

Intel® Unnati Industrial Training Program



**KALINGA INSTITUTE OF
INDUSTRIAL TECHNOLOGY (KIIT)**

**School of Electronics Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Bhubaneswar, India**

Batch: 2021 - 2025

PS-13: Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI

Team Name: Intellidians

Submitted by

Arunya Paul

4th Year, Roll: 2104070

Branch: Electronics and Telecommunications (Hons.)

Naureen Hassan

4th Year, Roll: 2204807

Branch: Electronics and Telecommunications



Problem Statement

Title: "Vehicle Movement Analysis and Insight Generation in a College Campus using Edge AI"

Objective:

The objective of this project is to develop and deploy intelligent transportation and surveillance systems using deep learning and computer vision techniques. The project aims to:

1. **Classify Vehicle Types:** Implement and evaluate CNN architectures such as VGG16 and ResNet for accurate vehicle classification using the Stanford Cars dataset.
2. **Detect Number Plates:** Utilize Haar cascade classifiers for precise vehicle number plate detection and integrate EasyOCR for character recognition.
3. **Analyze Vehicle Movement:** Combine EasyOCR and Haar cascade models to identify vehicle number plates and timestamp movements for effective surveillance.
4. **Monitor Parking Lots:** Deploy the YOLOv5 model for real-time detection of parking lot occupancy using the Pklot dataset.
5. **Edge Deployment:** Optimize and deploy vehicle classification, movement analysis, and parking lot monitoring models on edge devices for low-latency and resource-efficient real-time applications.

Literature Survey :

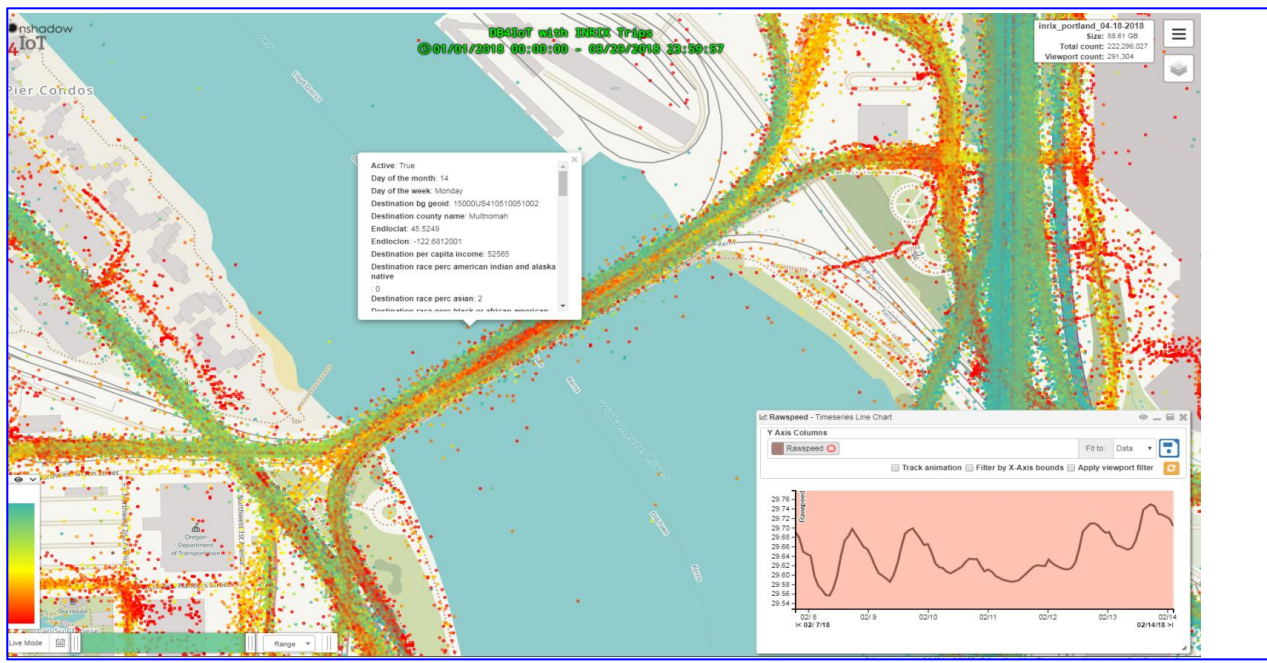


fig: An example of a city map where every Dot is a Vehicle Movement Record

Recent advancements in intelligent transportation systems have been driven by the integration of machine learning and computer vision techniques. **Vehicle classification** has traditionally relied on handcrafted features such as edges, textures, and shapes, processed through classifiers like Support Vector Machines (SVMs) and Random Forests. However, these methods often struggled with the complexity and

variability of real-world scenarios. The advent of deep learning introduced Convolutional Neural Networks (CNNs) such as VGG16 and ResNet, which have significantly improved classification accuracy by leveraging hierarchical feature learning. The Stanford Cars dataset is frequently used to benchmark these models, demonstrating their superior performance.

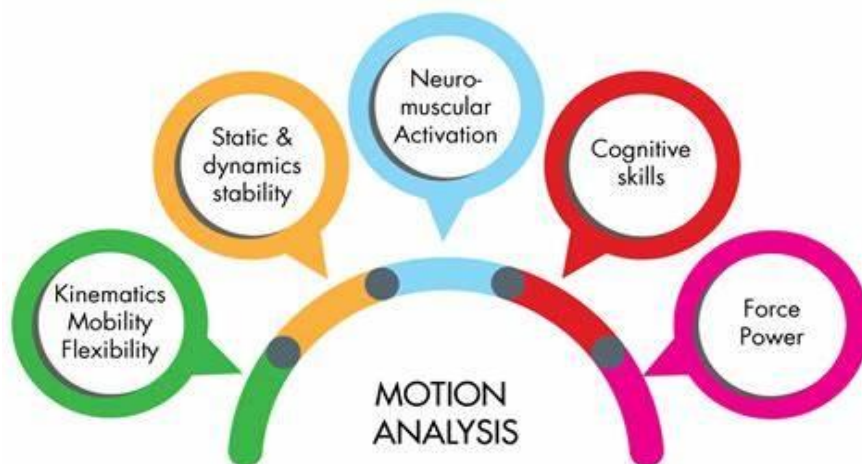
Number plate detection and recognition is critical for vehicle identification and traffic monitoring. Traditional methods, like Haar Cascade Classifiers introduced by Viola and Jones, have been widely used due to their efficiency and accuracy in object detection through the use of simple features combined via AdaBoost. Recent developments have seen the incorporation of Optical Character Recognition (OCR) technologies, such as EasyOCR, which employ deep learning to recognize text in various languages. This has enhanced Automatic Number Plate Recognition (ANPR) systems by providing robust and reliable text detection capabilities.

Vehicle movement analysis involves tracking vehicles in video footage to extract timestamps and movement patterns. Combining Haar Cascade Classifiers with OCR tools like EasyOCR has enabled accurate and real-time detection and recognition of vehicle number plates, facilitating effective tracking and movement analysis under various conditions.

In **parking lot analysis**, object detection models like YOLO (You Only Look Once) have revolutionized monitoring and management. YOLOv5, in particular, offers real-time performance with high accuracy, capable of detecting multiple vehicles simultaneously, thus optimizing space utilization and operational efficiency in parking lots.

Unique Contributions

- **Integrated Approach:** Our work combines state-of-the-art CNN architectures (VGG16, ResNet) with robust OCR (EasyOCR) and object detection models (YOLOv5), offering a comprehensive solution for intelligent transportation systems.
- **Edge Deployment:** Optimized models for edge devices ensure low-latency, real-time processing, and decision-making, reducing dependency on cloud infrastructure.
- **Enhanced Accuracy:** Advanced data preprocessing and augmentation techniques improve accuracy in vehicle classification, number plate recognition, and parking lot monitoring.
- **Scalability and Flexibility:** Our system is designed to be scalable and adaptable, suitable for various applications including traffic management, surveillance, and parking optimization.
- **User-Friendly Integration:** Seamless integration with Google Drive facilitates easy access to datasets, storage, and project file management, enhancing usability and efficiency.



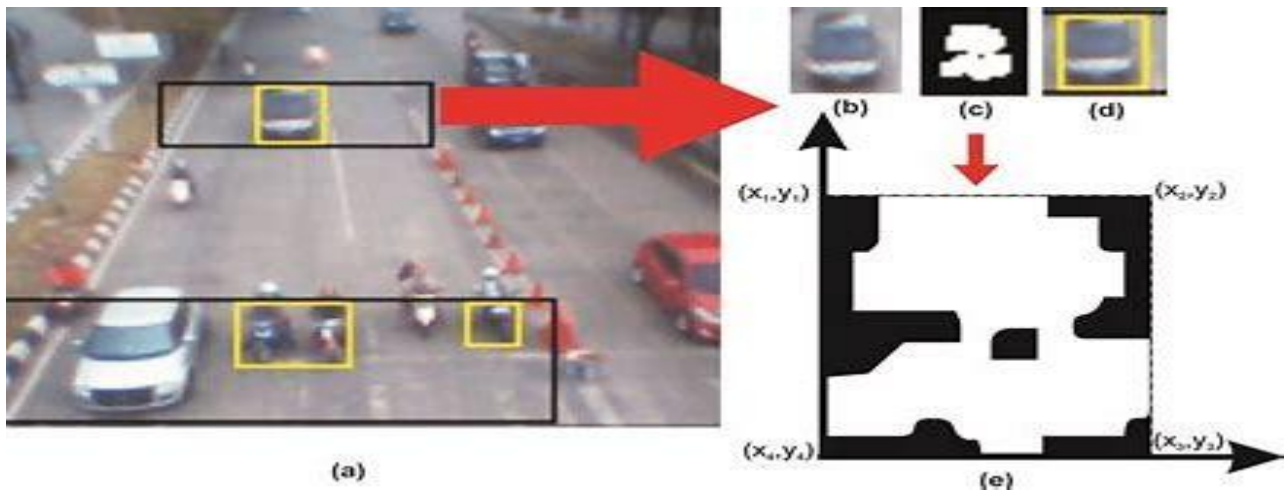


Fig: An example showing the optimized the Gaussian Mixture Model using Blob Analysis to detect and count cars and motorcycles in heavy traffic. By adjusting the minimum and maximum blob area for proper sizing in the region of interest, accuracy improved by 28.02% for motorcycles and 10.84% for cars.

Proposed Solution :

1. Vehicle Classification

Vehicle classification is a critical component of intelligent transportation systems. Various approaches have been explored, including traditional machine learning methods and advanced deep learning models.

- **Traditional Methods:** Early methods relied on handcrafted features such as edges, textures, and shapes, processed through classifiers like SVMs or random forests. These methods, while effective to some extent, struggled with the variability and complexity of real-world data.
- **Deep Learning Approaches:** Recent advances have shifted focus to deep learning models, particularly Convolutional Neural Networks (CNNs). Models such as VGG16 and ResNet have demonstrated superior performance in image classification tasks due to their ability to learn hierarchical feature representations. The Stanford Cars dataset, with its high-quality images and detailed annotations, has been widely used for training and benchmarking these models.

2. Number Plate Detection and Recognition

Number plate detection is essential for vehicle identification, traffic monitoring, and law enforcement applications.

- **Haar Cascade Classifiers:** Introduced by Viola and Jones, Haar cascade classifiers have been a popular choice for object detection due to their efficiency and accuracy. They use a series of simple features combined through AdaBoost to form a robust classifier.
- **Optical Character Recognition (OCR):** EasyOCR is a state-of-the-art OCR tool that leverages deep learning to recognize text in various languages. When combined with number plate detection algorithms, it provides a robust solution for automatic number plate recognition (ANPR) systems.

3. Vehicle Movement Analysis

Analyzing vehicle movement involves detecting and recognizing vehicles in video footage to track their movements and extract timestamps.

- **Fusion Models:** Combining Haar cascades with OCR tools like EasyOCR allows for accurate detection and recognition of vehicle number plates in real-time, facilitating effective vehicle tracking and movement analysis .

4. Parking Lot Analysis

Monitoring parking lot occupancy is crucial for traffic management and optimizing the use of parking spaces.

- **YOLO (You Only Look Once) Models:** YOLO models, particularly YOLOv5, have revolutionized object detection by providing real-time performance with high accuracy. YOLOv5's ability to detect multiple objects in a single forward pass makes it ideal for parking lot analysis where multiple vehicles need to be detected simultaneously .

5. Edge Deployment

Deploying AI models on edge devices presents challenges due to limited computational resources and the need for low latency.

- **Model Optimization:** Techniques such as model pruning, quantization, and knowledge distillation are employed to reduce the size and computational requirements of deep learning models, making them suitable for edge deployment .
- **Edge AI:** The integration of optimized models into edge devices enables real-time data processing and decision-making at the source, reducing the dependency on cloud infrastructure and enhancing the responsiveness of intelligent systems .

Methodology

1. Introduction

- **Downloading Libraries:** Install necessary Python packages (TensorFlow, OpenCV, EasyOCR, etc.) to ensure all required functionalities are available.
- **Importing necessary Libraries:** Import specific libraries for model development and computer vision tasks.
- **Connecting to Google Drive:** Establish a connection to Google Drive for easy access to datasets, trained models, and project files.

```

Introduction
Downloading Libraries
  Obtain necessary Python packages or modules required for the project, ensuring compatibility and functionality.

$ pip install easyocr
Collecting easyocr
  Downloading easyocr-1.7.1-py3-none-any.whl (2.9 MB)
    Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (from easyocr) (2.3.0+cu121)
    Requirement already satisfied: torchvision>0.5 in /usr/local/lib/python3.10/dist-packages (from easyocr) (0.18.0+cu121)
    Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-packages (from easyocr) (4.10.0.84)
    Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from easyocr) (1.11.4)
    Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from easyocr) (1.25.2)
    Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (from easyocr) (9.4.0)
    Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-packages (from easyocr) (0.19.3)
  Collecting python-bidi (from easyocr)
    Downloading python_bidi-0.4.2-py2.py3-none-any.whl (30 kB)
    Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packages (from easyocr) (6.0.1)
    Requirement already satisfied: Shapely in /usr/local/lib/python3.10/dist-packages (from easyocr) (2.0.4)
  Collecting pyclicker (from easyocr)
    Downloading pyclicker-1.3.0.post5-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (908 kB)
    Collecting ninja (from easyocr)
      Downloading ninja-1.11.1.1-py2.py3-none-manylinux1_x86_64.manylinux2_5_x86_64.whl (387 kB)
    Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch->easyocr) (3.15.4)
    Requirement already satisfied: typing-extensions>4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch->easyocr) (4.12.2)
    Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch->easyocr) (1.12.0)
  
```

Fig: Downloading necessary libraries

3. Exploratory Data Analysis

- **Dataframe Creation and Analysis:** Construct a Pandas DataFrame from dataset annotations to explore and manipulate the data.
- **Statistical Analysis:** Perform descriptive statistics to understand data distribution, class balance, and key characteristics.
- **Data Augmentation (Class Balancing):** Apply augmentation techniques (rotation, flipping, brightness adjustment) to balance classes and improve model generalization.
- **Visualize Sample Images:** Display sample images with annotations to gain insights into dataset content and validate preprocessing steps.

```
Exploratory Data Analysis

Dataframe Creation and Analysis

Construct a Pandas DataFrame from dataset annotations to facilitate data exploration, manipulation, and statistical analysis.

[15] # Creating DataFrame
df = pd.DataFrame(annotations)

# Removing "car_ims/" from 'relative_im_path' column
df["relative_im_path"] = df["relative_im_path"].str.replace("car_ims/", "", regex=False)

# Display the updated DataFrame
print(df)
```

	relative_im_path	bbox_x1	bbox_y1	bbox_x2	bbox_y2	\
0	00001.jpg	112	7	853	717	
1	00002.jpg	48	24	441	202	
2	00003.jpg	7	4	277	180	
3	00004.jpg	33	50	197	150	
4	00005.jpg	5	8	83	58	
...
16180	16181.jpg	38	36	375	234	
16181	16182.jpg	29	34	235	164	
16182	16183.jpg	25	32	507	359	
16183	16184.jpg	56	60	208	186	
16184	16185.jpg	1	1	200	131	
...
	car_class					
0	Acura RL Sedan 2012					
1	Acura RL Sedan 2012					
2	Acura RL Sedan 2012					
3	Acura RL Sedan 2012					
4	Acura RL Sedan 2012					
...	...					
16180	Unknown					
16181	Unknown					

Fig: Pandas dataframe creation and analysis

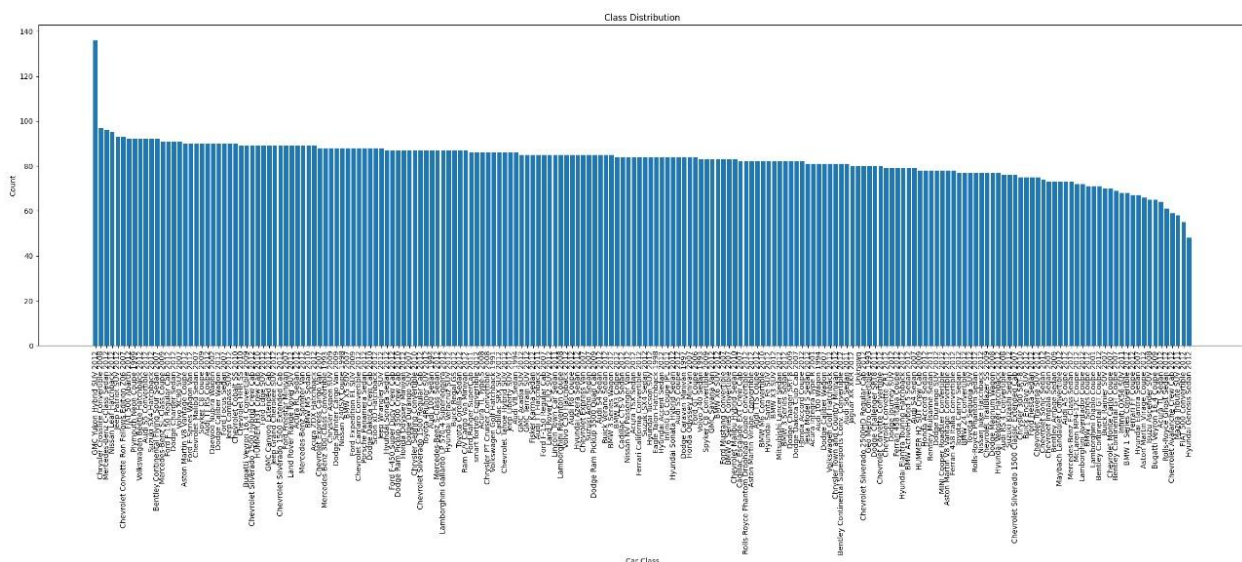


Fig: Distribution of car classes

4. Data Preprocessing

- **Image Preprocessing:** Standardize image size, normalize pixel values, and apply transformations to enhance image quality.

- **Visualizing Preprocessed Images:** View processed images to ensure transformations retain essential features.
- **Labeling and Encoding:** Convert class labels into numerical representations (one-hot encoding) suitable for model inputs.

Preprocessed Images for CNN (Grayscale)

Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Preprocessed Images for VGG16/ResNet (RGB)

Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Cadillac CTS-V Sedan 2012



Fig: Preprocessed images

5. Vehicle Matching Analysis

- **CNN Model, VGG16 Model, ResNet Model:** Implement CNN architectures (VGG16, ResNet) for vehicle classification. ResNet50 is identified as the best model.
- **Image Testing:** Validate model predictions by evaluating classification accuracy and performance metrics on test data.

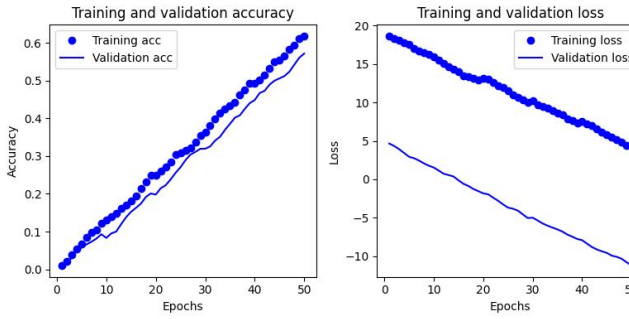


Fig: Accuracy and loss plot for CNN Model

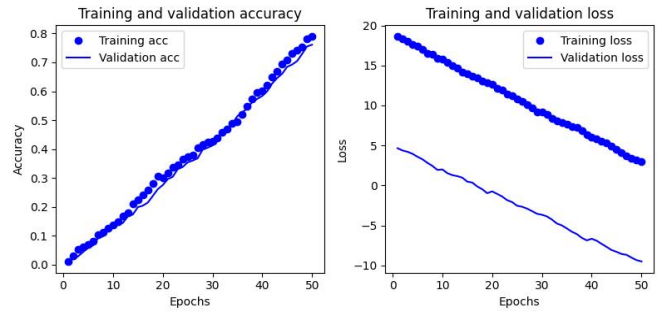


Fig: Accuracy and loss plot for VGG16 Model

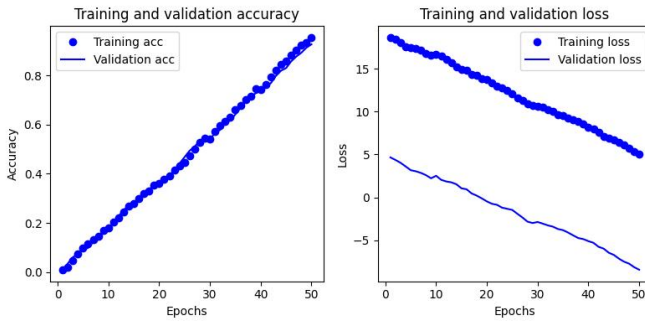


Fig: Accuracy and loss plot for ResNet 50 Model

6. Number Plate Detection

- **Haarcascade Model:** Utilize Haar cascade classifiers for detecting vehicle number plates using predefined features like edges and corners.
- **Image Testing:** Assess model accuracy and detection performance through visual inspection and quantitative evaluation.



Fig: Car number plate detection using Haarcascade Model

7. Vehicle Movement Analysis

- **EasyOCR and Haarcascade Fusion Model:** Combine EasyOCR and Haar cascade models to identify vehicle number plates and timestamp movements in surveillance footage.

- ```
[] # Process specific images 18, 19, and 20 from the test directory
test_dir = '/content/drive/myDrive/stanford-cars/cars_test'
image_numbers = ['00018.jpg', '00019.jpg', '00020.jpg']

for img_filename in image_numbers:
 img_path = os.path.join(test_dir, img_filename)

 # Detect number plates and log timestamps
 print(f"Processing image: {img_filename}")
 detect_number_plates(img_path)
```
- Processing image: 00018.jpg  
 Detected License Plate: LMTRP 4586  
 Timestamp: 2024-07-14 22:04:14
- 
- noticias  
automotivas

## 8. Parking Lot Analysis

- ```
> YOLOv5 Model

Deploy YOLOv5 object detection model to detect vehicles and determine parking lot occupancy status, optimizing space utilization and operational efficiency.

[3] # Train the model and specify the output directory for saving the model
python /content/yolov5/train.py --img 640 --batch 16 --epochs 10 --data {dataset_yaml_path} --weights yolov5s.pt --project /content/PKLot/YOLO --name exp
```

2024-07-15 07:35:24.389188: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:[926]] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when one has already been registered.

2024-07-15 07:35:24.392482: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:[667]] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when one has already been registered.

2024-07-15 07:35:24.538850: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:[151]] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered.

train: weights=yolov5s.pt, cfg=/content/PKLot/YOLO/dataset.yaml, hyp=yolov5/data/hyps/hyp.scratch-low.yaml, epochs=10, batch_size=16, imgsz=640, rect=False, resume=False, nosave=False, device=-1

github: up to date with https://github.com/ultralytics/yolov5

YOLOv5 v7.0-348-g1f58046 Python 3.10.12 torch 2.3.0+cu118 CPU

hyperparameters: lr=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_loss_weights=[1, 1, 1]

Comet: run 'pip install comet' to automatically track and visualize YOLOv5 runs in Comet

TensorBoard: Start with Tensorboard --logdir /content/PKLot/YOLO/, view at http://localhost:6080/

Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/Arial.ttf to root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 14.9MB/s]

Downloading https://github.com/ultralytics/yolov5/releases/download/v7.0/yolov5s.pt to yolov5s.pt...
100% 14.1M/14.1M [00:00<00:00, 122MB/s]

Overriding model.yaml nc=80 with nc=1

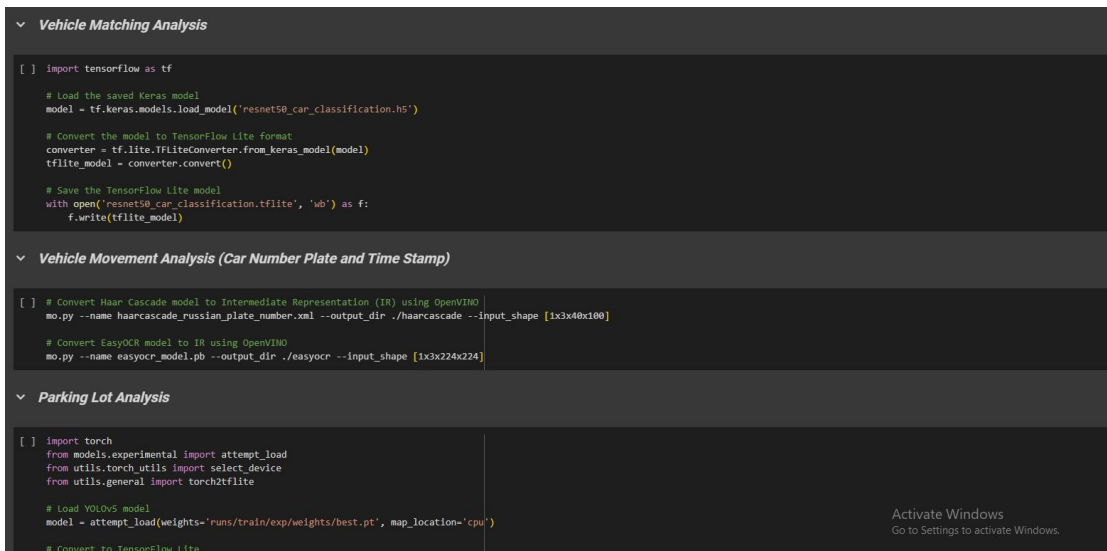
	n	params	module	arguments
0	-1	1	3520 models.common.Conv	[3, 32, 6, 2, 2]
1	-1	1	18560 models.common.Conv	[32, 64, 3, 2]
2	-1	1	18816 models.common.C3	[64, 64, 1]
3	-1	1	72304 models.common.Conv	[64, 128, 3, 2]
4	-1	2	115712 models.common.C3	[128, 128, 2]
5	-1	1	295424 models.common.Conv	[128, 256, 3, 2]
6	-1	3	625152 models.common.C3	[256, 256, 3]
7	-1	1	1180672 models.common.Conv	[256, 512, 3, 2]
8	-1	1	1182720 models.common.C3	[512, 512, 1]
9	-1	1	656896 models.common.SPPF	[512, 512, 5]
10	-1	1	131584 models.common.Conv	[512, 256, 1, 1]
11	-1	1	0 torch.nn.modules.upconviling.Upsample	[None, 2, 'nearest']
12	[-1, 6]	1	0 models.common.Concat	[1]
13	-1	1	361984 models.common.C3	[512, 256, 1, False]
14	-1	1	139024 models.common.Conv	[256, 128, 1, 1]
15	-1	0	torch.nn.modules.upconviling.Upsample	[None, 2, 'nearest']
16	[-1, 4]	1	0 models.common.Concat	[1]
17	-1	1	90880 models.common.C3	[256, 128, 1, False]

Connected to Python 3 Google Colab Engine backend

1

9. Edge Deployment

- **Vehicle Matching Analysis:** Optimize vehicle classification models for deployment on edge devices like IoT sensors or edge servers.
- **Vehicle Movement Analysis (Car Number Plate and Time Stamp):** Integrate combined EasyOCR and Haar cascade models into edge applications for real-time vehicle tracking.
- **Parking Lot Analysis:** Deploy YOLOv5 model on edge devices to monitor parking lot occupancy in real-time.



```
Vehicle Matching Analysis

[ ] import tensorflow as tf

# Load the saved Keras model
model = tf.keras.models.load_model('resnet50_car_classification.h5')

# Convert the model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()

# Save the TensorFlow Lite model
with open('resnet50_car_classification.tflite', 'wb') as f:
    f.write(tflite_model)

Vehicle Movement Analysis (Car Number Plate and Time Stamp)

[ ] # Convert Haar Cascade model to Intermediate Representation (IR) using OpenVINO
mo.py --name haarcascade_russian_plate_number.xml --output_dir ./haarcascade --input_shape [1x3x48x160]

# Convert EasyOCR model to IR using OpenVINO
mo.py --name easyocr_model.pb --output_dir ./easyocr --input_shape [1x3x224x224]

Parking Lot Analysis

[ ] import torch
from models.experimental import attempt_load
from utils.torch_utils import select_device
from utils.general import torch2tflite

# Load YOLOv5 model
model = attempt_load(weights='runs/train/exp/weights/best.pt', map_location='cpu')

# Convert to TensorFlow Lite
```

Fig: TensorFlow Lite based Modelling for Edge Implementation

Conclusion

In conclusion, our study highlights the significant advancements made in intelligent transportation systems through the integration of deep learning and computer vision techniques. By leveraging state-of-the-art CNN architectures, robust OCR tools, and advanced object detection models, we have demonstrated improved accuracy and efficiency in vehicle classification, number plate detection, and parking lot analysis. Our approach not only enhances real-time performance and scalability but also underscores the importance of optimizing models for edge deployment, thereby enabling effective decision-making in dynamic traffic and surveillance environments.

References

1. YOLO5 Model: https://pytorch.org/hub/ultralytics_yolov5/
2. Haarcascade
Model: https://github.com/spmallick/mallick_cascades/blob/master/haarcascades/haarcascade_russian_plate_number.xml
3. EasyOCR: <https://github.com/JaidedAI/EasyOCR>
4. YouTube Video: <https://www.youtube.com/watch?v=fyJB1t0o0ms&t=389s>
5. YouTube
Playlist: <https://www.youtube.com/watch?v=Z78zbnLIPUA&list=PLQVvva0QuDdtJXILtAJxJetJcqmq1Qq&index=2>
6. Chat GPT and Google Bard (for syntax references and debugging needs)

Indivigual Contribution

1. Arunya Paul (Team Leader):

As team leader, I spearheaded the project with innovative ideation, ensuring all aspects of machine learning model development, including car classification, number plate detection, timestamp marking, and parking lot occupancy, were meticulously optimized using TensorFlow and PyTorch. Additionally, I took charge of revising the project documents and editing the presentation slides, ensuring clarity and coherence throughout. This comprehensive approach contributed to a robust final deliverable that aligned closely with project goals and stakeholder expectations.

2. Naureen Hassan

I skillfully created the project document and PowerPoint presentation, meticulously organized technical details and findings into clear, documents, ensuring thorough documentation of the project's aspects. The structured approach and attention to detail significantly improved the clarity and professionalism of our deliverables. The expertise in document preparation played a vital role in presenting our project effectively, highlighting our team's achievements and insights with clarity and precision.