PROJECT

SPACE SOFTWARE

By:
Arda Erdal
Aral Deniz Uzun
Talha Demirbilek
Ensar Demirbilek
Mürsel Efe Can Çalışkan

Our Challenge

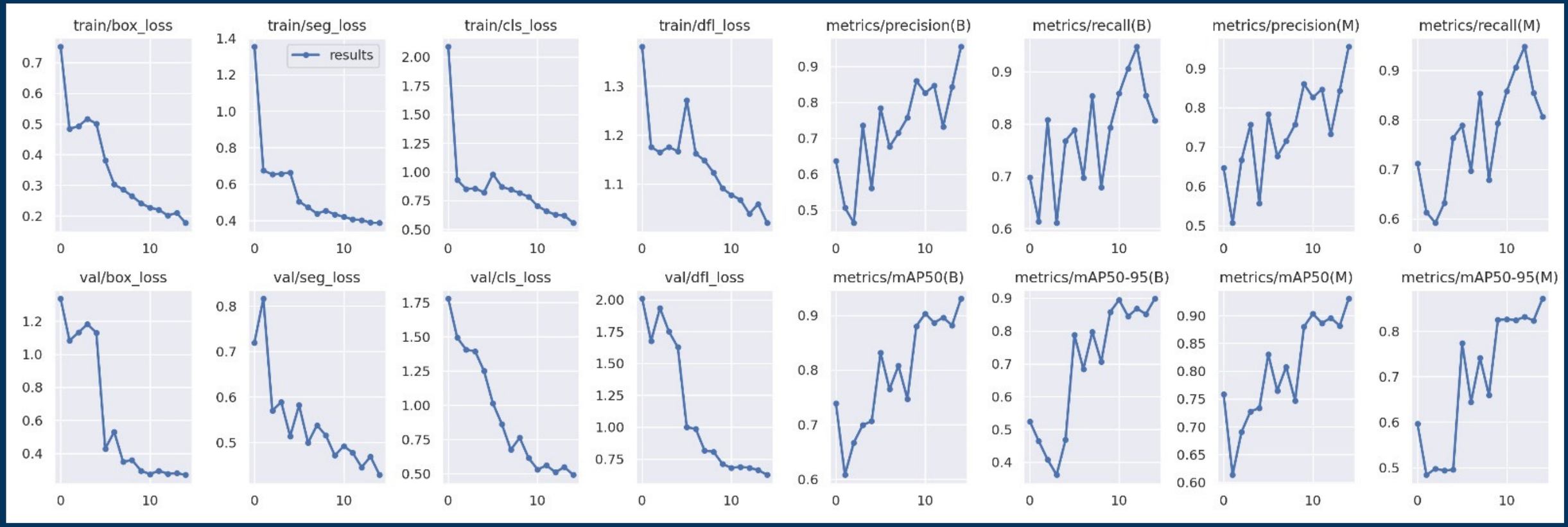
GeoAI Reimagined

Our challenge is to develop an accurate AI model that could be useful in a range of critical geospatial applications, such as disaster recovery operations, environmental change monitoring geological data discovery, and more.

Our Goal

In this project, our goal is to leverage the power of AI to enable a range of critical geospatial applications such as environmental planning, environmental change monitoring, geospatial data discovery and more.

....CHART CREATED BY THE ARTIFICIAL INTELLIGENCE (AI) WE WROTE USING THE PYTHON PROGRAMMING LANGUAGE



Availability

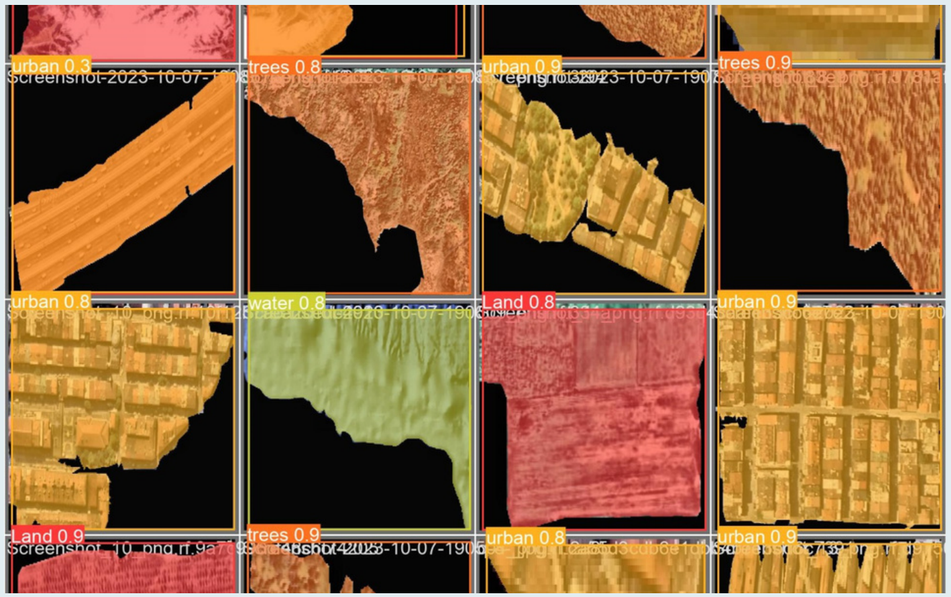
What Makes We Different?

In our program, everyone will be able to create and observe the designed visual without difficulty. With a simple interface and design we aim to make it accessible and understandable to everyone.

Where can we use this program in daily life?

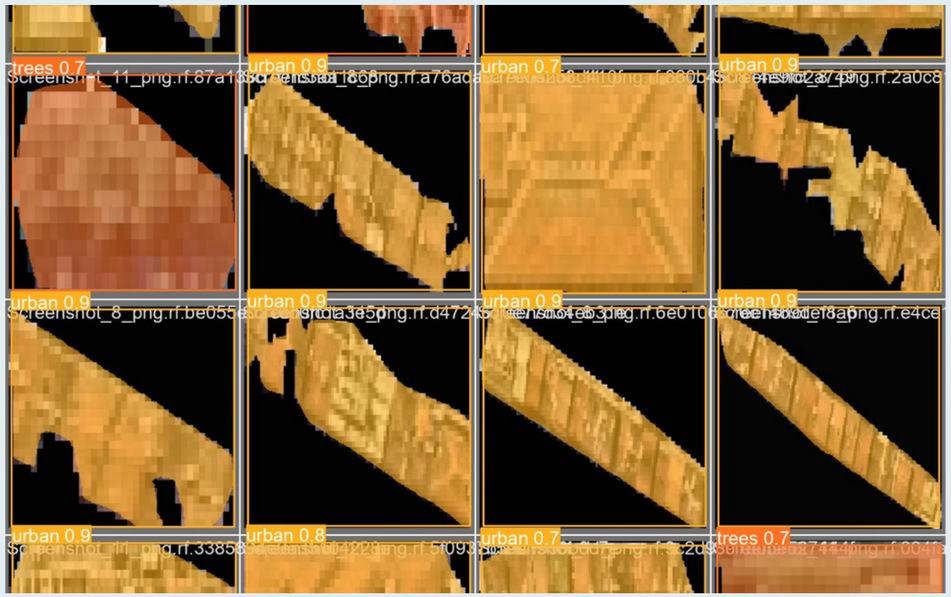
It can be used in many areas such as school projects, so that civil engineers can more easily obtain information about where to build, and the availability of the place to build.

CONFUSİON MATRİX-PREDİCTİON GRAPH THAT EMERGED WHİLE DOİNG OUR PROJECT



We used different colors

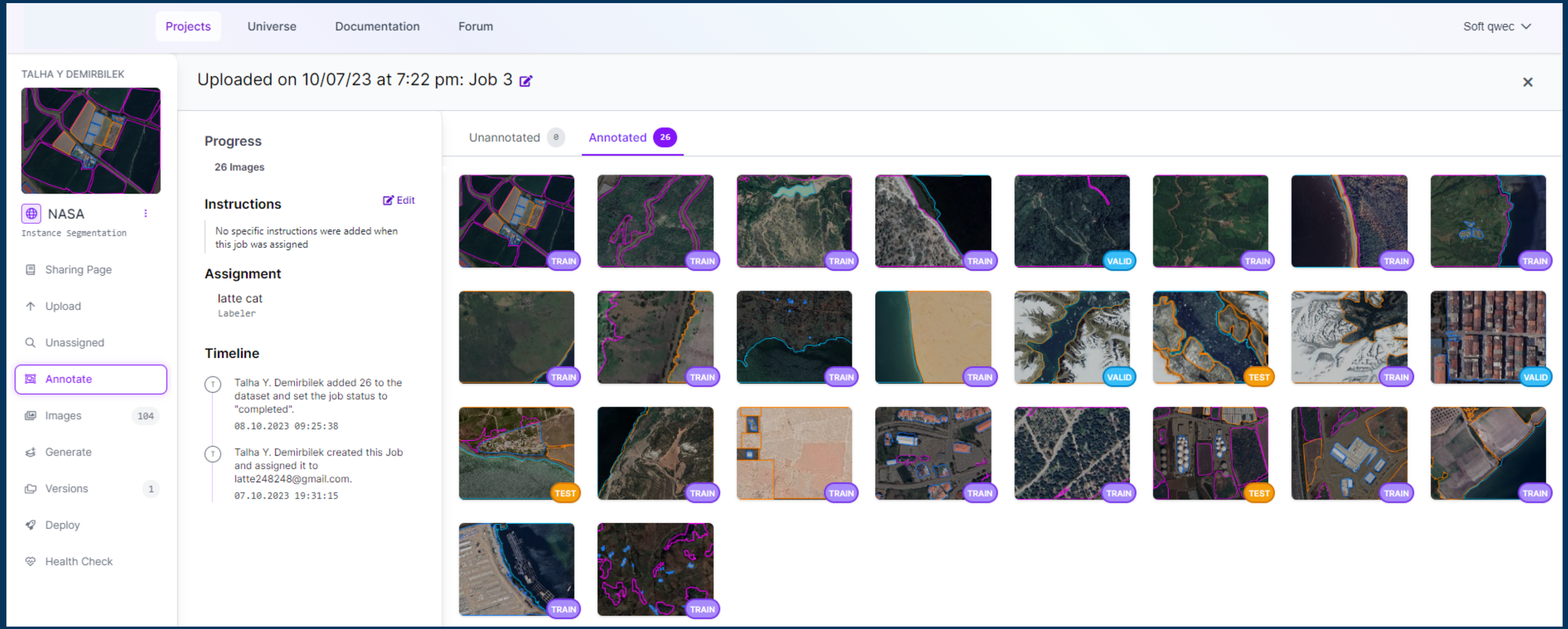
While making the program, structures that are different in the area searches are labelled with different colors.



Similar structures, Similar colors

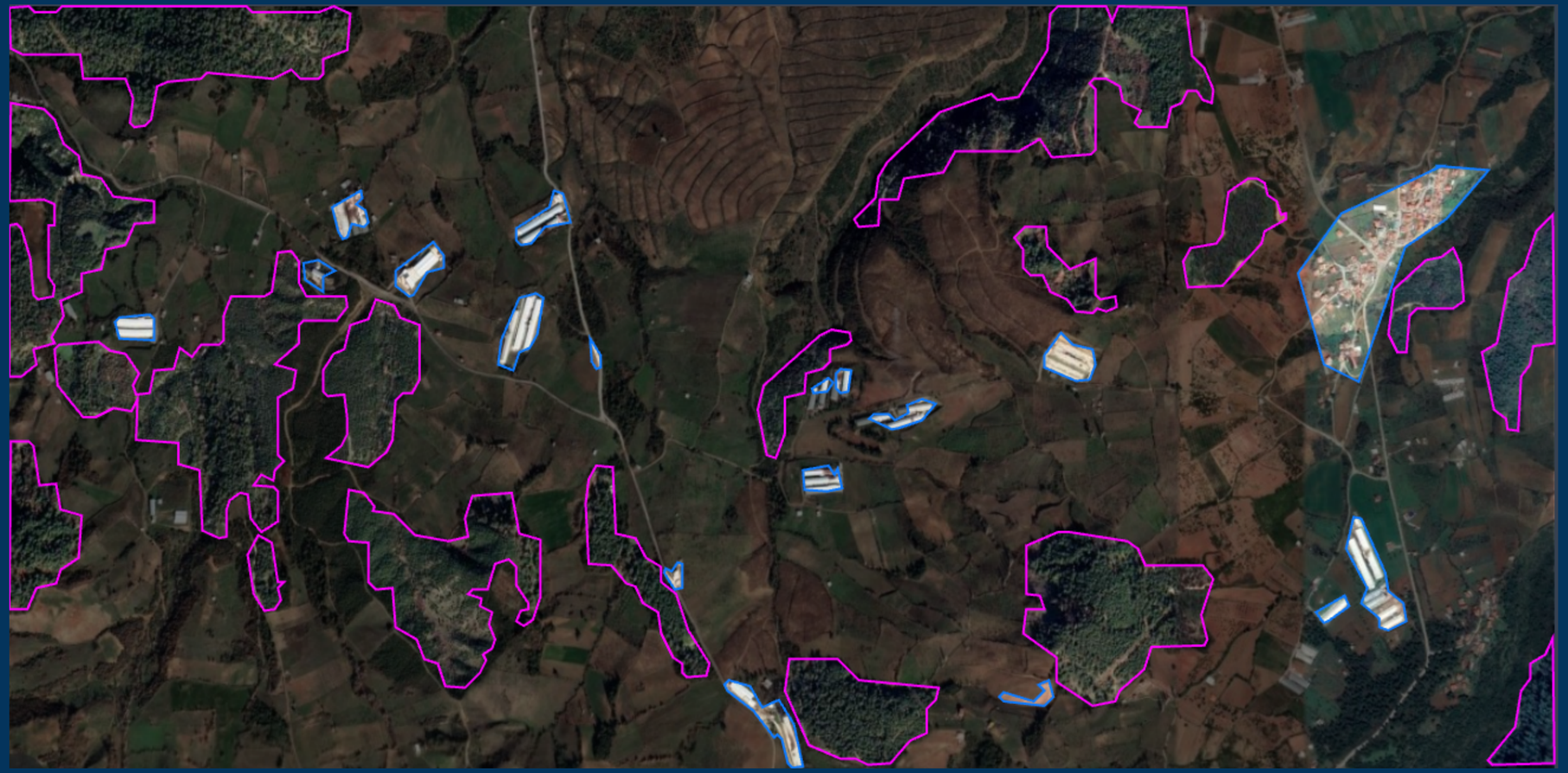
We used similar colors when modeling similar or identical structures.

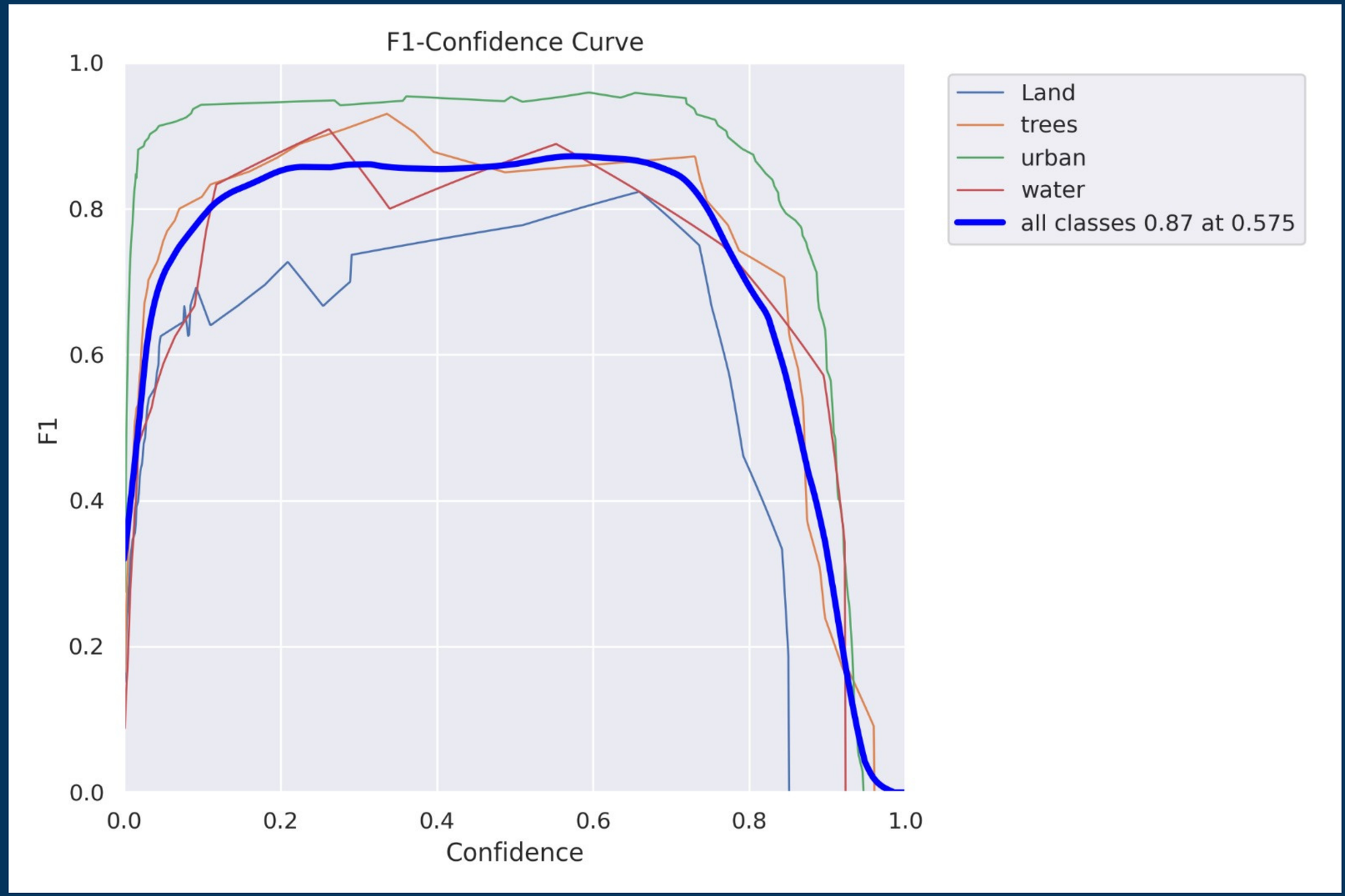
Labeling studies on data in the demo phase



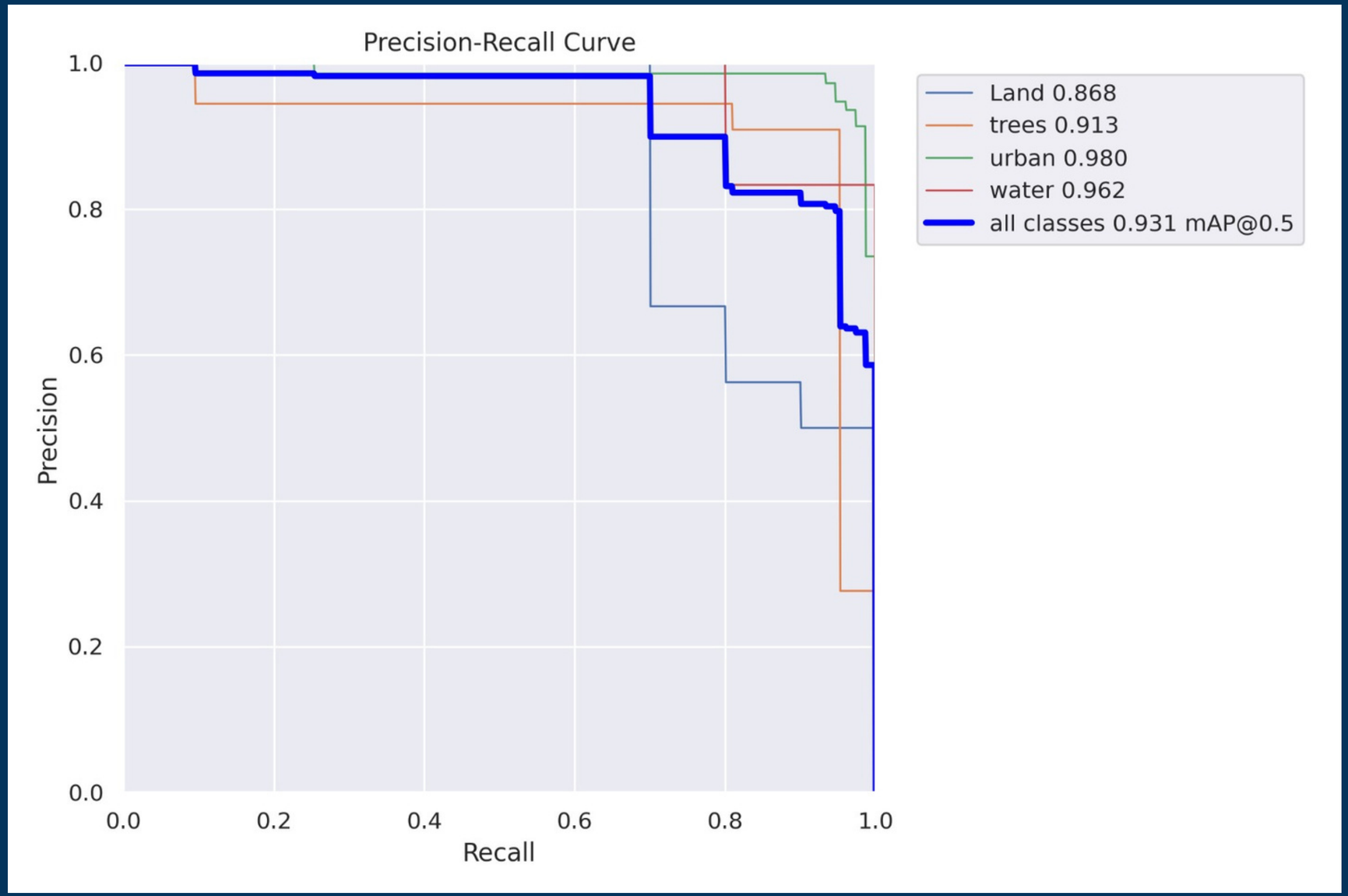
LABELLING EXAMPLES

A LABELİNG EXAMPLE THAT WE STUDİED İN DETAİL





F1-CONFIDENCE CURVE

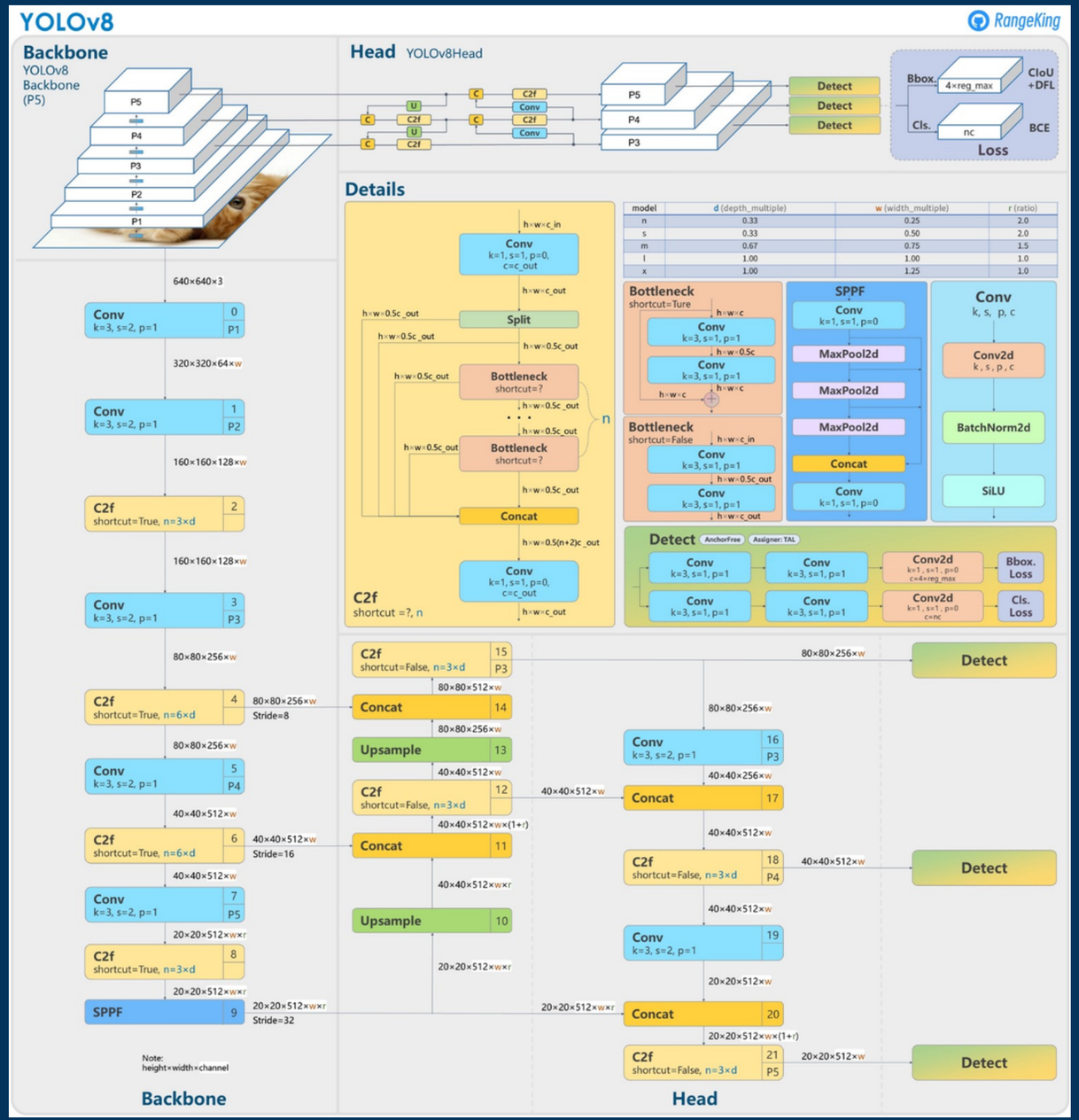


Precision-Recall Curve

HOW DOES IT WORK

YOLO algorithm is an algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in one run of the Algorithm.

HOW DOES IT WORK



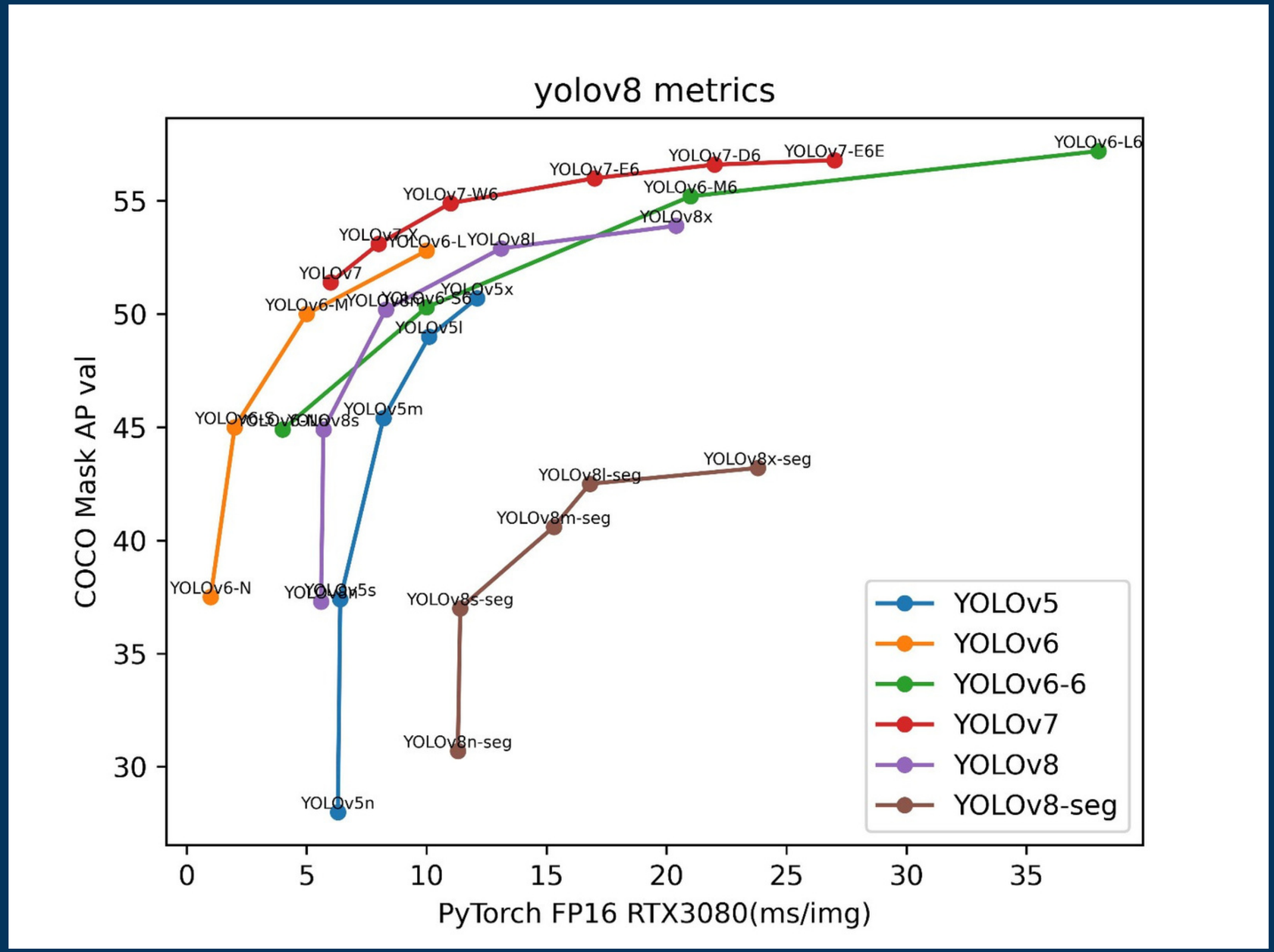
Object detection is a popular task in computer vision.

It deals with localizing a region of interest within an image and classifying this region like a typical image classifier. One image can include several regions of interest pointing to different objects. This makes object detection a more advanced problem of image classification.

YOLO (You Only Look Once) is a popular object detection model known for its speed and accuracy. It was first introduced by Joseph Redmon et al. in 2016 and has since undergone several iterations, the latest being YOLO v7.

In this article, we will discuss what makes YOLO v7 stand out and how it compares to other object detection algorithms.

Yolov8 vs older versions of Yolo



Data about how we train AI

```
%cd {HOME}

!yolo task=segment mode=train model=yolov8s-seg.pt data=/content/datasets/NASA-1/data.yaml epochs=15 imgsz=800 plots=True
```

OK

```
      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       1/15      7.57G     0.7522      1.354       2.09       1.38         56        800: 100% 111/111 [01:51<00:00,  1.01s/it]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.18it/s]
                   all        111        111      0.637      0.698      0.739      0.524      0.647      0.711      0.758      0.596
                  Land        111         10      0.507          1      0.851      0.665      0.506          1      0.851      0.734
                 trees        111         21      0.525      0.619      0.623       0.41      0.565      0.667      0.681        0.5
                 urban        111         75      0.764      0.573      0.661      0.399       0.77      0.579      0.682      0.456
                 water        111          5      0.752        0.6      0.819      0.624      0.748        0.6      0.819      0.695

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       2/15      9.53G     0.4841     0.6753     0.9307      1.175         55        800: 100% 111/111 [01:46<00:00,  1.04it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.31it/s]
                   all        111        111      0.507      0.613      0.608      0.464      0.507      0.613      0.613      0.484
                  Land        111         10      0.613        0.6      0.601       0.51      0.613        0.6      0.601      0.544
                 trees        111         21      0.498      0.619       0.63      0.516      0.498      0.619      0.646      0.546
                 urban        111         75      0.354       0.96      0.833      0.596      0.354       0.96      0.839      0.616
                 water        111          5      0.562      0.275      0.368      0.236      0.562      0.275      0.368      0.229

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       3/15      9.53G     0.4926     0.6533     0.8514      1.165         56        800: 100% 111/111 [01:45<00:00,  1.06it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 4/4 [00:02<00:00,  1.36it/s]
                   all        111        111      0.464      0.808      0.667      0.408      0.667      0.592      0.691      0.498
                  Land        111         10      0.423        0.8      0.627      0.366      0.631        0.6      0.627      0.457
                 trees        111         21      0.316      0.725      0.576      0.351      0.517      0.714      0.664      0.439
                 urban        111         75      0.621      0.907       0.88      0.536      0.804      0.853      0.903      0.638
                 water        111          5      0.498        0.8      0.586      0.378      0.716        0.2      0.568      0.456

      Epoch    GPU_mem   box_loss   seg_loss   cls_loss   dfl_loss  Instances       Size
       4/15      9.53G     0.5167     0.6571     0.8571      1.176         51        800: 100% 111/111 [01:44<00:00,  1.06it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 4/4 [00:03<00:00,  1.27it/s]
                   all        111        111      0.735      0.611      0.699      0.361      0.757      0.632      0.728      0.494
                  Land        111         10      0.693      0.679      0.731      0.312      0.693      0.678      0.731      0.491
                 trees        111         21      0.538      0.619      0.621      0.328      0.582      0.667      0.681      0.434
                 urban        111         75      0.711      0.747      0.719       0.49      0.755      0.787      0.773      0.502
                 water        111          5          1      0.399      0.726      0.315          1      0.397      0.726      0.547
```

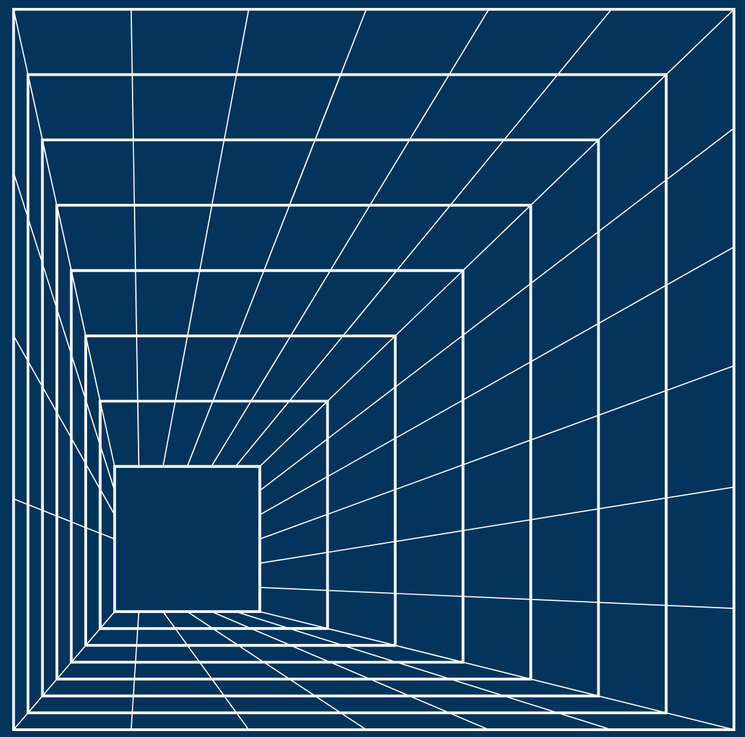
VALIDATION OF THE TRAINED MODEL

```
[ ]  %cd {HOME}

     !yolo task=segment mode=val model={HOME}/runs/segment/train2/weights/best.pt data=/content/datasets/NASA-1/data.yaml
```

```
/content
2023-10-08 10:28:09.512572: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
2023-10-08 10:28:10.407520: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
YOLOv8s-seg summary (fused): 195 layers, 11781535 parameters, 0 gradients, 42.4 GFLOPs
val: Scanning /content/datasets/NASA-1/valid/labels.cache... 111 images, 0 backgrounds, 0 corrupt: 100% 111/111 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95)     Mask(P          R      mAP50  mAP50-95): 100% 7/7 [00:08<00:00,  1.25s/it]
                   all        111        111      0.955      0.807      0.931      0.899      0.955      0.807      0.931       0.87
                  Land        111         10      0.937        0.7      0.868      0.856      0.937        0.7      0.868      0.837
                 trees        111         21      0.915       0.81      0.913      0.913      0.915       0.81      0.913      0.851
                 urban        111         75      0.969      0.947       0.98      0.977      0.969      0.947       0.98      0.913
                 water        111          5          1       0.77      0.962      0.848          1       0.77      0.962      0.879
Speed: 4.8ms pre-process, 24.2ms inference, 0.0ms loss, 5.3ms post-process per image
```

PREDICTION/TESTING

```
%cd {HOME}
!yolo task=segment mode=predict model={HOME}/runs/segment/train2/weights/best.pt conf=0.25 source=/content/datasets/NASA-1/test/images save=True
```

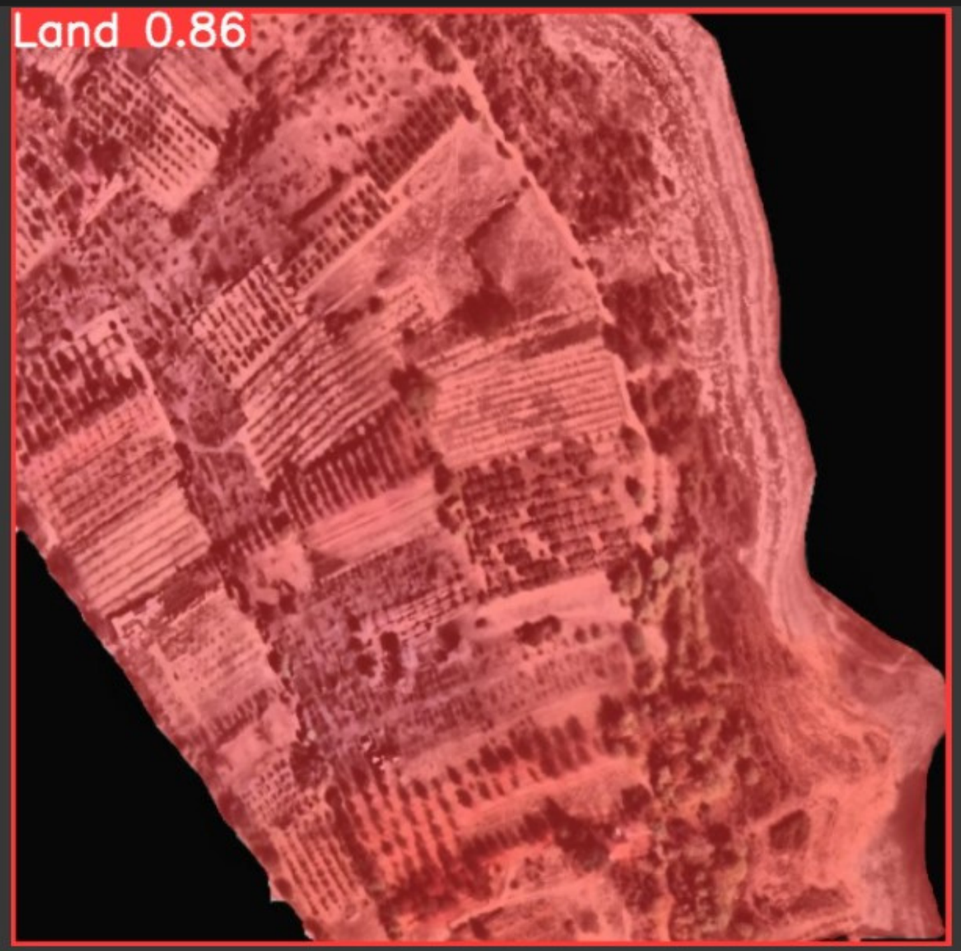


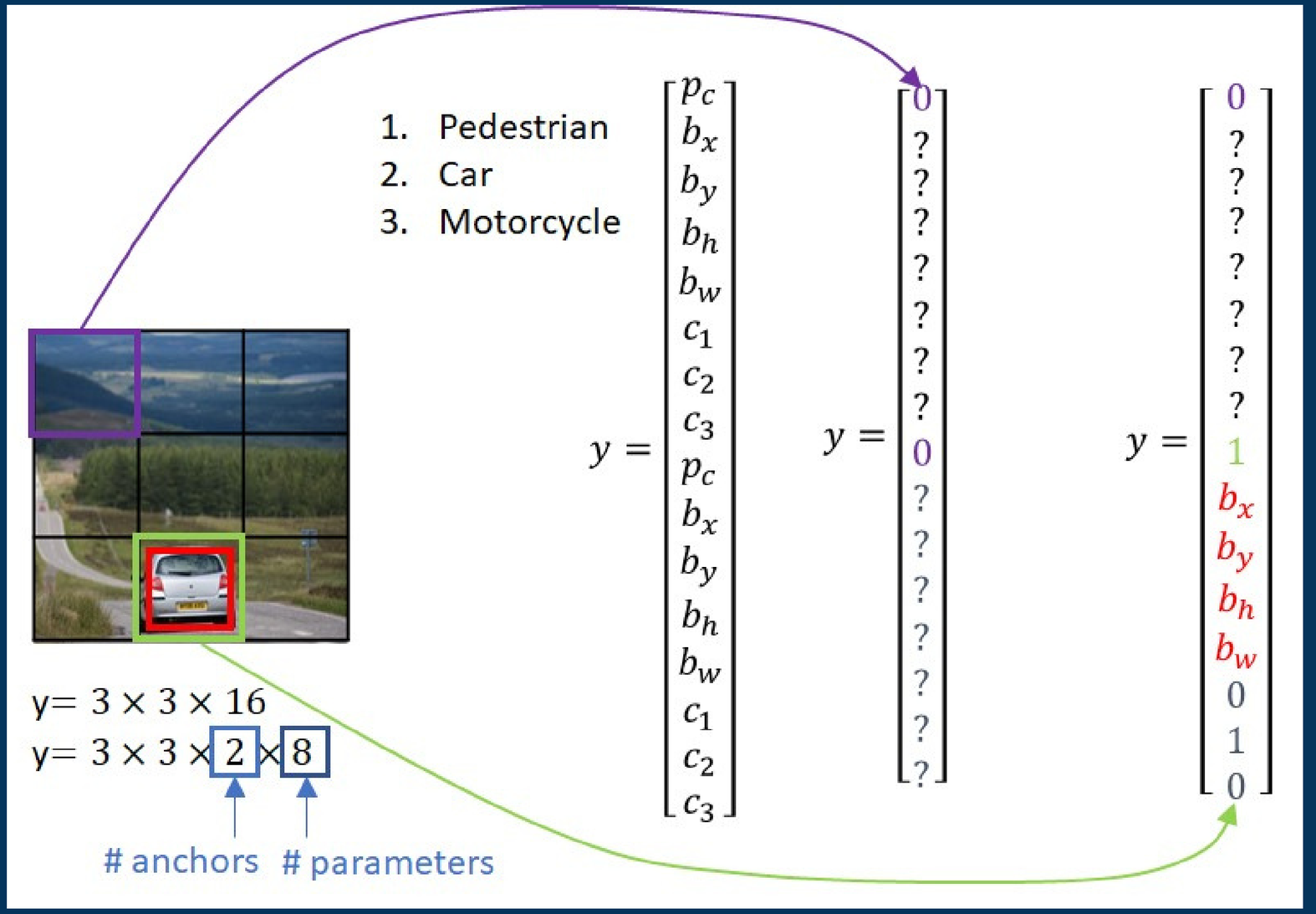
Importing The Dataset

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="Personel-API")
project = rf.workspace("talha-y-demirbilek").project("nasa-nrx2h")
dataset = project.version(1).download("yolov8")
```

TESTING SAMPLE VISUALIZATION

```
import glob
from IPython.display import Image, display

for image_path in glob.glob(f'{HOME}/runs/segment/predict3/*.jpg')[5:30]:
      display(Image(filename=image_path, width=600))
      print("\n")
```




1. Pedestrian
2. Car
3. Motorcycle
$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$
$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$
$y = \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$
y= 3 × 3 × 16
y= 3 × 3 × 2 × 8
# anchors
# parameters

Where is object detection used today?

Today, object recognition is the core of most vision-based AI software and programs. Object detection plays an important role in scene understanding, which is popular in security, construction, transportation, medical, and military use cases.

The End

We thank you for reading our presentation until the end and wish everyone success.

NASA