Using the **"2. model_train.ipynb"** notebook, I finetuned a ***yolov8n*** model on the developed dataset on Kaggle. I imported the dataset from ROBOFLOW to KAGGLE directly.

The ***YOLOv8n.pt*** architecture is a convolutional neural network designed for object detection, primarily consisting of a "backbone" for feature extraction, a "neck" for feature fusion, and a "head" for generating final predictions, with the "n" indicating a smaller, faster model variant, utilizing advanced techniques like Cross-Stage Partial Connections (CSP) to improve efficiency and accuracy while maintaining real-time performance.

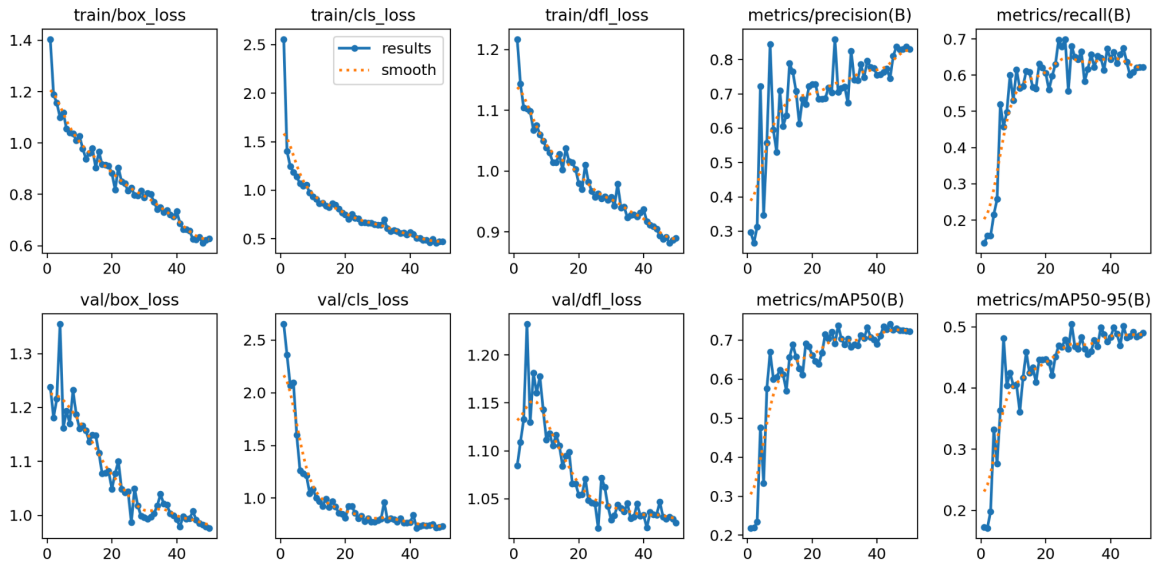
Below are the finetuning parameters.

```
results = target_model.train(
    data = DATA_YAML_PATH,
    epochs = 50,
    imgsz = 640,
    conf = 0.1,
    device = 'cuda:0'
)
```

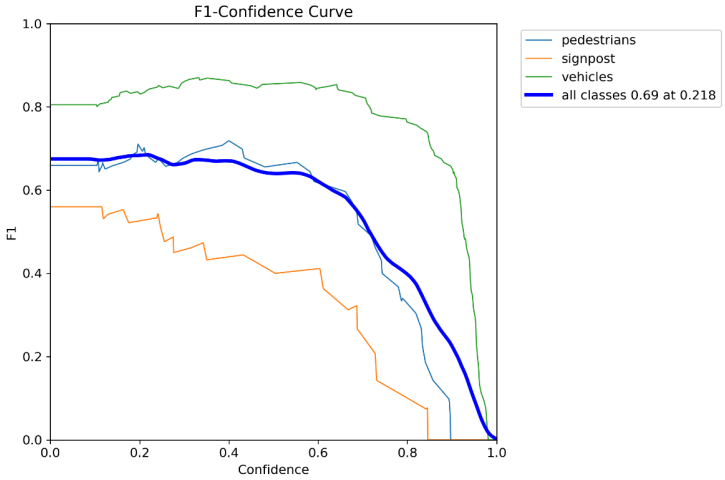
After being done with the finetuning, Looking at this generated model summary report, We can get a breakdown of key information:

1. 168 layers: The model consists of 168 trainable layers.
2. 3,006,233 parameters: The total number of model parameters, indicating its complexity.
3. 0 gradients: This suggests inference mode (no training happening).
4. 8.1 GFLOPs: Computational cost of processing one image.

I then plotted the train-valid loss curves to evaluate the performance of my model with the validation split and using these relevant metrics. The loss functions (box, cls, dfl) are decreasing, indicating that the model is learning effectively. The performance metrics (precision, recall, mAP) are increasing, suggesting improved detection capability.



Then I plotted the **F1-Confidence Curve** shown below which illustrates the relationship between confidence thresholds and the corresponding F1-score for different object classes in a classification or detection task.



I later showed all 15 predictions of the model from the validation split in in batch representation like shown below.

Class-wise Performance:

Classes	Precision	Recall	mAP50	mAP50-95
All	0.705	0.68	0.732	0.501
Pedestrians	0.73	0.692	0.758	0.485
Signposts	0.581	0.48	0.535	0.27
Vehicles	0.805	0.868	0.903	0.747

Finally, I saved the model in pth and pkl format also as well as converted it into onnx format.

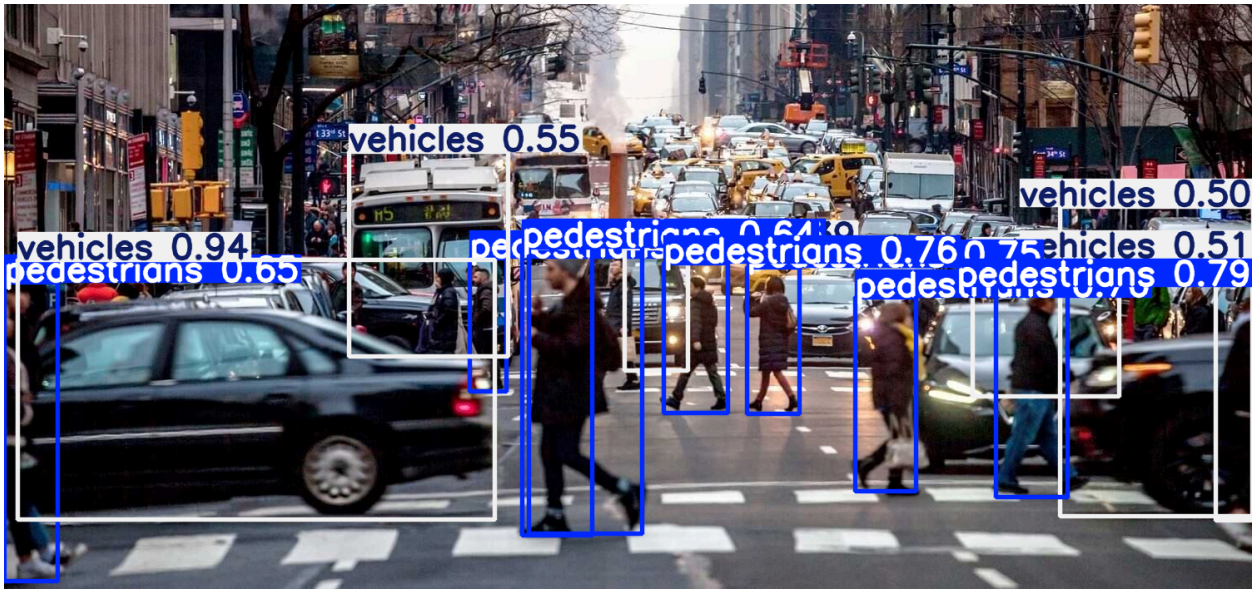
Observations:

1. Vehicles have the highest accuracy with the best mAP scores.
2. Signposts have the lowest performance, likely due to smaller object sizes or fewer instances in the dataset.
3. Overall mAP50 (0.732) and mAP50-95 (0.501) indicate decent model performance, but there's room for improvement in detecting smaller or less-represented objects.

Inferences:

Using the **"3. inference_notebook.ipynb"** notebook, I took inferences from the finetuned ***yolov8n*** model on some random images form the internet like shown below.


East 42nd St
vehicles 0.53
vehicles 0.33
pedestrians 0.71
pedestrians 0.70
pedestrians 0.85
vehicles 0.84
pedestrians 0.75


vehicles 0.55
vehicles 0.50
vehicles 0.94
pedestrians 0.65
pedestrians 0.76
vehicles 0.51
pedestrians 0.79