Group 9: ACCIDENT DETECTION AND ALERTING SYSTEM

Rahul Aggarwal Akash Shriwas Mohit Singh

Komal Yadav Abhinav Kuruma

22111403, 22111006, 22111402, 22111031, 22111401

{rahulag22, ashriwas22, mohitsingh22, komaly22,akuruma22}@iitk.ac.in

Indian Institute of Technology Kanpur (IIT Kanpur)

Abstract

According to worldwide statistics, traffic accidents are the cause of a high percentage of violent deaths. The time taken to send the medical response to the accident site is largely affected by the human factor and correlates with survival probability. As per NCRB's ADSI-2021Statistics report, the total number of accidental deaths in the country was 3.97 lakh in 2021. The number of deaths due to road accidents in 2021 was 1.56 lakh compared to 1.55 lakh in 2019 and 1.33 lakh in 2020. Most of the deaths which were happened due to accidents were because of not knowing the information to the health center immediately and so in our project we are detecting the accident after it occurs and sending the details to their respective family members so that they can take the action immediately and may save a life. In this paper convolution-Neural network is used to train the model whether an accident has taken place or not.

1 Introduction

According to statistics in India, there is an average of 13 deaths per hour, that is, 140,000 deaths per year. Nowadays road accidents are the main reason for losing lives every day. Late response time from the health centers is the main cause of it. An effective road accident detection and information communication system is required in respect of saving injured persons So Here, we have done an architecture such that it detects the accident when occurred and within seconds information is sent to nearby health care and authorities can save them who are involved in an accident. Nowadays, Deep Learning -based approaches have shown high performance in computer vision tasks that involve the relationship of a complex feature. Therefore, this work develops an automated DL-based method capable of detecting traffic accidents on video. The main goal is to use convolutional neural networks to classify accidents and non-accident and as the CNN algorithm is very accurate in image classification we can undoubtedly go for this architecture. The accidents occurred mainly because of the following reasons:

1. geometry of the road
2. climate of the area
3. Drunken drivers
4. Speeding and overtaking
5. Weather conditions
6. Red light jumping

Using more layers in deep learning we will be able to classify accurately. But the problem with CNN is that it needs more and more data and due to this curse of dimensionality problems will arise

But CNN solves that too. Several convolution layers help to extract relevant visual features within the same data set. Recurrent neural networks are also involved so that they can store previous process data and can have a temporal relationship. Existing methods detect accidents by analyzing sound data or video data. They use machine learning/deep learning techniques, such as support vector machines, Gaussian mixture models, and learning vector quantization to determine accidents.

Our main motive is that we trained the model with CNN and when an accident is predicted then we send the message to nearby health centers automatically so that they can take the necessary action.

The rest of the document is organized as follows:
2. Related work which contains a literature review of previous methods and their ideas.
3. proposed idea which we discuss the current idea we have implemented.
4. Methodology which contains clear details of our idea, data set, learning algorithms, and implementation details of the project.
5. Results contains accuracy and other details which we obtained.
6. Discussion and future work contains the work which can be added in the future to improve accuracy and other features.

2 Related Work

Ghosh et al.[1] did a brief work on predicting the accident by using machine learning models. They include various deep learning algorithms with smartphones, GSM and GPS technologies, vehicular ad-hoc networking, and on-route mobile applications. The methods timely inform emergency services about the accident area, in the process of decreasing the death rate due to accidents and saving precious lives. In such methods, there are some limitations like low reliability, less accuracy, and hardware malfunctions. So, an opportunity is open to work on efficient accident detection methods.

Jae Gyeong Choi et al.[2] proposed a car crash detection system, based on ensemble deep learning and multimodal data from dashboard cameras, for an emergency road call service that recognizes traffic accidents automatically and allows immediate rescue after transmission to emergency recovery agencies. In particular, the proposed car crash detection system uses video data from dashboard cameras and two different types of audio data from dashboard cameras, audio features, and spectrogram images, in order to improve the per- performance of car crash detection.

Robles-Serrano et al. [**?**]proposed based on techniques used in video analytics. In particular, deep learning neural network architectures trained to detect the occurrence of a traffic accident are used. They have used neural networks by taking the input of a video but the problem is that they have not made the dataset at different places like hilly areas, night, and such weird cases, and as CNN uses lots of data to get good accuracy they have to have more and more data but they have used frequent places that too in china road and areas. and they are getting an accuracy of 98% but the dataset they used was humungous and for training the model they have used large GPUs such that they can train those many datasets.

Kaplan et al. [3]did the research to detect accidents if occurred and inform the respective authority or person through wireless technologies automatically. The proposed system detects the casualties by utilizing several sensors to sense the physical changes due to the accidents. This system is used to recognize the location of the accident with the help of a GPS sensor and transmit the coordinates to a backup team using the GSM module.

3 Proposed Idea

NOTE: For our convenience, we have used QR codes to read data since our remote control cars are very small and it is difficult to read the number plate with the camera we had . Normally, we can use the car number plate with the OCR recognition model.

1. We have made an ML model to predict whether an accident has occurred or not from the video that we will get from our surveillance camera.

2. After prediction, if our model predicts it as a non-accident, then we will do nothing, but if the prediction is yes, then we will first collect the QR codes of all the cars present in the frame and store data in the list, and then we try to recollect the data after 2 seconds and match it with previous data we have stored. Now, if any data matches the previous data, it means there are high chances of an accident.

3. Collecting the data after 2 seconds also confirms that an accident has occurred, or if there is a minor accident, that the case will be eliminated by itself.

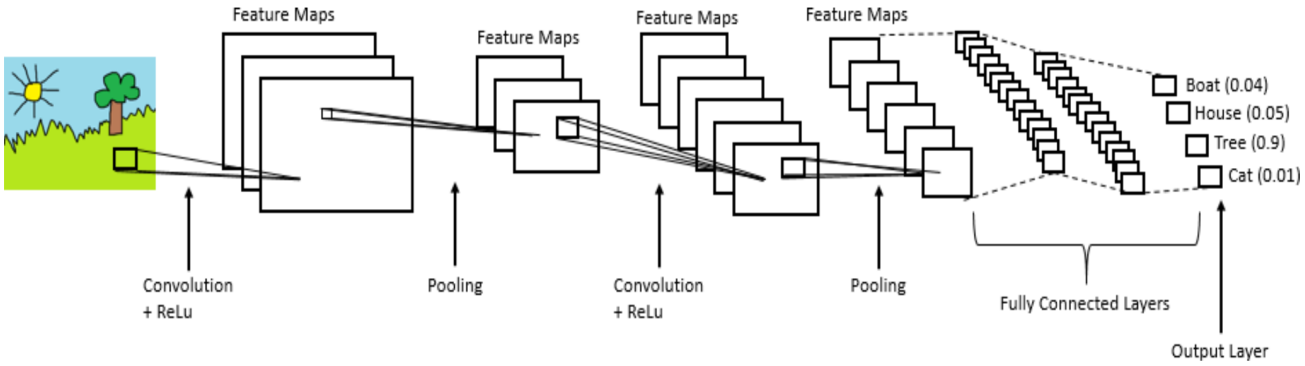
4. We will send an alert message to the user's emergency number and to the ambulance along with the accident screenshot link.

4 Methodology

4.1 The data sets that we have used

We have merged 2 datsets. One is our own. And another is taken from kaggle.

4.2 Pre-processing done on the data and exploratory data analysis



We have used teras *image_dataset_from_directory* function. And following attributes:

- image_size = (,), to resize the images.
- batch_size = ' ', which defines the number of frames propagated through the network at a time.
- Seed = ' ', to reproduce same training/ validation dataset while we use different hyperparameters.

We did search extensively for datasets but was unable to find them. Especially, one that fits our toys car (as objects).

4.3 Details of different machine learning algorithms used

We have used mainly two Machine Learning algorithms:

1. Multi-layer Perceptron Classifier (MLPClassifier)
2. VGG-19 which used database ImageNet

With [1] we trained our datset on image-size which was (200, 200) and batch-size was 10. In a CNN, each layer has two kinds of parameters : weights and biases. The total number of parameters is just the sum of all weights and biases. Which can be calculated using below given equation:

$$W_C = (K^2) * C * N$$
$$B_C = N$$
$$P_C = W_C + B_C$$

where:
$W_c = Numberofweightsoftheconvlayer.$
$B_c = Numberofbiasesoftheconvlayer.$
$P_c = Numberofparametersoftheconvlayer.$
$K = Size(width)ofkernelsusedintheconvlayer.$
$N = Numberofkernels.$
$C = Numberofchannelsoftheinputimage.$
For example in first layer $K = 16, C = 3, N = 96(default)$. Hence we can calculate total numbers of parameters.

Our architecture is a 5 layered model which looks like:

```
Model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.BatchNormalization(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(128, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(256, activation='relu'),
  layers.Dense(len(class_names), activation= 'softmax')
])
```

And with [2] parameters was provided using ImageNet data. Which are in millions.
We trained our dataset on image-size which was (480, 640) and batch-size was 5. Our architecture looks like:

```
vgg = VGG19(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
for layer in vgg.layers:
    layer.trainable = False
x = Flatten()(vgg.output)
prediction = Dense(3, activation='softmax')(x)
model = Model(inputs=vgg.input, outputs=prediction)
```

Loss function was same for both the models i.e sparse categorical crossentropy , which is shown below:

```
model.compile(
  loss='sparse_categorical_crossentropy',
  optimizer="adam",
  metrics=['accuracy']
)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 360, 480, 3)	0
conv2d (Conv2D)	(None, 360, 480, 16)	448
batch_normalization (BatchNormalization)	(None, 360, 480, 16)	64
conv2d_1 (Conv2D)	(None, 360, 480, 32)	4640
max_pooling2d (MaxPooling2D)	(None, 180, 240, 32)	0
conv2d_2 (Conv2D)	(None, 180, 240, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 90, 120, 64)	0
conv2d_3 (Conv2D)	(None, 90, 120, 128)	73856
max_pooling2d_2 (MaxPooling2D)	(None, 45, 60, 128)	0
flatten (Flatten)	(None, 345600)	0
dense (Dense)	(None, 256)	88473856
dense_1 (Dense)	(None, 2)	514

Total params: 88,571,874
Trainable params: 88,571,842
Non-trainable params: 32

(a) CNN

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 640, 480, 3)]	0
block1_conv1 (Conv2D)	(None, 640, 480, 64)	1792
block1_conv2 (Conv2D)	(None, 640, 480, 64)	36928
block1_pool (MaxPooling2D)	(None, 320, 240, 64)	0
block2_conv1 (Conv2D)	(None, 320, 240, 128)	73856
block2_conv2 (Conv2D)	(None, 320, 240, 128)	147584
block2_pool (MaxPooling2D)	(None, 160, 120, 128)	0
block3_conv1 (Conv2D)	(None, 160, 120, 256)	295168
block3_conv2 (Conv2D)	(None, 160, 120, 256)	590080
block3_conv3 (Conv2D)	(None, 160, 120, 256)	590080
block3_conv4 (Conv2D)	(None, 160, 120, 256)	590080
block3_pool (MaxPooling2D)	(None, 80, 60, 256)	0
block4_conv1 (Conv2D)	(None, 80, 60, 512)	1180160
block4_conv2 (Conv2D)	(None, 80, 60, 512)	2359808
block4_conv3 (Conv2D)	(None, 80, 60, 512)	2359808
block4_conv4 (Conv2D)	(None, 80, 60, 512)	2359808
block4_pool (MaxPooling2D)	(None, 40, 30, 512)	0
block5_conv1 (Conv2D)	(None, 40, 30, 512)	2359808
block5_conv2 (Conv2D)	(None, 40, 30, 512)	2359808
block5_conv3 (Conv2D)	(None, 40, 30, 512)	2359808
block5_conv4 (Conv2D)	(None, 40, 30, 512)	2359808
block5_pool (MaxPooling2D)	(None, 20, 15, 512)	0
flatten (Flatten)	(None, 153600)	0
dense (Dense)	(None, 3)	460803

Total params: 20,485,187
Trainable params: 460,803
Non-trainable params: 20,024,384

(b) $VGG-19$

Figure 1: Models Summary

4.4 Details on the experimental setup we have used for the project.

We have used a camera. More specifically a Amazon Basics 1080p webcam with full HD resolution. And we pasted QR code on top of the robot control cars.

4.5 Implementation details of our project

And we are making them crash in the frame of camera. While cars are in frame we are capturing the QR code on them and giving it to our model . Our model then waits for 2 secs and take the QR codes of car now present in the frame. It then matches the QR codes and if the codes matches it declare that there's been an accident.

5 Discussion and Future Work

The results of the proposed solution show that using a QR code for re-confirmation of accident and non-accident cases is not feasible because it cannot be implemented in a real-world scenario. It was considered in the proposed solution because the used vehicles in this case were small robotic cars and the configuration of the used camera was not up to the mark as per the requirements, so it was difficult to detect the number plate of the vehicle. Also, the data set used for the training of the multilayer perceptron model wasn't enough because it was prepared manually using the robotic cars.

The proposed solution can be implemented in the real world, as in the real scenario, the size of the real vehicle is good enough compared to robotic cars so that the number plate can be detected with the camera having a good configuration instead of using the QR code for the confirmation of whether an accident happened or not after getting the prediction result from the model, and also, the data set for training the model is easily available in a huge amount.

The alerting system can benefit from further improvements. A Python library, PyWhatKit, is used in the proposed solution for alerting about the accident via WhatsApp, which is not suitable for cases where the internet is not available to get a notification at the right time and is obviously not feasible for real-world scenarios. It is feasible to use cellular networks to send an alert message in the event of an accident. The MLP model used in the solution gives an accuracy of around 80, which can also be improved further. To enhance this solution further, one more important thing can be done: instead of training the model with the data set consisting of images, the data set consisting of videos can be used, but for the implementation of the same, a highly configured system should be used.

6 Results

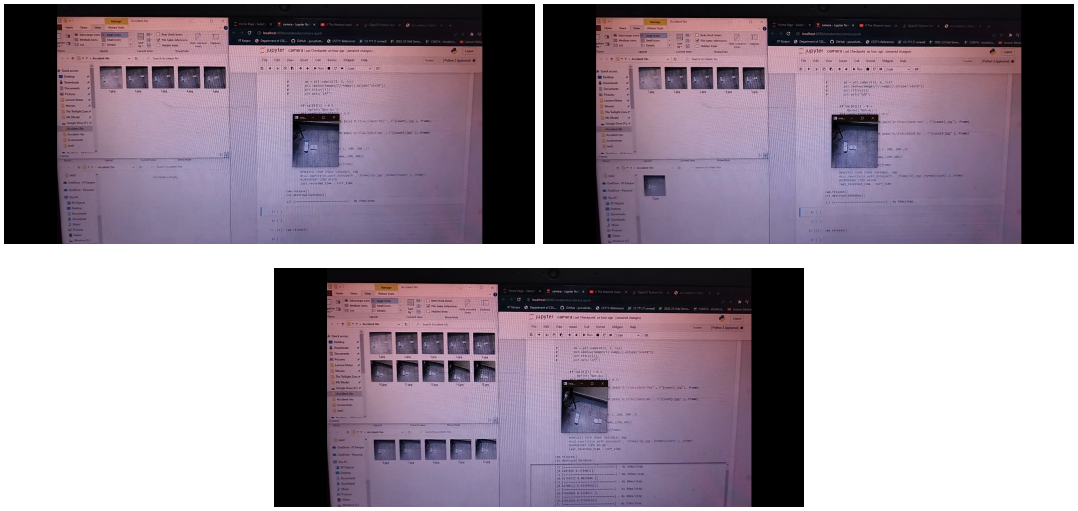


Figure 2: Accident Detection

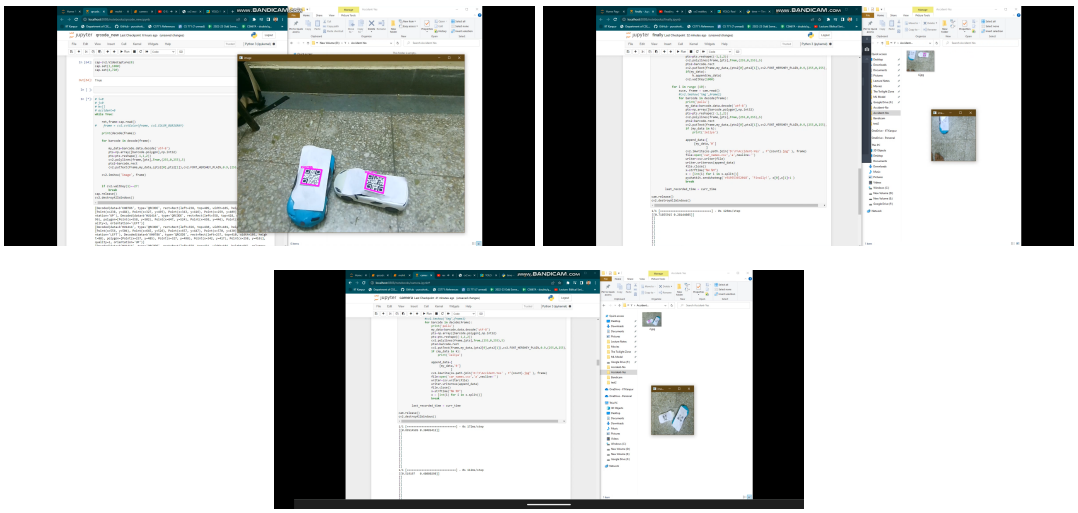


Figure 3: QR Code reading

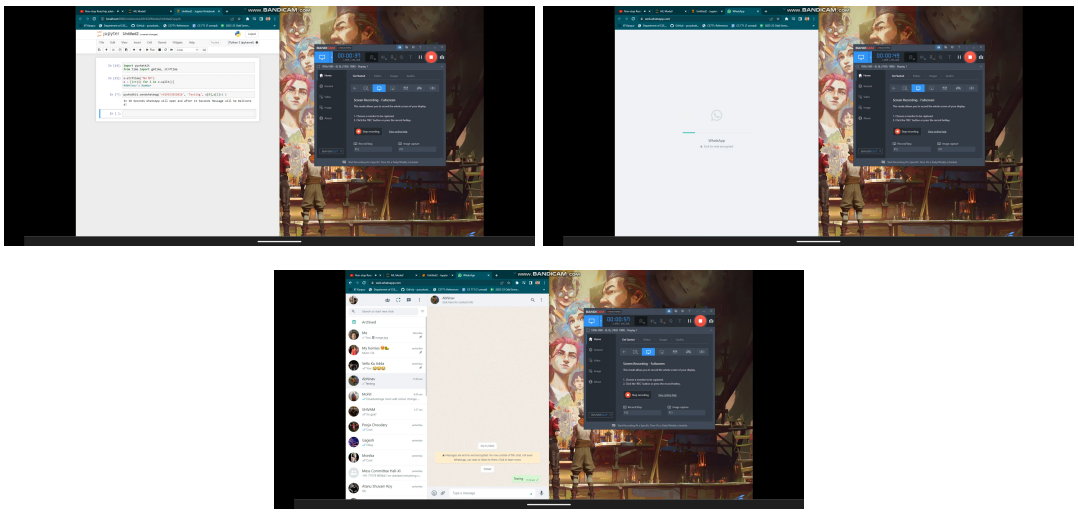


Figure 4: Message Sending

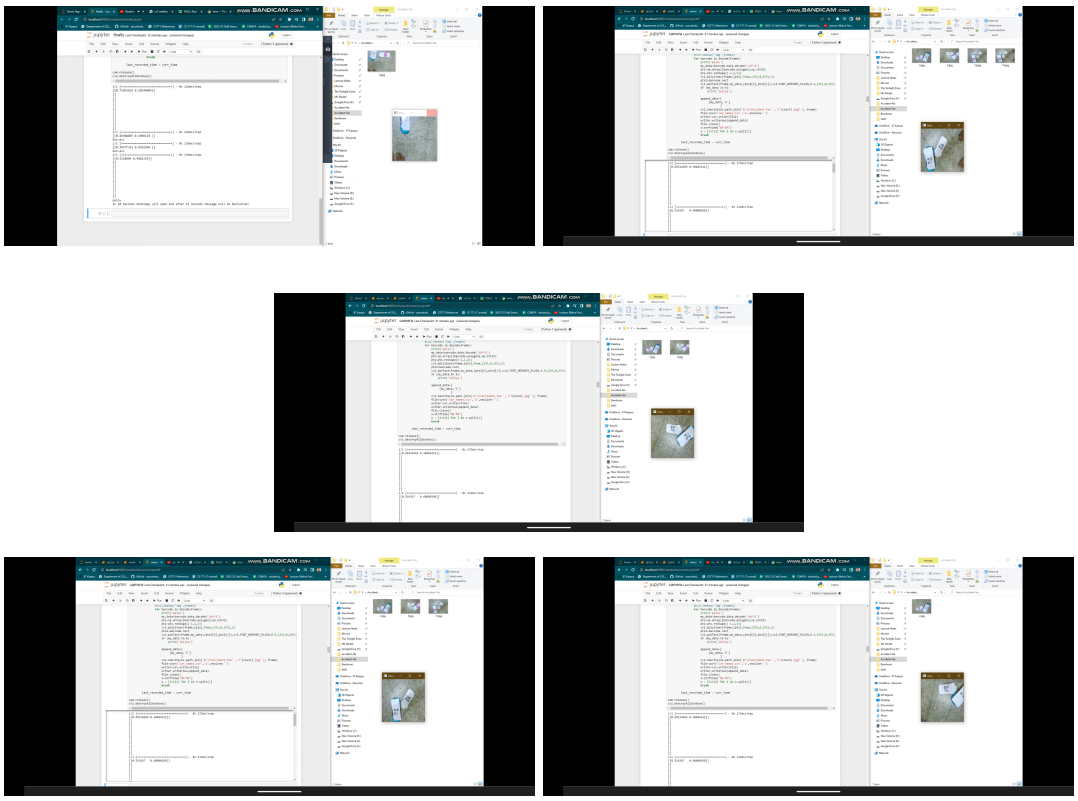


Figure 5: Final Prediction

7 Conclusion

We discovered that the CNN model, i.e., MLPCLassifier, has the highest accuracy of 80% while testing our model on different machine learning algorithms. Our proposed idea is good enough to be implemented in real time for accident detection and sending a notification for the same with the usage of real-time data sets for training the model and number plate detection instead of QR codes, as in a real-life scenario the used camera will be of high configuration and the number plate can be detected easily, which will be helpful for confirming that an accident has happened and also for sending an alert to an associated contact number with a number plate of a particular vehicle. The proposed solution can play a great role in reducing the number of deaths that happen in accidents due to the lack of treatments at the required time.

8 Individual Contributions

Team Automation		
Team Members	Work Done	Contribution
Mohit Singh	Idea proposal. And setting up the camera and reading of QR Code and setting up the resolution of camera. And doing simulations.	20%
Akash Shriwas	Helped in creating our own datset. And trained the model on VGG-19.	20%
Ahinav Kuruma	Fishing dataset from internet. Prepared and presented PPTs and report. Changing the batch size and re-training the model.	20%
Rahul Aggarwal	Helped in training the model on CNN (MLPClassifier). And writing the code and doing simulations.	20%
Komal Yadav	Helped in creating our own dataset. And helped in obtaining API to transfer message to emergency contact	20%

9 Our GitHub Link: GitHub

References

[1] S. Ghosh, S. J. Sunny, and R. Roney, "Accident detection using convolutional neural networks," in *2019 International Conference on Data Science and Communication (IconDSC)*, pp. 1–6, 2019.

[2] J. G. Choi, C. W. Kong, G. Kim, and S. Lim, "Car crash detection using ensemble deep learning and multimodal data from dashboard cameras," vol. 183, p. 115400, 2021.

[3] K. Kapilan, S. Bandara, and T. Dammalage, "Vehicle accident detection and warning system for sri lanka using gnss technology," in *2020 International Conference on Image Processing and Robotics (ICIP)*, pp. 1–5, 2020.