International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 2, Mar-Apr 2025

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Ambulance Detection in traffic and alerting System through Deep learning and IOT

Mr.Asharfali Shaikh[1], Kapse Vaibhav[2], Thombare Krushna[3], Kale Gauri[4], Pokale Manasvi[5]

[*1]Ass.prof. Department of Computer Technology, Padmashri Dr. Vitthalrao Vikhe Patil Institute of Technology & Engineering (Polytechnic), Loni, Maharashtra, India

[*2,3, 4 5]Department of Computer Technology, Padmashri Dr. Vitthalrao Vikhe Patil Institute of Technology & Engineering (Polytechnic), Loni, Maharashtra, India

----------**********----------

Abstract:

Early Alert System It is an urgent need to use image processing to notify other drivers so that lives can be spared. For the simple reason that traffic congestion has been becoming worse over the past few years due to the ever-increasing number of vehicles on the road. Countries like India and Japan are experiencing severe traffic problems due to their expansive and lengthy highways, which make it impossible to carve out a special lane for emergency vehicles. Because of this, thousands of people have perished. The suggested system uses an artificial intelligence model called the MobileNet deep learning technique to detect when an ambulance or fire engine is approaching at specific intersections, allowing drivers of cars and other four-wheeler vehicles to get to them as quickly as possible. When emergency vehicles like ambulances and fire trucks are spotted, a digital board will show an alarm message at every intersection up ahead.

Keywords: Image Processing, CNN, Mobile net, Deep Learning models, Object Detection. Neural Network

----------**********----------

I. INTRODUCTION

Teachers and schools face a formidable problem in A significant number of automobiles on the road has rapidly increased the amount of vehicle traffic in cities. Due to the heavy traffic, there are frequently traffic delays on the roadways, which can result in human lives being lost when emergency medical vehicles like ambulances and fire engines become stopped in the gridlock. Current traffic control systems are static, requiring cars to wait until the microcontroller turns on the green light for that lane after a predetermined length of time. The traffic police can provide precedence to the ambulance if it is stuck close to a traffic light by displaying the appropriate symbols or signals to the passing vehicles, allowing the ambulance to exit the traffic as swiftly as feasible. Additionally, if the emergency vehicles are blocked in a lane distance from the traffic signal, the ambulance's siren won't be able to reach the traffic police, forcing us to wait until the traffic is cleared or rely on other cars to move aside, which is not always possible in traffic. The use of Image Processing technology has been implemented in the proposed system. This system uses a CNN (Convolutional neural network) and MobileNet modules. Following the predetermined stage, the features are taken from the provided input picture using different layers of the CNN. And MobileNet also same work but accuracy is more than and used to express the view that one should make the most of the present moment without worrying about the future. The detection outcome's projected bounding boxes are used. The ambulances seen in the traffic data are the product of the process. The CNN network is the basis for the proposals. We will do an image detection following the image capture in

order to further define the detection process and focus it on the ambulance. The next phase of the procedure is bilateral filtering, where the intensity value of each pixel is saved using the weighted intensity of the nearby pixels that was collected in the previous step. In this stage, the edges' pixel values are saved. The data from this dataset file is sent on to the following CNN layer for additional filtering. This line contains the target. The ambulance is detection next stage signals is controlled then main target is to send the message into the traffic controller.

[1] A pioneering investigation on the effects of the ongoing COVID-19 pandemic on the TT of firefighter ambulances serving hospitals in the Doubs-France region is presented by Selene Cerna et al. As soon as the number of confirmed and suspected cases of the disease started to climb in 2020, there was a noticeable increase in the AvTT per day. Further evidence that the daily AvTT increased as a result of the COVID-19 pandemic is the fact that this increment is out of the ordinary when compared to prior years. The shortage of ambulances and people at the SDIS 25 centers will have a direct and detrimental domino impact on the firefighters' service in 2020, leading to an increase in breakdowns. Accordingly, the authors of this research argue that a data-driven approach is necessary to predict which TTs will have ambulances at specific hospitals. When staff members announce their intention to visit a healthcare facility, this forecast could be made. More lives could be saved if fire brigades and, on a worldwide scale, EMSs could activate proactive decision-making using the resources they have. Limitation\ Stay tuned for more updates! We'll be extending the author's findings and forecasts to cover the second half of 2020, including the second wave of COVID-19 in France. The author also plans to test the methodology with other ML techniques and enhance it by adding new features and studying their influence (feature selection). Finally, there is also another direction for further research, and that is about finding the nearest hospital with a shorter TT, considering internal hospital data.

[2] in S. Rafiq et al explain the existing system doesn't provide a transparent path for emergency vehicles during traffic congestion. Traffic Density Analysis, Ambulance, and Accident detection System Using Image Processing has been discussed in this proposed system. According to our literature review, an RFID-based smart traffic control system can alleviate traffic congestion and efficiently direct emergency vehicles to their designated lanes (thanks to the author's implementation of vital data sharing with hospitals). In order to store patient vital parameters and capture the current status of traffic signals in different paths, the author upgraded the Arduino uno board and added an additional system at the junction. This system scans the lane density repeatedly, allowing the system to automatically allow the lane with the highest density. As a result, emergency vehicles are able to avoid traffic jams and get to their destinations faster, saving countless lives.

[3] the third Patel, R et al introduced Emergency vehicle priority (EVP) systems could greatly reduce the traveling time for emergency vehicles like ambulances, which in turn could save many lives. The presented EVP used a neural network-based classifier to detect an ambulance siren. A mobile app was used to track the location of the ambulance. A decentralized network of IoT devices equipped with visual indicators was used to inform traffic of an incoming ambulance. Hence a makeshift emergency lane was set up on the roads. author’s EVP system didn’t require any modifications to be made to existing ambulances. Nodes could be added to the IoT network easily and without affecting the response speed of the system. As author’s system used audio signals and WiFi communication to function, it was not affected by factors like rain, snow, fog, etc. author expect that author’s proposed system enables ambulances and other emergency vehicles to reach their destinations as quickly as possible

[4] Vehicle detection can be classified using Random forest model as mentioned by Shubham gade et al. By identifying the pattern of EV charging station location and energy use using the imputated data from the preprocessing step, the author is able to increase predicted accuracy and control overfitting. This is accomplished by training several decision tree classifiers on separate sub-samples of the dataset using averaging in a random forest, which is a meta estimator. Optimal splitting is employed by forest trees; this is the same as building a Decision Tree Regressor with the splitter value as the "best" parameter. After setting the bootstrap value to the default Boolean value 'True,' you'll be able to manage the sub-sample size with the max_samples option. or else the whole dataset is used in the building of each tree. To get an idea of how many trees there are in the forest, author look at the criteria. In order to separate the maximum number of samples between training and testing stages, author use the train_test_split function. To train the model, author use 85% of the data, while to test it, author use 15%. The standard scaler class is used to normalize features by eliminating the mean and scaling the data to the unit variance. author use the train and test data lists that are obtained for this purpose. During the designated training epochs, authoruse the random forest regressor to shape the random forest model. After training the random forest model with the data, author apply K-fold cross-validation using the cross_val_score class. Figure 2 shows the schematic diagram of the Random Forest regression model, which may be used to understand how the model operates.

The study is structured into five parts. We present a comprehensive overview of ITS in the second section. A concise introduction to deep learning and its uses is given in Section 3. Section 4 details how smart cities and ITS make use of deep learning to detect pedestrians, summarizes the work of several academics in the area, and lays out the obstacles to further study in each subfield. Next, in Section 5, we shall present the final verdict.

II. LITERATURE SURVEY

[5] In order to categorize photos of emergency vehicles, kherraki, Amine et al. conducted a comparison study utilizing eight well-known CNN architectures. Automatic vehicle classification in traffic scenes has been accomplished by the development and deployment of a very effective deep learning system. The Vidhya Emergency Vehicle dataset was preprocessed to standardize image sizes. For each convolutional neural network (CNN) design, the author has incorporated three layers: a "GlobalAveragePooling2D" layer to expedite training by reducing the input image's dimensions, a "Dropout" layer to prevent overfitting, and a "Dense" layer to specify the number of output classes. Simulating each architecture later on, the author found that DenseNet121, with its 95.14% accuracy, 93.87% F1 score, and 27.5 MB average order memory, is the best model for real-time emergency vehicle classification. Applications that employ the author's top design for emergency vehicle classification will undoubtedly benefit much from the achieved results.

[6] Sunil M et al. suggested a method to save a patient's life more quickly. Because it reduces reaction time, it's useful for those in dire situations. By utilizing the video and detection algorithm, this application expedites the ambulance's ability to reach patients. More lives are saved by this system. Ambulance detection is made easier and at a lower cost. The problem is that it will be extremely difficult to notice the ambulance if it becomes stuck after traveling more than 300 meters. Limitation\ Projects in the future involving Eventually, this technology can be used to many types of emergency vehicles, including fire engines, police cars, and others. author can easily transmit patient data to the healthcare facility.

[7] In their study, Usaid, M. et al. detail how they created a collection of traffic and ambulance siren sounds in ".wav" format. The Librosa module in Python is used to extract features from the acquired audio signals after they have been preprocessed.

Spectral centroid, spectral bandwidth, zero-crossing rate, Chromastft, and twenty-one Mel-frequency Cepstral Coefficients are retrieved. An MLP model was trained using a GPU; the model attained 90% accuracy throughout training. Feature extraction and dataset design have been the primary areas of concentration, with the primary goal of differentiating ambulance sound from road noise. The data has not been tested in a real-world setting by the author yet. The author intends to record additional audio datasets in the future so that they can test the data in real-time. After improving the dataset and testing in real-world circumstances, the resulting model can be put into practice. Any microcontroller device can be configured to use the trained model in real-world applications to maximize response time to ambulance sirens. Given its relative simplicity, the well-established MLP-model architect can be executed on any microprocessing device or board that does not have a Graphical Processing Unit.

[8] The authors, Bhoomika G. M. et al., accomplished the following: Create a dataset that the program could use for testing and training. The application made use of the design that distinguished the ambulance from the majority of vehicles. When the author realized the captured vehicle was an ambulance, the computer changed the light from red to green, and "a notification was sent to the traffic controller" was sent. Other traffic lights remained red. Emergency vehicles like the ambulance can use the camera's prediction and prevention features to go where they need to go in only a few minutes. It is possible to successfully finish the system-wide simulation after the capture, training, and detection work activities are done. All emergency vehicles can be included in the future expansion of the research scope.

[9] According to Mahmud, Umar et al., traffic management poses a significant challenge in emerging nations. To control the flow of traffic, both automated and human-operated technologies are used. Wasted fossil fuel, pollution of the urban environment, and inconvenient delays are all outcomes of traffic congestion. Accidents involving fast electric vehicle travel through traffic intersections are more likely to occur as a result of these delays. Using a Raspberry PI and a few basic sensors—such as infrared sensors to pick up on emergency lights and directional microphones to pick up on sirens—this study suggests a decentralized A IoT-based EV transit system. This system ensures that electric vehicles navigate crossings smoothly by making decisions based on predetermined guidelines. The traffic light will remain green until an incoming lane detects an EV and the EV has safely cleared the intersection. Detection of the departure lane allows for the sharing of this information with nearby intersections, significantly reducing travel time. The method has led to a 62.5 percent rise in the percentage of successful EV transit, up from 25% before.

[10] According to Yarra Kavitha et al., the main goal of this project's design is to reduce the amount of time electric vehicles have to wait at the junction before they can cross, which in turn reduces the likelihood of accidents that could cause serious injuries or even fatalities. No human intervention from the driver is necessary for the transmission of SRM requests to the RSU; the system operates entirely autonomously in this instance. Thanks to DSRC's dependable communication, transmission and reception happen substantially faster than the GSM technique, which is a major drawback. Additionally, the system is less expensive since this method does not utilize extra detectors and sensor circuits that other approaches employ to find the vehicle's location or to identify when the vehicle enters the pre-emption point. A system of automatic messaging can also be included to let the driver know when there is an open lane at an intersection leading to their intended location. To alert approaching intersections of the urgent need for action in congested locations, decision support systems may initiate communication. It is possible to install this system in non-emergency vehicles, like motorbikes, so they can help the user in an emergency. In the future, this system could be enhanced by making a web or mobile app that allows for more intelligent automatic and manual

control.

[11] Digital control of traffic signs calibrated to the route's actual traffic volume can reduce the human labor of the road safety officer, according to Dr. P. Sankar Babu et al. Since the system is entirely automated, human intervention is not necessary. If an officer is present at the intersection, the light will turn red when a stolen car is spotted. Additionally, they will be alerted through text message when it is time to prepare to stop the stolen vehicle at the next intersection. Emergency vehicles, such fire engines and ambulances, must respond quickly in the event of an emergency. If they are unable to escape traffic, they put the lives of countless others in jeopardy. The light will turn green as soon as the all-clear is given, provided that a rescue vehicle is waiting at the junction. It will turn red once the emergency vehicle has gone through. Using longer-range RFID scanners to test the prototype might help refine it. The stolen automobile detecting module might also have a GPS component.

[12]the eleventh In order to identify emergency vehicles in congested traffic areas, Rawand Sulayman et al. presented a model. There is an excess of traffic in densely populated areas like Kurdistan, which causes emergency vehicles like fire trucks and ambulances to get stuck in the middle of the road. This problem will be solved by the author's model. Incorporating closed-circuit television allows for the monitoring of emergency vehicles and the assignment of passing priority on that route. This automated process eliminates the need for human intervention in such a situation. The author employed a modified version of the Yolov5 object detection algorithm in their study. You Only Look Once, or YOLO for short, is a grid-based object detection technique. The onus for object detection inside each grid cell falls on those cells individually. High-performance object detection makes use of YOLO models, which have 84 classes for detecting and differentiating 84 distinct objects. The author's approach is structured around four classes: firetrucks, ambulances, police cars, and regular cars. Impressively, the author's model has been able to recognize and identify several types of emergency vehicles. For police cars, the results were 98%, for fire trucks, 96%, for ambulances, 89%, and for normal cars, 97%. Future Tasks Restrictions - The findings of this study offer a solid groundwork for additional investigations into awareness and peripheral displays, as well as for other areas that require further investigation. Future work by the author will focus on developing a Sound Recognition System that can identify emergency vehicles based on their siren sounds. When an alarm goes off or an emergency vehicle is summoned, it will make a unique sound called the Siren Sound. In order to warn other motorists and pedestrians, emergency vehicles emit sirens when they carry out their duties.

[13] An innovative approach to improving urban traffic management is put forth by Santosh D. Pandure et al., who show how to prioritize traffic signals for emergency vehicles. Adaptive signal control, multi-modal data analysis, and deep learning all working together could reduce traffic jams and make emergency responses faster. Limitation\ In the Long Run: Practical implementation and system optimization for wider urban deployment will be the primary goals of future research.

[14] Research by Amrutasagar et al. When the YOLOv7 ambulance detection algorithm is put into action, accurate data regarding urban traffic can be used, which enhances emergency services even more. Accuracy of 1.00 at a confidence of 0.816, mAP of 0.925 at IoU of 0.5, and F1 score of 0.90 at a confidence level of 0.444 and higher demonstrate the model's best performance, which is to say that it identifies ambulances in traffic with few false alarms. Limitation\ Future plans - A YOLOv7 success story in ambulance detection has opened up new possibilities for the model's future use and improvement.

[15] In order to save patients' lives, Noor, A. et al. propose a new architecture for an ambulance detecting system that turns the traffic light green and prioritizes ambulance trucks. The system can

process and store reports of many kinds of signal detection in the cloud, regardless of their origin or format. For example, the author uses a YOLOv8 model to identify when an ambulance is approaching and then triggers the green lights to hasten the ambulance's predicted arrival at the hospital. To the author's knowledge, they are the first to use a YOLOv8 model for ambulance detection. The author gathered and annotated three thousand photographs of ambulances from ten nations, representing a wide range of domains and languages, in order to showcase the efficacy of her YOLOv8 model. Computer vision models trained and evaluated for cross-lingual tasks can benefit from this dataset. The author also ran 22 separate tests, 10 of which used pre-trained and 10 of which used non-pre-trained subjects. For each of the nations mentioned, the author ran two more trials that she refers to as the universal model. In addition, the author compared the YOLOv8 model's performance to that of other models, such as YOLOv5 and YOLOv7, that have been published in the literature. With average scores of 0.982 for accuracy, 0.976 for precision, 0.958 for recall, and 0.967 for F1 score, the experimental findings show that the YOLOv8 universal model is doing really well. Thanks to the massive amounts of data used for training, the universal model was able to gain a deeper understanding of the ambulance vehicles and achieved the top result among all countries. The author plans to modify the network layers to fit their datasets, leading to a model with great efficiency, which is a limitation regarding future scope. To top it all off, the authors intend to create a novel segmentation method called semantic segmentation. This will be the pixel-level analogue of object detection, which uses a square box to identify items.

[16] The suggested RTAIAED system by Mecocci, A. et al. demonstrates excellent accuracy and performance at a reasonable cost. Proving to have wide-ranging practical uses, it stands out as a flexible solution that can handle various real-life problems, especially in cases where alternative solutions are too costly or complicated, such handling short-term crises. The planned upgrades for the future will expand the system's capabilities and make it more compatible with other smart technologies. They will also show how intelligent transportation systems are improving and will have a big impact on optimizing emergency responses and traffic management in both urban and suburban areas. Limitation \ Looking ahead, - The development of RTAIAED, or Real-Time Ambulance in Emergency Detection, is expected to involve numerous important projects: To improve interoperability, it is recommended to start developing an API. This will allow the RTAIAED and other smart systems to share data easily. This will open the door for other types of smart apps to use the RTAIAED's real-time data, which will improve their capabilities and ability to respond to crises. The groundbreaking work of Barbosa et al. will serve as an inspiration for future endeavors aimed at enhancing the RTAIAED's capacity to integrate a broader array of emergency vehicles. The goal of this improvement is to improve the operational efficacy of different emergency services by creating a structured priority framework. • Managing Conventional Traffic Flows in an Experimental Manner: The exciting route for further exploration that RTAIAED's possible application in routine traffic management offers is still in the conceptual phase.

METHODOLOGY

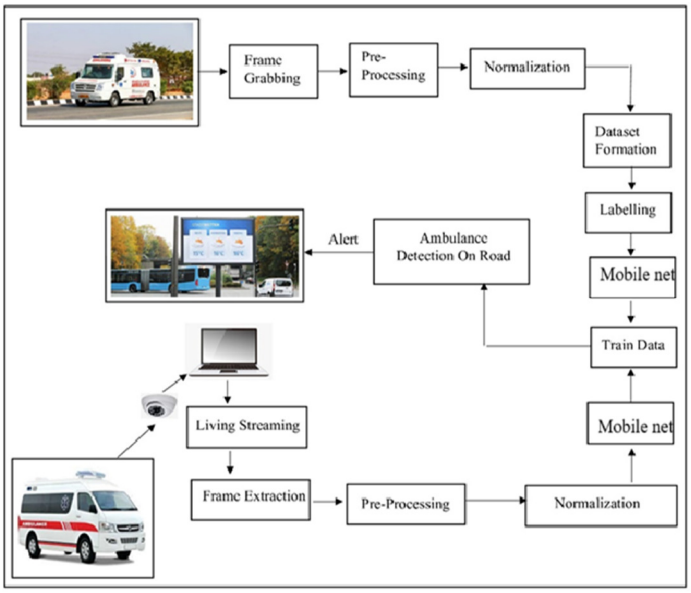

Figure 1: System overview Diagram

The method that has been suggested to establish a Ambulance Detection depicted in the system overview in Figure 1 up top. The suggested method was based in part on the execution of the procedures detailed below.

Step 1: Dataset Generator: Photographing the various automobiles on the road using OpenCV is the first step of the process. For the aim of detecting them during model testing, the Video Capture method inside the CV2 package takes pictures of the intended ambulance. A complete set of ambulance photos is housed in the dataset folder. Working on models such as the ambulance is the model's current focus. In preparation for the model's next stage, the acquired photos are saved in a folder.

Step 2: Data Labelling: Here, the labeling software is employed to assign labels to the photos that were collected in the previous stage. By importing the image into the labeling software, the user can indicate the coordinates of the top right corner and the bottom left to create x1, y1, and x2, y2 of the bottom right corner, respectively, of the rectangle. To train the model using a deep learning network called MobileNet, the acquired coordinates are saved in an.xml file.

Step 3: Installation of APIs: This Google Colab instance requires the TensorFlow Object Detection API, which must be installed first. To accomplish this, you'll need to execute a few installation instructions after cloning the [TensorFlow models repository](https://github.com/tensorflow/models). The following code portions can be run by clicking the play button. The next step in downloading and extracting cuDNN files is to install the conda environment. Next, grab the tensorflow models repository file from GitHub and clone it. After that, the Object Detection API was installed.

Step 4: Upload Image Dataset and Prepare Training Data: Here, we'll get the TensorFlow training data ready by uploading our training photos and running the scripts for TF Record generation. Once we have uploaded all of our photographs, we can divide them into separate folders for training, validation, and testing. Then, we can execute the scripts that will create TF Records from our data. To begin, on our local PC, create a single folder named "images.zip" and bundle all of our training images and XML files into it. Please ensure that the files are located inside the zip folder. After uploading, we need to execute certain commands to extract the zip file and configure our picture folders. In this particular instance, the file system's /content subdirectory is where these directories are generated. To access the file system, we can use the "Files" icon located on the left side of the screen.

Step 5: Split images into train, validation, and test folders : Locate our "images.zip" file among the listed ones; it's the folder icon on the left side of the screen. Unzipping the dataset and making folders to store the photographs are the next steps after uploading the dataset. In this particular instance, the file system's /content subdirectory is where these directories are generated. To access the file system, we can use the "Files" icon located on the left side of the screen.

The following step is to divide the photos into three sets: train, validation, and test. The purpose of each set is as follows: Ride the rails: The photos utilized

to train the model are these very ones. The neural network is fed a batch of photos from the "train" collection at each training stage. Objects in the photos are classified and their locations are predicted by the network. The loss is computed by the training method, which then uses back propagation to modify the network weights.

Validation: The training algorithm can utilize images from the "validation" set to verify how well training is going and to tweak hyperparameters (such as learning rate). These photos, in contrast to the "train" images, are only utilized on a periodic basis throughout the training process, specifically, once every specific number of steps. Warning: The neural network will never view these pictures while it is being trained. The ultimate testing of the model's accuracy is meant to be done by a human using these.

Step 6: Create TF Records : The last step is to transform the photos into TF Records, a data file format that TensorFlow uses for training. Automated data conversion to TF Record format is being accomplished by means of Python programs. We must first establish a class label map before we can run them. You can generate a class list in a "labelmap.txt" file using the code portion below. Substitute our classes for the 'class1' text, for instance "Ambulance," then insert a new line for each class. To run the code, hit the play button. The result is a "labelmap.txt" file that specifies which classes the object detection model should look for.

Step 7: Set Up Training Configuration : We are configuring the training of an SSD-MobileNet model in this section. Here, we're naming the TensorFlow 1 Object Detection Model that we'd like to employ as our starting point. An accompanying configuration file for each model specifies where to find certain files, allows the user to alter training parameters (such learning rate and total number of steps), and more. We are making changes to the custom training job's configuration file. In the first part of the code, you can see a list of models that are available in the TF1 Model Zoo. Following that, you can see the filenames that will be used to download the model and configuration file. Keeping track of which model is being used and adding more models in the future becomes much easier with this setup. Put the name of the model we want to use for training into the "chosen_model" field. The "ssd-mobilenet-v1-quantized" model is now selected to be used. After that, specify the parameters for the pre-trained model and the configuration file, and then press play in the following three steps to download them.

After getting the model and configuration files downloaded, the next step is to add some general training parameters to the configuration file. Training phases are controlled by the following variables: number of steps: The overall number of steps to train the model using. For an initial target, 40,000 steps is a solid choice. If, once training is complete, we see that the loss measures are still going down, we can add extra stages. Training takes longer as the number of stages increases. Additionally, training can be terminated before the designated number of steps if the loss level reaches a plateau. batch_size: The picture count for each training run. Although training a model with a higher batch size requires fewer steps, the size can only be as big as the GPU RAM that is available for training. It is generally recommended to use 16 GPUs for Colab instances.

(Used exclusively for training that takes quantization into account) quant_delay_steps The training algorithm will then imitate quantization by inserting "fake" quantization nodes into the network after these many steps. Halving the total number of training steps is a decent place to start. [This article](https://neuralet.com/article/quantization-of-tensorflow-object-detection-api-models/) has more details. At this stage, you also provide additional training data, such as the total number of classes, the location of the configuration file, and the file containing the pre-trained model. In order to apply the training parameters that we have just defined, we will need to edit the configuration file. In order to create our own "pipeline_file.config" file, the following code will automatically edit the

downloaded.config file to include our specific parameters. Combine the default pipeline file with our dataset, model checkpoint, and training settings to create a custom configuration file. The subsequent block updates the training script such that checkpoints are saved every 1000 steps. If we want to store checkpoints more or less frequently, we can change 'num_eval_steps' accordingly. You can view the mobile-network architecture in table 1

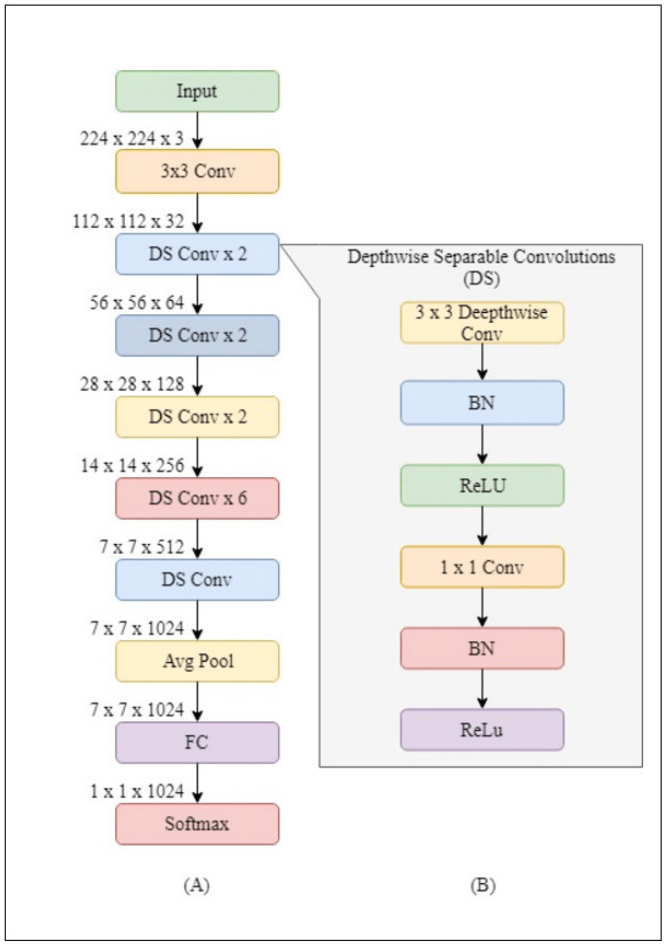

Table 1: Mobile-Net Body Architecture

Step 8: Train Custom TF1 Object Detector: Our object identification model training has begun! In order to train the model, the TF Object Detection API's "model_main_tf2.py" script is utilized. All of the arguments and parameters needed by'model_main_tf2.py' have previously been defined in earlier portions of this Colab. Time required for training can range from two to six hours, with exact timing dependent on model, batch size, and total number of steps. For the purpose of running model tests, this will generate a.tflite file.

Step 9: Testing the model for Ambulance detection : In this stage, the Python software uses the Droid Cam app, which is compatible with both laptops and mobile phones, to record video and, by extension, frames from the mobile phone's camera. A file containing the trained model. In order to identify the ambulance in the frames of the live broadcast, tflite is employed. The digital board will show the presence of the ambulance as soon as it is detected in the traffic.

III. RESULTS AND DISCUSSIONS

The proposed method for Ambulance Detection in traffic and alerting System through Deep learning and IOT.was developed using the Anaconda framework, Python, and the Spyder IDE. The development computer has 1 terabyte of secondary memory and 8 gigabytes of main RAM. A number of factors have been considered in order to determine how feasible the proposed plan is. In this part, we detail the results of the experimental study.

The obtained results for confusion matrix are depicted below in the following figures.

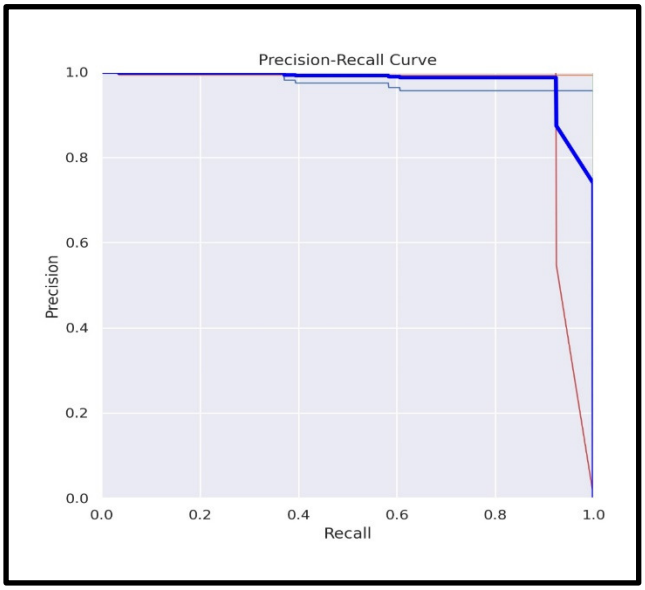

Figure 2: Precision-Recall Curve

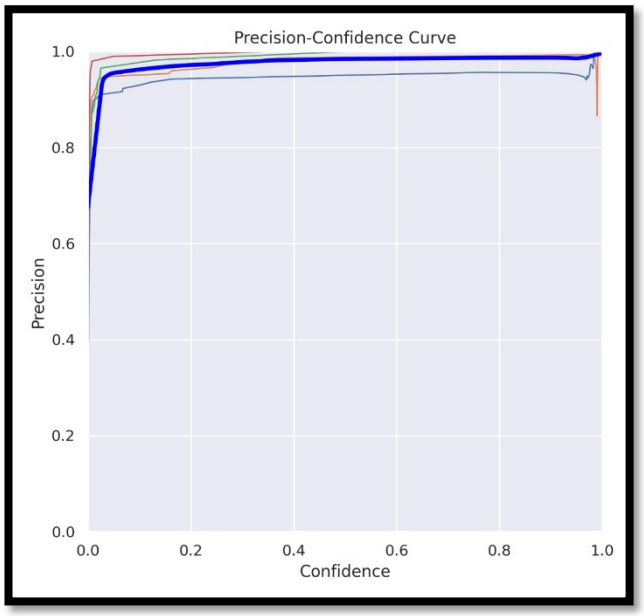

Figure 3:Precision-Confidance Curve

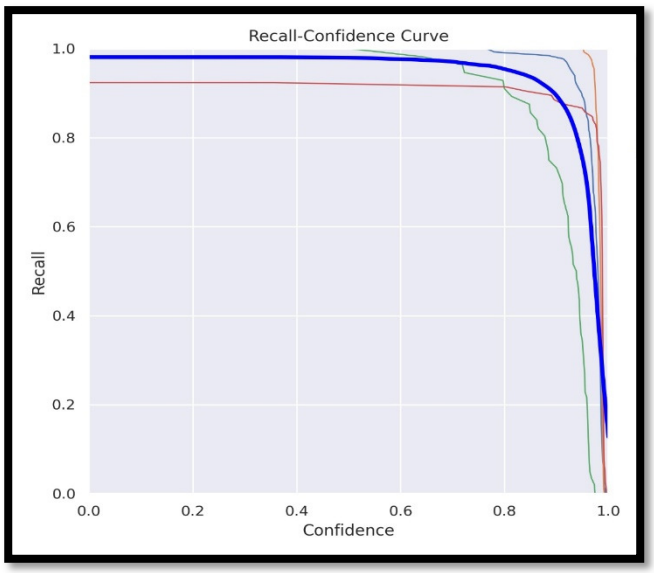

Figure 4: Recall-Confidence Curve

IV. CONCLUSION AND FUTURE SCOPE

Gather data in a way that the program can use it for testing and training. The application made use of the design that was made to distinguish the fire engine and ambulance from the majority of other vehicles. To help cars and four-wheelers find the best way to get to hospitals quickly, the proposed system tracks their locations at certain intersections and uses Res-Net deep learning, an AI model, to identify when they're getting close. When an emergency vehicle's siren is heard, a digital board will display an alarm message at every intersection that is about to be crossed.

REFERENCES

[1] Selene Cerna, Héber H. Arcolezi, Christophe Guyeux, Guillaume Royer-Fey, Céline Chevallier, Machine learning-based forecasting of firemen ambulances' turnaround time in hospitals, considering the COVID-19 impact, Applied Soft Computing, Volume 109, 2021, 107561, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2021.107561.

[2] S. Rafiq and M. A. Khanum (2021) Review on Minimization of Ambulance Response Time Using Image Processing and Critical path Mapping Based on Traffic Control, Vol. 02, Iss. 02, S. No. 002, pp. 1 7, 2021. https://doi.org/10.54060/JIEEE/002.02.002

[3] Patel, R., Mange, S., Mulik, S. et al. AI based emergency vehicle priority system. CCF Trans. Pervasive Comp. Interact. 4, 285–297 (2022). https://doi.org/10.1007/s42486-022-00093-7

[4] Shubham Gade , Vinayak Jadhav , Rohit N Galande and Dr. Swapnil Pokharkar, Optimizing Renewable energy harvesting process for EV Charging Stations through machine learning , IJNRD Vol.10, Issue 1, page no.b226-b237, January-2025 ISSN:2456-4184

[5] KHERRAKI, Amine; EL OUAZZANI, Rajae. Deep convolutional neural networks architecture for an efficient emergency vehicle classification in real-time traffic monitoring. IAES International Journal of Artificial Intelligence (IJ-AI), [S.l.], v. 11, n. 1, p. 110-120, mar. 2022. ISSN 2252-8938. Available at:

<https://ijai.iaescore.com/index.php/IJAI/article/view/21104>,doi:http://doi.org/10.11591/ijai.v11.i1.pp110-120.

[6] Sunil M, V Yashaswini Naidu, Vignesh R, Vishwas P, Amitha S, " SMART TRAFFIC MANAGEMENT FOR AMBULANCE, " Volume:04/Issue:12/December-2022 Impact Factor- 6.752 www.irjmets.com

[7] Usaid, M., Muhammad Asif, Tabarka Rajab, Munaf Rashid, & Syeda Iqra Hassan. (2022). Ambulance Siren Detection using Artificial Intelligence in Urban Scenarios. Sir Syed University Research Journal of Engineering & Technology, 12(1), 92–97. Retrieved from https://sirsyeduniversity.edu.pk/ssurj/rj/index.php/ssurj/article/view/467

[8] Bhoomika G M et al., "Ambulance Detection using Image Processing ," DOI 10.48175/IJARSCT-5667

[9]Mahmud, Umar, Hussain, Shariq, Sarwar, Amber, Toure, Ibrahima Kalil, A Distributed Emergency Vehicle Transit System Using Artificial Intelligence of Things (DEVeTS-AIoT), Wireless Communications and Mobile Computing, 2022, 9654858, 12 pages, 2022. https://doi.org/10.1155/2022/9654858

[10] Yarra Kavitha, Penke Satyanarayana, Shafi Shahsavar Mirza, Sensor based traffic signal pre-emption for emergency vehicles using efficient short-range communication network, Measurement: Sensors, Volume 28, 2023, 100830, ISSN 2665-9174, https://doi.org/10.1016/j.measen.2023.100830.

[11] Dr. P. Sankar Babu, K. Meenendranath Reddy, P. Naga Timmaiah, & V. Srikanth. (2023). Intelligents Traffic Light Controller for Ambulance. Journal of Image Processing and Intelligent Remote Sensing, 3(04), 19–26. https://doi.org/10.55529/jipirs.34.19.26

[12] Rawand Sulayman, Salih Rajab and Bawer Kareem. Emergency Vehicle Detection with Computer Vision. ScienceOpen Preprints. 2023. DOI: 10.14293/PR2199.000508.v1

[13] Intelligent Traffic Signal Prioritization for Emergency Vehicle Diversion in Urban Environments using Multi-modal Deep Learning - Santosh D. Pandure, Pravin L. Yannawar - IJFMR Volume 6, Issue 4, July-August 2024. DOI 10.36948/ijfmr.2024.v06i04.23981

[14] Amrutasagar, K.; Manoj, Pera; Divya, Morla; Mahesh Babu, Meethukulla; Gangotri, Lavudiya, "Enhanced Traffic Signal Adaptation with Ambulance Identification and Distance Computation," International Journal of Computing and Digital Systems ISSN (2210-142X) Int. J. Com. Dig. Sys. #, No.1, 1-10 (March-2024)

[15] Noor, A.; Algrafi, Z.; Alharbi, B.; Noor, T.H.; Alsaeedi, A.; Alluhaibi, R.; Alwateer, M. A Cloud-Based Ambulance Detection System Using YOLOv8 for Minimizing Ambulance Response Time. Appl. Sci. 2024, 14, 2555. https://doi.org/10.3390/app14062555

[16] Mecocci, A.; Grassi, C. RTAIAED: A Real-Time Ambulance in an Emergency Detector with a Pyramidal Part-Based Model Composed of MFCCsandYOLOv8. Sensors 2024, 24, 2321. https:// doi.org/10.3390/s24072321

*****

