STM JOURNALS

Trends in
Machine Design

ISSN: 2455-3352
Volume 11, Issue 2, 2024
May–August
DOI (Journal): 10.37591/TMD

http://engineeringjournals.stmjournals.in/index.php/TMD/index

Research TMD

Traffic Detection Algorithms Analysis using ML

Subanshika[1], Shivam Patil[2], Nikita Jain[3,*]

Abstract

It is difficult to watch traffic on crowded roads. Traffic monitoring procedures are time-consuming, expensive, labor-intensive, and require human operators. The limited accessibility hindered the storing and processing of large-scale video streams. Nonetheless, it is now possible to employe video feeds from traffic monitoring systems for number plate recognition, object tracking, traffic behavior analysis, and surveillance. Static image recognition and vehicle identification in a traffic surveillance system are very useful and easily adjustable to a range of tasks. The techniques for processing automated vehicle detection and recognition will be covered in this article. Cars' notable discriminating qualities are discovered to be correlated with minimum bounding box environment-related data. Vehicle detection is a crucial task in traffic surveillance, which has several applications in transportation systems, public safety, and urban planning. In traffic surveillance film, machine learning approaches have demonstrated significant potential for effectively identifying moving cars. In this research, we offer a machine learning-based methodology for effective moving vehicle detection in traffic surveillance. Data collection, preprocessing, feature extraction, model selection, training, assessment, and deployment are all steps in the technique. We demonstrate the effectiveness of the proposed methodology by evaluating the performance of several machine learning models on a large traffic surveillance dataset.

Keywords: Traffic detection algorithms analysis using ML

INTRODUCTION

Image processing is significantly used by artificial intelligence to understand the environment. Before being used for interpretation by artificial intelligence, environment-related data must first go through a number of processes. Environment-related data must first pass through several steps before being utilized by artificial intelligence for interpretation. Typically, these phases involve obtaining data (images, videos, etc.), analyzing it, and then reformatting it in a way that allows a computer to understand it [1].

In this study, we employed a data set and camera to recognize and classify autos under the assumption that the camera is steady, or estabilized. We examined the vehicles on the security cameras for this study. Software for traffic control must be able to categorize cars. In this study, we use a camera and data set to recognize and categorize cars; the camera is assumed to be stationary (or stabilized). Two crucial procedures in image processing are object recognition and classification. Every piece stands out on its own for being unique. gridlock, accidents, and traffic jams. Without being explicitly trained to do so, machine learning algorithms create a model from sample data, also known as training data, to produce predictions or judgements.

This paper examines how machine learning algorithms are used in a wide range of applications, such as computer vision, speech recognition, email filtering, medicine, and agriculture, when developing conventional algorithms that can do the

***Author for Correspondence**
Nikita Jain
E-mail: nikita.jain@poornima.org

[1]Student, Department of Computer Engineering, Poornima College of Engineering, Jaipur, India
[2]Student, Department of Computer Engineering, Poornima College of Engineering, Jaipur, India
[3]Professor, Department of Computer Engineering, Poornima College of Engineering, Jaipur, India

Received Date: May 09, 2024
Accepted Date: June 04, 2024
Published Date: July 17, 2024

Citation: Subanshika, Shivam Patil, Nikita Jain. Traffic Detection Algorithms Analysis using ML. Trends in Machine Design. 2024; 11(2): 9–15p.

required tasks is challenging or impossible. Data from a range of sources, including unstructured data (like text, photos, or audio) and structured data (like tables or databases), can be used to train a machine learning model [2].

In this study, we will discuss a variety of machine learning methodologies, including semi-supervised learning, reinforcement learning, supervised learning, and unsupervised learning. Each strategy has advantages and disadvantages, and the best one to use will depend on the particular issue at hand. To improve efficiency, machine learning models can profit from a number of techniques, such as data augmentation, transfer learning, and pruning. These techniques can reduce the training time, memory requirements, and computational complexity of the models while maintaining their accuracy.

This paper examines the effective use of machine learning approaches for moving vehicle recognition in traffic surveillance, which is a potential strategy for improving safety and traffic management. More advanced and precise techniques for identifying vehicles in traffic surveillance footage are probably in store as machine learning algorithms advance. In which we found a significant increase in the use of machine-learning techniques for traffic surveillance. One of the most popular deep learning models for vehicle detection in traffic surveillance footage is convolutional neural networks, or CNNs. CNNs are very good at tracking and identifying cars because of their ability to automatically extract attributes from the input data. Recent advances in deep learning and machine learning have completely changed traffic surveillance and other computer vision applications. Convolutional neural networks (CNNs), in particular, are deep learning algorithms that have demonstrated impressive performance in a variety of vision tasks, such as object detection and recognition [3].

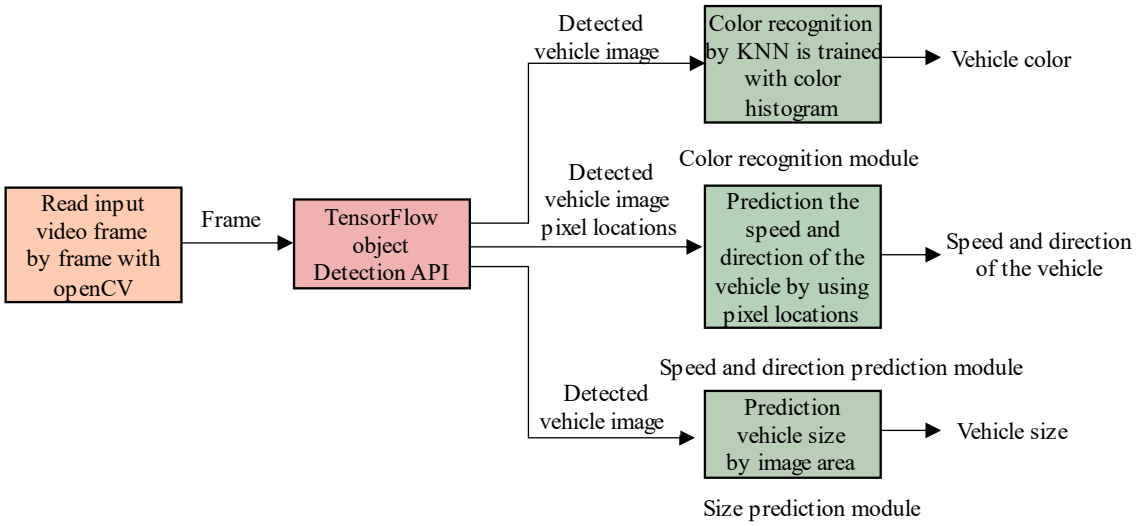


Figure 1. System Architecture.

METHODS & MATERIAL

This section outlines the methodology and materials utilized in the development to four vehicle detection in traffic surveillance using machine learning techniques, focusing on the import libraries starts by importing the required libraries. In this case, it imports the cv2 (OpenCV) library for image processing and computer vision tasks. The development process applies background subtraction techniques to detect moving objects (cars) and then counts the number of cars based on their movement [4].

System Design and Database Architecture

Initially, we designed the application's architecture with scalability and real-time data management in mind. Then we starts with the program by importing the required libraries. In this case, it imports the cv2 (OpenCV) library for image processing and computer vision tasks. The count_cars function is

defined to perform the car counting operation. It takes the path to the video file as input and uses background subtraction to count the cars [5].

Backend Subtraction

The function creates a background subtractor using the cv2.createBackgroundSubtractorMOG2() function (or cv2.createBackgroundSubtractorMOG2() if available) to separate the moving foreground objects (cars) from the static background of the video. This function creates an instance of the MOG2 (Mixture of Gaussians) background subtractor. It's an advanced algorithm that models each pixel's distribution as a mixture of Gaussians over time. This helps in adaptively updating the background model to accommodate gradual changes in lighting conditions and other factors [6].

Video Capture

The function opens the video file using cv2.VideoCapture(video_path) and checks if it can be opened successfully. A video capture object that can read frames from the video file is initialised by this function.

Car Counting Loop

The function processes each frame in the video until the end of the video is reached. In order to identify the moving objects, background removal is used when each frame is received using cap.read().

Noise removal

To remove the noise and enhance the detection, morphological operations (erosion and dilation) are applied using cv2.morphologyEx.

a. *Contour detection:* Contours are the closed curves that represent the shapes of detected objects.
b. *Filtering and Counting:* The function iterates through the detected contours and filters out small contours to avoid detecting noise [7].
c. *Display and User Interaction:* The processed frame with the bounding boxes is displayed using cv2.imshow.
d. *Release Resources:* After processing, the video capture is released with cap.release() and the OpenCV windows are closed with cv2.destroyAllWindows().

PROPOSEDSCHEME

The proposed scheme aims to address the challenges of efficient moving vehicle detection in traffic surveillance using machine learning techniques. The program uses the OpenCV library in Python to count the number of cars in a given video file. It applies background subtraction techniques to detect moving objects (cars) and then counts the number of cars based on their movement. The goal is to subtract the static background from each frame of the video, leaving only the moving objects (such as cars). This is crucial for tasks like object detection and tracking in videos [8].

Following the function creates an instance of the MOG2 (Mixture of Gaussians) background subtractor. It's an advanced algorithm that models each pixel's distribution as a mixture of Gaussians over time. This aids in updating the backdrop model adaptively to take into account small variations in lighting and other variables. The schema was optimized for quick queries and updates, essential for real-time applications. The function opens the video file using cv2.VideoCapture(video_path) and checks if it can be opened successfully [9]. The function processes each frame in the video until the end of the video is reached. It reads each frame using cap.read() and applies background subtraction to detect the moving objects. A moving target can have several properties extracted from it, including texture, colour, and shape. These characteristics can be loosely categorised as spatial characteristics or features of time.

The convolutional neural network receives the points from the area of interest and uses them to track moving objects. The proposed approach for vehicle detection locates every car in each scenario under

various environmental conditions, and its accuracy and processing speed are excellent. The observed vehicle is tracked using the CNN-based tracking method. The suggested algorithm's characteristics include sensitivity (92% detection rate), accuracy (approximately 91.8%), and specificity (about 91.2%).

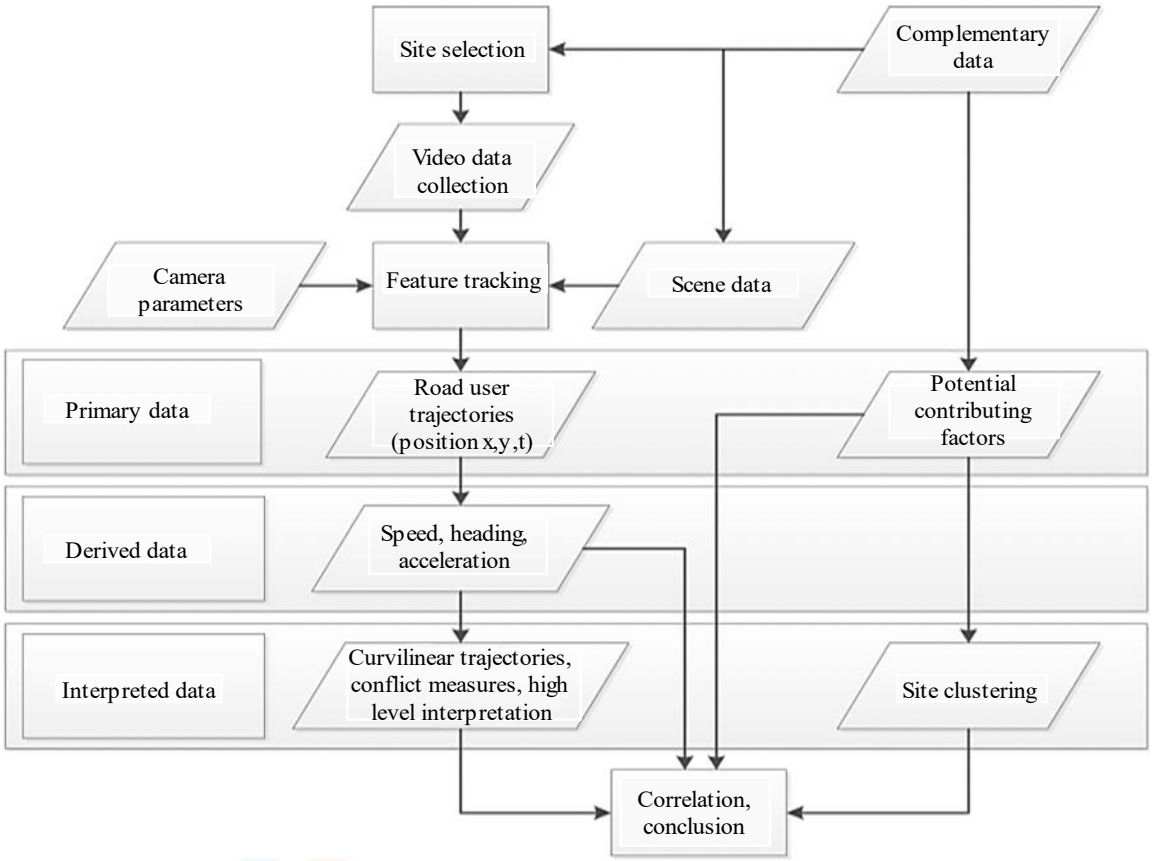


Figure 2. Flow chart.

The basic goal of ML research is to develop general-purpose learning techniques that are more effective across a wide range of domains and are more efficient (in terms of both time and space). Together with time and space complexity, the effectiveness with which a technique uses data resources is a key performance aspect in machine learning. Better prediction accuracy and predictor rules that are easy to understand are also very important.

ML algorithms have an advantage over manual or direct programming since they are entirely data-driven and can analyze a lot of data in short bursts of time. Also, they are frequently more accurate and immune to human prejudice. Software creation etc. Anybody can quickly identify a picture of a letter by the alphabet it belongs to, but creating an algorithm to do this is challenging software customization for the environment in which it is used. Take voice recognition software as an example, which must be tailored to the demands of the user. In Vehicle identification and recognition remain challenging challenges in the field of intelligent transportation surveillance systems. In this research, we introduced a cascade of boosted classifiers that can be used to detect vehicles in photos of on road scenes based on the properties of the vehicle images.

Vehicle identification and recognition remain challenging challenges in the field of intelligent transportation surveillance systems. Based on the characteristics of the vehicle photographs, we developed a cascade of boosted classifiers in this study that can be used to identify cars in pictures of on-road situations.

RESULTS & DISCUSSIONS

Following the deployment of our vehicle detection in using machine learning techniques. The program will process the video, detect cars using background subtraction, draw bounding boxes around the detected cars, and display the video with the bounding boxes in real-time. When you run the program, it will open a window displaying the processed video. The software will count automobiles continually and print the total number of cars found at the conclusion. You can use the 'q' key to halt the counting and close the video window.

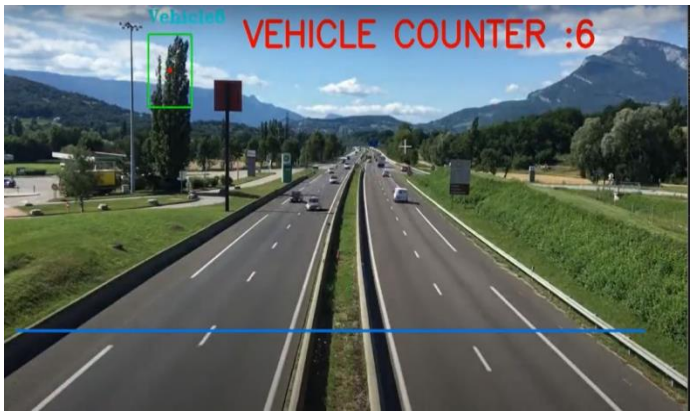
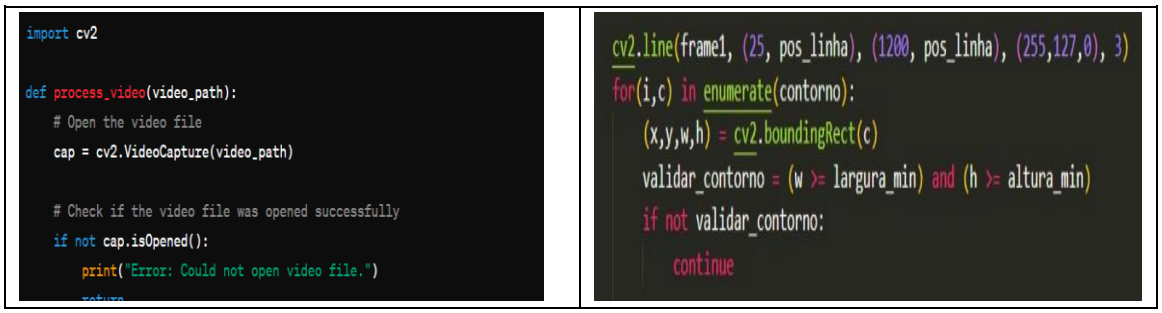


Figure 3. Traffic Monitoring.

This output indicates that the program detected a total of 6 cars in the video. The actual number will vary depending on the accuracy of the car detection process. The number of cars detected may also be affected by factors such as lighting conditions, camera angle, and the presence of other moving objects in the video.

The vehicle detection classifier was then built using Hare-like features and an Ada-boost method, which is different from other published research on the subject. The results showed that it will continuously count the number of cars and print the total number of cars detected at the end. To stop the counting process and close the video window, you can press the 'q' key.



```
import cv2

def process_video(video_path):
    # Open the video file
    cap = cv2.VideoCapture(video_path)

    # Check if the video file was opened successfully
    if not cap.isOpened():
        print("Error: Could not open video file.")
        return
```

```
cv2.line(frame1, (25, pos_linha), (1200, pos_linha), (255,127,0), 3)
for(i,c) in enumerate(contorno):
    (x,y,w,h) = cv2.boundingRect(c)
    validar_contorno = (w >= largura_min) and (h >= altura_min)
    if not validar_contorno:
        continue
```

Figure 4. Algorithm to detect traffic.

The algorithm modifies its behavior to maximize the reward after receiving input in the form of rewards or penalties for its activities. Algorithms for semi-supervised learning combine supervised and unsupervised learning in which some data is labeled and some are not [10].

Artificial intelligence (AI) technologies known as machine learning enable computer systems to automatically learn from their experiences and advance without explicit programming. These algorithms are made to analyze enormous volumes of data, spot patterns, and base choices or predictions on those patterns. This training data is then used by the model to create predictions about fresh, unforeseen data.


```
        cv2.rectangle(frame1,(x,y),(x+w,y+h),(0,255,0),2)
        centro = pega_centro(x, y, w, h)
        detec.append(centro)
        cv2.circle(frame1, centro, 4, (0, 0,255), -1)

        for (x,y) in detec:
            if y<(pos_linha+offset) and y>(pos_linha-offset):
                carros+=1
                cv2.line(frame1, (25, pos_linha), (1200, pos_linha), (0,127,255), 3)
                detec.remove((x,y))
                print("car is detected : "+str(carros))

    cv2.putText(frame1, "VEHICLE COUNT : "+str(carros), (450, 70), cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 0, 255),5)
    cv2.imshow("Video Original" , frame1)
    cv2.imshow("Detectar",dilatada)

    if cv2.waitKey(1) == 27:
        break

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

Figure 5. Output.

Machine learning algorithms come in a variety of forms, including reinforcement learning, unsupervised learning, semi-supervised learning, and supervised learning. When training a model with known input and output variables, supervised learning techniques employ labeled data.

On the other hand, unsupervised learning algorithms don't rely on labeled data. Instead, they use techniques like clustering and dimensionality reduction to analyze the data to find patterns and links. In general, machine learning algorithms are playing a bigger role in a variety of applications, such as fraud detection, natural language processing, recommendation systems, picture and speech recognition, and many more.

The discussion of the program in the following steps are: Open the video file and create a background subtractor. Process each frame, apply background subtraction, and find contours to detect moving objects (cars). Filter out small contours to avoid noise. Increment the car count for each detected car. Draw bounding boxes around the detected cars to visualize the results. Display the video with the detections and wait for the user to stop the process. Print the total number of cars detected.

Calculate the area of each contour to filter out small contours as noise. Draw a bounding box around the detected car to visualize the result. Wait for a key press (30 milliseconds) and check if the 'q' key is pressed if so, exit the loop and stop counting. To guarantee that the code inside the block is only run if the script is run directly (rather than imported as a module), this is a frequent Python idiom.

CONCLUSION

Vehicle identification and recognition remain major challenges in the field of intelligent transportation surveillance systems. Based on the characteristics of the vehicle photographs, we developed a cascade of boosted classifiers in this study that can be used to identify cars in pictures of on-road situations. Subsequently, the vehicle detection classifier was constructed utilising an AdaBoost technique using Hare-like characteristics, which deviates from previous published studies on the topic. The histogram intersection method was then used to determine the similarity between different LGBP Histogram Sequences. The final classification, which is remarkably unaffected by changes in appearance caused by lighting or vehicle pose, was based on the closest neighbourhood of the Euclidean distance. On a realistic data set, we have tested this technique.

Vehicle identification and recognition remain challenging challenges in the field of intelligent transportation surveillance systems. In this research, we introduced a cascade of boosted classifiers that can be used to detect vehicles in photos of on-road scenes based on the properties of the vehicle images. The vehicle detection classifier was then built using Hare-like features and an Ada-boost method, which is different from other published research on the subject. The histogram intersection approach was then

utilised to ascertain the degree of similarity between different LGBP Histogram Sequences, and the nearest neighbourhood of the Euclidean distance was employed for the final classification resulting from lighting or vehicle posture.

We might also include facilities that enable users to upload video, continuously detect and monitor traffic at specified intervals, and take additional action in response to the observed traffic's average volume enlarge the area of operation. For the purposes of robotics, security surveillance, and self-driving cars, we can add more crucial functions. The model could also propose additional routes and actions that should be performed as well as avoided based on the volume of traffic.

Overall, developing an Android application requires planning, design, development, testing, and publishing. You can successfully develop an Android application that fulfils the requirements and expectations of your users by following these steps. One of the key issues in computer vision and machine learning is the effective use of machine learning approaches for moving vehicle recognition in traffic surveillance. The objective is to create a system that can quickly and precisely identify cars in a traffic surveillance camera's video feed. This has numerous applications, including traffic monitoring, vehicle tracking, and automated toll collection.

REFERENCES

1. S. Yu, Y. Wu, W. Li, Z. Song, and W. Zeng, “A model for fine-grained vehicle classification based on deep learning,” Neurocomputing, vol. 257, pp. 97–103, 2017.
2. H. Asaeda, A. Aara, and M. Belleek, “Shadow elimination and vehicles classification approach in traffic video surveillance context,” Journal of Visual Languages & Computing, vol. 25, no. 4, pp. 333–345, 2014.
3. G. Yan, M. Yu, Y. Yu, and L. Fan, “Real-time vehicle detection using histograms of oriented gradients and AdaBoost classification,” Opti, vol. 127, no. 19, pp. 7941–7951, 2016.
4. W. Sun, G. Zhang, X. Zhang, X. Zhang, and N. Ge, “Fine-grained vehicle type classification using a lightweight convolutional neural network with feature optimization and joint learning strategy,” Multimedia Tools and Applications, pp. 1–14, 2020.
5. H. Kim, “Multiple vehicles tracking and classification system with a convolutional neural network,” Journal of Ambient Intelligence and Humanized Computing, pp. 1–12, 2019.
6. Y. Nam and Y. C. Nam, “Vehicle classification based on images from visible light and thermal cameras,” EURASIP Journal on Image and Video Processing, vol. 2018, no. 1, 2018.
7. C. R. Kumar and R. Anuradha, “Feature selection and classification methods for vehicle tracking and detection,” Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 3, pp. 4269–4279, 2020.
8. X. Wang, X. Chen, and Y. Wang, “Small vehicle classification in the wild using the generative adversarial network,” Neural Computing and Applications, vol. 33, no. 10, pp. 5369–5379, 2020.
9. B. Zhang, “Reliable classification of vehicle types based on cascade classifier ensembles,” IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 1, pp. 322–332, 2012.
10. N. Shavit, A. Hasnaa, A. Meacher, and A. Naik, “Accurate classification for automatic vehicle-type recognition based on ensemble classifiers,” IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 3, pp. 1288–1297, 2019.

