From Zero to YOLO: Car Detection Without Pre-Trained Models

SAPIENZA
UNIVERSITÀ DI ROMA

Master's Degree in Artificial Intelligence and Robotics

Palumbo Simone 1938214
Spena Francesco 1911090

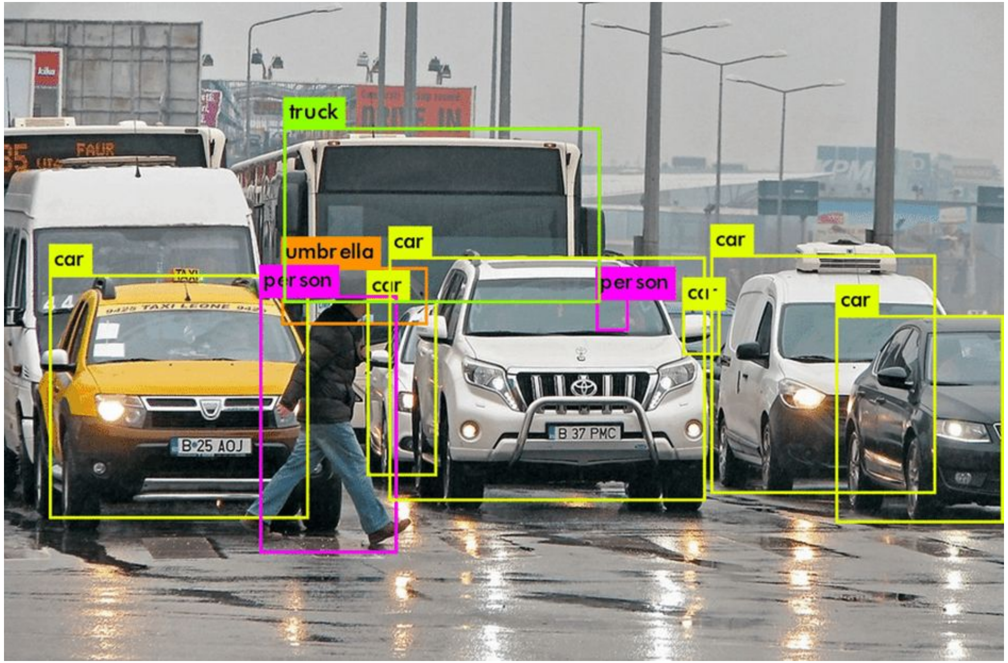
https://github.com/CogSP/Yolov8-Car-Detection

Outline

- Introduction
- Proposed Method
- Dataset and Metrics
- Experimental Results
- Conclusions and References

Object Detection: Localize and Classify Objects in Images

- Combines **object classification** and **localization**
- Outputs: **bounding boxes**, **class labels**, and **confidence scores**
- Critical for applications like **autonomous vehicles**, **surveillance**, and **medical imaging**

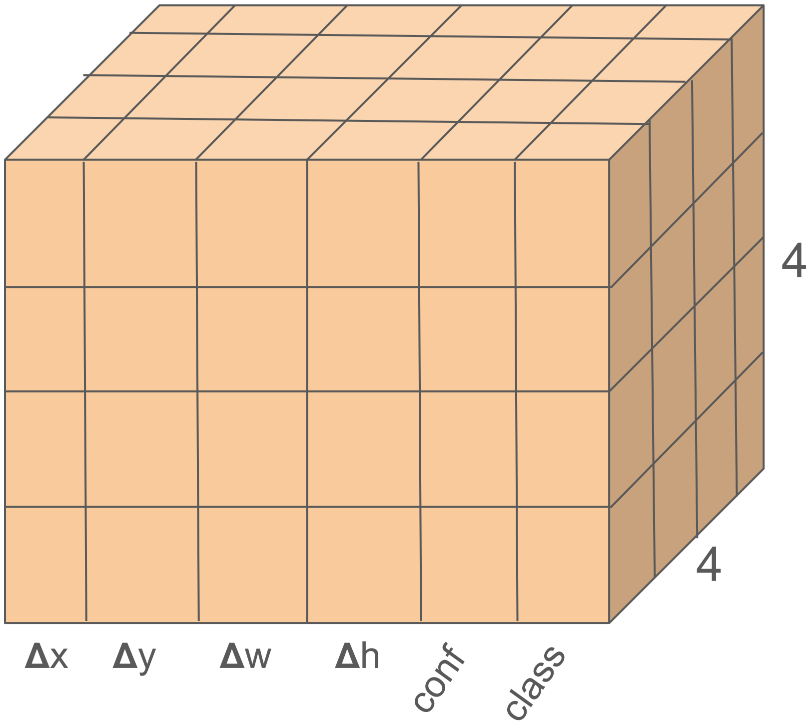


YOLO (You Only Look Once): Real-Time Object Detection

- **Single-pass** neural network for object detection
- Processes images in **real-time** with high accuracy
- Efficiently balances **speed** and **accuracy**

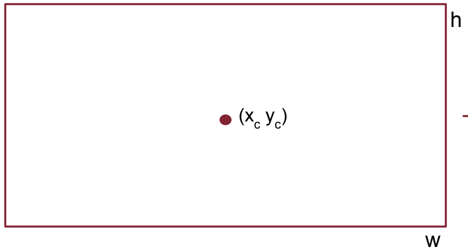
Input encoding

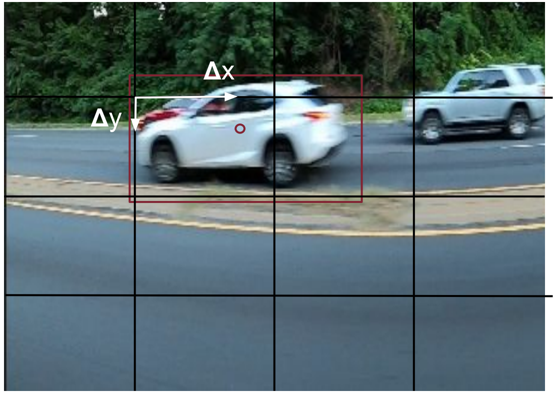
1. After **resizing** the image, divide into **grid cells**
2. Given the center of the object, calculate the **cell** in which the **center** lies. This cell is the one responsible for calculating the bbox.
3. Transform the (x_c, y_c, w, h) in $(\Delta_x, \Delta_y, \Delta_w, \Delta_h)$
4. Add confidence and car class probabilities of 1
5. Put all **zeros** in all the other cells, in which we do not have objects



Coordinate Transformations

(x_a, y_a): the coordinate of left-top point





$$x_c = \frac{x_{min}+x_{max}}{2}$$
$$y_c = \frac{y_{min}+y_{max}}{2}$$
$$w = x_{max} - x_{min}$$
$$h = y_{max} - y_{min}$$

$$\Delta x = (x - x_a)/64 \qquad \Delta w = w/448$$
$$\Delta y = (y - y_a)/64 \qquad \Delta h = h/448$$

YOLOv8 Architecture

YOLOv8 Variants

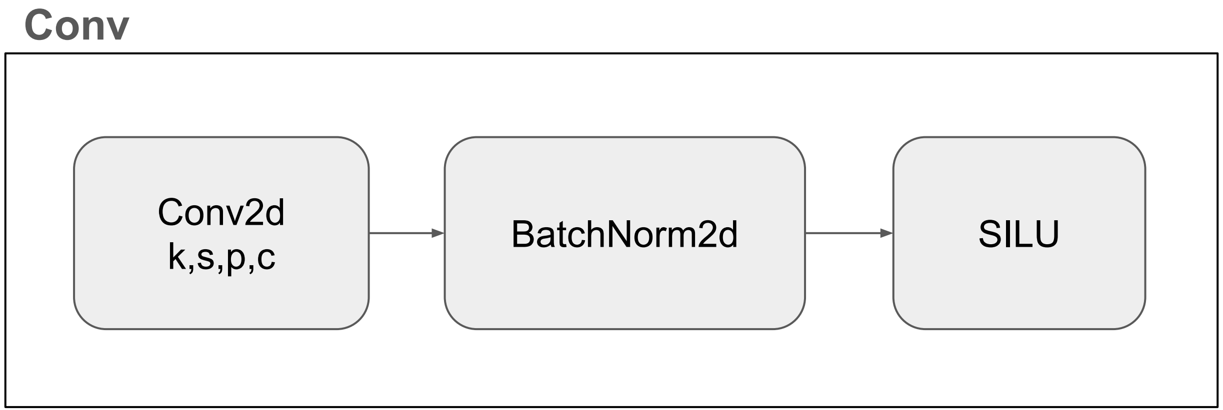
Model Variant	depth_multiple	width_multiple	max_channels
n	0.33	0.25	1024
s	0.33	0.50	1024
m	0.67	0.75	768
l	1	1	512
xl	1	1.25	512

- YOLOv8 **family** includes variants that **balance** accuracy, speed and size
  - **depth_multiple**: scales the number of bottleneck in c2f
  - **width_multiple**: scales the number of channels in the convolutional layers
  - **max_channels**: upper limit of allowed channel to prevent the model from becoming too wide. Can also prevent overfitting.
- We chose the **large** (l) model: high accuracy but slow inference

YOLOv8 Building Blocks

- **Convolutional** Block
- **Bottleneck**
- **C2f** Block
- **SPPF** Block
- **Detect** Block

Convolutional Block

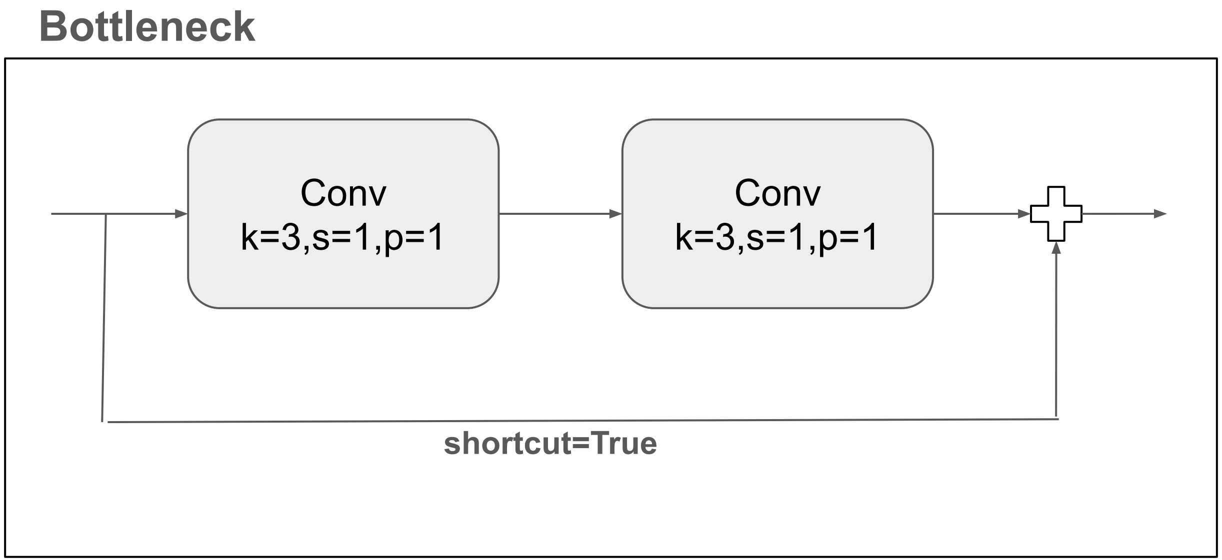


Features extraction

Normalized data (Local efficiency)

Smooth gradients (No vanish)

Bottleneck

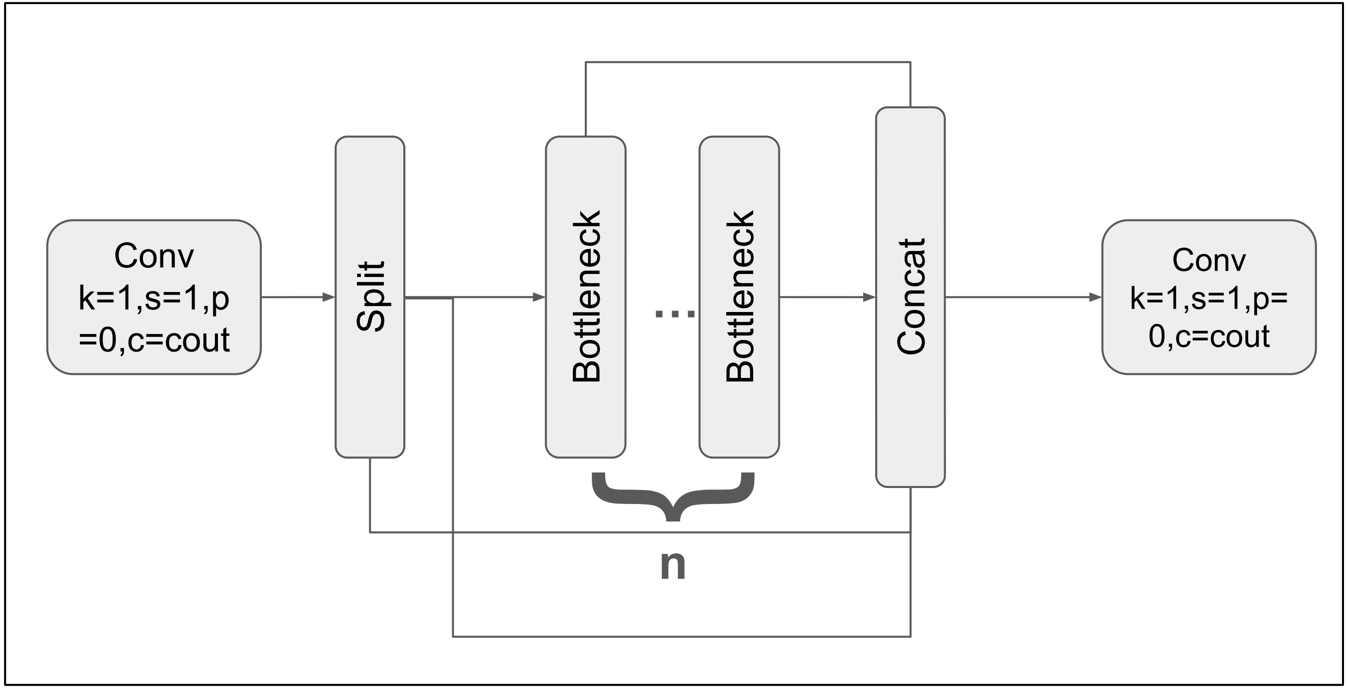


Residual block for gradient efficiency

$$r(x) = f(x) + x \implies \nabla r(x) = \nabla f(x) + I$$

C2f Block

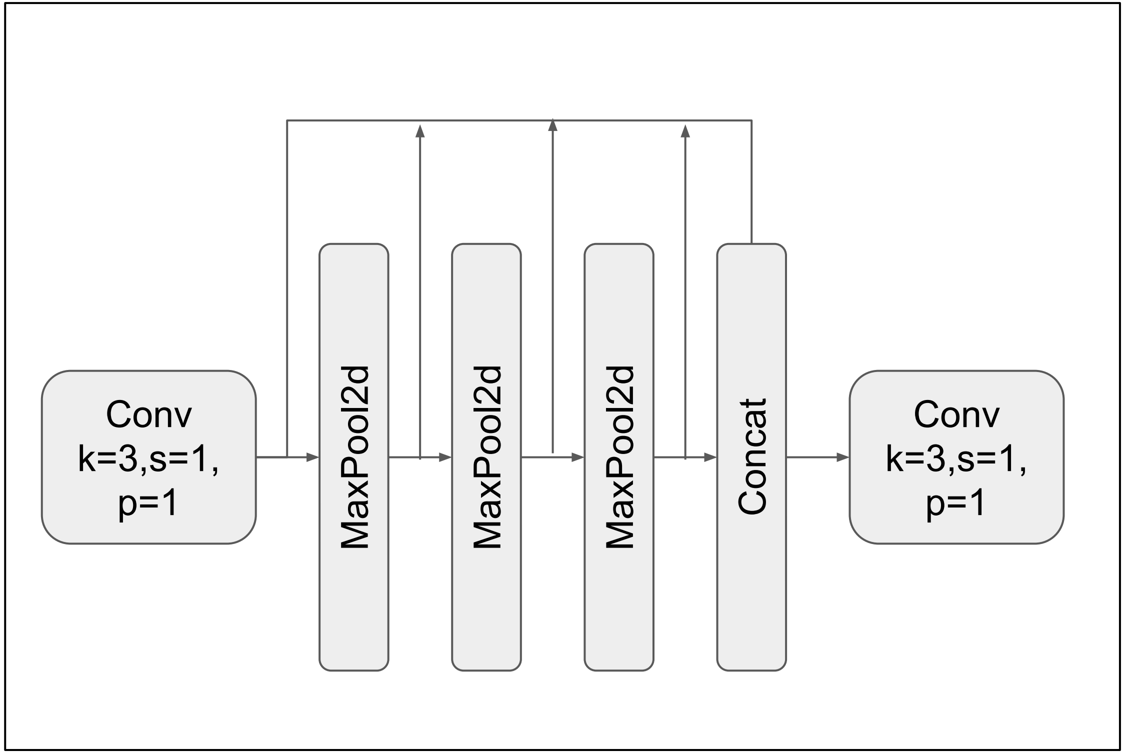
C2f Block



Enhance feature extraction

Half of the input goes through the n (**depth_multiple**) Bottleneck, while the other half is passed to the **Concat**

Spatial Pyramid Pooling Fast (SPPF) Block



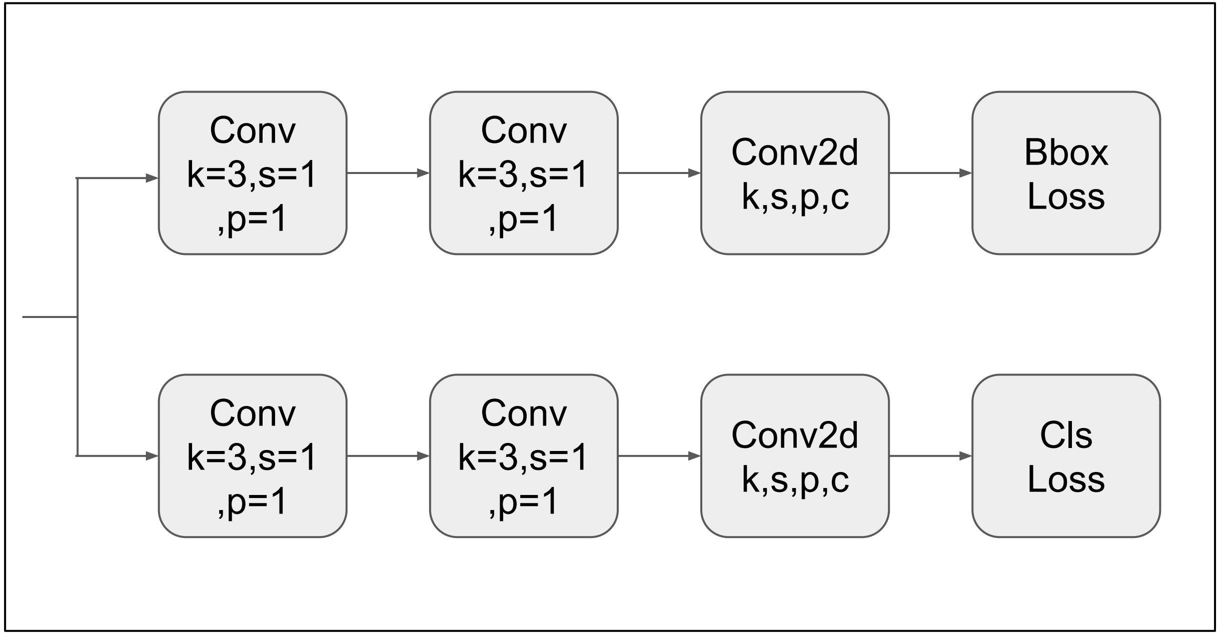
SPPF

Divides images in **grid** and **pools** features from each cell, handling object of different sizes and capturing **multi-scale** information

SPP-**Fast** uses a single **fixed-size** kernel instead of multiple levels (SPP): **tradeoff** between accuracy and efficiency.

Detect Block

Detect

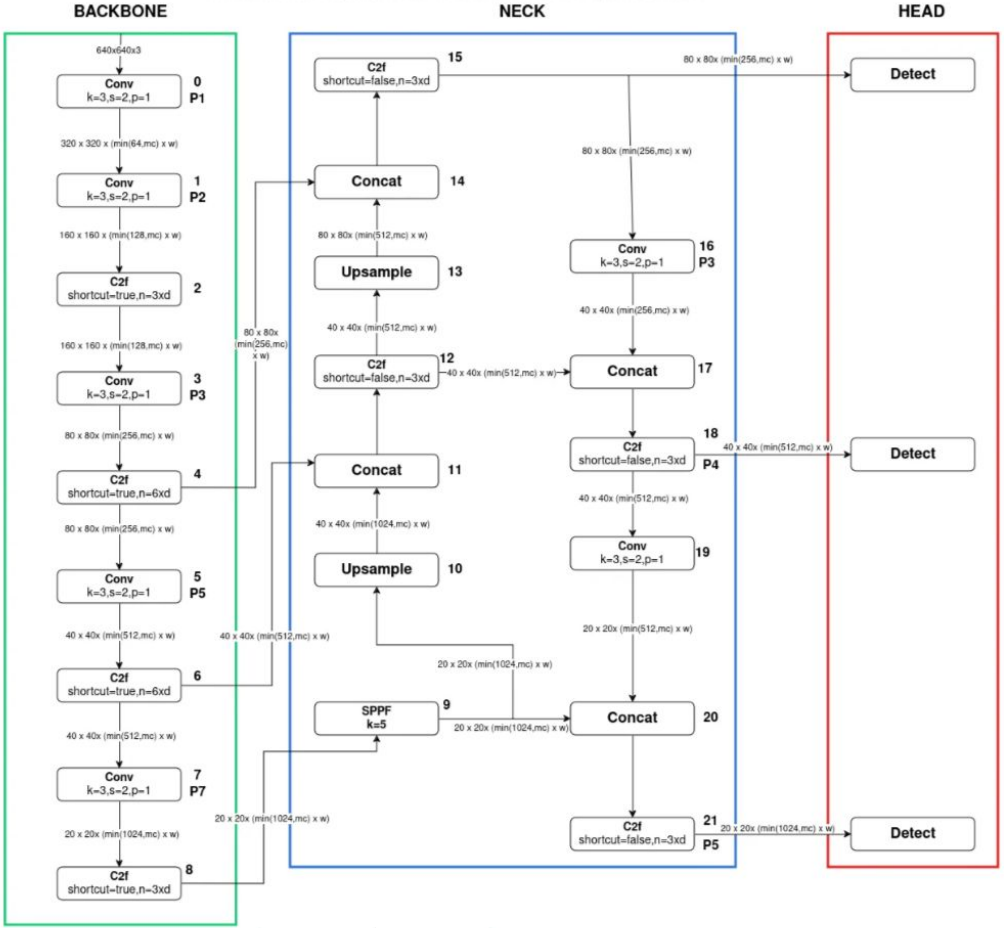


Responsible for the object **detection**

Two tracks for bbox and class prediction ➠ We merged them in a **single branch**

YOLOv8 is **anchor-free**: faster post-processing after inference

The Overall Architecture



Backbone

- **Deep** part of the model
- **Different** levels of features extraction

Neck

- **Upsampling** process
- **Merge** features from different layers
- **No** resolution changing

Head

- **Final** output:
  - **class** label
  - **bounding boxes**
  - **confidence**
- **Multiple** Detect block to improve multi-scale detection

Loss Function

YOLOv1 Loss with a small change: we added a **cls** square differences for **no-object** cell

< 1 so **more** importance on grid cells that contain object than cells which don't

Total Loss = $\sum_{x,y} (L_{obj} + \boxed{\lambda_{no-obj}} L_{no-obj})$

$$L_{obj} = \boxed{\lambda_{coord}} L_{obj-box} + L_{obj-conf} + L_{obj-class}$$

> 1 so **more** importance on box parameters than **cls** and **confidence**

Loss Function (2)

With:

$$L_{obj-box} = (\hat{\Delta x} - \Delta x)^2 + (\hat{\Delta y} - \Delta y)^2 + (\sqrt{\hat{\Delta w}} - \sqrt{\Delta w})^2 + (\sqrt{\hat{\Delta h}} - \sqrt{\Delta h})^2$$

$$L_{obj-conf} = (\hat{C} - C)^2$$

$$L_{obj-cls} = (\hat{p} - p)^2$$

Having **C = 1** and **p = 1**. Same holds for **no-obj** loss, but using **C = 0** and **p = 0**

Dataset and Metrics

Dataset

We used the Kaggle Car Object detection dataset, which includes

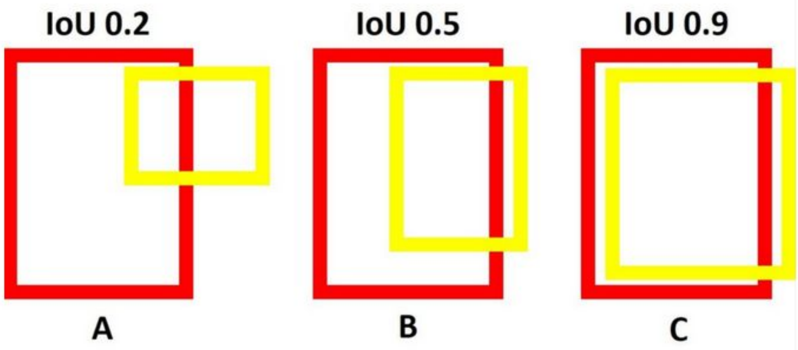
- images of cars in all views
- CSV files containing the ground truth label in the format: $(id, x_{max}, y_{max}, x_{min}, y_{min})$

Evaluation Metrics

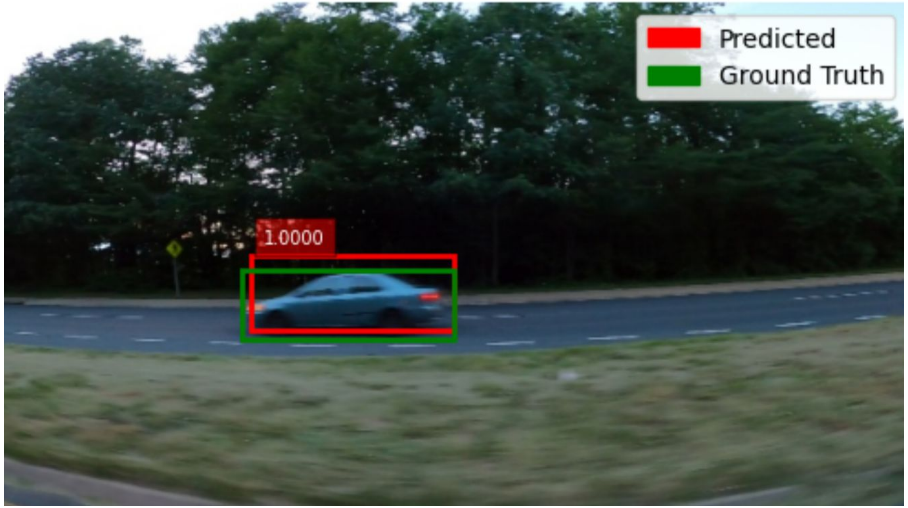
Intersection over Union (IoU):

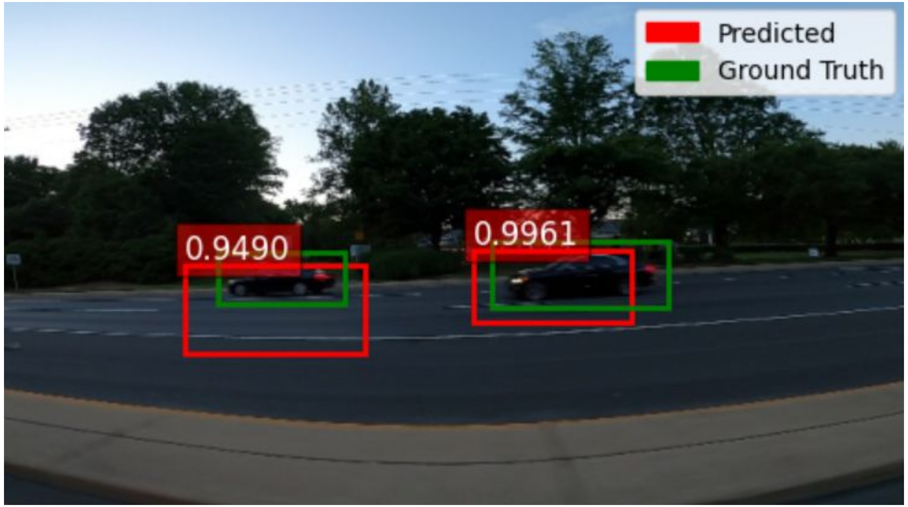
- used to evaluate the alignment between ground-truth and predicted boxes

$$IoU = \frac{IntersectionArea}{UnionArea}$$



Results





The generated images are provided along with their corresponding **IoU values**.

References

- R. Varghese, M. Sambath. *YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness,* 2024, International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)
- J. Redmon, S. Divvala, R. Girshick, A. Farhadi. *You Only Look Once: Unified, Real-Time Object Detection, 2016*
- J. Terven, D. Esparza, J. Gonzales. *A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS, 2024*
- D. Reis, J. Hong, J. Kupec, A. Daoudi. *Real-Time Flying Object Detection with YOLOv8*
- *Kaggle Car Object Detection Dataset*