YOLO

"You Only Look Once" - Object Detection

Group 6: Joaquin Arias, Era Gërbeshi, Mohamed Mourad Ouazghire, Naomi Mukuhi

What is our project?

Real-Time Object Detection

Locating and identifying:
- Duckiebots
- Ducks
- Stop signs
- Road signs

Real-world application:
- Autonomous driving

DUCKIETOWN

X

YOLO

Introduction

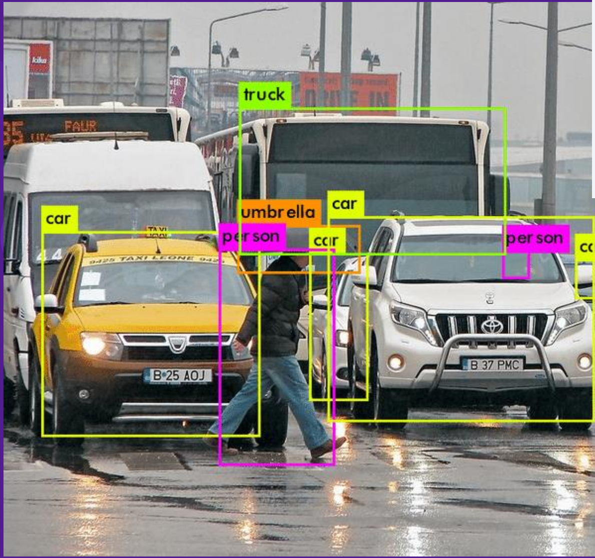
Object Detection

What is it?

Technique to identify and locate objects in an image or video.

Applications

- Surveillance
- Self-driving cars
- Robotics



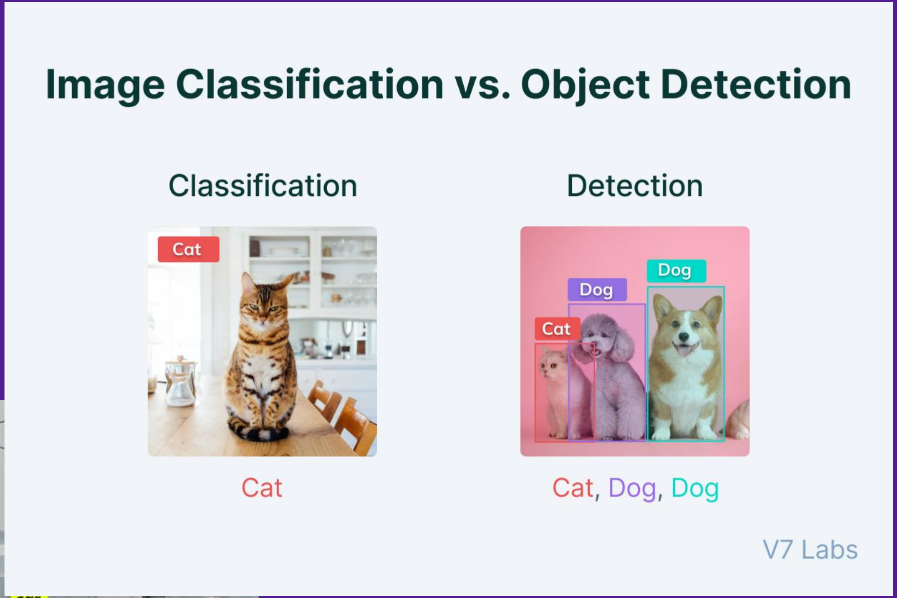


Image Classification vs. Object Detection

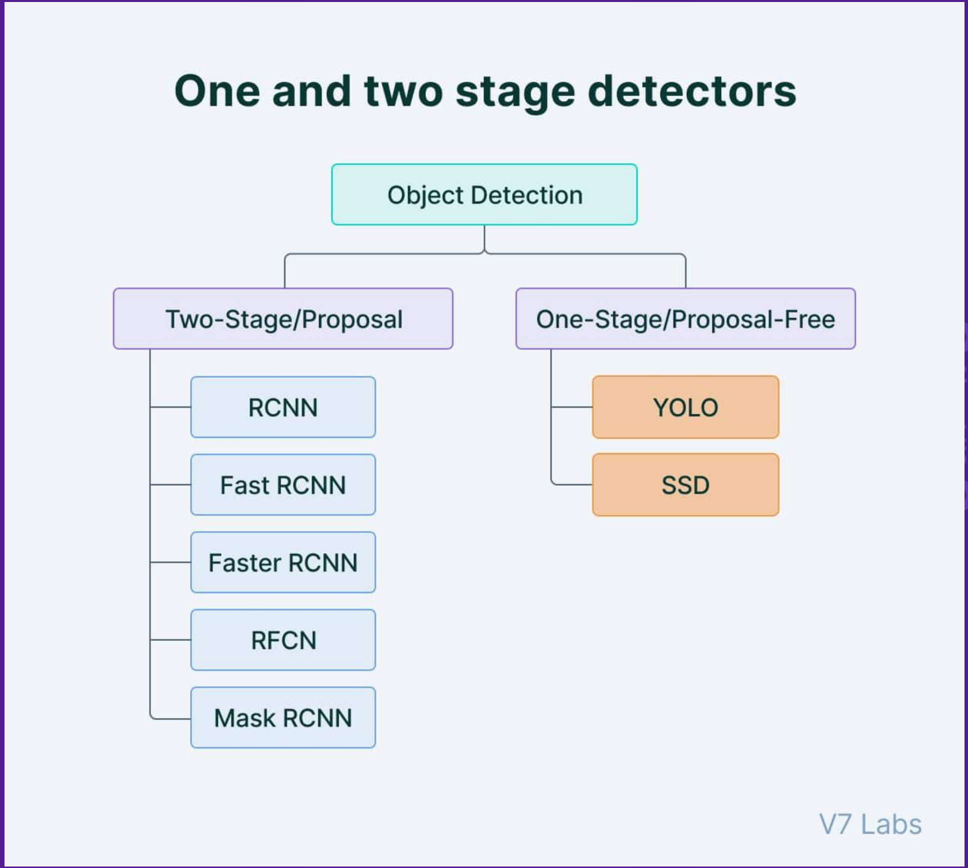
Object Detection

Two-Stage

- Two passes of the input image:
    - 1st: Proposal of set of regions that might contain the object.
    - 2nd: classify and refine regions.
- Computationally expensive.

One-Stage

- One pass of the input image.
- Predicts the bounding step without region proposal.
- Computationally efficient and faster.



One and two stage detectors

V7 Labs

(Kundu, 2023)

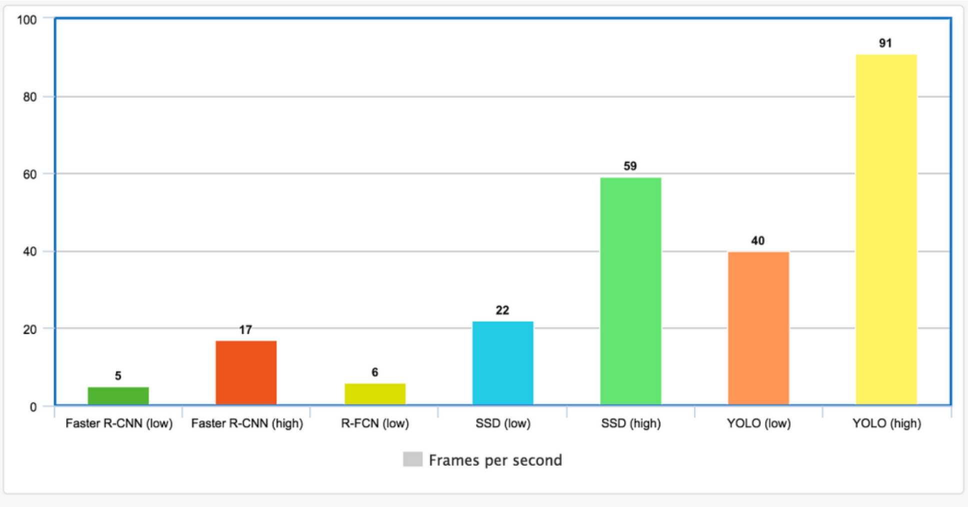
You Only Look Once

- Widely adopted.
- Easy to implement and adapt to new tasks due to its simple architecture.
- Struggles with small objects

Fast

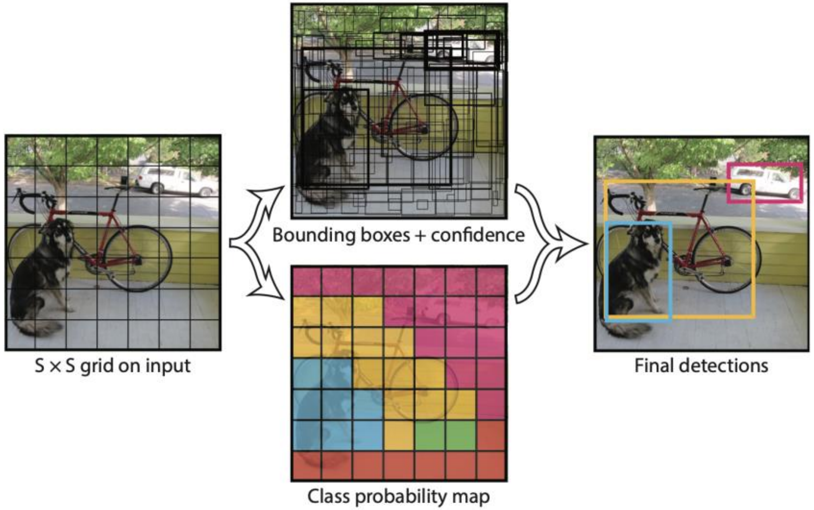
Efficient

Simple



(Hui, 2018)

How does it work?



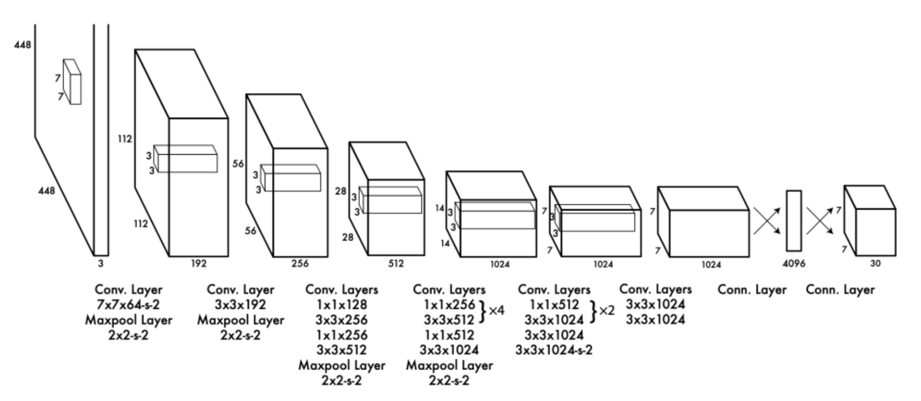
(Redmon et al., 2016)

The Model

- Divide image into an S x S grid.
- Predict B bounding boxes and the confidence for them.
- Predict class probabilities.
- Output final detections.

How does it work?

The Architecture



(Redmon et al., 2016)

Convolutional Layers

24

Extract features and create features map.

Connected Layers

2

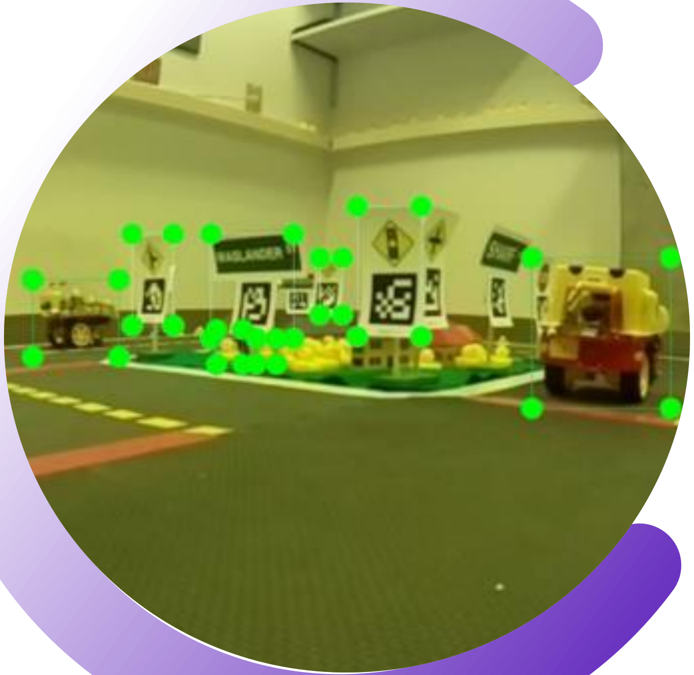
Predict bounding boxes and class probabilities for each cell in the grid.

Output

1

A list of bounding boxes with their respective probabilities.

Implementation



DUCKIETOWN

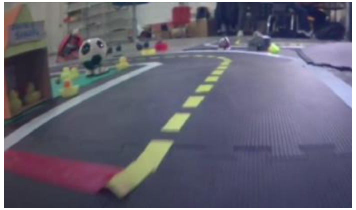
X

YOLO

Our Progress

Implementation

+500 similar images



The Dataset

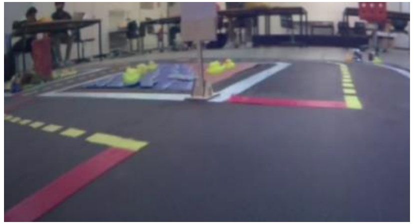
- 507 compressed images from the duckiebot's camera, capturing the different road signs, duckies and duckiebots during its movement.

Separation of blurry images from non-blurry ones.

The Dataset

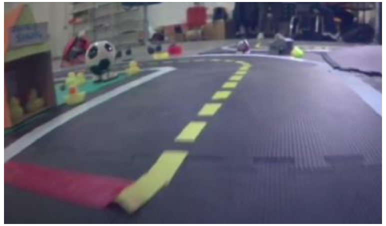
- For optimizing our object detection efficiency, we built a python script to only conserve the non-blurry images, increasing the model training quality, and reducing the training time span.

- With a filtering threshold of 200, the total number of images from the dataset gets reduced from +1500->507



Kept

✓

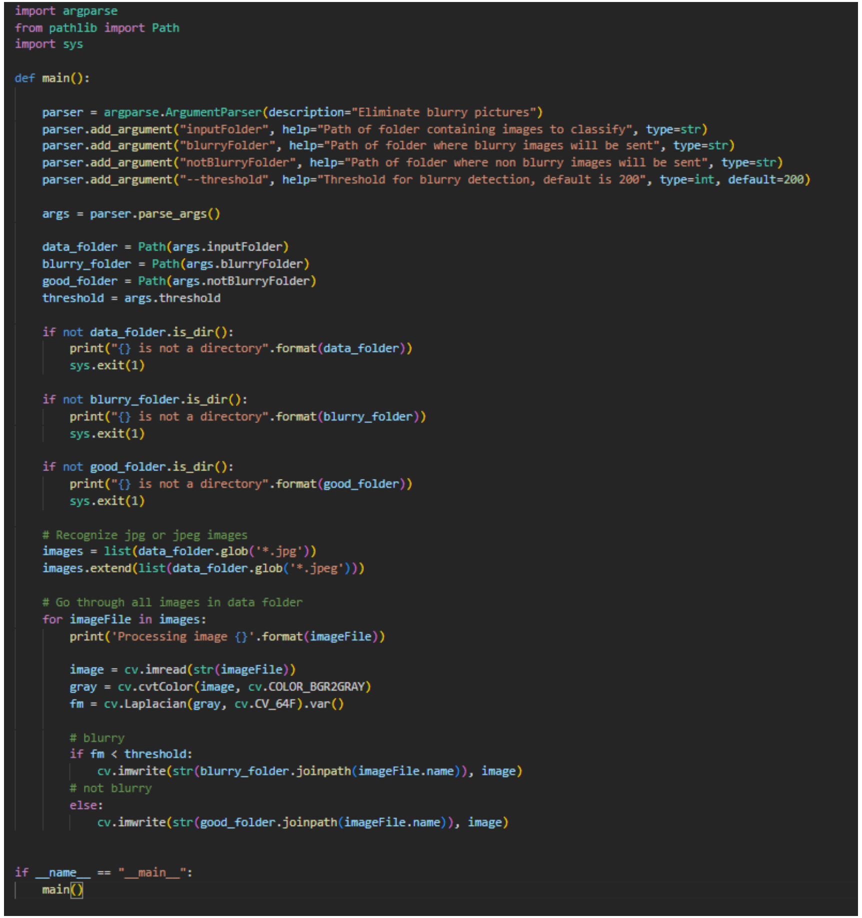


Removed

✗

The Laplacian Filter was used for this purpose

detect_blurry_img.py



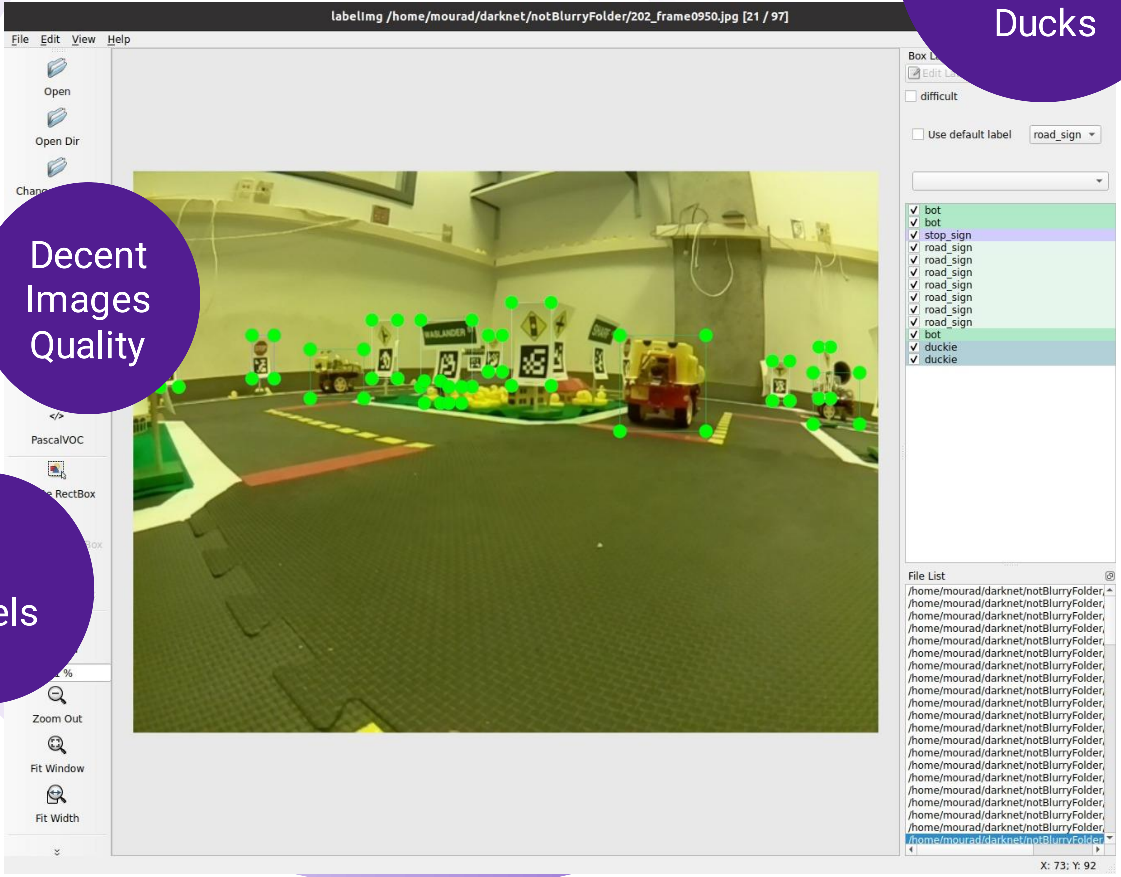
```
import argparse
from pathlib import Path
import sys

def main():

    parser = argparse.ArgumentParser(description="Eliminate blurry pictures")
    parser.add_argument("inputFolder", help="Path of folder containing images to classify", type=str)
    parser.add_argument("blurryFolder", help="Path of folder where blurry images will be sent", type=str)
    parser.add_argument("notBlurryFolder", help="Path of folder where non blurry images will be sent", type=str)
    parser.add_argument("--threshold", help="Threshold for blurry detection, default is 200", type=int, default=200)

    args = parser.parse_args()

    data_folder = Path(args.inputFolder)
    blurry_folder = Path(args.blurryFolder)
    good_folder = Path(args.notBlurryFolder)
    threshold = args.threshold

    if not data_folder.is_dir():
        print("{} is not a directory".format(data_folder))
        sys.exit(1)

    if not blurry_folder.is_dir():
        print("{} is not a directory".format(blurry_folder))
        sys.exit(1)

    if not good_folder.is_dir():
        print("{} is not a directory".format(good_folder))
        sys.exit(1)

    # Recognize jpg or jpeg images
    images = list(data_folder.glob('*.jpg'))
    images.extend(list(data_folder.glob('*.jpeg')))

    # Go through all images in data folder
    for imageFile in images:
        print('Processing image {}'.format(imageFile))

        image = cv.imread(str(imageFile))
        gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
        fm = cv.Laplacian(gray, cv.CV_64F).var()

        # blurry
        if fm < threshold:
            cv.imwrite(str(blurry_folder.joinpath(imageFile.name)), image)
        # not blurry
        else:
            cv.imwrite(str(good_folder.joinpath(imageFile.name)), image)


if __name__ == "__main__":
    main()
```

Labeling

Stop Signs
Road Signs
Duckiebots
Ducks

Decent Images Quality

4 Labels



labelImg /home/mourad/darknet/notBlurryFolder/202_frame0950.jpg [21 / 97]

□ difficult

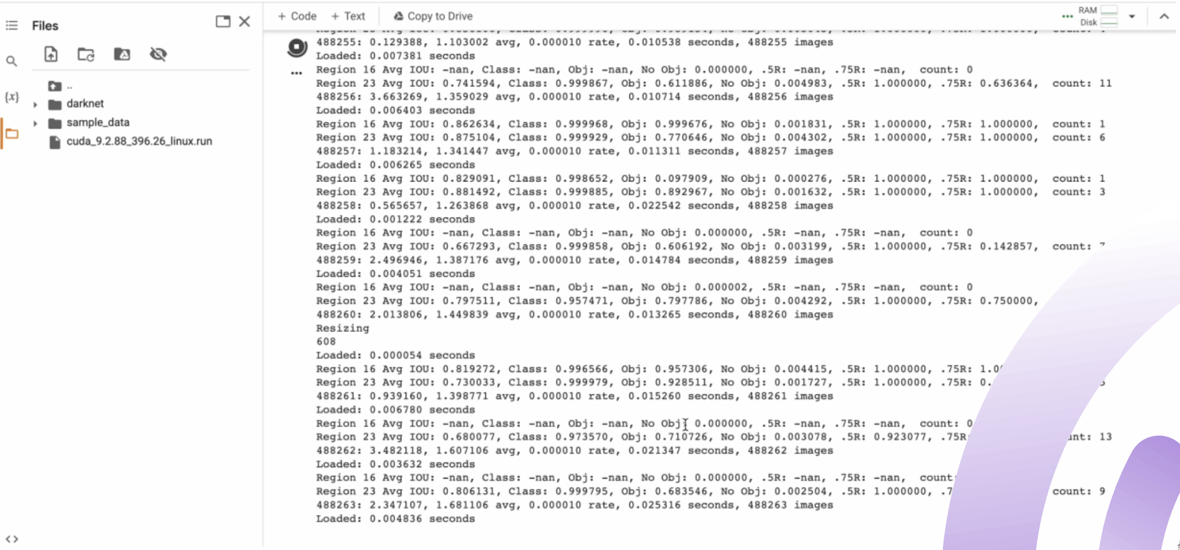
□ Use default label road_sign

- ✓ bot
- ✓ bot
- ✓ stop_sign
- ✓ road_sign
- ✓ road_sign
- ✓ road_sign
- ✓ road_sign
- ✓ road_sign
- ✓ road_sign
- ✓ road_sign
- ✓ bot
- ✓ duckie
- ✓ duckie

File List

/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder,
/home/mourad/darknet/notBlurryFolder

X: 73; Y: 92

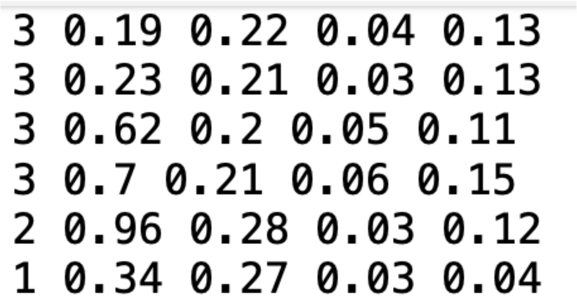


```
488255: 0.129388, 1.103002 avg, 0.000010 rate, 0.010538 seconds, 488255 images
Loaded: 0.007381 seconds
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 23 Avg IOU: 0.741594, Class: 0.999867, Obj: 0.611886, No Obj: 0.004983, .5R: 1.000000, .75R: 0.636364,  count: 11
488256: 3.663269, 1.359029 avg, 0.000010 rate, 0.010714 seconds, 488256 images
Loaded: 0.006403 seconds
Region 16 Avg IOU: 0.862634, Class: 0.999968, Obj: 0.999676, No Obj: 0.001831, .5R: 1.000000, .75R: 1.000000,  count: 1
Region 23 Avg IOU: 0.875104, Class: 0.999929, Obj: 0.770646, No Obj: 0.004302, .5R: 1.000000, .75R: 1.000000,  count: 6
488257: 1.183214, 1.341447 avg, 0.000010 rate, 0.011311 seconds, 488257 images
Loaded: 0.006265 seconds
Region 16 Avg IOU: 0.829091, Class: 0.998652, Obj: 0.097909, No Obj: 0.000276, .5R: 1.000000, .75R: 1.000000,  count: 1
Region 23 Avg IOU: 0.881492, Class: 0.999885, Obj: 0.892967, No Obj: 0.001632, .5R: 1.000000, .75R: 1.000000,  count: 3
488258: 0.565657, 1.263868 avg, 0.000010 rate, 0.022542 seconds, 488258 images
Loaded: 0.001222 seconds
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 23 Avg IOU: 0.667293, Class: 0.999858, Obj: 0.606192, No Obj: 0.003199, .5R: 1.000000, .75R: 0.142857,  count: 7
488259: 2.496946, 1.387176 avg, 0.000010 rate, 0.014784 seconds, 488259 images
Loaded: 0.004051 seconds
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000002, .5R: -nan, .75R: -nan,  count: 0
Region 23 Avg IOU: 0.797511, Class: 0.957471, Obj: 0.797786, No Obj: 0.004292, .5R: 1.000000, .75R: 0.750000,
488260: 2.013806, 1.449839 avg, 0.000010 rate, 0.013265 seconds, 488260 images
Resizing
608
Loaded: 0.000054 seconds
Region 16 Avg IOU: 0.819272, Class: 0.996566, Obj: 0.957306, No Obj: 0.004415, .5R: 1.000000, .75R: 1.0
Region 23 Avg IOU: 0.730033, Class: 0.999979, Obj: 0.928511, No Obj: 0.001727, .5R: 1.000000, .75R: 0.
488261: 0.939160, 1.398771 avg, 0.000010 rate, 0.015260 seconds, 488261 images
Loaded: 0.006780 seconds
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj 0.000000, .5R: -nan, .75R: -nan,  count: 0
Region 23 Avg IOU: 0.680077, Class: 0.973570, Obj: 0.710726, No Obj: 0.003078, .5R: 0.923077, .75R          unt: 13
488262: 3.482118, 1.607106 avg, 0.000010 rate, 0.021347 seconds, 488262 images
Loaded: 0.003632 seconds
Region 16 Avg IOU: -nan, Class: -nan, Obj: -nan, No Obj: 0.000000, .5R: -nan, .75R: -nan,  count
Region 23 Avg IOU: 0.806131, Class: 0.999795, Obj: 0.683546, No Obj: 0.002504, .5R: 1.000000, .7            count: 9
488263: 2.347107, 1.681106 avg, 0.000010 rate, 0.025316 seconds, 488263 images
Loaded: 0.004836 seconds
```

Training

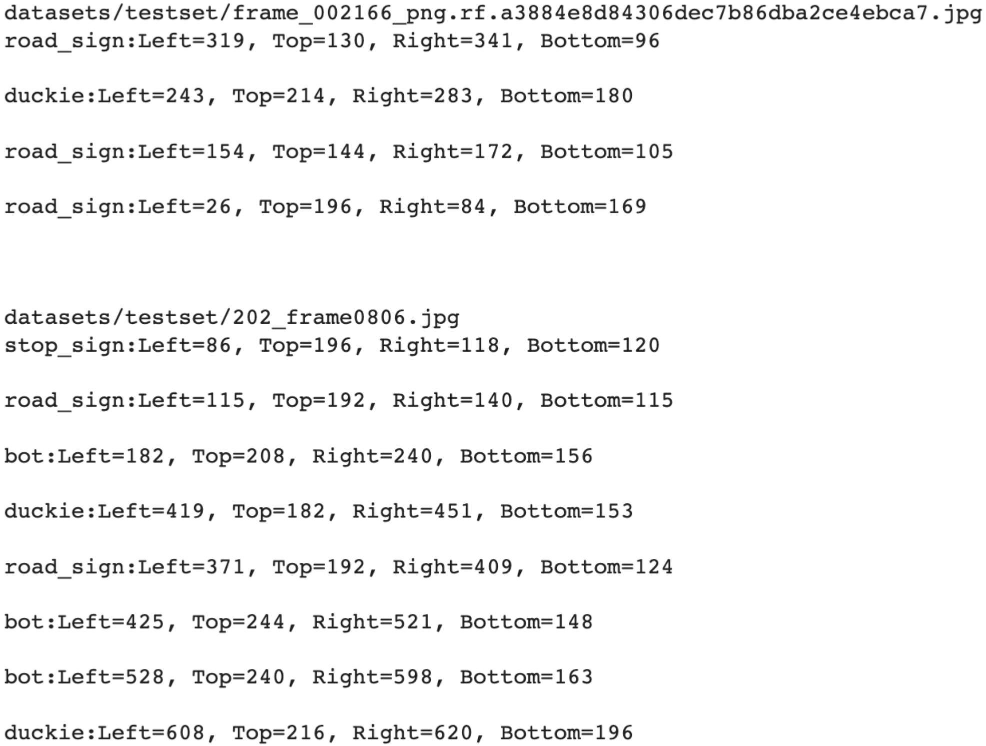
- Prepare the data for training:
  - Training, Testing, Evaluation sets.
- Prepare config. Files:
  - Classes, model selection, determine classes.
- Fork Darknet repository

Ground Truth Bounding Boxes



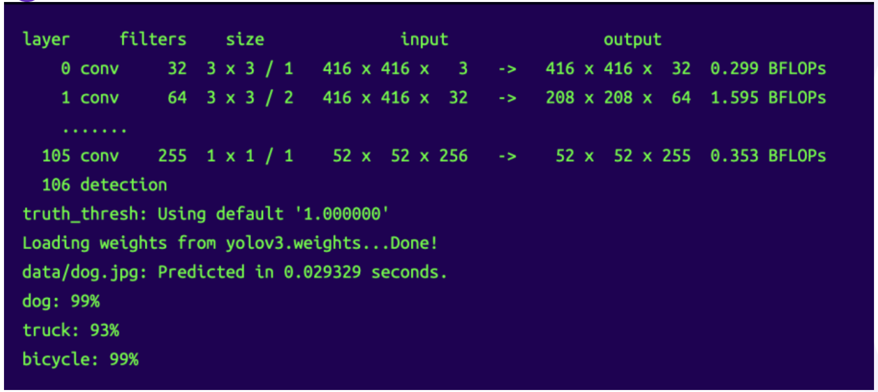
```
3 0.19 0.22 0.04 0.13
3 0.23 0.21 0.03 0.13
3 0.62 0.2 0.05 0.11
3 0.7 0.21 0.06 0.15
2 0.96 0.28 0.03 0.12
1 0.34 0.27 0.03 0.04
```

Label



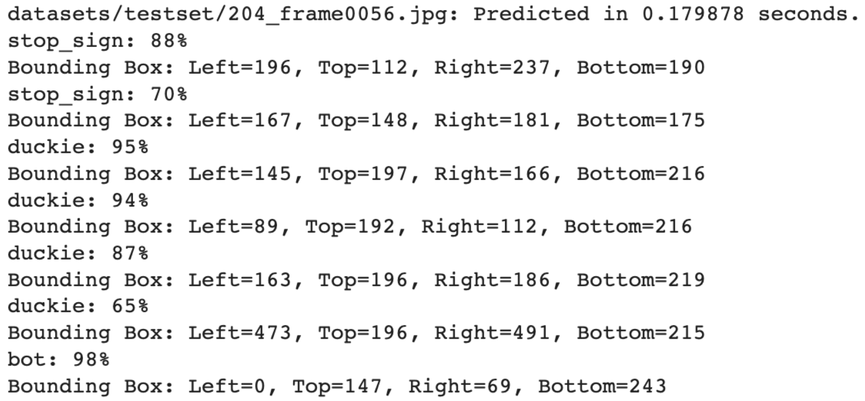
```
datasets/testset/frame_002166_png.rf.a3884e8d84306dec7b86dba2ce4ebca7.jpg
road_sign:Left=319, Top=130, Right=341, Bottom=96

duckie:Left=243, Top=214, Right=283, Bottom=180

road_sign:Left=154, Top=144, Right=172, Bottom=105

road_sign:Left=26, Top=196, Right=84, Bottom=169



datasets/testset/202_frame0806.jpg
stop_sign:Left=86, Top=196, Right=118, Bottom=120

road_sign:Left=115, Top=192, Right=140, Bottom=115

bot:Left=182, Top=208, Right=240, Bottom=156

duckie:Left=419, Top=182, Right=451, Bottom=153

road_sign:Left=371, Top=192, Right=409, Bottom=124

bot:Left=425, Top=244, Right=521, Bottom=148

bot:Left=528, Top=240, Right=598, Bottom=163

duckie:Left=608, Top=216, Right=620, Bottom=196
```

Editing the source code



```
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  32  0.299 BFLOPs
    1 conv     64  3 x 3 / 2   416 x 416 x  32   ->   208 x 208 x  64  1.595 BFLOPs
    .......
  105 conv    255  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 255  0.353 BFLOPs
  106 detection
truth_thresh: Using default '1.000000'
Loading weights from yolov3.weights...Done!
data/dog.jpg: Predicted in 0.029329 seconds.
dog: 99%
truck: 93%
bicycle: 99%
```

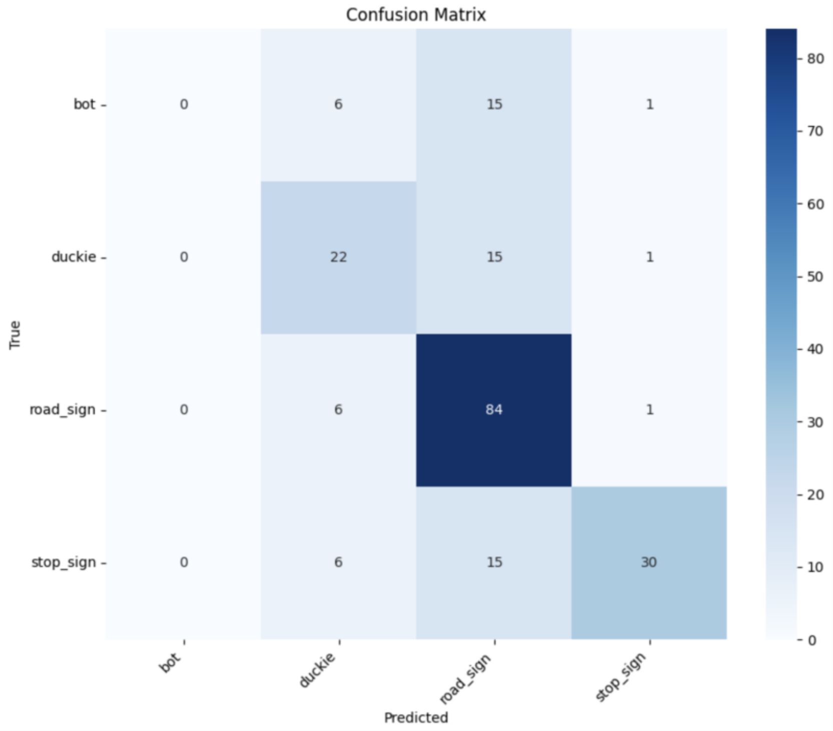
Before



```
datasets/testset/204_frame0056.jpg: Predicted in 0.179878 seconds.
stop_sign: 88%
Bounding Box: Left=196, Top=112, Right=237, Bottom=190
stop_sign: 70%
Bounding Box: Left=167, Top=148, Right=181, Bottom=175
duckie: 95%
Bounding Box: Left=145, Top=197, Right=166, Bottom=216
duckie: 94%
Bounding Box: Left=89, Top=192, Right=112, Bottom=216
duckie: 87%
Bounding Box: Left=163, Top=196, Right=186, Bottom=219
duckie: 65%
Bounding Box: Left=473, Top=196, Right=491, Bottom=215
bot: 98%
Bounding Box: Left=0, Top=147, Right=69, Bottom=243
```

After

Results


Confusion Matrix

True \ Predicted	bot	duckie	road_sign	stop_sign
bot	0	6	15	1
duckie	0	22	15	1
road_sign	0	6	84	1
stop_sign	0	6	15	30

Class: bot
True Positives (TP): 0
True Negatives (TN): 0
False Positives (FP): 0
False Negatives (FN): 44

Class: duckie
True Positives (TP): 22
True Negatives (TN): 0
False Positives (FP): 6
False Negatives (FN): 68

Class: road_sign
True Positives (TP): 84
True Negatives (TN): 0
False Positives (FP): 15
False Negatives (FN): 40

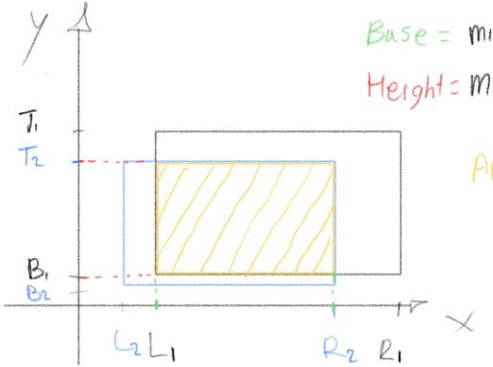
Class: stop_sign
True Positives (TP): 30
True Negatives (TN): 0
False Positives (FP): 1
False Negatives (FN): 13

Intersection

Base = $\min(R_1, R_2) - \max(L_1, L_2)$

Height = $\min(T_1, T_2) - \max(B_1, B_2)$

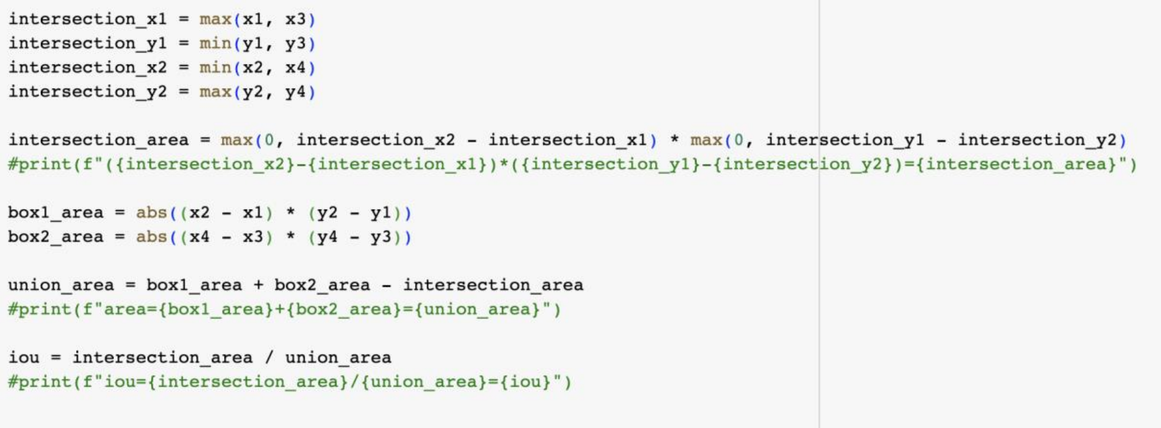
Area = Base × Height



$$IoU = \frac{Area}{Area}$$

Union:

$$Area = (T_1 - B_1)(R_1 - L_1) + (T_2 - B_2)(R_2 - L_2) - Area$$



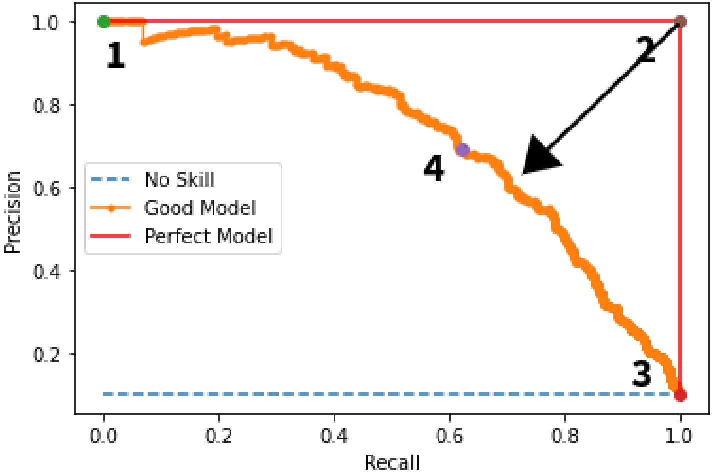
```
intersection_x1 = max(x1, x3)
intersection_y1 = min(y1, y3)
intersection_x2 = min(x2, x4)
intersection_y2 = max(y2, y4)

intersection_area = max(0, intersection_x2 - intersection_x1) * max(0, intersection_y1 - intersection_y2)
#print(f"({intersection_x2}-{intersection_x1})*({intersection_y1}-{intersection_y2})={intersection_area}")

box1_area = abs((x2 - x1) * (y2 - y1))
box2_area = abs((x4 - x3) * (y4 - y3))

union_area = box1_area + box2_area - intersection_area
#print(f"area={box1_area}+{box2_area}={union_area}")

iou = intersection_area / union_area
#print(f"iou={intersection_area}/{union_area}={iou}")
```

Results

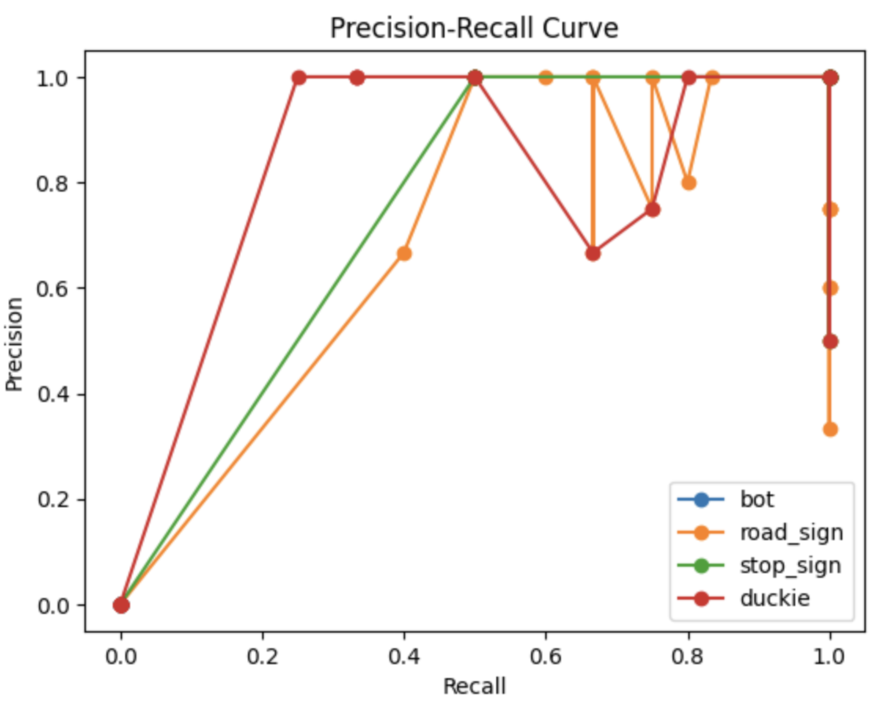
Precision: how often does the model predict correctly?
Precision = TP/(TP+FP)

Recall: has the model predicted every time that it should have predicted?
Recall = TP/(TP+FN)

Average precision: Area under the curve.



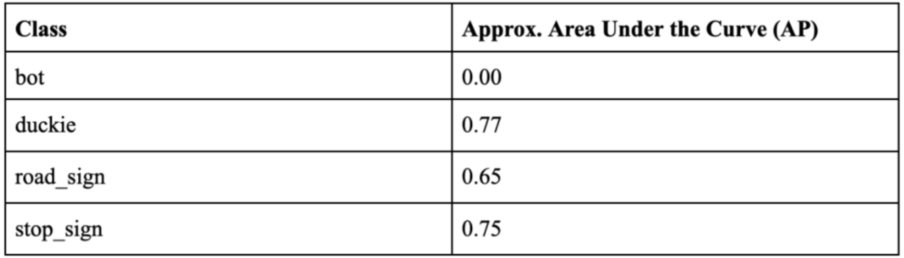
Reference



Results

Results

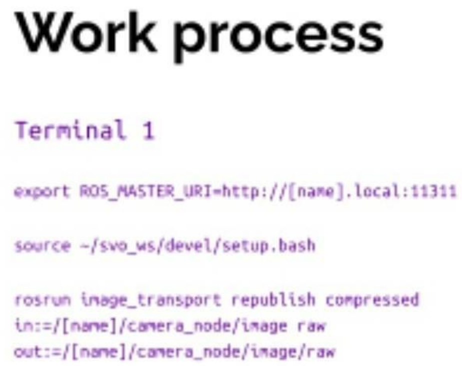
Average precision



Class	Approx. Area Under the Curve (AP)
bot	0.00
duckie	0.77
road_sign	0.65
stop_sign	0.75

Mean Average Precision (mAP) = 0.54

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$



Work process

Terminal 1

```
export ROS_MASTER_URI=http://[name].local:11311

source ~/svo_ws/devel/setup.bash

rosrun image_transport republish compressed
in:=/[name]/camera_node/image raw
out:=/[name]/camera_node/image/raw
```

Challenges and Recommendations

Assessment Problems

- The "Bot" class only showed False Negatives (FN), and the model failed to detect any instances of this class

Duckiebot

- Wiring problems caused problems turning it on and off.
- Problems with Jetson Nano OS

darknet_ros node

- Problems accessing the node

THANK YOU

"You only live once - YOLO."
:)