System To Detect And Classify Vehicles In Urban Street Scenes

- **Dataset Name: coco-2014-dataset (80 unique categories)**
- **filtered classes ==>> 'car', 'bus', 'truck'**
- **Custom method Object Detection using Bounding Boxes based on annotations of data**

❖ Object Detection: I chose YOLO model version 8 'yolov8n.pt'

- **Pre-trained YOLO model doesn't need to be trained which trained globally in big datasets like official coco datasets**
- **which is best for real-time performance rather than any other model**
- **don't need to download wights which is online for libraries.**

❖ Classification: using Custom CNN Using Torch vision(nn) & pycocotools Libraries

Some method of enhancements: series of transformations:

1. **Convert to PIL image**: The image is first converted to a PIL format for easier manipulation.
2. **Resize**: The image is resized to a fixed size (default is 224x224 pixels) to ensure consistency across inputs.
3. **Convert to Tensor**: The image is then converted to a PyTorch tensor for further processing.
4. **Normalize**: The image is normalized using mean and standard deviation values typically used in pre-trained models like ResNet. This step standardizes pixel values to improve model performance.
5. **Add batch dimension**: The image is unsqueezed to add a batch dimension, making it ready for input into a neural network.

❖ optimization techniques:

- **optimizer = optim.Adam(vehicle_classifier.parameters(), lr=0.001)**:
  The Adam optimizer is used to update the model's weights. It adjusts the parameters based on the gradients computed during backpropagation, with a learning rate of 0.001, balancing convergence speed and stability.
- **criterion = nn.CrossEntropyLoss()**:

Cross-entropy loss is chosen as the loss function, commonly used for multi-class classification. It measures the difference between the predicted output probabilities and the true class labels, guiding model optimization.

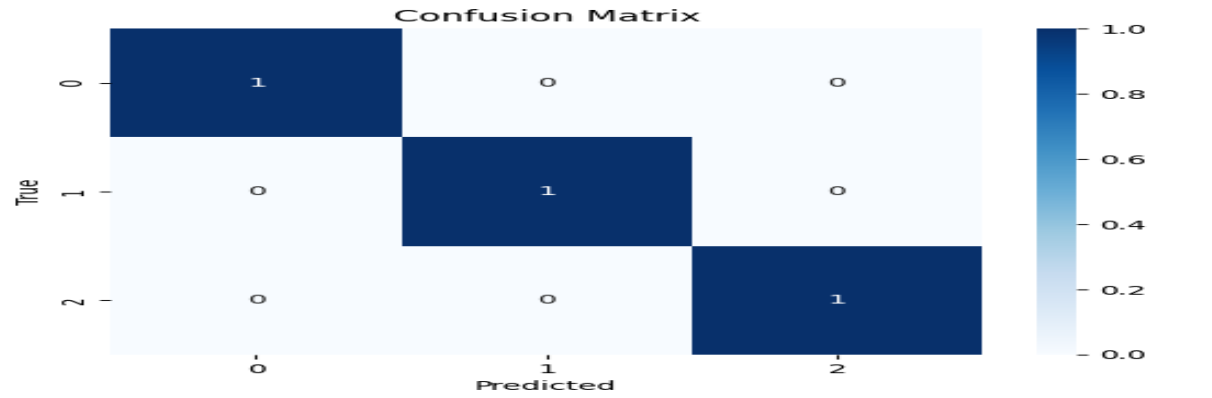
❖ Evaluation:

(F1 Score and Confusion Matrix) & Classification Report

```
F1 Score: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1
           2       1.00      1.00      1.00         1

    accuracy                           1.00         3
   macro avg       1.00      1.00      1.00         3
weighted avg       1.00      1.00      1.00         3
```



Simple Ui (CLI):

Due to working on Kaggle platform and visibility of dataset online,

I can't use Thinker or any other method of presenting UI which required to be in local pc program (PyCharm or notebook Juber), Only implement simple ui manage us to put path of image then do the test on it.

```
# Get user input
image_path = input("Enter the path to the image file: ")
process_and_display_image(image_path)
```

```
Enter the path to the image file:
```

```
Enter the path to the image file:  /kaggle/input/coco-2014-dataset-for-yolov3/coco2014/images/train2014/COCO_train2014_000000001518.jpg

0: 480x640 4 cars, 2 traffic lights, 9.5ms
Speed: 1.4ms preprocess, 9.5ms inference, 1.4ms postprocess per image at shape (1, 3, 480, 640)
```