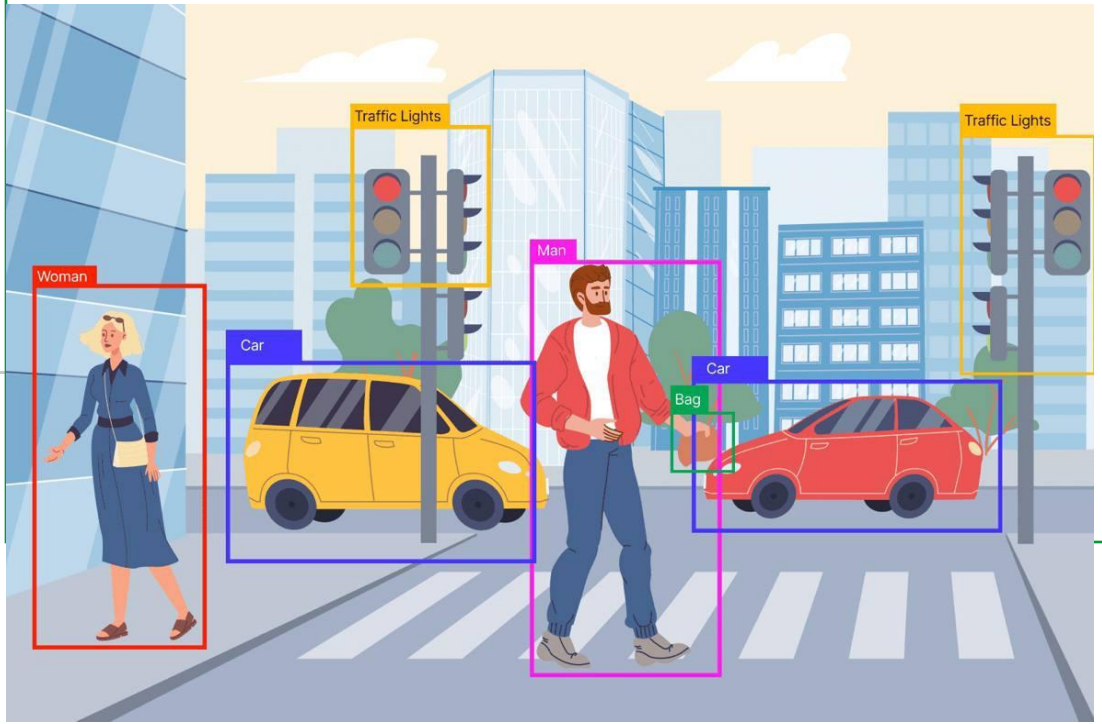
YOLO Algorithm

For Object Detection



GitHub Link

https://github.com/arkarapetyan/nuImages-yolo

Slide Contents

1. The Object Detection Problem
2. nulmages Dataset
3. Training Procedure
4. Evaluation
5. Conclusion

1. The Object Detection Problem

Contents

Object Detection

It is a phenomenon in computer vision that involves the detection of various objects (eg. people, cars, chairs, buildings, and animals) in digital images or videos.

This phenomenon answers two basic questions:

- What is the object?

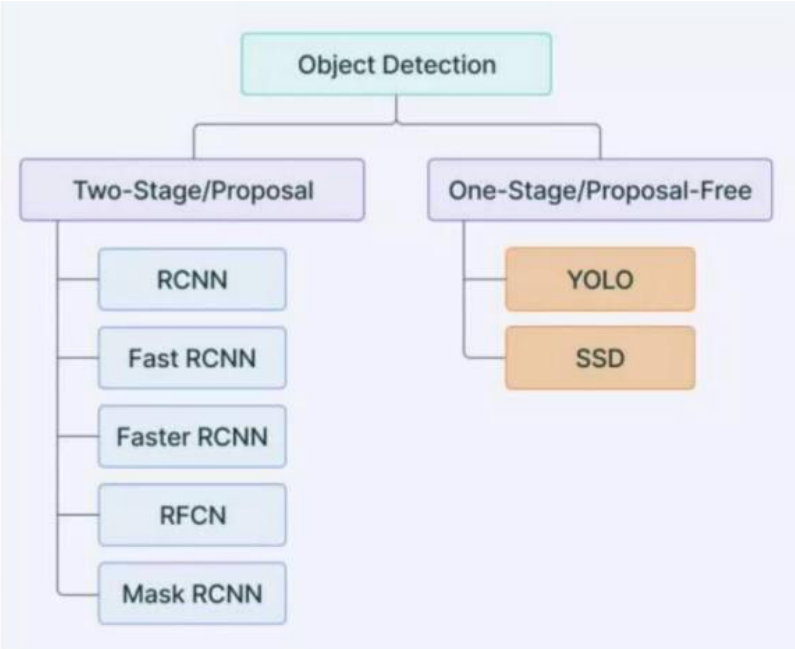
  This question seeks to identify the object in a specific image.

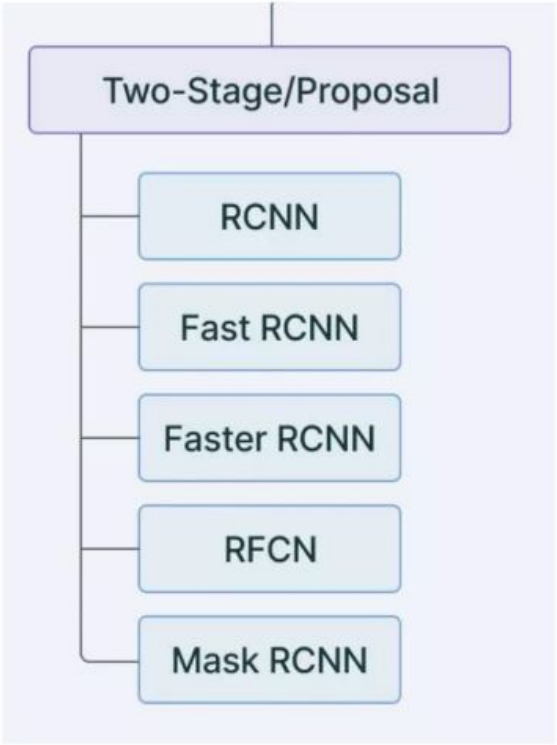
- Where is it?

  This question seeks to establish the exact location of the object whitin the image.

Two-stage object detection

- Two-stage object detection refers to the use of algorithms that break down the object detection problem statement into the following two-stages:
- Detecting possible object regions.
- Classifying the image in those regions into object classes.





Popular two-step algorithms like Fast-RCNN and Faster-RCNN typically use a Region Proposal Network that proposes regions of interest that might contain objects.

The output from the RPN is then fed to a classifier that classifies the regions into classes.

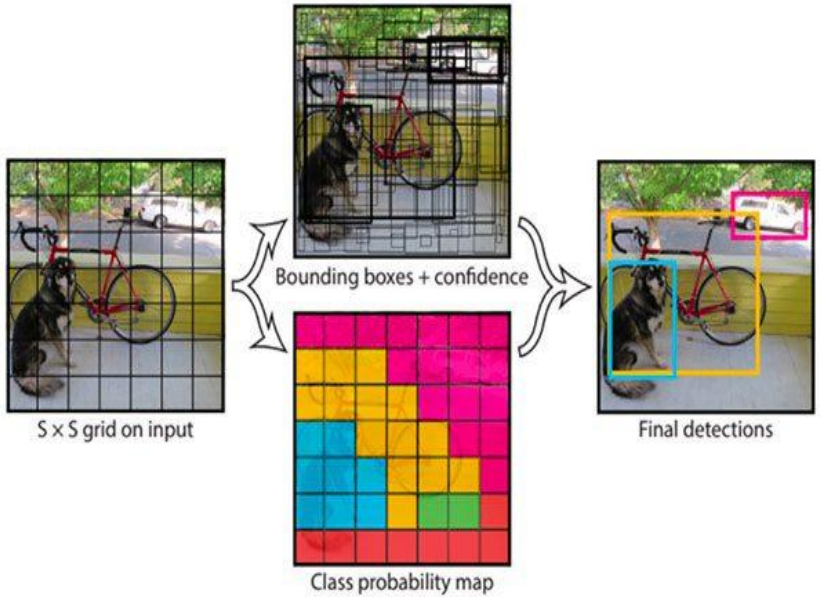
While this gives accurate results in object detection with a high mean Average Precision (mAP), it results in multiple iterations taking place in the same image, thus slowing down the detection speed of the algorithm and preventing real-time detection.

What is YOLO? *You Only Look Once*

- YOLO is an algorithm proposed by by Redmond et. al. in a research article published at the TEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) as a conference paper, winning OpenCV People's Choice Award
- In Comparison with other object detection algorithms, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once.

How does YOLO work?

- The YOLO algorithm works by dividing the image into N grids, each having an equal dimensional region of SxS.
- Each of these N grids is responsible for the detection and localization of the object it contains.
- Correspondingly, these grids predict B bounding box coordinates relative to their cell coordinates, along with the object label and probability of the object being present in the cell.

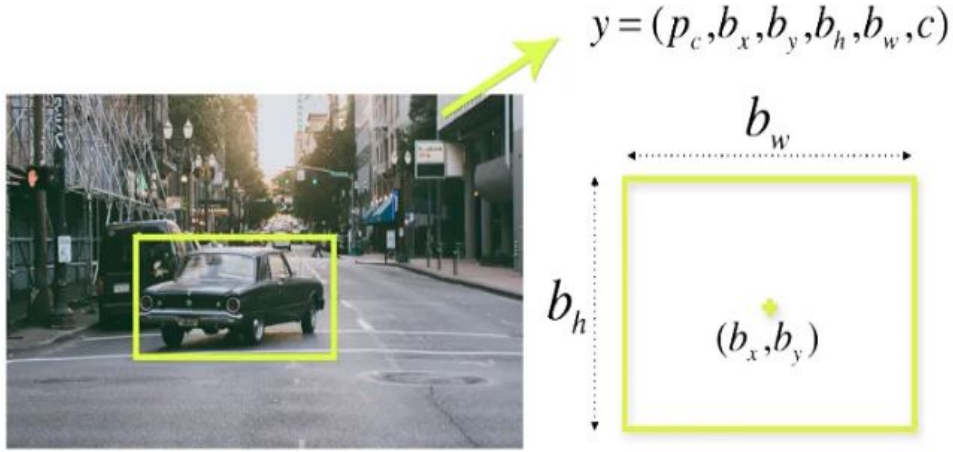


Bounding box regression

A bounding box is an outline that highlights an object in an image.

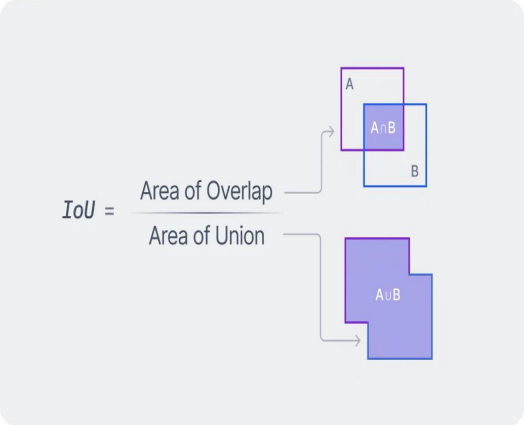
Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.) - This is represented by the letter C.
- Bounding box center (bx,by)



Intersection over Union

- Intersection over Union is a popular metric to measure localization accuracy and calculate localization errors in object detection models.
- To calculate the IoU with the predictions and the ground truth, we first take the intersecting area between the bounding boxes for a particular prediction and the ground truth bounding boxes of the same area. Following this, we calculate the total area covered by the two bounding boxes—also known as the Union.
- The intersection divided by the Union, gives us the ratio of the overlap to the total area, providing a good estimate of how close the bounding box is to the original prediction.



Non Maximum Suppression

Other methods brings forth a lot of duplicate predictions due to multiple cells predicting the same object with different bounding box predictions.

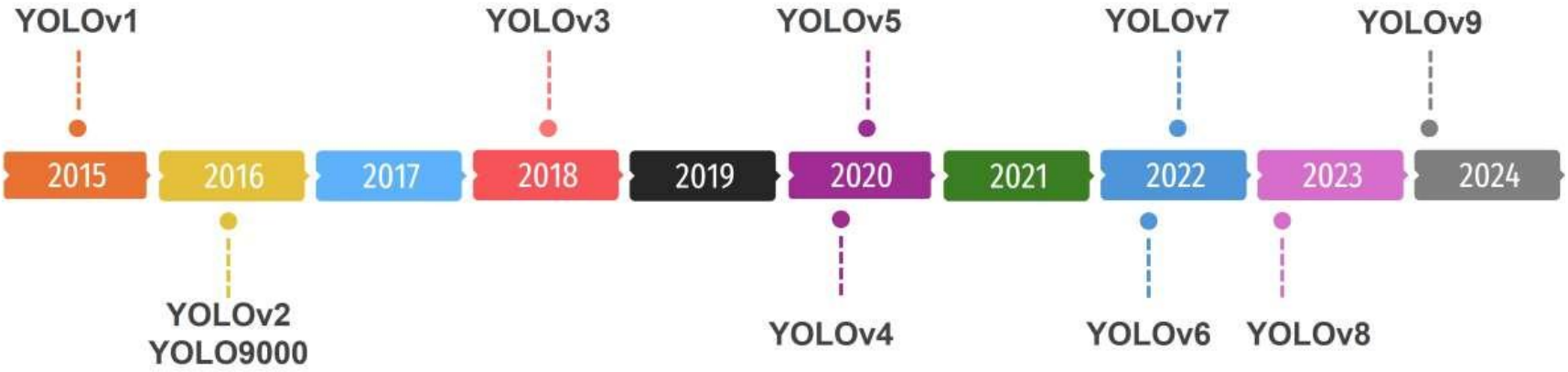
YOLO makes use of Non Maximum Suppression to deal with this issue.

- In Non Maximal Suppression, YOLO suppresses all bounding boxes that have lower probability scores.
- YOLO achieves this by first looking at the probability scores associated with each decision and taking the largest one. Following this, it suppresses the bounding boxes having the largest IoU with the current high probability bounding box.
- This step is repeated till final bounding boxes are obtained.

Average Precision (AP)

- Average Precision is calculated as the area under a precision vs recall curve for a set of predictions.
- Recall is calculated as the ratio of the total predictions made by the model under a class with a total of existing labels for the class.
- On the other hand, Precision refers to the ratio of true positives with respect to the total predictions made by the model.
- The area under the precision vs recall curve gives us the Average Precision per class for the model. The average of this value, taken over all classes, is termed as mean Average Precision (mAP).

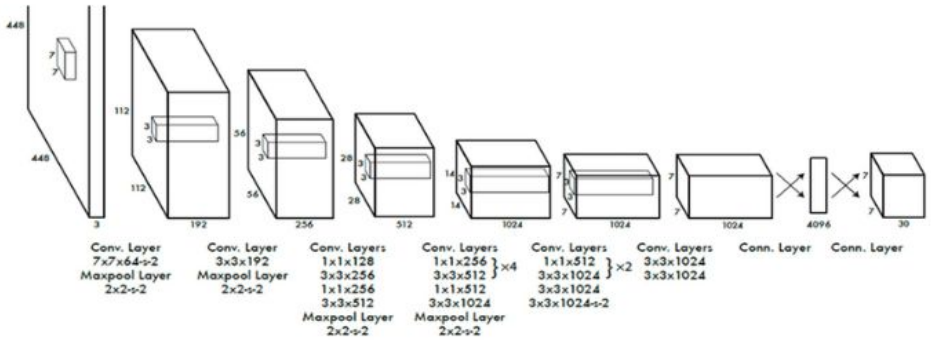
YOLO Timeline



https://www.ikomia.ai/blog/what-is-yolo-introduction-object-detection-computer-vision

YOLO Architecture

Inspired by the GoogleNet architecture, The first YOLO architecture has a total of 24 convolutional layers with 2 fully connected layers at the end.



YOLOv8 Architecture

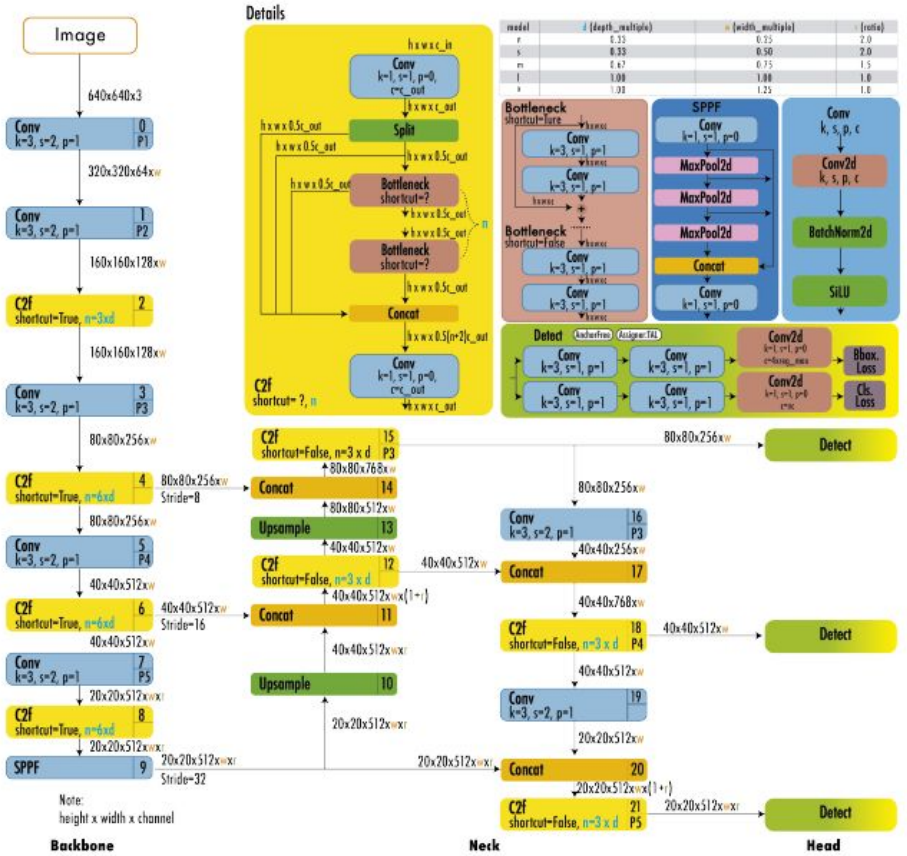


Figure 17: YOLOv8 Architecture. The architecture uses a modified CSPDarknet53 backbone. The C2f module replaces the CSPLayer used in YOLOv5. A spatial pyramid pooling fast (SPPF) layer accelerates computation by pooling features into a fixed-size map. Each convolution has batch normalization and SiLU activation. The head is decoupled to process objectness, classification, and regression tasks independently. Diagram based in [115].

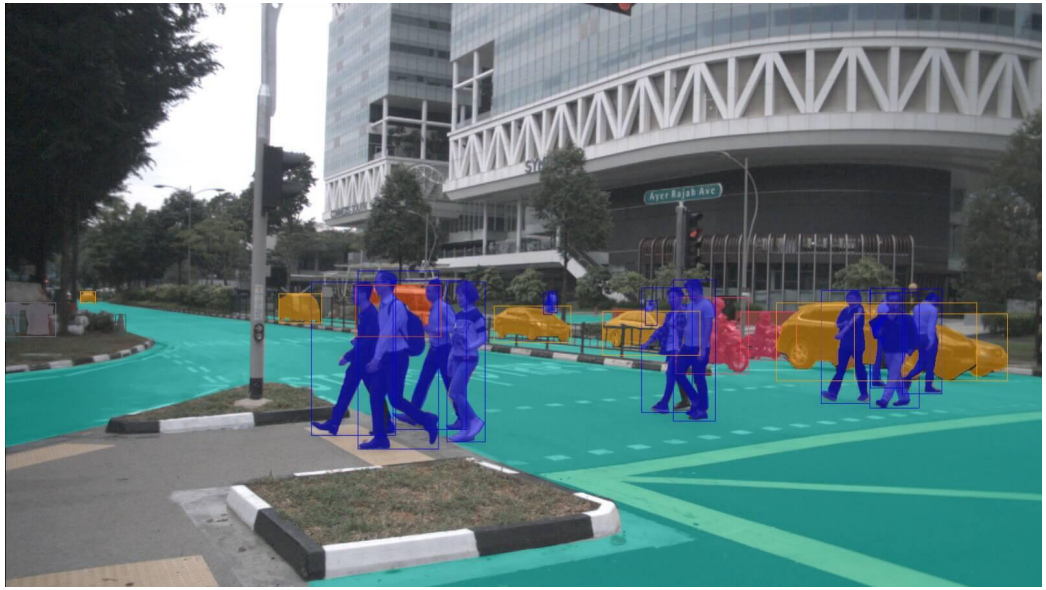
https://arxiv.org/pdf/2304.00501v6

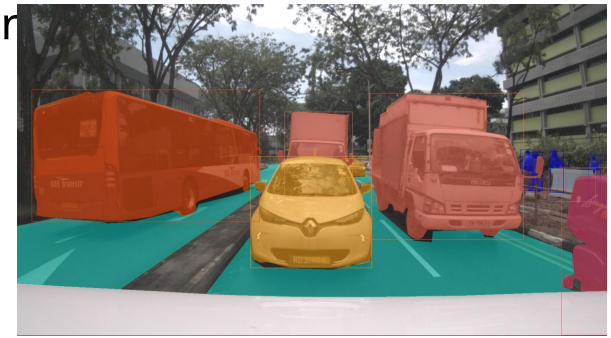
Importance of YOLO

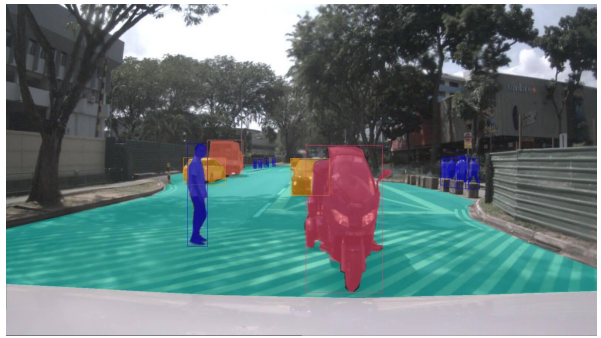
- YOLO algorithm is important because of the following reasons:
- Speed: This algorithm improves the speed of detection because it can predict objects in real-time.
- High accuracy: YOLO is a predictive technique that provides accurate results with minimal background errors.
- Learning capabilities: The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

2. nulmages Dataset

The NuImages dataset is a comprehensive and large-scale dataset designed for autonomous driving and computer vision research. It is part of the larger nuScenes dataset suite and provides annotated images captured from a variety of camera sensors mounted on autonomous vehicles.







Key features of the nuImages dataset include:

≈67k train samples, ≈16k val. samples and ≈9k test samples of size 1600x900.

1. Diverse Annotations: Contains annotations for multiple object categories including vehicles, pedestrians, traffic signs, and other road users, aiding in robust object detection tasks.

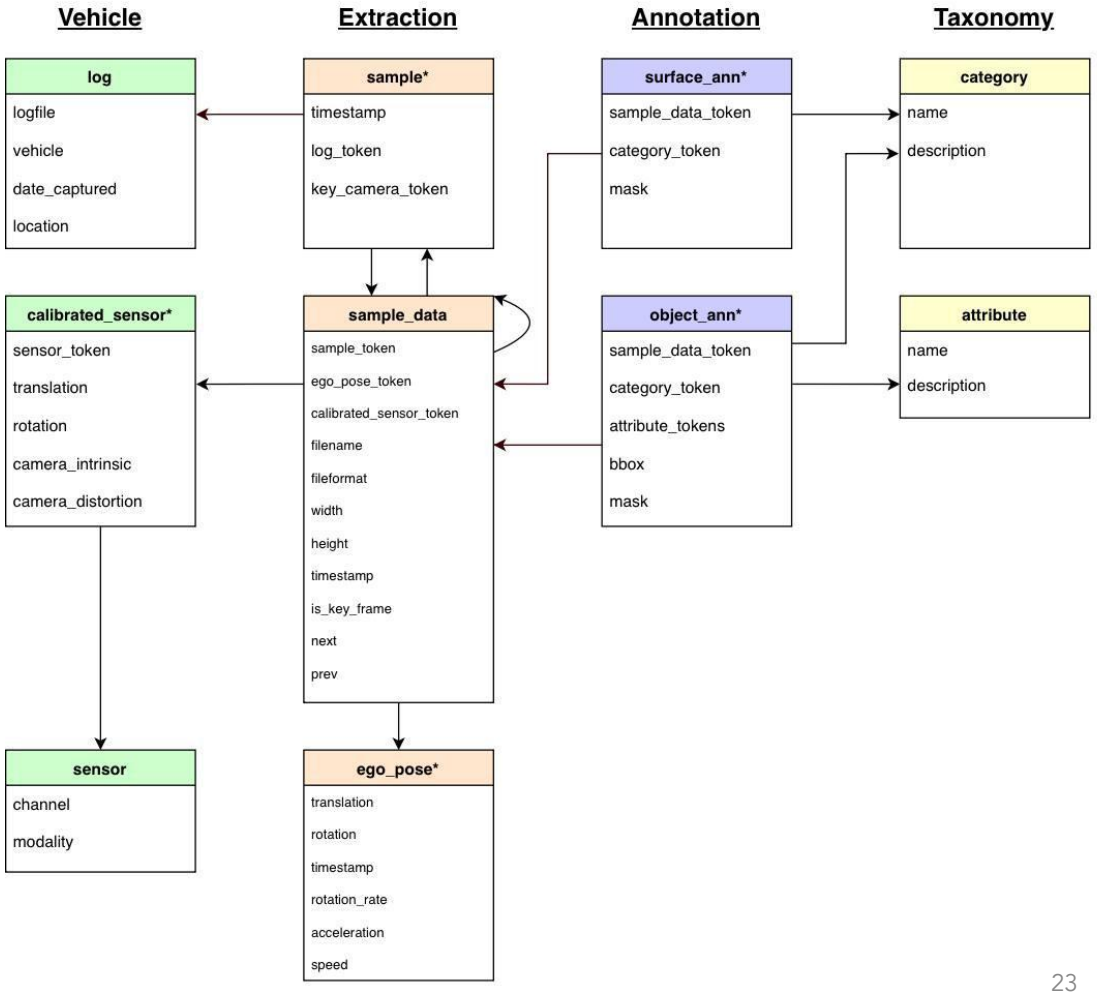
2. Variety of Scenes: Offers a wide range of urban, suburban, and highway scenes captured under different weather conditions and times of day, providing a rich and diverse set of images.

3. High-Quality Data: Includes high-resolution images with precise annotations for object detection, segmentation, and other vision tasks.

4. Extensive Metadata: Accompanied by detailed metadata such as timestamp, sensor information, and calibration data, which are crucial for aligning images with other sensor data in multi-modal research.

5. Open Access: Freely available for academic and commercial research, encouraging advancements in autonomous driving technologies and computer vision applications.

nulmages schema



object classes

```yaml
names:
  0: animal
  1: flat.driveable_surface
  2: human.pedestrian.adult
  3: human.pedestrian.child
  4: human.pedestrian.construction_worker
  5: human.pedestrian.personal_mobility
  6: human.pedestrian.police_officer
  7: human.pedestrian.stroller
  8: human.pedestrian.wheelchair
  9: movable_object.barrier
  10: movable_object.debris
  11: movable_object.pushable_pullable
  12: movable_object.trafficcone
  13: static_object.bicycle_rack
  14: vehicle.bicycle
  15: vehicle.bus.bendy
  16: vehicle.bus.rigid
  17: vehicle.car
  18: vehicle.construction
  19: vehicle.ego
  20: vehicle.emergency.ambulance
  21: vehicle.emergency.police
  22: vehicle.motorcycle
  23: vehicle.trailer
  24: vehicle.truck
```

3. Training Procedure

Training Description

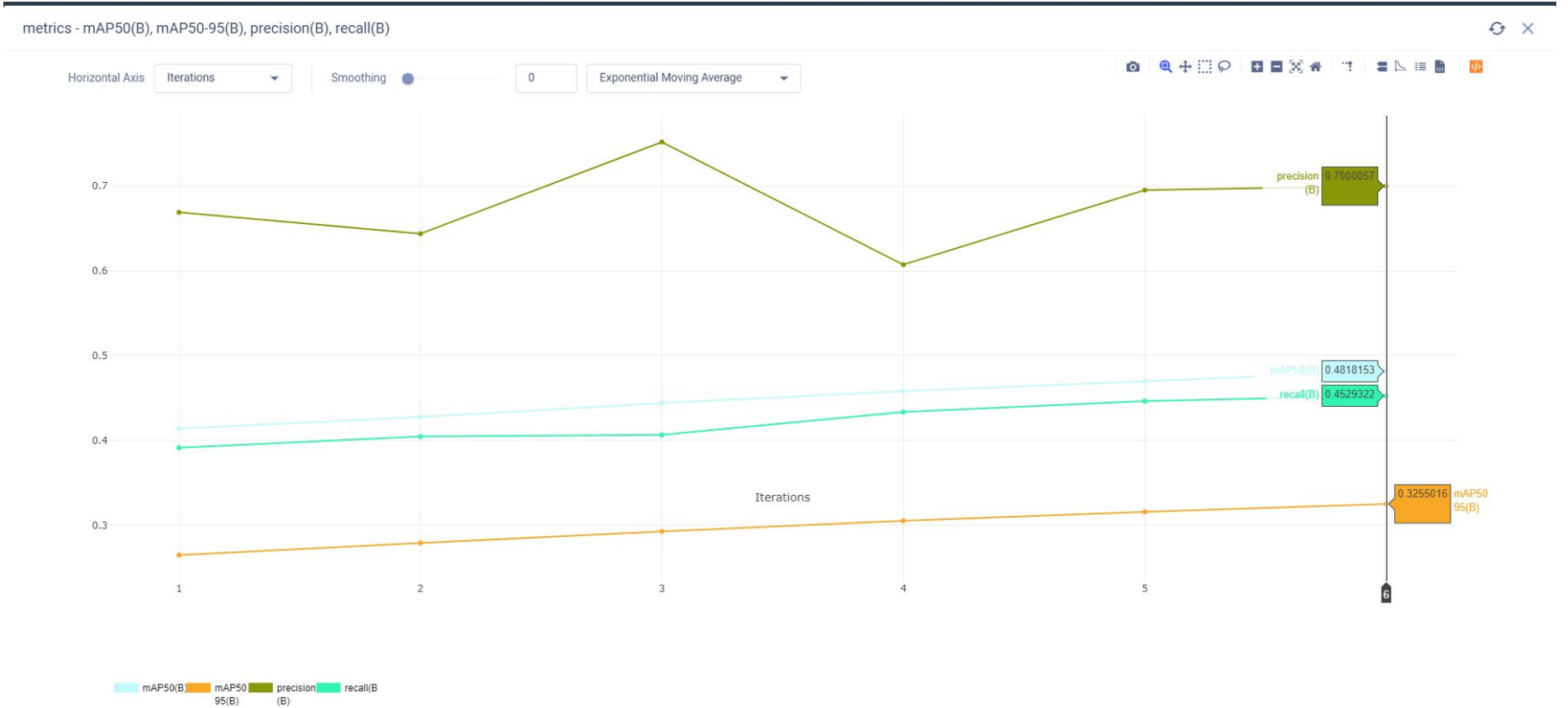
1. Using Ultralytics Yolov8 model.
2. Need to save the dataset in COCO format.
3. 16 epochs on 2x Nvidia T4 GPUs (Kaggle Notebook)
4. Mostly with cosine learning rate, then gradually decreasing.
5. Image size = 960 (original was 1600x900)
6. Played with different loss weights.

Training Description

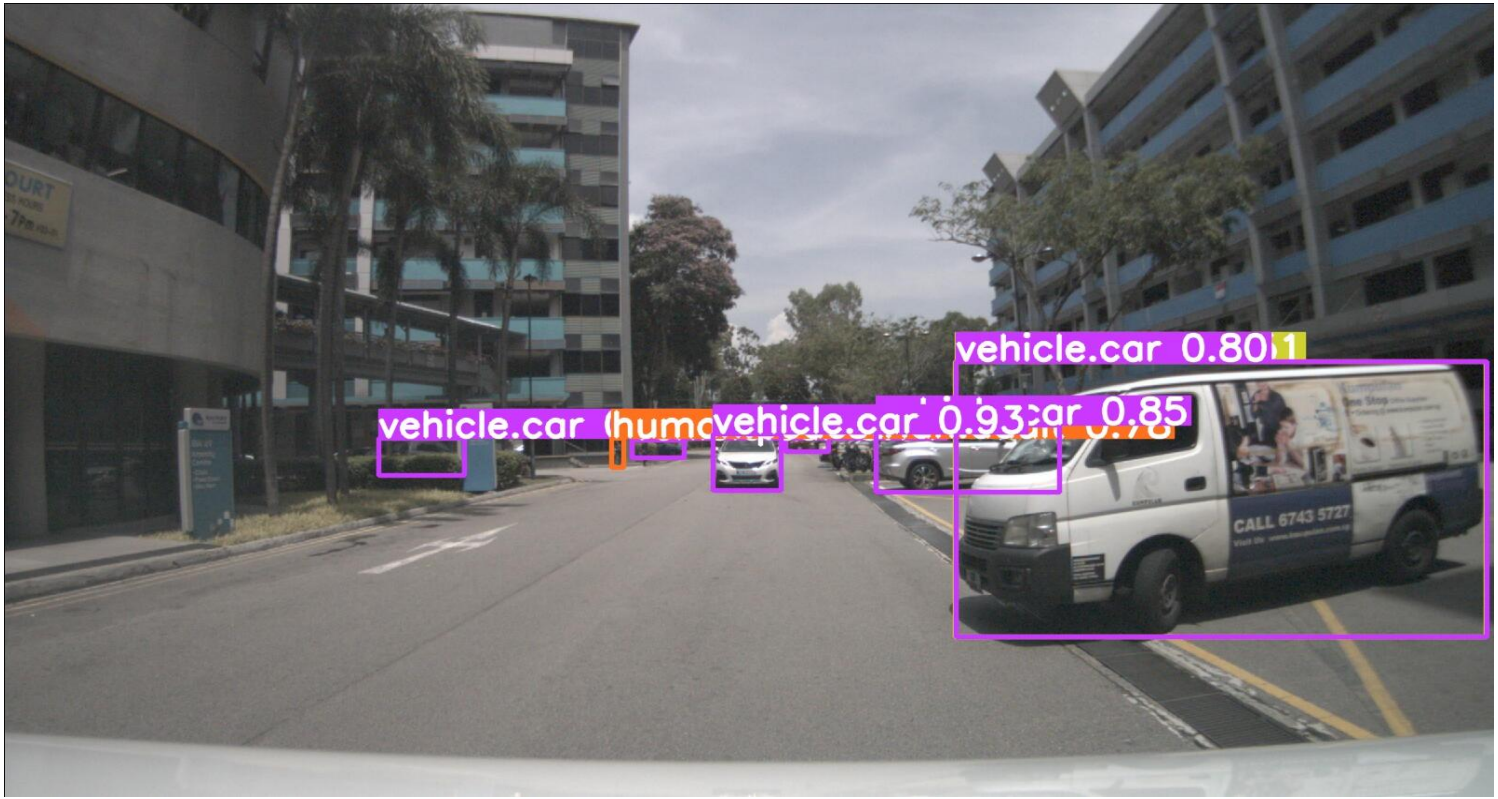
https://app.clear.ml/projects/ab8bcec9e518452fb2448a5f595a8f6d/

4. Evaluation

Metrics


metrics - mAP50(B), mAP50-95(B), precision(B), recall(B)
Horizontal Axis
Iterations
Smoothing
0
Exponential Moving Average
0.7
0.6
0.5
0.4
0.3
1
2
3
4
5
6
Iterations
precision (B) 0.7000057
mAP50(B) 0.4818153
recall(B) 0.4529322
0.3255016 mAP50 95(B)
mAP50(B)
mAP50 95(B)
precision (B)
recall(B

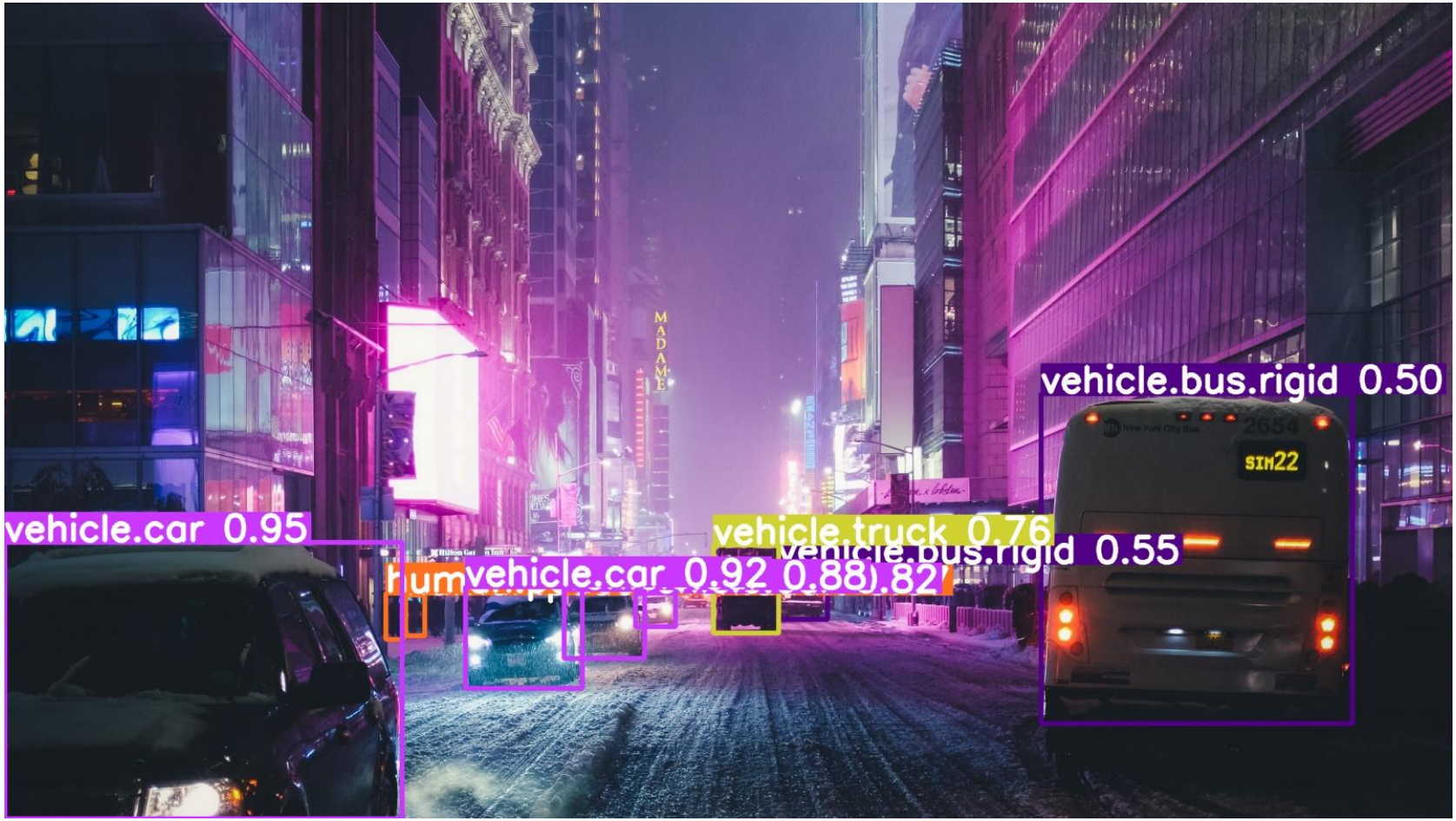
Predictions



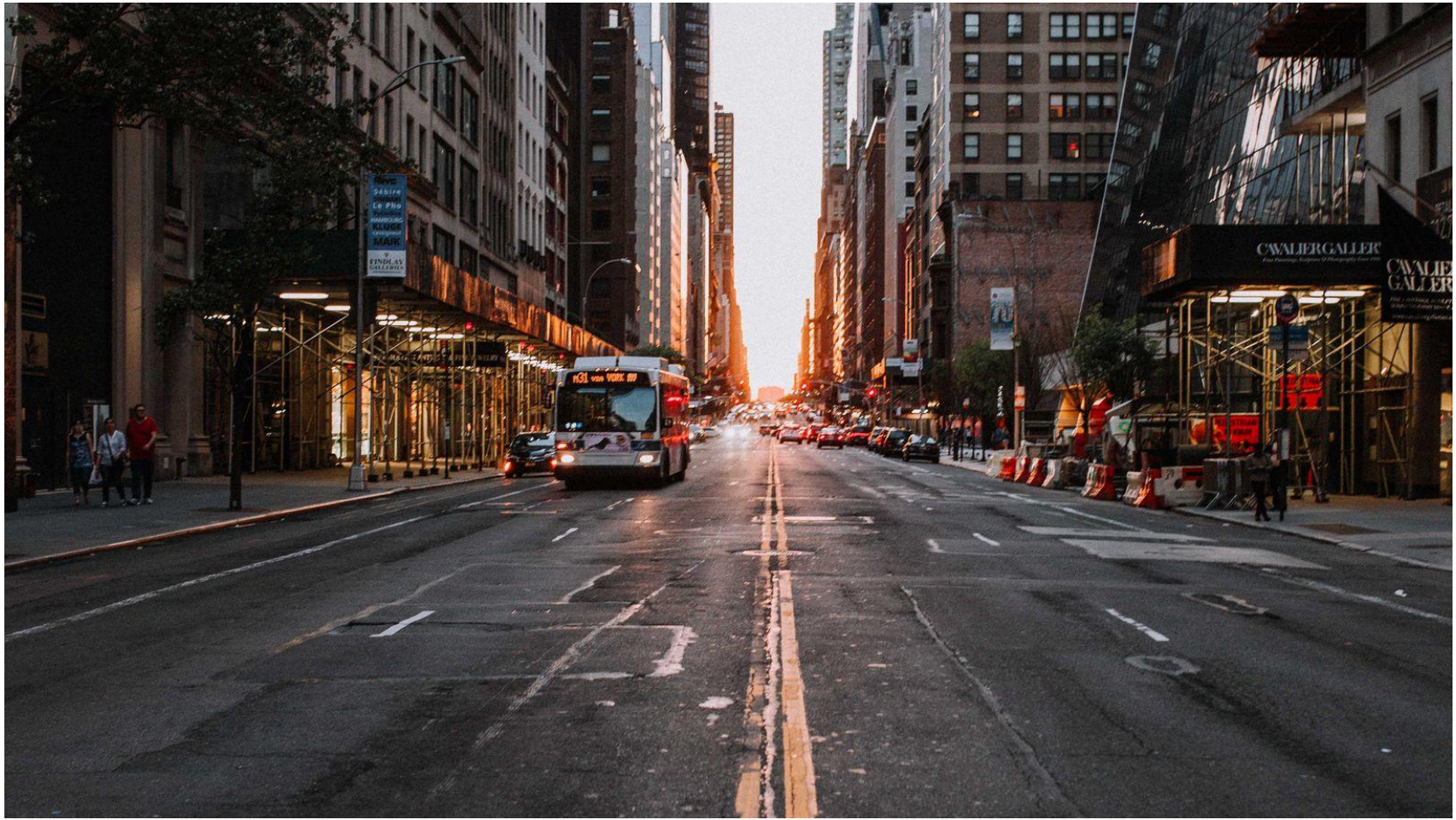

vehicle.car 0.92
ac.com


vehicle.car 0.57
human.ped

Out of distribution samples


vehicle.bus.rigid 0.50
vehicle.car 0.95
vehicle.truck 0.76
vehicle.bus.rigid 0.55
vehicle.car 0.92
0.88
0.82
SIM22
MADAME


vehicle.bus.rigid 0.50
vehicle.car 0.95
vehicle.truck 0.76
vehicle.bus.rigid 0.55
vehicle.car 0.92
0.88
0.82
SIM22


CWALIER GALLER
CWALIE
GALLER


vehicle.bus.rigid 0.87
human.pedestrian.adult 0.86
vehicle.car 0.87
vehicle.car 0.87

5. Conclusion

What Have We Learned?

1. Using cloud technologies for saving big datasets.
2. Frameworks for object-detection and DL model training.
3. Fine Tuning is very sensitive to learning rates.
4. Cosine l.r. is useful with many epochs.

Could we do better?

Yes, here's how:

1. Better learning rate scheduling.
2. Playing with hyperparameters.
3. Data Augmentation.

Thank You.