Automated Parking System

Problem Statement:

The existing parking systems especially if it is a paid parking place, are either fully manual which involves the use of human attendants near the entrance and the exit to help the drivers throughout the whole process, or at least involve some kind of ticketing system to keep track of the amount of time the vehicle has been parked based on which the user will be charged while exiting the parking place, however, this method is time-consuming and often leads to congestion and frustration among drivers. Our parking system aims to provide a user-friendly and seamless experience for drivers by eliminating the entire ticketing system.

Functionalities:

1. Real-time and Automated License Plate Detection: YoloV8 allows us to retrieve and analyse license plates in real-time.
2. Automated Ticketing System: The process is quick and easy, and the user only needs to pay at the exit.
3. No workforce: Our project fully eliminates the need for human assistants.

Architecture & Design:

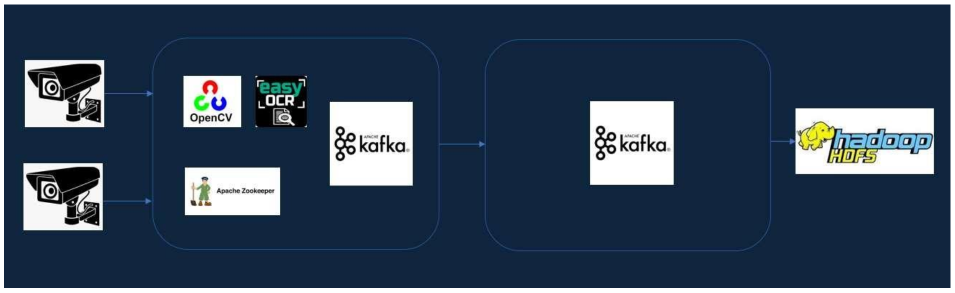
Architecture Flow:

1. Using the CCTV footage which is placed near the entrance of the parking place, we will be feeding the video to our Machine learning model which is currently running on Google Colab. The model is being trained using YOLO object detection algorithm and Easy OCR for optical character recognition for license plate detection.
2. The Kafka producer which will send cropped license plate images with the file name which is actually the number plate value to the Kafka Consumer.
3. Kafka Consumer will write them to HDFS with timestamp, license plate number and cropped license plate image
4. Using the CCTV footage which is placed near the exit of the parking place, we will be feeding the video to our Machine learning model which is currently running on Google Colab.

5. The Kafka producer which will send the license plate images to the Kafka Consumer.
6. Kafka consumer will be used to read the detected license plate numbers from Kafka topic and fetch the value from the HDFS with a timestamp and license plate number and calculate the amount the driver has to pay while exiting based on the amount of time the vehicle has been parked.

Hadoop HDFS Architecture:

1. One Data Node
2. One Name Node



GitHub Location of Code:

https://github.com/tejaasreddy001/Automated_Parking_System

Deployment Instructions:

Requirements:

1. Google Colab with GPU runtime
2. Python 3.11.0 or higher
3. Hadoop Distributed File System (HDFS)
4. Apache Kafka
5. Apache ZooKeeper

Tools and Technologies:

1. YOLOv8: a real-time object detection and image segmentation model based on deep learning.
   - The YOLO (You Only Look Once) object detection method is used to quickly identify and categorize objects in an image.
   - The fully convolutional neural network that forms the foundation of the YOLO algorithm separates the input image into a grid of cells.
   - The bounding boxes, confidence scores for each box, and class probabilities for each object are all predicted in each cell of the grid. A loss function that penalizes failures in both localization and classification is optimized by the algorithm over time.

2. EasyOCR: a Python module for extracting text from images.
    - The EasyOCR module is a Python-based OCR (Optical Character Recognition) engine that can extract text from images and turn it into machine-readable text.
    - The module supports more than 70 languages and is based on deep learning.
    - To increase the model's accuracy, The model is trained on a large dataset of images and text.
    - The EasyOCR module can recognize text in a variety of fonts and sizes and enables multi-language text recognition.
    - Additionally, it can handle low-quality photos and fix any skewing or distortion to increase accuracy.

3. OpenCV: a library of programming functions mainly used for real-time computer vision tasks.
    - OpenCV (Open Source Computer Vision) is an open-source library of computer vision algorithms and tools.
    - It is a cross-platform library that offers many different methods and algorithms for processing images and videos, finding and tracking objects, and machine learning.
    - OpenCV is written in C++ and also provides Python bindings, making it accessible to a larger group of developers.
    - It has a sizable and vibrant community of contributors who are always updating and expanding the library's functionality.
    - Some of the key features of OpenCV include: Image and video processing, Object detection and tracking, Machine learning, Camera calibration and 3D reconstruction, User interface.

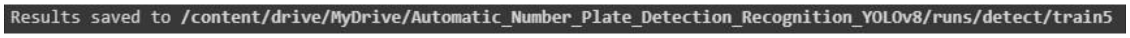
4. Convolutional Neural Networks (CNNs): a class of artificial neural networks commonly applied to analyse visual imagery and used in deep learning for computer vision tasks.
    - CNNs consist of several layers, including convolutional, pooling, and fully connected layers.
    - An image is used as the network's input and is processed through several convolutional layers.
    - A series of filters are applied by each convolutional layer to the input picture, which then undergoes a convolution process to create feature maps.
    - The edges, corners, and textures are among the various features of the image that these feature maps capture.
    - CNNs are trained using a process called backpropagation that modifies the network's weights to reduce the discrepancy between expected and actual results.
    - A huge number of labeled images are sent into the network during training, and the output is then compared to the ground truth label.
    - CNNs are a powerful type of neural network that can learn hierarchical representations of images by applying convolutional filters and pooling operations.

Steps to Run the Application:

For Machine Learning Part

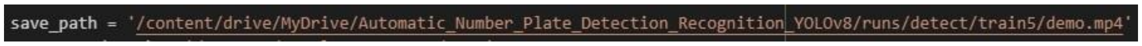
1. The GitHub links contains the below file
    a. Automatic_Number_Plate_Detection_Recognition_YOLOv8_new.zip
    b. Colab Code.ipynb
    c. Colab Code To Test.ipynb
    d. CroppedImages.zip
    e. cars.mp4
    f. consumer.ipynb
    g. demo.mp4
    h. predict.py
    i. producer.ipynb

2. Open Colab Code.ipynb on google colab (Runtime as GPU)
3. Unzip and load Automatic_Number_Plate_Detection_Recognition_YOLOv8_new.zip folder onto the google colab (Runtime as GPU)
4. Run all the cells of Colab Code (Runtime as GPU)
5. For running cell 6 if you face any issues, please create a roboflow account for accessing their dataset. An api will be assigned which you can use in the code.
6. While running 18th cell note where the results are saved



```
Results saved to /content/drive/MyDrive/Automatic_Number_Plate_Detection_Recognition_YOLOv8/runs/detect/train5
```

- Modify the save path variable with the value on 20th cell. Note: keep the .mp4 same as your sources .mp4 name



```
save_path = '/content/drive/MyDrive/Automatic_Number_Plate_Detection_Recognition_YOLOv8/runs/detect/train5/demo.mp4'
```

- Same goes while running the cell 22nd and 24th cell.
- **Note:** If you get an error while downloading any files from drive please use the file given on github
    - predict.py file is uploaded
    - cars.mp4 file is uploaded
    - demo.mp4 file is uploaded

7. For Hadoop Part:

    Install Hadoop HDFS on your local machine:

    a. Delete the files in datanode folder
    b. Delete the files in namenode folder
    c. Run the following commands
        i. #hadoop namenode -format
        ii. #star-all.cmd
        iii. #jps

8. <u>For Apache Kafka Part:</u>

   Install Apache Kafka & Zookeeper

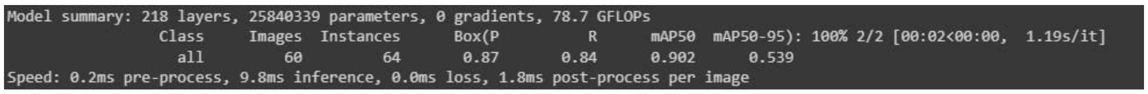
   a. Run the following commands
      i. C:\kafka>.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
      ii. C:\kafka>.\bin\windows\kafka-server-start.bat .\config\server.properties
      iii. C:\kafka\bin\windows>kafka-topics.bat --create --bootstrap-server localhost:9092 --topic test
      iv. C:\kafka\bin\windows>kafka-console-producer.bat --broker-list localhost:9092 --topic test
      v. C:\kafka\bin\windows>kafka-console-consumer.bat --topic test --bootstrap-server localhost:9092 --from-beginning
   b. Download the CroppedImages.zip and unzip it from github and store it locally
   c. Download the producer.ipynb and consumer.ipynb file from github
      i. Modify the path of the folder_path = 'C:/Users/tejas/OneDrive/Desktop/Latest/DB/CroppedImages/CroppedImages' based on where you have downloaded and unzipped CroppedImages folder
      ii. Run consumer.ipynb first and then run producer.ipynb
      iii. You will see the data being loaded to HDFS
      iv. Run producer code again and you will see the amount being displayed in consumer.ipynb

<u>In Github: Readme file has google colab link for our project for license plate detection.</u>

<u>Test Results:</u>

<u>Accuracy of Model:</u>

1. After training the model for 120 epochs our model has achieved the below accuracy for detecting the license plate



```
Model summary: 218 layers, 25840339 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 2/2 [00:02<00:00,  1.19s/it]
                   all         60         64       0.87       0.84      0.902      0.539
Speed: 0.2ms pre-process, 9.8ms inference, 0.0ms loss, 1.8ms post-process per image
```

2. Our model achieved a mAP50 of 0.902 and an mAP50-95 of 0.539
3. mAP50 (mean average precision at 50% intersection over union threshold) measures the overall precision of the model across different object categories, with a threshold of 50%.
   An mAP50 score of 0.902 indicates that your model performs well in detecting objects and localizing their bounding boxes with a high degree of precision.
4. mAP50-95 is another evaluation metric that measures the model's precision across different IoU thresholds (from 50% to 95%).
   An mAP50-95 of 0.539 suggests that your mdel has some room for improvement in terms of correctly identifying and localizing objects across different IoU thresholds.

EasyOCR:

1. The results show that EasyOCR has resulted in more than 95% accuracy for predicting the number plate when compared to Tesseract OCR which has only resulted in 90% accuracy.

Conclusion:

In conclusion, the present parking systems, particularly in paid parking zones, mainly rely on time-consuming manual procedures or ticketing systems that aggravate and annoy drivers. Our approach tackles these issues by implementing a seamless parking system that is user-friendly and does away with the requirement for tickets. We provide drivers with a simplified experience by utilizing cutting-edge technologies, such as automated license plate detection using YOLOv8, which minimizes waiting periods and the involvement of human attendants. In order to make parking convenient and hassle-free for all users, we want to completely transform the parking experience.