnandan/Mini_Project_II/blob/main/MAIN_CODE/roadaccidentvideo1.gif

Result for another video:

Here also the model is detecting the accident in the frame correctly and with high probability

Working demo:
https://github.com/nidinvnandan/Mini_Project_II/blob/main/MAIN_CODE/roadaccidentvideo2.gif

Part 2 : Detecting the accident and sending the Alert message:

Here for the detection of the accident we are employing the same method and for sending the alert message we are using one of the famous messaging platform telegram

For this alert sending purpose we will create a particular channel in the telegram and we will use an API key provided by the telegram inorder to send the alert message to the channel.

We will send the image of the accident also along with the alert message so that the person who is responding to the emergency situation can verify that the accident is occurred there
The model will send the alert message only if the probability of the accident detected within the frame goes above 80%, this is done in order to make sure that under any circumstances the model will not send the false alert message

For sending the alert we use a function called send_telegram_message_to_channel for sending the message along with the accident photo.

The code snippet is:

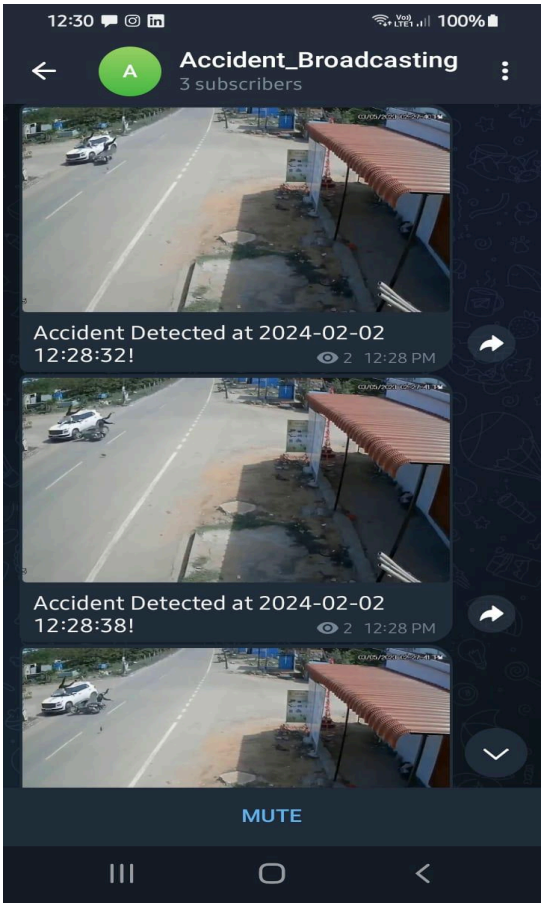
“

```
def send_telegram_message_to_channel(bot_token, chat_id, message, photo=None):
                url = f"https://api.telegram.org/bot{bot_token}/sendPhoto" if photo else f"https://api.telegram.org/bot{bot_token}/sendMessage"
    params = {
        "chat_id": chat_id,
        "caption": message
    }
    files = {'photo': ('accident.jpg', photo)} if photo else None
    responses = requests.post(url, params=params, files=files)
    if responses.status_code == 200:
        print("accident alert sent successfully!")
    else:
        print(f"Failed to send alert. Status code: {response.status_code}")”
```

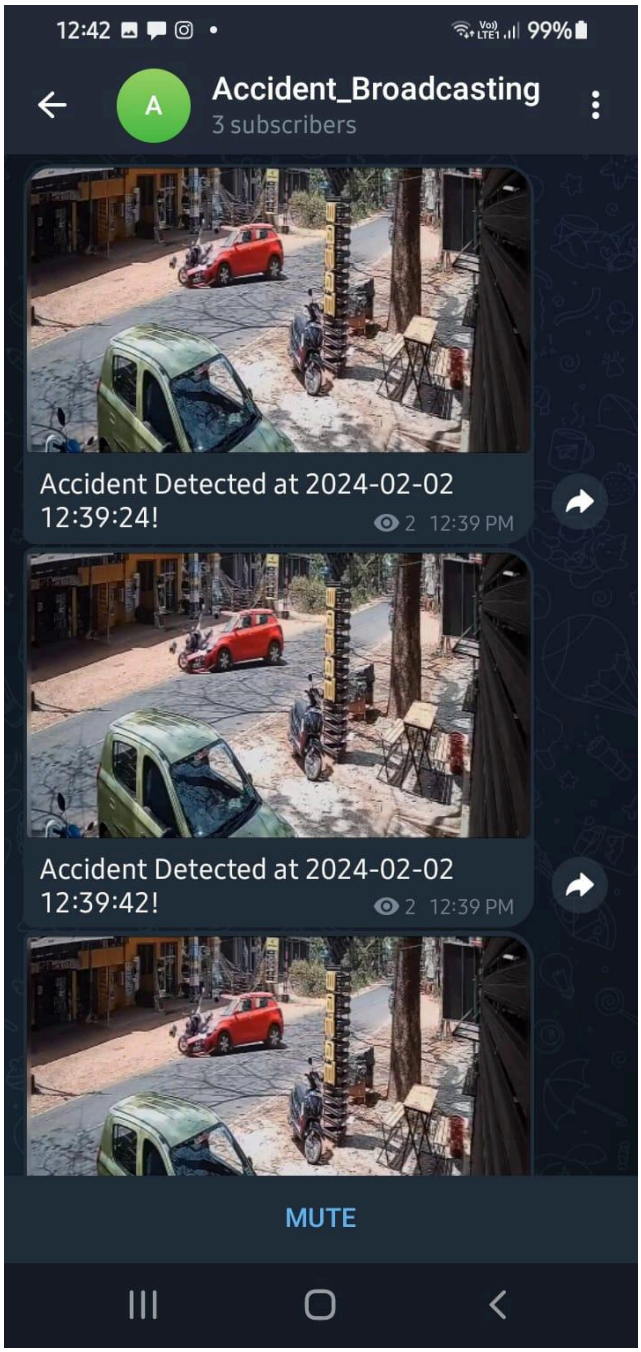
Result for the accident detection and alert part:

Whenever we are detecting the accident in the frame and if the probability of that class belonging to accident is greater than 80% then we will utilize the opencv library to create a .jpg format file that contains the accident frame image. This accident image along with the accident occurring time and date will be attached with the message and then using the api key provided by the telegram we will send this information to the channel named Accident_Broadcasting.

We can see that the model is performing well as it is correctly sending the accident photos along with the alert message and time to the telegram channel.



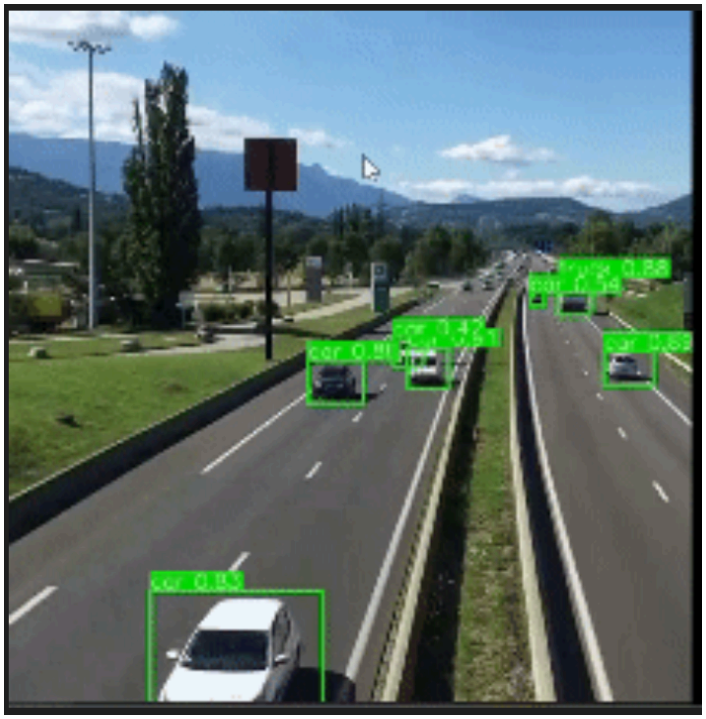
Doing it with another sample video:



Here also we can see that whenever the model detects the accident it will send the accident frame as well as the alert message into the channel. In this when the bike hits the car the probability of accident becomes greater than 80% and at that point the OpenCv library will capture the accident image from the frame and then using the datetime the date and time of the accident will be taken and then both these will be combined along with a message Accident Detected and using the API key provided by the Telegram it will be sended to the created telegram channel called Accident_Broadcasting for giving the message that a accident was detected

It is also evident from the picture of the telegram channel that the model is correctly detecting the accident scenarios and send the alert message to the telegram channel

Testing the model with videos having no accident part:



Demo link:
https://github.com/nidinvnandan/Mini_Project_II/blob/main/MAIN_CODE/normaldetection.gif

Now in this video there is no accident visual and it just shows the passing of vehicles on the road. Here our model is not recognizing any accident part and it is just detecting the cars that are passing through the road. As there is no accident detected it will not sent any alert message to the telegram channel

So this is the working of the model,whenever it detects an accident it will immediately capture the accident frame and attach it with the time of the accident and then send it to the channel as an alert.

CONCLUSION

In summary, this project demonstrates the viability and effectiveness of using advanced computer vision techniques to detect road accidents. Through the utilization of the YOLOv5 model, we achieve real-time accident detection and analysis from video footage, contributing to improved road safety and faster emergency response times. The successful integration of this detection system with Telegram ensures seamless communication and timely dissemination of alerts to relevant parties.

Moreover, this project highlights the significance of employing state-of-the-art technologies to address important issues like road safety. The deployment of deep learning algorithms such as YOLOv5 shows the potential of artificial intelligence in transforming transportation systems and emergency services. As technology continues to advance, there are opportunities for further enhancements and refinements to the system, including the addition of more features for comprehensive accident analysis and intervention.

Overall, the road accident detection system showcased in this project underscores the value of collaboration between computer science and transportation engineering. By leveraging machine learning and computer vision capabilities, we make significant progress in creating safer roads and minimizing the impact of accidents on public safety. Looking ahead, ongoing research and innovation in this field promise to further enhance road safety measures and ultimately save lives.

REFERENCES

https://ieeexplore.ieee.org/document/9801568

https://sh-tsang.medium.com/brief-review-yolov5-for-object-detection-84cc6c6a0e3a

https://psujit775.medium.com/ihow-to-send-telegram-message-with-python-e826b94f1d9b