SMART PARKING MANAGEMENT SYSTEM

INTRODUCTION

The project is an example of a computer vision application of AI technology. The aim of this project is to automate the process of identifying and tracking vehicles in a parking lot by using their number plates. This system can be used to manage parking facilities at universities, companies, and other institutions.

The process involves capturing images of the number plates of incoming vehicles using a camera system that is equipped with image recognition software. This software can then analyze the images and identify the alphanumeric characters on the number plate. The information extracted from the number plate can be used to determine if the vehicle belongs to staff, students or other authorized personnel. This allows the parking management system to grant or deny access to the parking lot and track the movement of vehicles within the facility.

AI plays a key role in this project as it allows the software to learn and improve over time. The more data the system is trained on, the better it becomes at recognizing and identifying number plates accurately. AI algorithms such as machine learning and deep learning can be used to improve the accuracy of the system and reduce errors. Here we have used CNN and YOLO v4 for this project.

Overall, this project has the potential to greatly improve the efficiency and security of parking facilities by automating the identification and tracking of vehicles. It can save time for staff and students by reducing the need for manual entry and validation of information, and can help to prevent unauthorized access to the parking lot.

LITERATURE SURVEY

We have briefed 6 scientific papers in our document related to our project. All these papers discuss different methods to detect plates with different accuracy.

1. Computer Vision based License Plate Detection for Automated Vehicle Parking Management System:

   This paper proposes a computer vision-based license plate detection system for automated vehicle parking management systems. The system is able to detect and recognize license plates of vehicles entering and exiting the parking area. The algorithm is based on the Haar feature-based cascade classifier and utilizes the OpenCV library. The system is able to detect license plates with an accuracy of 98.4% and recognize the characters on the plate with an accuracy of 95%.

2. An Automated Vehicle License Plate Recognition System:

   This paper describes an automated vehicle license plate recognition system that uses a combination of edge detection, segmentation, and template matching techniques to detect and recognize license plates in real-time. The system is implemented using the OpenCV library and is able to detect license plates with an accuracy of 96.8% and recognize the characters on the plate with an accuracy of 92.4%.

3. Vehicle Number Plate Detection System for Indian Vehicles:

   This paper proposes a vehicle number plate detection system for Indian vehicles. The system is designed to handle the challenges posed by Indian license plates, such as nonstandard plate sizes, non-uniform fonts, and complex backgrounds. The system uses a combination of edge detection, morphological operations, and connected component analysis to detect and recognize license plates. The system is able to detect license plates with an accuracy of 97.6% and recognize the characters on the plate with an accuracy of 92.3%.

4. License Plate Detection using Computer Vision technique with Artificial Intelligence:

   This paper proposes a license plate detection system using a combination of computer vision techniques and artificial intelligence. The system uses deep learning-based object detection algorithms, such as You Only Look Once (YOLO) and Faster R-CNN, to detect license plates. The system is able to detect license plates with an accuracy of 96.7% and recognize the characters on the plate with an accuracy of 94.1%.

5. Car Number Plate Detection using Deep Learning:

   This paper proposes a system for detecting and recognizing car license plates using deep learning techniques. The system uses a convolutional neural network (CNN) for license plate detection, and a combination of CNN and a Recurrent Neural Network (RNN) for license plate recognition. The proposed system achieves high accuracy in detecting and recognizing license plates in real-world scenarios.

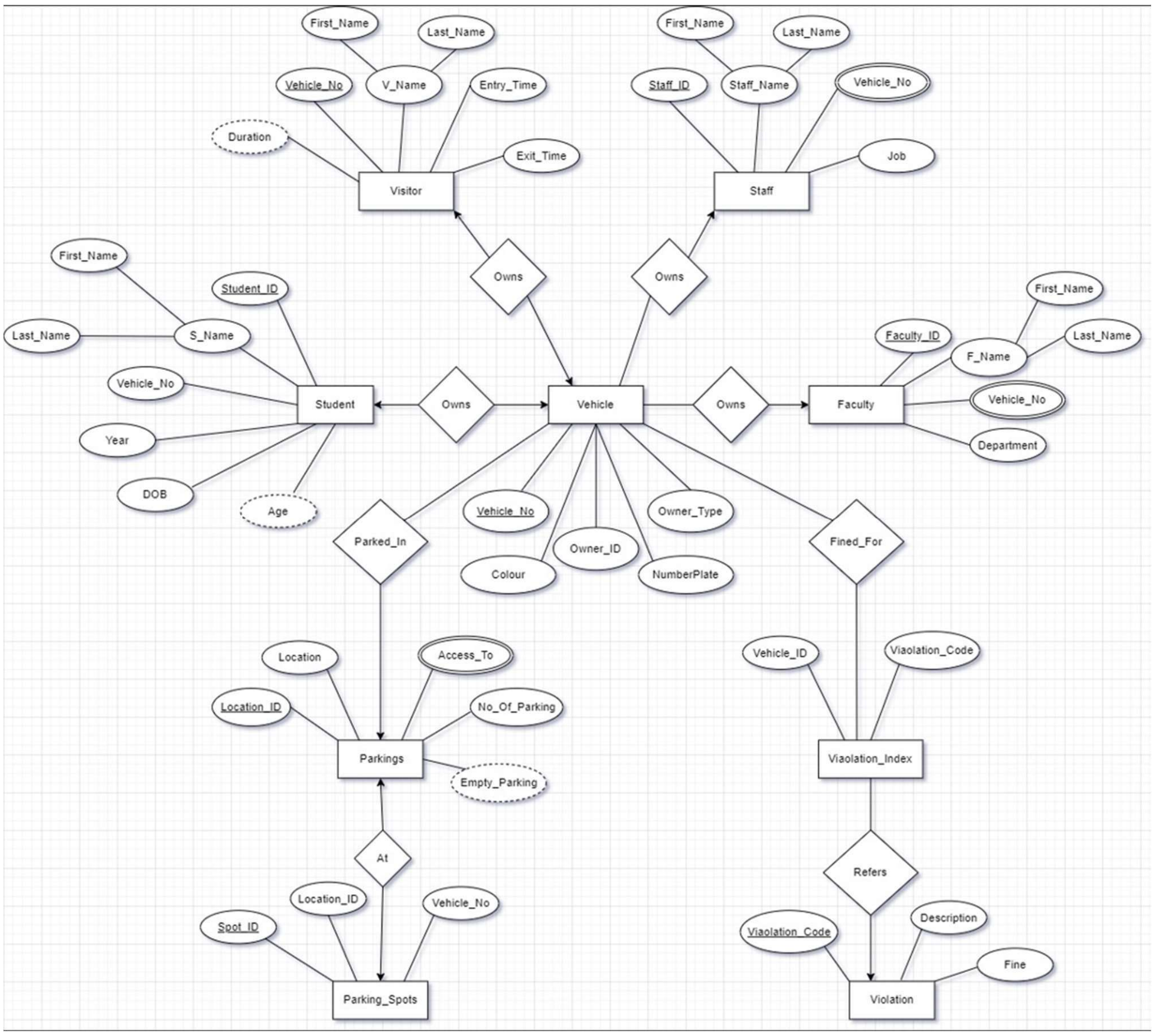
6. Performance Analysis of Vehicle Number Plate Recognition System Using Template Matching Techniques:

   This paper compares the performance of various template matching techniques for license plate recognition. The authors evaluate the accuracy of four different template matching algorithms (Normalized Cross-Correlation, Sum of Absolute Differences, Sum of Squared Differences, and Phase-Only Correlation) on a dataset of vehicle images. The results show that the Phase-Only Correlation technique achieves the highest accuracy in recognizing license plates.

Overall, these papers propose different approaches to license plate detection and recognition using computer vision techniques, with varying levels of accuracy and applicability to different types of license plates. These systems have potential applications in automated parking systems, traffic management, and law enforcement.

ER DIAGRAM

Our ER table diagram according to which our entries will be made to table after identifying number in the number plates.



METHODOLOGY

Explanation of FUNCTIONS used:

1. cv2.VideoCapture(): This function is used to create a VideoCapture object that can read video frames from a video file or a camera.
2. cv2.dnn.readNetFromDarknet(): This function is used to load the neural network architecture and its pre-trained weights from the disk.
3. net.setPreferableBackend(): This function sets the preferred computation backend for the neural network. In this case, the OpenCV backend is used.
4. net.setPreferableTarget(): This function sets the preferred target device for the neural network. In this case, the CPU is used.
5. cv2.dnn.blobFromImage(): This function is used to preprocess the input image before passing it through the neural network. It performs scaling, normalization, and resizing of the image to the desired size.
6. net.setInput(): This function sets the input blob for the neural network.
7. net.getLayerNames(): This function returns the names of all the layers in the neural network.
8. net.getUnconnectedOutLayers(): This function returns the indices of the output layers in the neural network.
9. net.forward(): This function performs forward propagation in the neural network and returns the output feature maps of the output layers.
10. cv2.dnn.NMSBoxes(): This function performs non-maximum suppression on the bounding boxes generated by the neural network to eliminate duplicate detections.
11. cv2.rectangle(): This function draws a rectangle on an image at the specified location and with the specified color and thickness.
12. cv2.putText(): This function draws text on an image at the specified location and with the specified font, scale, color, and thickness.
13. cv2.cvtColor(): This function converts an image from one color space to another.
14. easyocr.Reader(): This function creates an instance of the OCR reader from the EasyOCR library.
15. reader.readtext(): This function performs OCR on an image and returns the text detected in the image.

16. cap.release(): This function releases the VideoCapture object and frees up the resources it was using.
17. cv2.destroyAllWindows(): This function destroys all the windows created by the OpenCV library.
18. most_frequent_string(arr): This function retruns the most frequently occurring license plate number.
19. Remove_special_charachter(): This function removes any special characters detected by mistake from the number plate.

Overall, the code reads video frames from a file, preprocesses them, passes them through a neural network, performs object detection and OCR on the detected objects, and displays the results. If no text is detected in any frame, the code exits.

Step By Step Explanation:

The code uses computer vision techniques and a pre-trained deep learning model to detect license plates in a video stream and perform optical character recognition (OCR) to extract the license plate number.

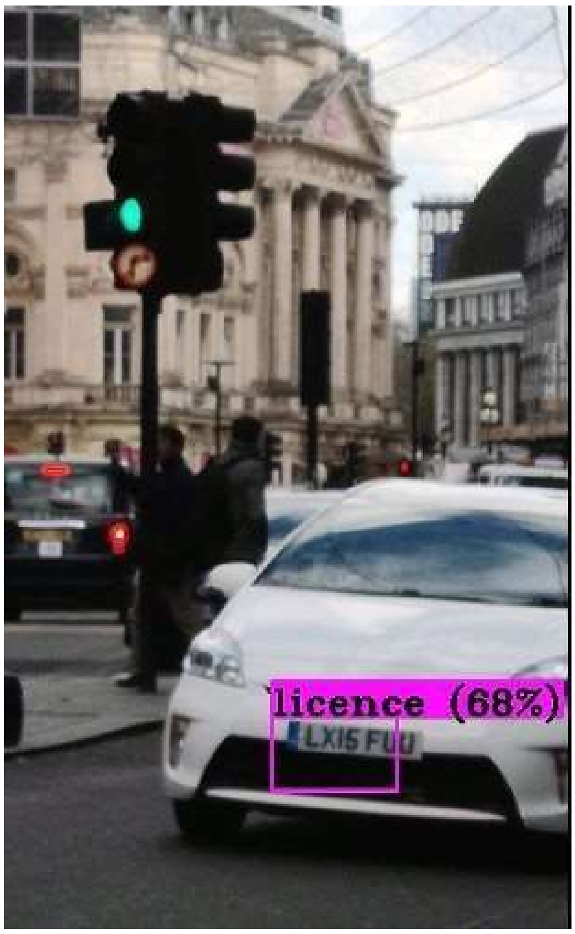
The code step by step:

1. Train your custom YOLO V4 model using darknet on custom dataset, download the generated weights and custom cfg file to run detection using this model.

2. Import the required libraries - cv2 for computer vision tasks, numpy for numerical operations, time to measure time, os.path for file path operations, easyocr for OCR, matplotlib for visualization, and sqlite3 for database operations. If using local env install pytorch as easyocr requires it, also install CUDA if you want to use your gpu to accelerate the process.
3. Set the configuration parameters for the deep learning model, such as the input image size (whT), the confidence threshold (confThreshold), and the non-maximum suppression threshold (nmsThreshold).
4. Read the class names from a file (classesFile) and load the pre-trained YOLOv4 model from disk using the configuration file (modelConfiguration) and the weights file (modelWeights).
5. Define a function findObjects that takes the YOLOv4 model outputs and the input image as inputs and uses them to extract the bounding boxes, class IDs, and confidence scores of the detected objects. Non-maximum suppression is applied to remove overlapping boxes, and the function returns the cropped license plate image.
6. Set up a video capture object (cap) to read frames from a video file (./tvideo2.mp4).

7. In a while loop, read a frame from the video stream and preprocess it by resizing it to the input size required by the YOLOv4 model and normalizing its pixel values.
8. Pass the preprocessed image through the YOLOv4 model and extract the outputs.
9. Call the findObjects function on the model outputs and input image.
10. If the license plate is successfully detected and recognized, print the license plate number.
11. If the license plate is not detected or recognized, continue reading the next frame until a license plate is detected and recognized or the end of the video stream is reached.
12. Use Remove_special_charachter() to remove any special characters from the license plate number detected and save all the detected license plate number in an array.
13. Using most_frequent_string(arr) finds the most frequently occurring number for precise detection.
14. The detected cars details will be stored in database by use of sqlite3.
15. The vehicles license plate and entry time will be captured at time of entry and at the time of exit exit time will be printed.
16. Release the video capture object and close all windows.

DETECTION OUTPUT



CNN

Yolo is an object detection algorithm it is uses convolution neural network layer to detect and localize objects. CNNs are one of the various types of neural networks like ANN and RNN.

A deep learning CNN consists of three layers: a convolutional layer, a pooling layer and a fully connected (FC) layer.

Convolutional layer is used to set filters on the image, after each iteration the value of the filter is updated a dot product is calculated between input pixels and filter. Then the image is converted into numerical values.

Pooling layer is mostly used to make the CNN faster as the convolutional layers are slow, we need pooling layer to reduce the information it sometime leads to information loss but mostly makes the CNN more efficient. It should be used carefully and properly

Fully connected layer here the image classification takes place in the CNN based on the features extracted in the previous layers i.e., convolutional and pooling layer.

There are two types of detector algorithms: -

1. Two-stage detectors

2. One-stage detectors

<u>YOLO</u>

Yolo comes under a one stage detector.

Single-shot object detection uses a single pass of the input image to make predictions about the presence and location of objects in the image. It makes these algorithms less effective in identifying small objects but faster and can be used to identify objects in real-time.

It stands for you only look once. It uses a single fully connected layer to perform all its predictions. It is the best real time object detecting algorithms.

The first 20 convolution layers of the model are pre-trained by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection. final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. To find the best bounding box yolo uses IOU the bounding box with the highest IOU is kept others are discarded.

YOLO v4:

YOLOv4 (You Only Look Once version 4) is a state-of-the-art object detection system that was introduced in 2020. It is an improvement over previous versions of YOLO and is considered one of the most accurate real-time object detection models available.

The main improvements of YOLOv4 over previous versions are its architecture, which includes a spatial pyramid pooling module, a path-aggregation network, and a modified residual network, all of which help improve the model's accuracy. YOLOv4 also includes improvements in training techniques, such as a better data augmentation strategy and a new loss function that helps to reduce false positives.

Key features of YOLOv4 include its ability to detect multiple objects in a single image and to classify those objects accurately. It is also capable of detecting small objects and objects in cluttered environments. Also, YOLOv4 is known for its fast processing time, making it ideal for real-time applications. Overall, YOLOv4 is a powerful tool for object detection and has been widely adopted in various fields, including autonomous driving, security, and robotics.

FLOW of the CODE

1. Import necessary libraries and define constants and variables.
2. Load the YOLOv4 model architecture and weights.
3. Load the input image and perform pre-processing.
4. Pass the pre-processed image through the YOLOv4 model and obtain bounding box predictions and associated class probabilities.
5. Apply non-max suppression to remove overlapping bounding boxes.
6. Draw the remaining bounding boxes and associated class labels on the image.
7. Display the resulting image.
8. Clean up and release resources.

DATASET

https://www.kaggle.com/datasets/andrewmvd/car-plate-detection

https://www.kaggle.com/datasets/saisirishan/indian-vehicle-dataset

we have combined these datasets taken from KAGGLE.

And also, we have many random videos of cars from parking lot and from online videos.

Car Plate detection

The final dataset contains 1035 images with bounding box annotations of the car license plates within the image.

Annotations are provided in the PASCAL VOC format.

The pascal VOC format was converted to yolo v5 format [class x_center y_center width height (in a txt file)] using python script.

RESULT

The outcomes or results of this project can include:

1. Improved security: By using AI to identify staff, students, and other people based on their number plates, the parking lot can be made more secure. Unauthorized individuals can be detected, and the system can alert security personnel or deny access.
2. Reduced wait times: Since the system can quickly identify individuals, it can reduce wait times at the entrance and exit of the parking lot.
3. Enhanced efficiency: By automating the identification process, the system can reduce the need for manual checks, which can improve the efficiency of parking management.
4. Data analytics: The system can also provide data analytics about parking usage, such as peak times and occupancy rates. This information can be used to optimize parking lot management and inform future planning decisions.

Overall, this project can enhance the security, efficiency, and overall management of parking lots by utilizing AI and number plate recognition technology.

FUTURE SCOPE

1. Improved accuracy: One area of improvement for the project could be to increase the accuracy of the object detection algorithm. This can be achieved by experimenting with different architectures or by fine-tuning the existing YOLOv4 model.
2. Real-time object detection: Another potential future scope could be to optimize the object detection algorithm to enable real-time detection of objects in video streams or live camera feeds.
3. Multi-object tracking: Our project detects objects in individual frames. Future work can involve into multi-object tracking algorithm to track objects across multiple frames.
4. Object classification: Currently, the model detects objects but does not classify them. Incorporating a classification algorithm could enable the model to identify the type of object detected.
5. Cloud deployment: The current implementation is designed to run locally on a machine. A potential future scope could be to deploy the model on cloud infrastructure for easy access and scalability.

6. Integration with other technologies: The object detection project could be integrated with other technologies such as robotics or autonomous vehicles to enable automated decision-making based on the detected objects.

These are just a few potential areas of future work for the object detection project. The direction of future work will depend on the specific needs and goals of the project.