Traffic Violation Detection System

1. Abstract

The rise in accidents due to reckless driving led us to adopt technology, specifically computer vision and machine learning, for a Real-time Traffic Violation Detection System. Our aim is to quickly spot traffic violations. This project stemmed from our mutual worry about road safety and the power of technology to make a difference. We developed this solution to match technological advancements with real safety issues in the world.

2. Introduction

We're employing computer vision and machine learning to combat the growing issue of accidents brought on by careless driving. Computer vision and machine learning are the foundation of our real-time traffic violation detection system. The aim is to promote road safety by immediately identifying traffic offenses.

We begin by looking at real-time video from traffic CCTV cameras. First, we use computer vision algorithms to identify the automobiles in these films. Then, we concentrate on identifying moving violations of traffic laws.

Our project is the result of a shared concern for traffic safety and a conviction in the potential of technology. Not only are we addressing current infractions, but also building a future where technology actively ensures road safety for everyone.

3. Literature Review

1. The paper on "**Automatic Traffic Red-Light Violation Detection Using AI**":
   Objective: The paper aims to develop a machine learning-based system for detecting traffic signal violations, particularly red-light violations, to address the growing number of road accidents caused by rule infractions.

   Methodology: The authors employ AI techniques, including the YOLOv5s model, to create a system that identifies violating vehicles, monitors red signal changes, and recognizes license plates. They optimize the system for accuracy using region-of-interest and vehicle location analysis.

   Model Comparison: The authors modify the YOLOv5s model and compare four variants (v1, v2, v3, and v4) in terms of parameters, inference times, and mean average precision (mAP). Model v4 is selected for its balanced accuracy and processing speed.

   Results: The system achieves high accuracy rates, with 82% for vehicle identification, 90% for detecting traffic signal status changes, and up to 86% for violation detection. Real-time results include detecting red-light violations, vehicle classification, and license plate extraction.

Conclusion and Future Work: The paper concludes that the proposed AI-based system holds promise for assisting traffic police in enhancing traffic safety. Future work may focus on reducing computation time and adapting the system to real-time CCTV cameras and various types of license plates.

2. The paper titled "Traffic Surveillance System and Criminal Detection Using Image Processing and Deep Learning":

Problem and Purpose: The paper addresses the challenges of traffic violations and criminal activities on roads, aiming to enhance security and safety. It proposes a system that utilizes image processing and deep learning to intelligently identify traffic offenders by recognizing license plates, detect accidents, and identify criminals.

System Components: The system comprises several components, including a traffic surveillance camera, object detection models, license plate extraction through Optical Character Recognition (OCR), criminal detection using facial recognition, and accident detection based on vehicle collisions. The system also integrates GPS, GSM, and Arduino for location tracking and emergency alerts.

Object Detection: The system employs RetinaNet, a one-stage object detection technique, to identify vehicles violating traffic rules. It localizes license plates through image processing operations, and OCR is used to extract text from the plates. Convolutional Neural Networks (CNNs) are employed to identify criminals.

Accident Detection: The system detects accidents by analyzing video frames from traffic cameras. It uses ResNet-50 for this purpose and determines accidents based on vehicle-to-vehicle or vehicle-to-object collisions. When an accident is detected, it sends alerts to the control room, emergency services, and vehicle owners using GPS, GSM, and Arduino.

Circuit and Experimental Results: The paper provides details of the system's circuit diagram, highlighting the integration of Arduino, GSM, and GPS modules. Experimental results demonstrate the effectiveness of the proposed system, including helmet detection, accident alerts, and criminal identification. RetinaNet is found to perform well, and the system shows promise for real-time applications.

4. Dataset

4.1 Dataset details

Link1 for vehicles:
https://s3.amazonaws.com/udacity-sdc/Vehicle_Tracking/vehicles.zip

Link2 for non- vehicles:
https://s3.amazonaws.com/udacity-sdc/Vehicle_Tracking/non-vehicles.zip

Link for License plate data:
https://www.kaggle.com/datasets/sarthakvajpayee/ai-indian-license-plate-recognition-data/download?datasetVersionNumber=1

Link for traffic signs:
https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-preprocessed

The license plate detection data is for Indian License Plates and The traffic signal data is already preprocessed. Before preprocessing training dataset was equalized making examples in the classes equal.

4.2 Data preprocessing

For data preprocessing, we have used the following techniques to process our image data to model usable data:-

Histogram of Oriented Gradients (HOG) is a key technique used for feature extraction from images. It works by capturing the distribution of gradients (edges) in different orientations within localized regions of an image.
Both photos of cars and photographs without cars were used to classify and identify HOG traits. Small cells are used to analyze how the edges of these images are oriented. The histograms of gradient orientations inside these blocks are generated when these cells are organized into bigger groups called blocks. The method effectively catches the crucial details of the texture and shape of objects in the photos, independent of their color, by taking into account these histograms.

The structural characteristics of things, in this case vehicles and non-cars, are represented by the HOG features. This method is essential for spotting different patterns in photos, especially when colour information alone might not be enough to classify an image correctly. HOG features enable the classifier to efficiently discriminate between cars and non-cars by teaching it crucial characteristics pertaining to the form and edges of vehicles.

5. Methodology
Vehicle Detection:

We use the Linear Support Vector Machine (SVM) and Logistic Regression and CNN machine learning classifiers to identify vehicles. These classifiers are essential for discriminating between photos with and without cars. Let's examine each one's functions and roles in more detail.

1. Linear Support Vector Machine (SVM): SVM is a potent classifier that is utilized for tasks like image recognition. SVM gains the ability to distinguish between features taken from photos of automobiles and non-cars in this situation.

2. Logistic Regression: Regression uses a logistic function to describe the likelihood that an input belongs to the positive class (vehicle). The input is categorized as a positive example if this probability is higher than a predetermined threshold, which is commonly 0.5; otherwise, it is categorized as a negative example.

Regression model can correctly predict whether or not a given area in a picture contains a car.

3. CNN:

Model Architecture:

- The model starts with a dense layer of 512 neurons using ReLU activation.
- A dropout layer is added to prevent overfitting by randomly dropping 50% of the neurons during training.
- Another dense layer with 128 neurons and ReLU activation follows.

- Finally, a dense layer with a single neuron and sigmoid activation is added for binary classification.

Compilation:

- The model is compiled using the Adam optimizer and binary crossentropy loss, which is suitable for binary classification tasks.
- The accuracy metric is chosen to monitor the model's performance during training.

Training:

- The model is trained on the training data (X_train and y_train) for 10 epochs with a batch size of 32.

Evaluation:

- The test set (X_test and y_test) is used to evaluate the model's performance.
- Test loss and accuracy are printed.

Prediction and F1 Score:

- The model predicts labels for the test set, and predictions are thresholded at 0.5.
- F1 score is computed using the predicted and true labels.

Sign Recognition:

Loading and Preprocessing Data:

- Loads data from a file named 'data2.pickle' using pickle.
- Converts the labels to categorical format using to_categorical.
- Transposes the dimensions of the image data arrays.

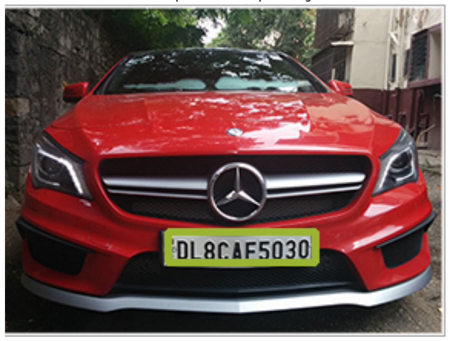
CNN Model Definition:

- Creates a Sequential model using Keras.
- Adds a 2D convolutional layer with 32 filters, a kernel size of 3x3, 'same' padding, and ReLU activation.
- Adds a 2D max pooling layer with a pool size of 2x2.
- Flattens the output to a 1D array.
- Adds a dense layer with 500 units and ReLU activation.
- Adds the output layer with 43 units (assuming it's a classification task) and softmax activation.
- Compiles the model using the Adam optimizer and categorical cross-entropy loss.

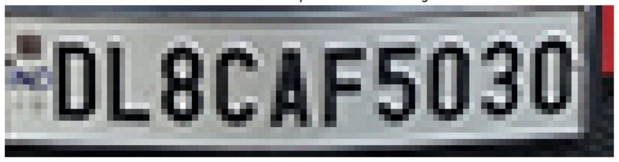
License Plate Detection:

- The detect_plate function takes an image as input and uses a pre-trained Haar Cascade classifier (plate_cascade) to detect license plates. It draws rectangles around the detected plates and optionally adds text.

detected license plate in the input image

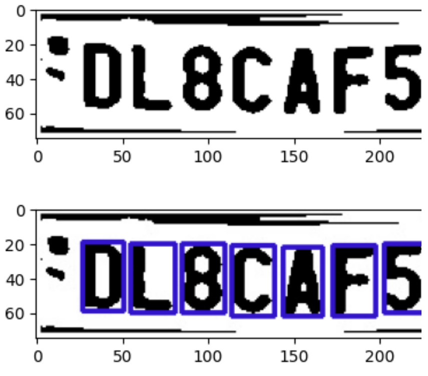


extracted license plate from the image

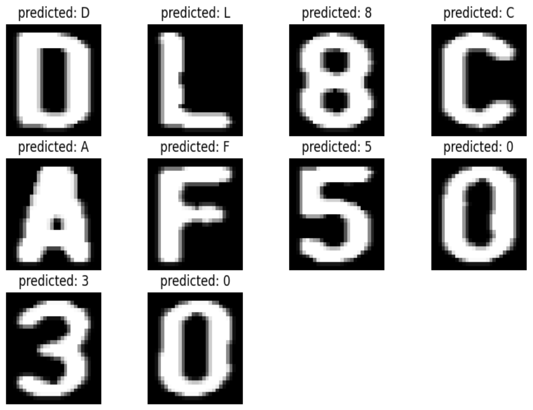


- Character Segmentation:
  - The segment_characters function takes a detected license plate image, processes it to enhance character visibility, and then uses contour detection to segment individual characters.
- Character Recognition Model:

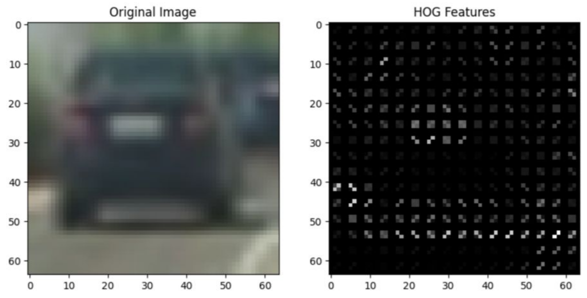
    - The code defines a convolutional neural network (CNN) using Keras for character recognition. The model architecture consists of convolutional layers, max pooling, dropout, and dense layers.

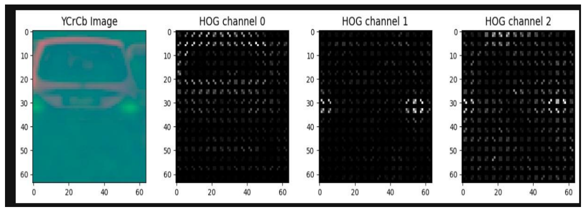


    - The model is compiled using the
    - sparse categorical cross-entropy loss function and the Adam optimizer with a specified learning rate.
    - Data augmentation is applied using the ImageDataGenerator to improve model generalization.
- Data Loading:
    - Training and validation data are loaded using the flow_from_directory method, with images resized to (28, 28) pixels.
- Custom F1 Score:
    - Custom F1 score functions (f1score and custom_f1score) are defined for model evaluation.
- Model Training:
    - The CNN model is trained using the fit_generator method with training and validation data generators. Training stops if the validation custom F1 score exceeds 0.99.
- Character Recognition Results:
    - The code segment responsible for character recognition, which involves predicting characters from the segmented images and displaying the results.



6. Results & Analysis

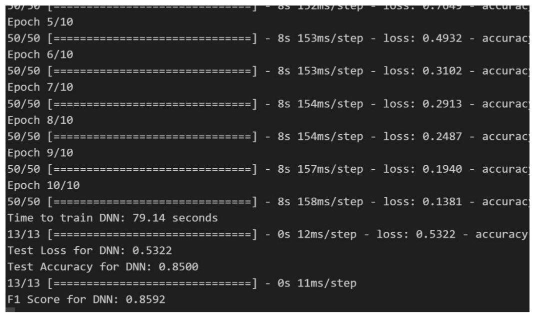




HOG (Histogram of Oriented Gradients) features are a type of feature descriptor used in computer vision and image processing for object detection. HOG features capture the distribution of gradients or edge directions in an image. Here's a brief explanation:

Gradient Computation:

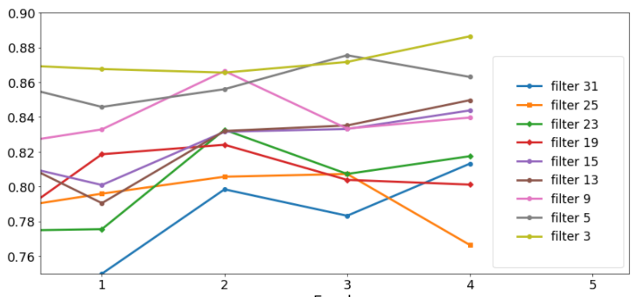
The first step is to compute the gradients of the image in both the horizontal and vertical directions. Gradients represent the intensity variations in different directions.
Cell Division:
The image is divided into small cells (e.g., 8x8 pixels). Each cell represents a local region in the image.
Histogram Calculation:
For each cell, a histogram of gradient directions is computed. This histogram represents the distribution of gradient orientations within the cell.
Block Normalization:
The cells are grouped into larger blocks (e.g., 2x2 cells). Normalization is applied to the histograms within each block. This helps invariance to changes in illumination and contrast.



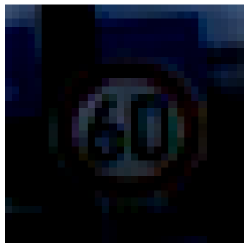
```
50/50 [=============================] - 8s 152ms/step - loss: 0.7649 - accurac
Epoch 5/10
50/50 [=============================] - 8s 153ms/step - loss: 0.4932 - accurac
Epoch 6/10
50/50 [=============================] - 8s 153ms/step - loss: 0.3102 - accurac
Epoch 7/10
50/50 [=============================] - 8s 154ms/step - loss: 0.2913 - accurac
Epoch 8/10
50/50 [=============================] - 8s 154ms/step - loss: 0.2487 - accurac
Epoch 9/10
50/50 [=============================] - 8s 157ms/step - loss: 0.1940 - accurac
Epoch 10/10
50/50 [=============================] - 8s 158ms/step - loss: 0.1381 - accurac
Time to train DNN: 79.14 seconds
13/13 [=============================] - 0s 12ms/step - loss: 0.5322 - accuracy
Test Loss for DNN: 0.5322
Test Accuracy for DNN: 0.8500
13/13 [=============================] - 0s 11ms/step
F1 Score for DNN: 0.8592
```

The previous models have accuracy of 69.62% accuracy in svm and 61.24% accuracy in logistic regression and 85% in CNN.

We used CNN for sign detection and achieved an accuracy of 92%. Along with an accuracy of 95% for Plate detection using CNN.



Output on terminal for this:
ClassId: 3
Label: Speed limit



7. Conclusion

In conclusion, our Real-time Traffic Violation Detection System, leveraging computer vision and machine learning, exhibits promising results with 85% accuracy in vehicle detection using CNN, 92% accuracy in traffic sign recognition, and a notable 95% accuracy in license plate detection.