VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

TARP Project Review

Topic - Detecting Road Accidents using Deep Learning for Road Safety Enhancement

Submitted To:

Dr. Ramani S

Team Members:

20BCE0943 - Lokesh Reddy B

20BCE0959 – Mahesh Reddy

20BCE0967 – Jayanth T

Abstract:

Traffic accidents pose a significant global threat, contributing to a substantial number of fatalities each year. Timely access to medical assistance at accident scenes is crucial in improving the chances of survival. To address this urgent issue, computer vision researchers have been exploring automated methods for the detection of traffic accidents in video footage. Deep learning, a powerful subset of artificial intelligence, has demonstrated remarkable effectiveness in handling intricate computer vision tasks. In this study, we introduce an automated deep learning-based approach for the detection of traffic accidents in video streams. Our approach is grounded in the assumption that traffic accidents can be characterized by specific visual features that manifest temporally within video sequences. To achieve this, our model is structured into two distinct phases: visual feature extraction and temporal pattern identification. During the training phase, the model learns to recognize both the visual cues that define accidents and the temporal patterns that they exhibit. This comprehensive learning process leverages custom-built datasets designed specifically for this study, as well as publicly available datasets, to ensure the model's robustness and generalization capabilities.

This research endeavours to enhance road safety by harnessing the potential of deep learning to automatically detect traffic accidents in video footage. The proposed method offers a promising avenue to facilitate the swift deployment of emergency medical assistance, ultimately contributing to the reduction of accident-related fatalities and improving overall road safety.

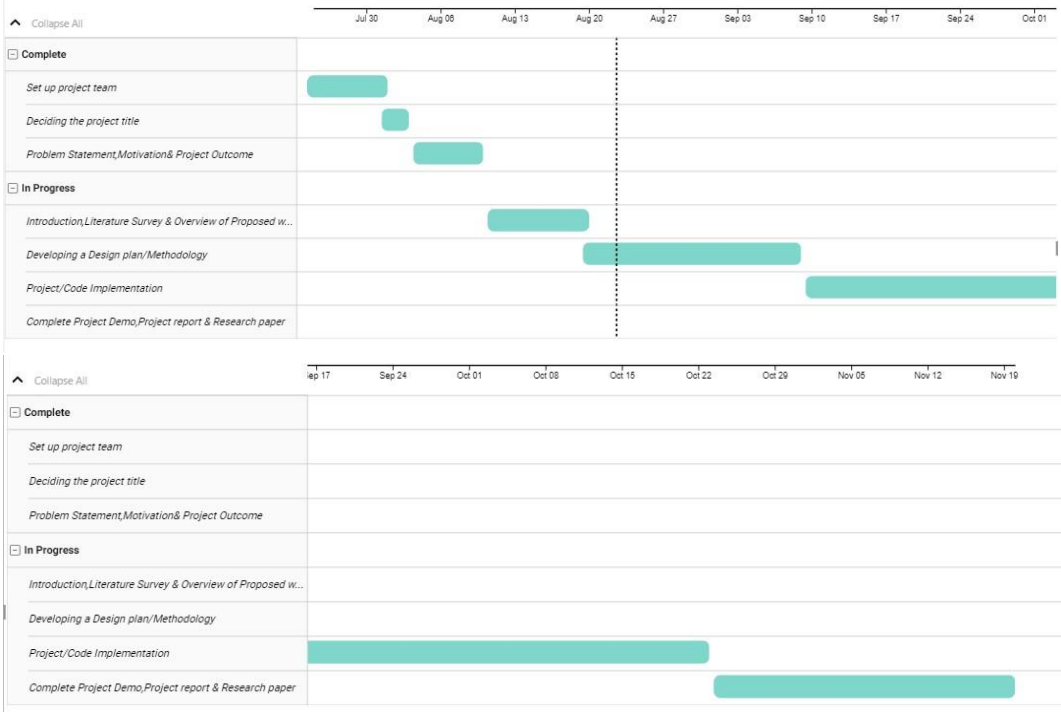
Description:

Traffic accidents are a leading cause of death worldwide. The timely arrival of medical assistance at accident sites can significantly improve the chances of survival. To address this, computer vision researchers are developing automated methods for detecting traffic accidents in videos. Deep learning (DL) has proven to be highly effective in complex computer vision tasks. In this study, we propose an automated DL-based method for detecting traffic accidents in videos. The approach assumes that traffic accidents can be described by visual features that occur temporally. The model architecture involves two main phases: visual feature extraction and temporal pattern identification.

The model learns both visual and temporal features during the training phase, using both custom-built datasets and publicly available ones.

Gantt chart:

Task Name	Status	Start Date	End Date
Set up project team	Complete	08/01/23	8/5/23
Deciding the project title	Complete	08/05/23	8/6/23
Problem Statement,Motivation& Project Outcome	Complete	08/07/23	8/10/23
Introduction,Literature Survey & Overview of Proposed work	In Progress	08/11/23	8/19/23
Developing a Design plan/Methodology	In Progress	08/19/23	9/9/23
Project/Code Implementation	In Progress	09/10/23	10/23/23
Complete Project Demo,Project report & Research paper	In Progress	10/24/23	11/20/23



Introduction:

Visual idioms are a powerful tool for understanding and communicating complex data and information. They are used to represent data and information in a visual format that is easy to understand and interpret. In recent years, there has been a growing interest in using visual idioms to analyse road accidents in India. This is since road accidents are a major problem in India, causing thousands of deaths and injuries each year. In this project, we will use a Deep learning model to analyse road accidents in India. Analysis of

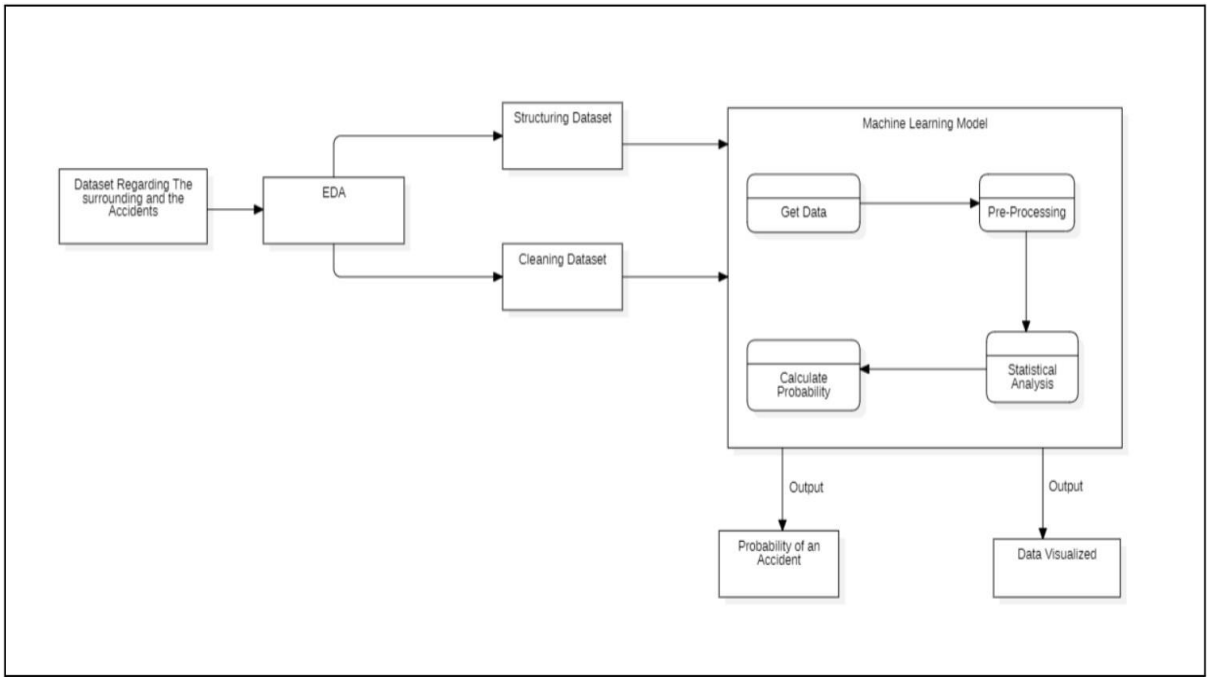
Road Traffic Casualties Using DV Techniques" With an exponential rise in population and growth in technology, most people now have access to the luxury of mobility.
The amount of traffic on the road is increasing every day, and so are the number of accidents. People do not take the necessary precautions and do not obey the guidelines. Furthermore, most of the time, these accidents are not reported to the appropriate authorities within the critical period when the victim's lives can be saved. As a result, the number of fatalities in car accidents has increased. Through the Analysis of Road Traffic Casualties, we aim to apply statistical analysis and data visualization algorithms on the Car Accident dataset to address the above problem.

Literature survey:

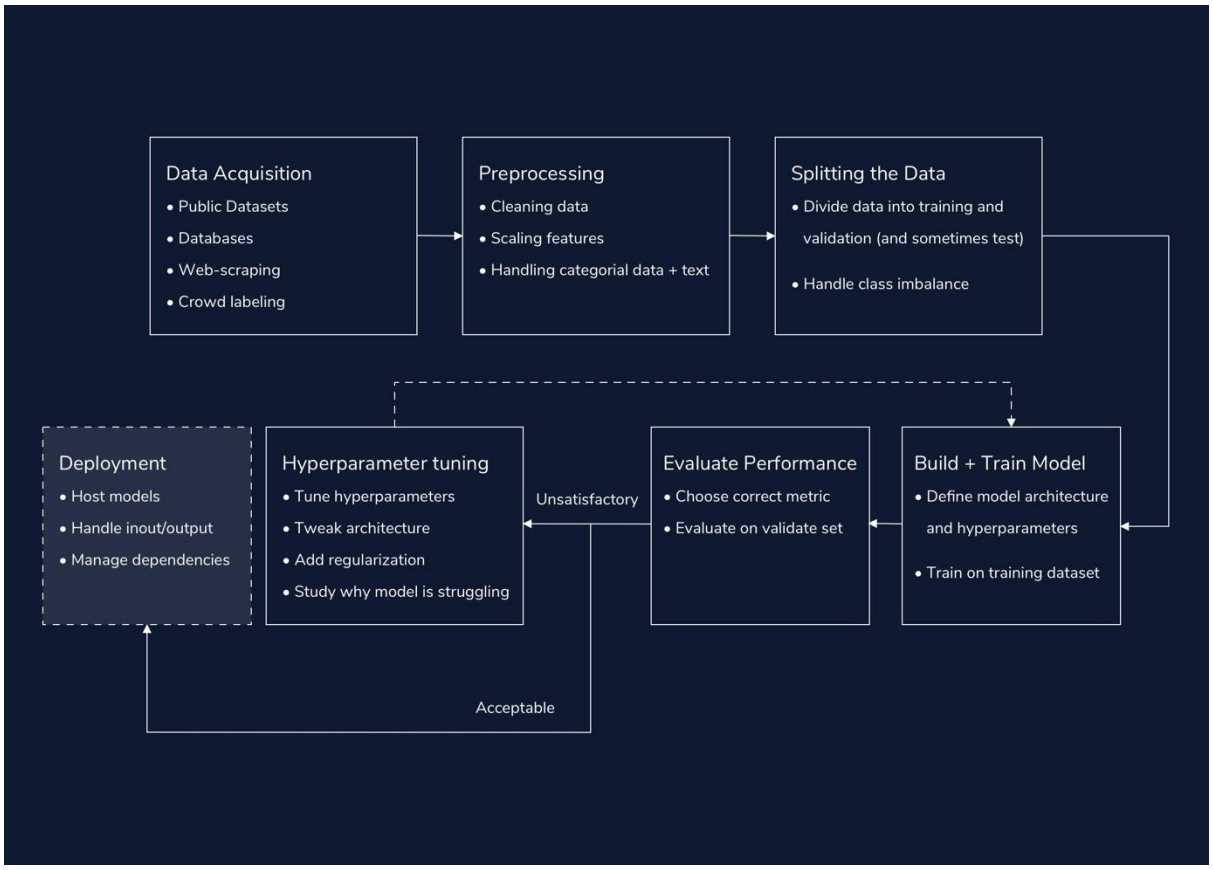
Serial No.	Authors and Year (Reference)	Title (Study)	Concept/ Framework	Conclusion/ Result	Limitations/ Future Research/ Gaps Identified
1.	Tiago Tamagusko, Matheus Gomes Correia , MinhAnh Huynh,Adelino Ferreira	Deep Learning applied to Road Accident Detection with Transfer Learning and Synthetic Images	This paper introduces an AI-driven solution for road accident detection using computer vision. The focus is on real-time monitoring using surveillance cameras and deep learning. Transfer learning with Convolutional Neural Networks, particularly EfficientNetB1 and MobileNetV2, is employed for accurate accident detection, even utilizing synthetic accident images due to their rarity.	The proposed solution leverages deep learning and transfer learning techniques to automatically detect road accidents from surveillance camera data. EfficientNetB1 and MobileNetV2 show promising results, achieving Mean Average Precision (MAP) values of 0.89, 0.88 and Matthews Correlation Coefficient (MCC) values of 0.77, 0.71, respectively, thereby contributing to enhanced road safety.	A challenge lies in acquiring real accident images for mode training. Synthetic images are used as a workaround. Future research could focus on refining the model's performance with a more diverse set of real accident images. Additionally, exploring real-time deployment, adapting to varied road conditions, and considering model robustness against environmental factors are areas worth investigating.
2.	Bulbula Kussia,Fengli Zhang,Ariyo Oluwasanmi,Forster Owusu	Vehicle Accident and Traffic Classification Using Deep Convolutional Neural Networks	The study addresses road traffic accidents' significance and economic impact. It explores deep learning for categorizing accident and traffic images, utilizing supervised domain knowledge for feature extraction through a deep Convolutional Neural Network (CNN). The CNN model effectively classifies images into predefined classes, enhancing road safety efforts.	The research employs deep CNN techniques to categorize traffic and accident images into predefined classes with high accuracy (94.4%). The model demonstrates effective generalization to real-world datasets, showcasing its potential in accurate accident and traffic classification, thus contributing to road safety initiatives.	While achieving impressive accuracy, the model's limitations might include potential difficulties in handling highly diverse or nuanced scenarios. Future research could focus on expanding the dataset to include more diverse conditions, addressing model robustness to various lighting and weather conditions, and exploring real-time implementation challenges.
3.	Miao Chong, Ajith Abraham and Marcin Paprzycki	Traffic Accident Analysis Using Machine Learning Paradigms	This paper investigates application of neural networks, decision trees and a hybrid combination of decision tree and neural network to build models that could predict injury severity.	The classification accuracy obtained in the experiments reveals that, for the non-incapacitating injury, the incapacitating injury, and the fatal injury classes, the hybrid approach performed better than neural network, decision trees and support vector machines.	The dataset doesn't provide enough information on the actual speed since speed for 67.68% of the data records' was unknown. If the speed was available, it is extremely likely that it could have helped to improve the performance of models studied in this paper.

4.	Maher Ibrahim Sameen,Biswajeet Pradhan	Severity Prediction of Traffic Accidents with Recurrent Neural Networks	The research introduces a Recurrent Neural Network (RNN) model for predicting injury severity in traffic accidents. It leverages sequential data analysis and temporal correlations inherent in accidents. The study focuses on the North-South Expressway (NSE), Malaysia, using deep learning, LSTM layers, and systematic network architecture optimization.	The RNN-based model shows promise in predicting injury severity in traffic accidents, outperforming Multilayer Perceptron (MLP) and Bayesian Logistic Regression (BLR) models with a validation accuracy of 71.77%. The RNN's ability to capture temporal relationships enhances its performance in comparison to traditional models, thus offering a valuable tool for such predictions.	While demonstrating improved accuracy, the RNN model may still have limitations in handling complex accident scenarios. Future research could explore expanding the dataset, incorporating external factors like weather, and fine-tuning model parameters for even higher predictive accuracy. Additionally, assessing the model's performance in other geographical regions could be beneficial.
5.	Md. Ebrahim Shaik, Md. Milon Islam, Quazi Sazzad Hossain	A review on neural network techniques for the prediction of road traffic accident severity	The paper reviews the utilization of neural network techniques for predicting road traffic accident severity, addressing the global rise in fatalities and injuries. Various neural network models, including Single Layer Perceptron (SLP), Multilayer Perceptron (MLP), Radial Basis Function (RBF), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN), are discussed in the context of impact prediction. The study explores their effectiveness and identifies challenges	Neural network models, particularly deep learning approaches like RNN and CNN, have shown high accuracy and efficiency in predicting road accident severity. Various types of neural networks are employed to identify influencing factors and assess injury risk. The review underscores their potential and highlights their application in road safety research.	While neural network models offer promising results, challenges such as data diversity, real-time implementation, and model interpretability persist. Future research could delve into refining model architectures, incorporating external factors, enhancing interpretability, and addressing model robustness in complex traffic scenarios. A comprehensive framework for integrating various neural network models could also be explored.

Block Diagram:



PROCESS-FLOW:



Methodology:

1. Data Collection:

Data collection is a crucial first step in any machine learning project. In this case, we have obtained a dataset from Kaggle containing images captured from CCTV cameras. The dataset is divided into a training split (images with and without accidents) and a test split for model evaluation. CCTV images can be diverse and collecting a substantial amount (over 600 images each) is essential for building a robust model**.**

2. Data Preprocessing:

Data preprocessing is essential to prepare the dataset for model training. This process involves several key steps. First, data cleaning is necessary to remove any corrupted or irrelevant images. Next, we mentioned scaling features, which typically involves normalizing pixel values to a common range (e.g., 0 to 1) to ensure consistency across images. Lastly, data

handling, which could include tasks like data augmentation (creating variations of existing images) to increase dataset size, can improve model generalization. We mentioned that VGG16 incorporates these preprocessing steps, which is true, as many pre-trained models handle these tasks internally.

3. Training the model:

Model training involves using the VGG16 architecture to extract features from the pre-processed images and then classify them into two categories: "Accident" and "Not Accident." VGG16 is a well-known convolutional neural network (CNN) architecture that has proven effective in image classification tasks. During training, the model learns to recognize patterns and features in the images that distinguish between accidents and non-accidents. This process involves forward and backward passes through the network to optimize its parameters using techniques like gradient descent.

VGG16:

The VGG16 model is a convolutional neural network (CNN) architecture that gained significant popularity for its effectiveness in image classification tasks. VGG16 is known for its simplicity and depth, consisting of 16 weight layers, including 13 convolutional layers and 3 fully connected layers.

Input Layer: The input layer receives the raw image data. In VGG16, typical input dimensions are 224x224x3, representing a 3-channel (RGB) image.

Convolutional Layers: VGG16 has 13 convolutional layers, each responsible for learning different features at multiple spatial scales. These layers use small learnable filters to convolve over the input, extracting patterns like edges, textures, and shapes.

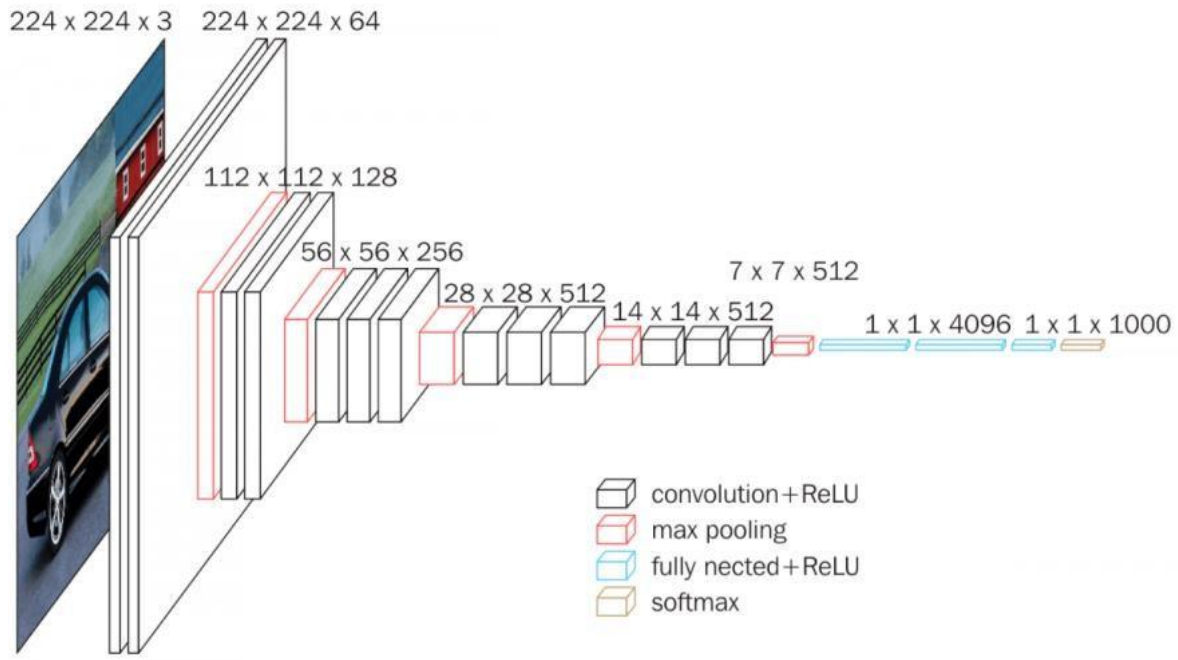
Max-Pooling Layers: After each set of convolutional layers, max-pooling layers reduce spatial dimensions by selecting the maximum value from a local region. This helps reduce computational complexity and enhance translation invariance.

Fully Connected Layers: The final layers of VGG16 are fully connected, densely connected neural network layers. They aggregate high-level features learned by the convolutional layers and perform the actual classification. The last fully connected layer typically has as many neurons as there are classes in the dataset.

SoftMax Layer: At the end of VGG16, a SoftMax activation layer is applied to produce class probabilities. It converts the raw output scores from the previous layer into a probability distribution, making it suitable for multiclass classification.

4. Model Tuning:

Model tuning is essential for optimizing the performance of our classifier. If VGG16 does not meet our expected accuracy, we will explore various techniques to improve it. This may include hyperparameter tuning (adjusting learning rates, batch sizes, etc.) and experimenting with different loss functions. If these efforts do not yield satisfactory results, we can explore other transfer learning models like ResNet, Inception, MobileNet, DenseNet, or Xception. These models have different architectures and may be better suited to our specific dataset.



5. Deployment:

Deployment is the final phase of your project, where we take our trained model and apply it to real-world scenarios. In this case, we mentioned testing the model on videos. we use OpenCV, a popular computer vision library, to extract individual frames from video streams. Then, by feeding these frames into your model, we can detect accidents in real-time or from recorded footage. This deployment phase is where the model's practical utility becomes evident, and it can be integrated into CCTV systems or traffic monitoring applications for accident detection. Monitoring model performance in this real-world context is crucial for ongoing refinement and improvement.

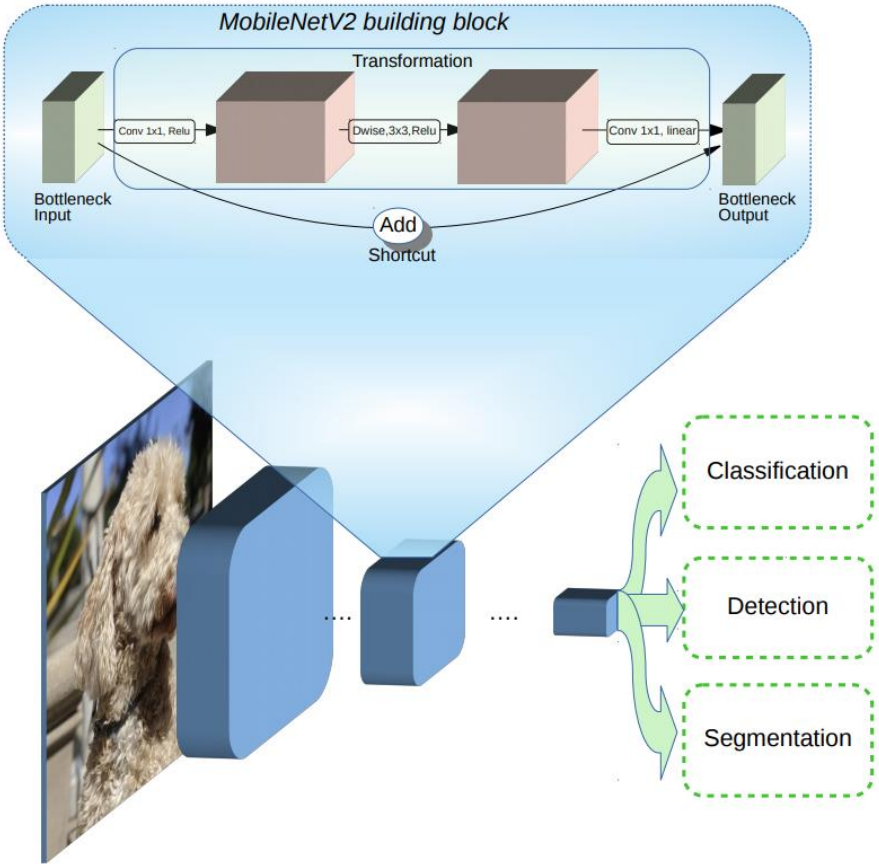
Output:

(Changes) VGG16 gave less accuracy. So, we are using mobile net v2.

MobileNet V2:

MobileNetV2 is a neural network architecture tailored for mobile and embedded vision applications. It's characterized by its efficiency and speed while maintaining solid accuracy. This is achieved through innovative features like inverted residual blocks, bottleneck design, and depthwise separable convolutions. Inverted residual blocks comprise lightweight depthwise separable convolutions followed by linear bottlenecks, reducing parameters and computational demands. The bottleneck design starts with a small number of filters and gradually increases them deeper into the network, preserving information flow while reducing costs. Additionally, shortcut connections and the ability to adjust the width and input resolution allow users to customize the model's trade-off between model size and performance, making MobileNetV2 a practical choice for real-time tasks like image classification, object detection, and segmentation on resource-constrained devices.

MobileNetV2 is particularly valuable for mobile and embedded applications, where computational resources are limited. Its focus on efficiency and its ability to maintain high accuracy make it a popular choice for various computer vision tasks, ensuring that these applications can run smoothly on devices with restricted processing power. Its versatility and balanced performance-to-size ratio have established MobileNetV2 as a go-to option for those seeking real-time image analysis on mobile platforms.



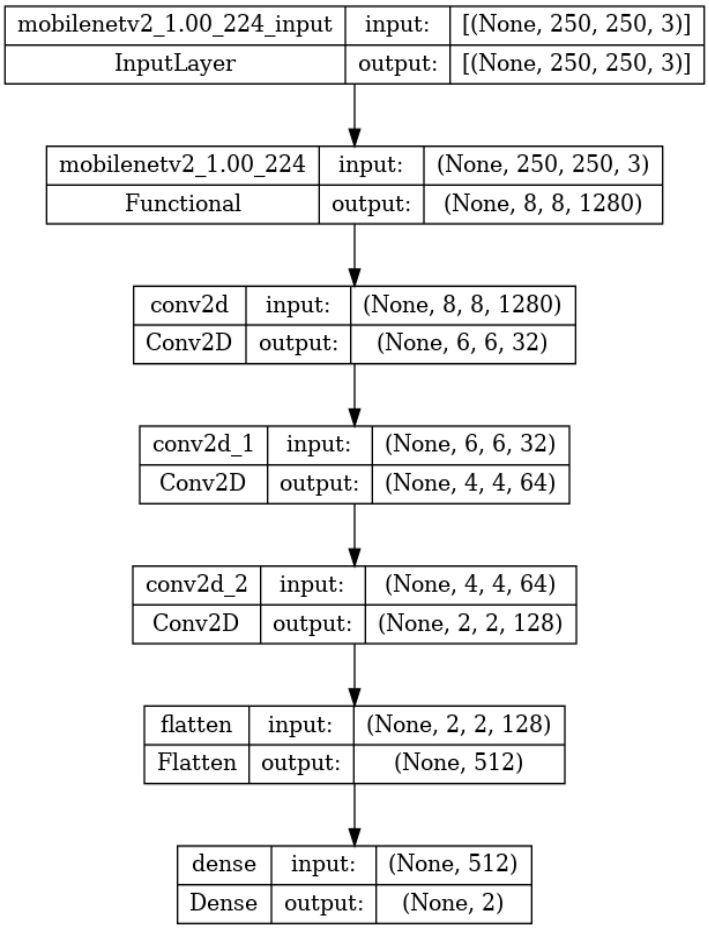
The model data source is a critical component of any deep learning project. The quality and quantity of data available can significantly impact the performance of the model. In the case of the accident detection system, the dataset is a collection of frames captured from YouTube videos involving accidents.
Train: 791
Test: 100
Validation: 98

One of the unique aspects of this dataset is that it includes consecutive frames of an accident, allowing the model to learn to differentiate between an accident and non-accident. This feature is particularly important because it provides the model with the necessary context to accurately identify and classify accidents.

Model Architecture

MobileNet is a well-known neural network architecture specifically designed for mobile devices with limited computational resources. This architecture is efficient and compact, making it suitable for image classification tasks. To further enhance the performance of MobileNet, three convolutional layers with Rectified Linear Unit (ReLU) activation have been added to fine-tune. ReLU is a commonly used activation function that introduces non-linearity to the neural network and can significantly improve the performance of deep learning models. Through the addition of these convolutional layers with ReLU activation, the accuracy of the model on the specific classification task has been improved.

The final network architecture consists of a MobileNet base, three convolutional layers with ReLU activation, and a fully connected layer for classification. This architecture has proved to be highly effective for the image classification task and has allowed the model to achieve state-of-the-art performance with relatively few parameters.



Model Training

During the training of the model, the optimizer used was Adam, a popular optimization algorithm in deep learning known for its efficient computation and adaptability to different learning rates during training. The model was trained for a total of 50 epochs, with each epoch consisting of forward and backward propagation, where the model updates its parameters based on the calculated gradients. The loss function used was categorical cross-entropy, which is commonly used for classification tasks. The neural network model was trained using a train test split of 70/30, meaning a division into training and validation sets, with 70% used for training and 30% reserved for testing the performance of the model.

To prevent overfitting, data augmentation techniques were applied during training. This involved randomly flipping and rotating the images to generate additional training data. The training data was also randomly shuffled and split into batches during each epoch to improve the efficiency of the training process. Throughout the training process, the model was regularly evaluated on a validation dataset to monitor its performance and detect any potential overfitting. If the validation loss did not improve after a certain number of epochs, the learning rate was reduced to help the model converge to a better solution. Overall, the training process was crucial in improving the accuracy of the model and ensuring that it could accurately classify images as either accidents or non-accidents.

Code:

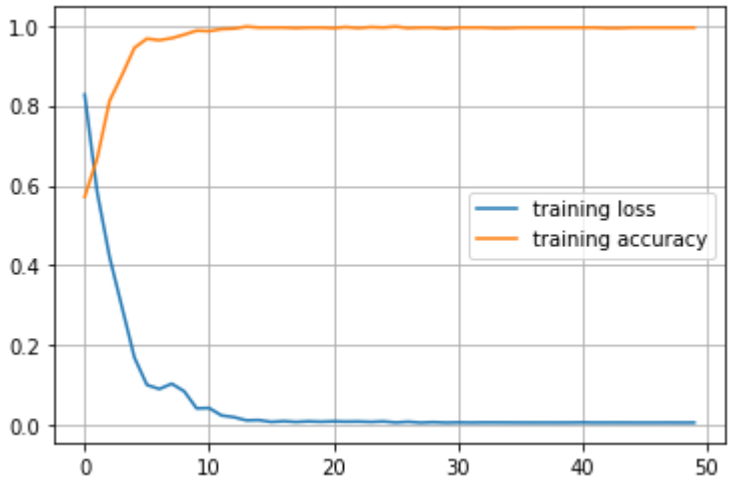
```
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras import layers

batch_size = 100
img_height = 250
img_width = 250

training_ds = tf.keras.preprocessing.image_dataset_from_directory(
    'data/train',
    seed=101,
    image_size= (img_height, img_width),
```

```
    batch_size=batch_size

)

testing_ds = tf.keras.preprocessing.image_dataset_from_directory(
    'data/test',
    seed=101,
    image_size= (img_height, img_width),
    batch_size=batch_size)

validation_ds =  tf.keras.preprocessing.image_dataset_from_directory(
    'data/val',
    seed=101,
    image_size= (img_height, img_width),
    batch_size=batch_size)

class_names = training_ds.class_names

AUTOTUNE = tf.data.experimental.AUTOTUNE
training_ds = training_ds.cache().prefetch(buffer_size=AUTOTUNE)
testing_ds = testing_ds.cache().prefetch(buffer_size=AUTOTUNE)

img_shape = (img_height, img_width, 3)

base_model = tf.keras.applications.MobileNetV2(input_shape=img_shape,
                                                include_top=False,
                                                weights='imagenet')

base_model.trainable = False

model = tf.keras.Sequential([
    base_model,
    layers.Conv2D(32, 3, activation='relu'),
    layers.Conv2D(64, 3, activation='relu'),
    layers.Conv2D(128, 3, activation='relu'),
    layers.Flatten(),
    layers.Dense(len(class_names), activation= 'softmax')
])

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

history = model.fit(training_ds, validation_data = validation_ds, epochs = 50)
plt.plot(history.history['loss'], label = 'training loss')
plt.plot(history.history['accuracy'], label = 'training accuracy')
plt.grid(True)
plt.legend()

plt.plot(history.history['val_loss'], label = 'validation loss')
plt.plot(history.history['val_accuracy'], label = 'validation accuracy')
```

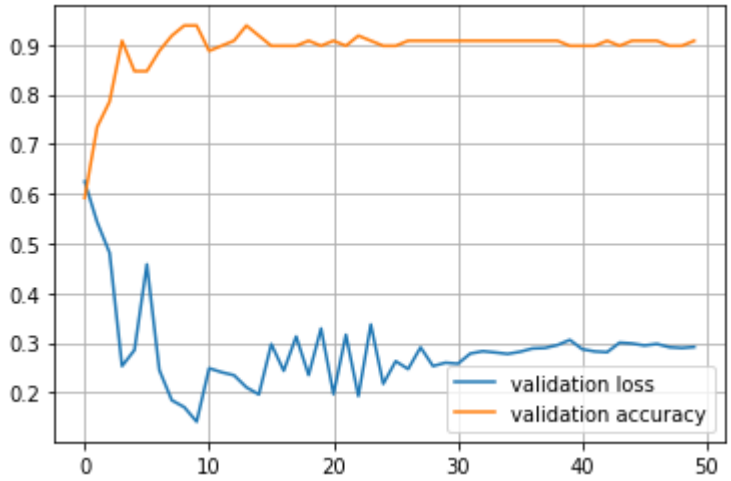
```
plt.grid(True)
plt.legend()

AccuracyVector = []
plt.figure(figsize=(30, 30))
for images, labels in testing_ds.take(1):
    predictions = model.predict(images)
    predlabel = []
    prdlbl = []

    for mem in predictions:
        predlabel.append(class_names[np.argmax(mem)])
        prdlbl.append(np.argmax(mem))

    AccuracyVector = np.array(prdlbl) == labels
    for i in range(40):
        ax = plt.subplot(10, 4, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title('Pred: '+ predlabel[i]+' actl:'+class_names[labels[i]] )
        plt.axis('off')
        plt.grid(True)

from keras.utils.vis_utils import plot_model
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True)

print(class_names)
```

Output:

Graph -> training Loss v/s Training Accuracy



Graph -> Validation Loss v/s Validation Accuracy



Accuracy: 98.5

For Videos- Prediction

Converting into TF-LITE Model and using it to predict the frame.

```
def predict_frame(img):
    img_array = tf.keras.utils.img_to_array(img)
    img_batch = np.expand_dims(img_array, axis=0)
    prediction=(model.predict(img_batch) > 0.5).astype("int32")
    if(prediction[0][0]==0):
        return("Accident Detected")
    else:
        return("No Accident")

import cv2
image=[]
label=[]

c=1
cap= cv2.VideoCapture('data/video.mp4')
while True:
    grabbed, frame = cap.read()
    if c%30==0:
        print(c)
        resized_frame=tf.keras.preprocessing.image.smart_resize(frame, (img_height,
img_width), interpolation='bilinear')
        image.append(frame)
        label.append(predict_frame(resized_frame))
        if(len(image)==75):
            break
    c+=1

cap.release()
```

```
print(label[10])
print(plt.imshow(image[10]))


# Converting into TFLITE model.
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the model.
with open('tf_lite_model.tflite', 'wb') as f:
    f.write(tflite_model)

# %%
interpreter = tf.lite.Interpreter(model_path = 'tf_lite_model.tflite')
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
print("Input Shape:", input_details[0]['shape'])
print("Input Type:", input_details[0]['dtype'])
print("Output Shape:", output_details[0]['shape'])
print("Output Type:", output_details[0]['dtype'])

# %%
interpreter.resize_tensor_input(input_details[0]['index'], (1, 250, 250,3))
interpreter.resize_tensor_input(output_details[0]['index'], (1, 2))
interpreter.allocate_tensors()
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
print("Input Shape:", input_details[0]['shape'])
print("Input Type:", input_details[0]['dtype'])
print("Output Shape:", output_details[0]['shape'])
print("Output Type:", output_details[0]['dtype'])

# %%
from PIL import Image
im=Image.open("data/train/Non Accident/5_17.jpg").resize((250,250))
img_array = tf.keras.utils.img_to_array(im)
img_batch = np.expand_dims(img_array, axis=0)

# %%
interpreter.set_tensor(input_details[0]['index'], img_batch)
interpreter.invoke()
tflite_model_predictions = interpreter.get_tensor(output_details[0]['index'])
print("Prediction results:", tflite_model_predictions)
print(plt.imshow(im))
```

Conclusion:

As a result, using CCTV data to create a deep learning model for accident probability prediction has shown to be a useful tool for both accident prediction due to the relative ease with which the model can notify and inform the first responders and the authorities to protect the people involved by sending aid and monitoring the traffic situation. The model's high accuracy reveals how well it can spot crucial details in the video that point to a crash.

To guarantee safe driving practices, it is important to follow the suggestions, which include wearing a seatbelt, obeying speed limits, avoiding distractions, and keeping a safe distance from other cars. While technology might help prevent accidents, it is ultimately the driver's obligation to drive safely and abide by the regulations of the road.

Hence, using this technology with conventional safety advice will help lower the number of collisions and fatalities on the road.

References:

[1] Sanjay Kumar Singh," Road Traffic Accidents in India: Issues and Challenges", research gate (2017) Transportation Research Procedia.

[2] Guillermo Casanova and Graciela Guerrero Idrovo," Analysis of video surveillance images using computer vision in a controlled security environment", research gate(2020) 2020 15th Iberian Conference on Information Systems and Technologies.

[3] Jayesh Patil and Mandar Prabhu, "Road Accident Analysis using Machine Learning",IEEE(2020) Pune Section International Conference .

[4] Mingyuan Xin and Yong Wang, "Research on image classification model based on deep convolution neural network",2019 EURASIP Journal on Image and Video Processing.

[5] Andrew G. Howard and Menglong Zhu," MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", research gate(2017).

[6] Avid Farhoodfar, "Machine Learning for Mobile Developers: Tensorflow Lite Framework", research gate (2019) by IEEE Consumer Electronics Society SCV.

[7] Nitin Garg , Adnan A Hyder," Road traffic injuries in India: a review of the literature", Scand J Public Health. 2006;34(1):100-9.
[8] Miao Chong, Ajith Abraham and Marcin Paprzycki, " Traffic Accident Data Mining Using Machine Learning Paradigms", Research Gate(2004)
[9] Kasimani, Ramesh & P, Muthusamy & M, Rajendran & Sivaprakash, Palanisamy. (2015). "A Review on Road Traffic Accident and Related Factors". International Journal of Applied Engineering Research.
[10] Mohammed, Ali & Ambak, Kamarudin & Mosa, Ahmed & Syamsunur, Deprizon. (2019). A Review of Traffic Accidents and Related Practices Worldwide. The Open Transportation Journal. 13. 65-83.
[11] Kakkar, Rakesh & Aggarwal, Pradeep & Kakkar, Monica & Deshpande, K. & Gupta, D. (2014). Road traffic accident: retrospective study. Indian J Sci Res. 5. 59-62.