Low-cost fast mobile object detection for robotic soccer

Why is this important?

Intelligent robots to win from Humans

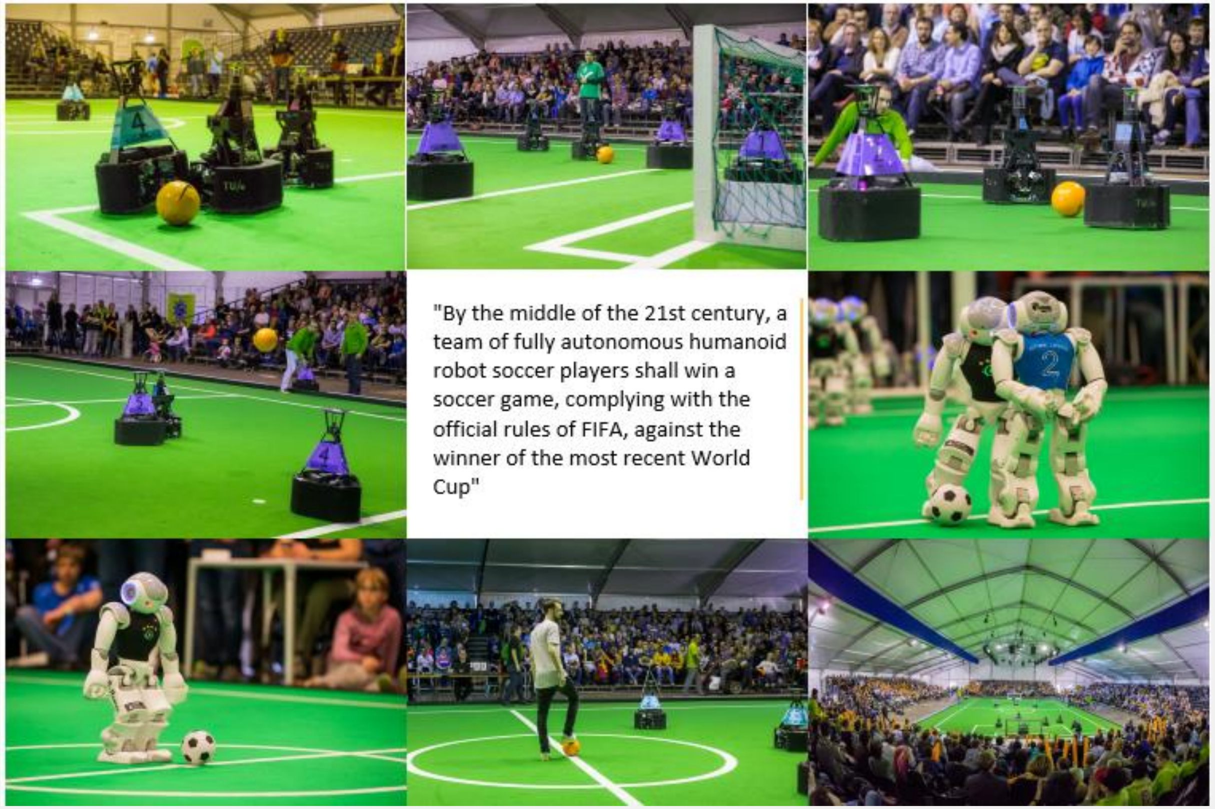
- If we want to win in 2050, we need adaptive intelligent soccer robots.

Bring the A.I. revolution to RoboCup

- State-of-the-art object detection algorithms bring immediate benefits

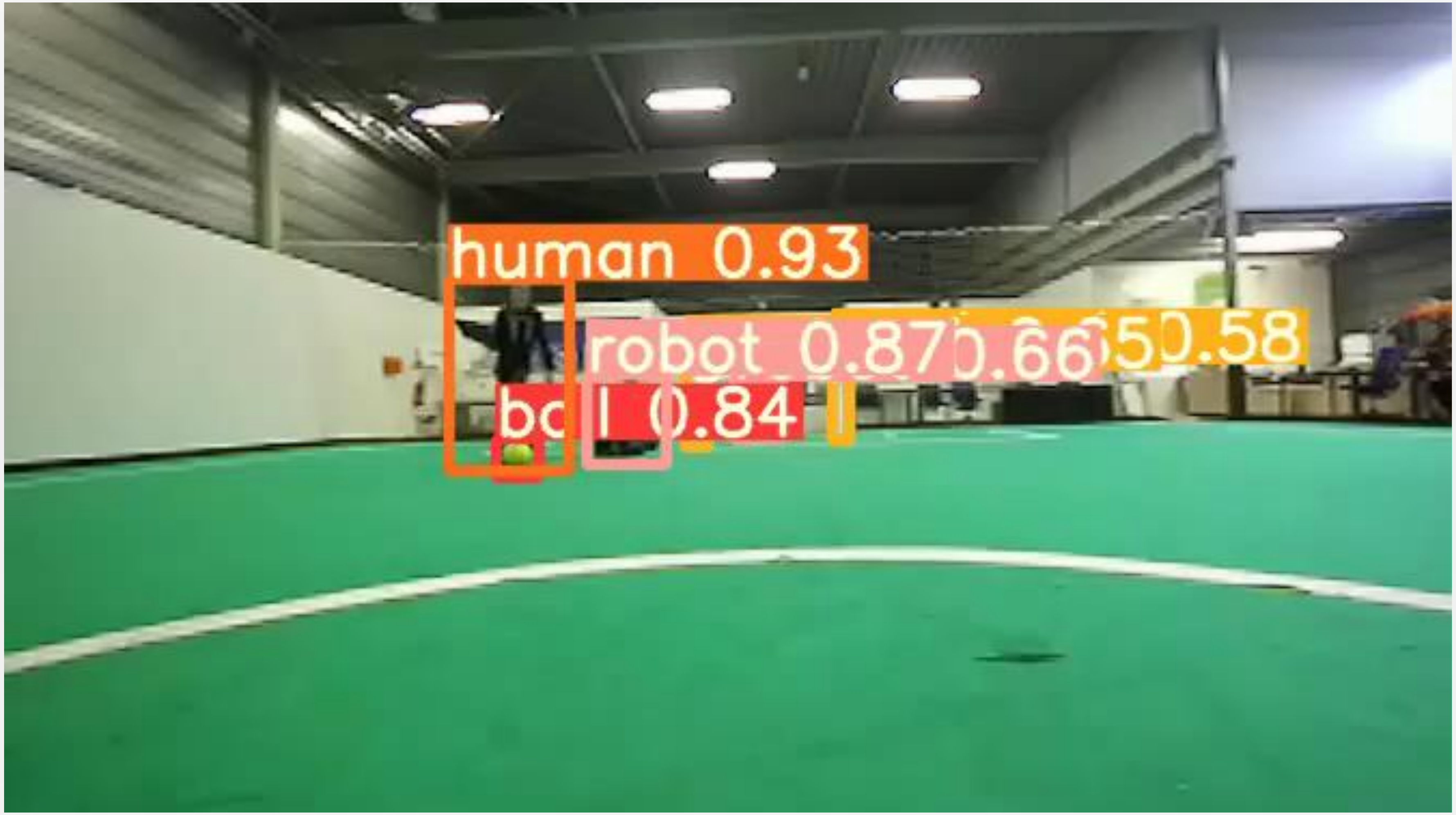
Low-cost fast mobile object detection for robotic soccer

- Why? Detect objects fast (and fast objects) on soccer field, based on local compute, affordable for teams



"By the middle of the 21st century, a team of fully autonomous humanoid robot soccer players shall win a soccer game, complying with the official rules of FIFA, against the winner of the most recent World Cup"

The results



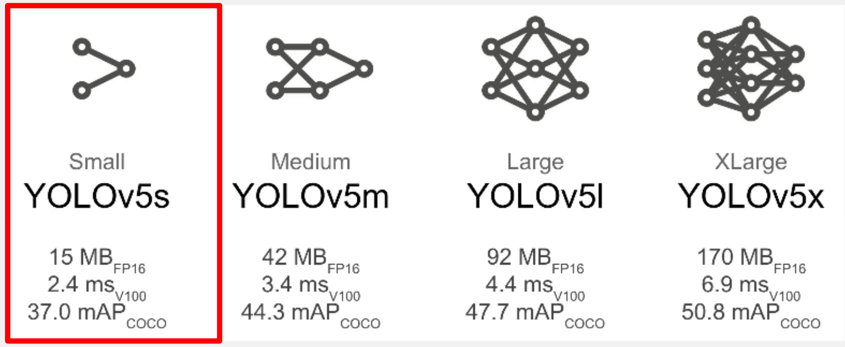
YOLO V5

Ultralytics' YOLOv5 "You Only Look Once" model family

- 5th generation of YOLO real-time object detection (June 2020)
- based on convolutional neural networks
- state-of-the-art, very fast, targeted at mobile devices
- easy to use, It's as easy as running a single pip install
- Pytorch framework

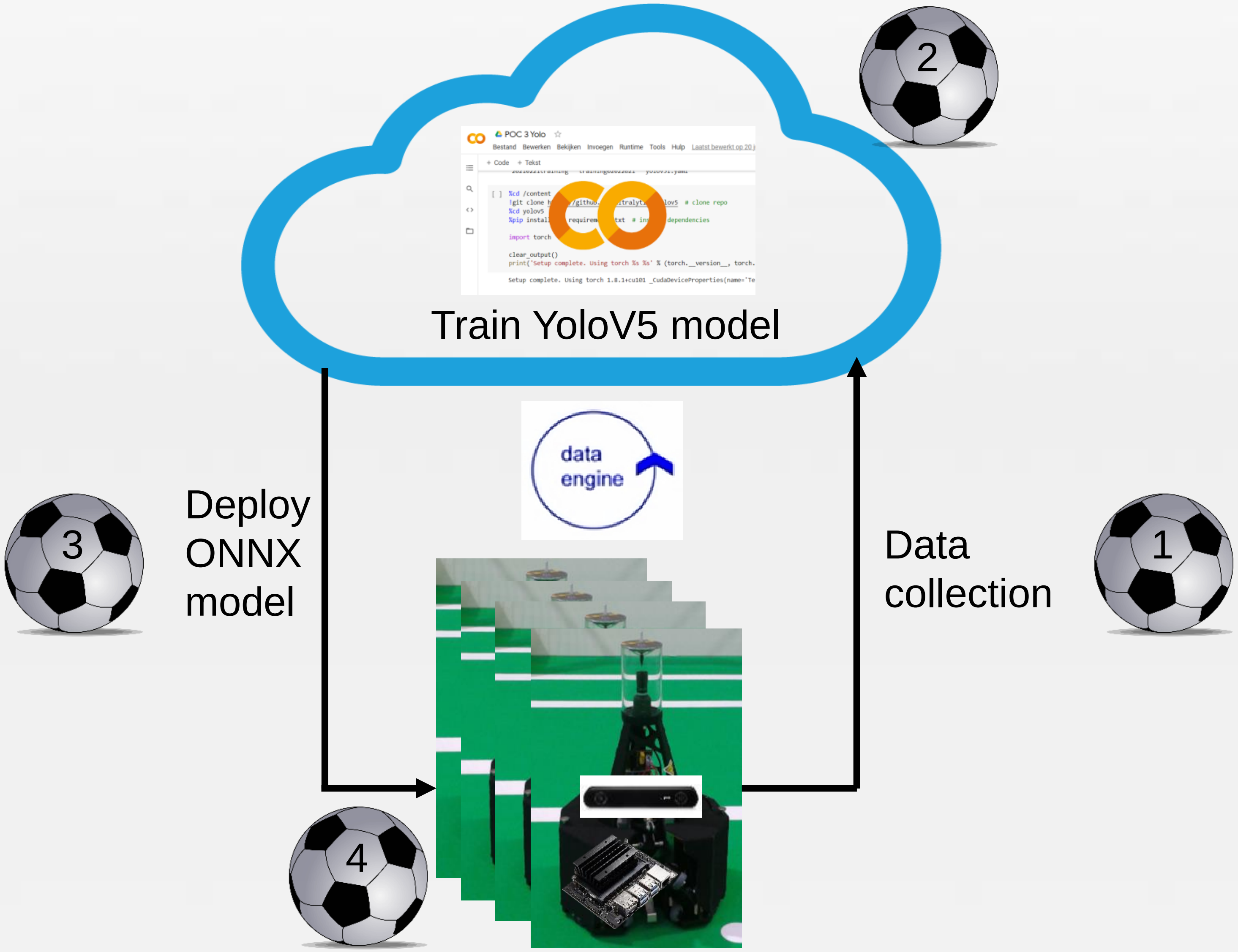
YoloV5-small tested in context of high FPS inference on mobile devices

- 15 MB footprint (90 percent smaller than YOLOV4 and 15x faster training)
- Fastest possible inference (2.4 ms on cloud GPU) without significant quality loss



Small	Medium	Large	XLarge
YOLOv5s	YOLOv5m	YOLOv5l	YOLOv5x
15 MB_{FP16} 2.4 ms_{V100} 37.0 mAP_{COCO}	42 MB_{FP16} 3.4 ms_{V100} 44.3 mAP_{COCO}	92 MB_{FP16} 4.4 ms_{V100} 47.7 mAP_{COCO}	170 MB_{FP16} 6.9 ms_{V100} 50.8 mAP_{COCO}

Continuous digital feedback loop



```
POC 3 Yolo
Bestand  Bewerken  Bekijken  Invoegen  Runtime  Tools  Hulp  Laatst bewerkt op 20 j
+ Code  + Tekst

[ ] %cd /content
    !git clone h     /github.     ltralyti     lov5  # clone repo
    %cd yolov5
    %pip install     requirem     txt  # ins     dependencies

    import torch

    clear_output()
    print('Setup complete. Using torch %s %s' % (torch.__version__, torch.

    Setup complete. Using torch 1.8.1+cu101 _CudaDeviceProperties(name='Te
```

Based on Tesla best practice

Optimized mobile Inference engine: Nvidia TensorRT

Data collection

Collected 1000 images from perspective of our robot camera

- Image size 672*376
- Image format JPG (batch converter PGN to JPG made available on github)
- Credits to ASML for making their data available for QuickStart

Annotated images on labelImg (free)

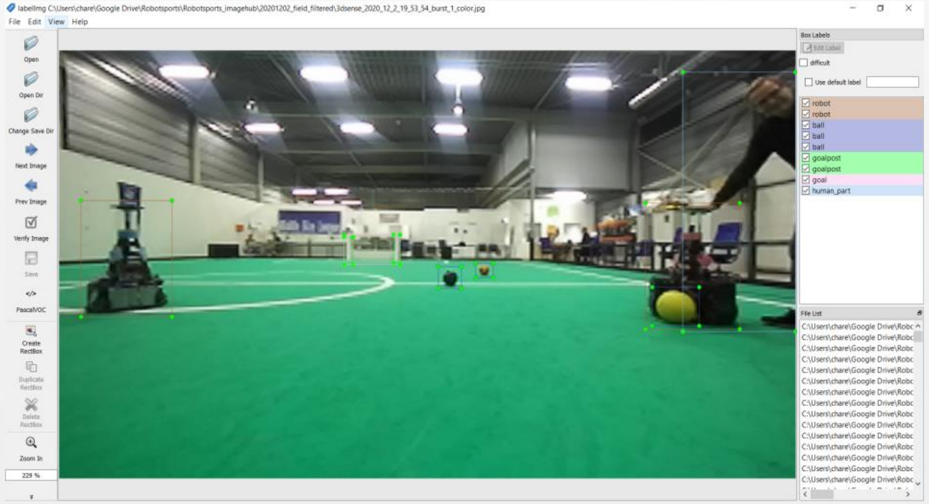
- Annotation VOC XML format (universal standard)
- Conversion SW to YOLO text file format (made available on github)

Labels:

- ball
- robot
- goalpost
- human

VDL Robotsports dataset made available on:

- https://github.com/Charelvanhoof/robocup_vdl
- https://www.kaggle.com/charel/robocup-images



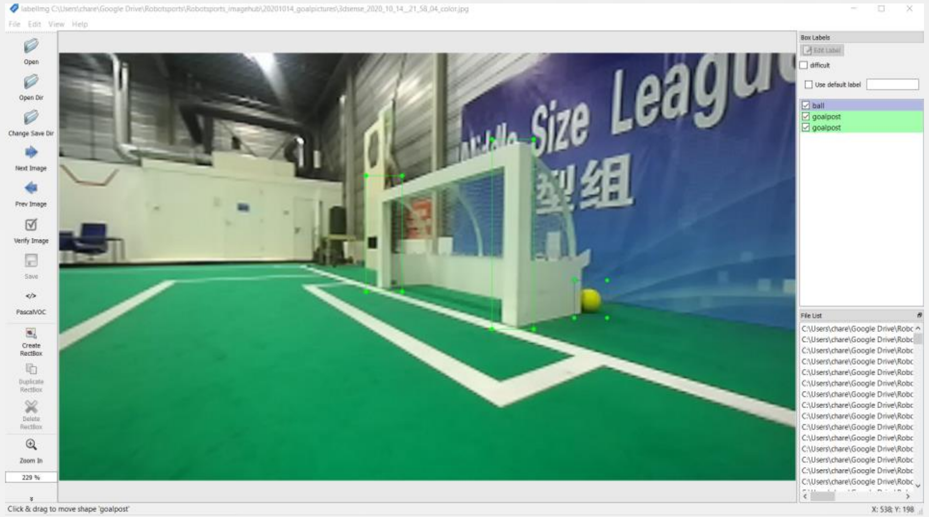
labelImg C:\Users\chare\Google Drive\Robotsports\Robotsports_imagehub\20201202_field_filtered\3dsense_2020_12_2_19_53_54_burst_1_color.jpg

Box Labels

☐ difficult

☐ Use default label

☑ robot
☑ robot
☑ ball
☑ ball
☑ ball
☑ goalpost
☑ goalpost
☑ goal
☑ human_part



labelImg C:\Users\chare\Google Drive\Robotsports\Robotsports_imagehub\20201014_goalpictures\3dsense_2020_10_14__21_58_04_color.jpg

Box Labels

☐ difficult

☐ Use default label

☑ ball
☑ goalpost
☑ goalpost

Click & drag to move shape 'goalpost'

X: 538; Y: 198

YoloV5 Training in the cloud

Training in the cloud

- Google Colab cloud GPU (Jupyter Notebook)
- Free of charge
- Many ML libraries pre-installed

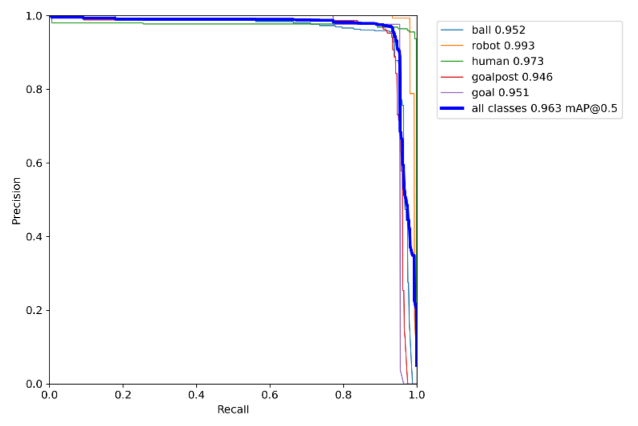
Neural network architecture

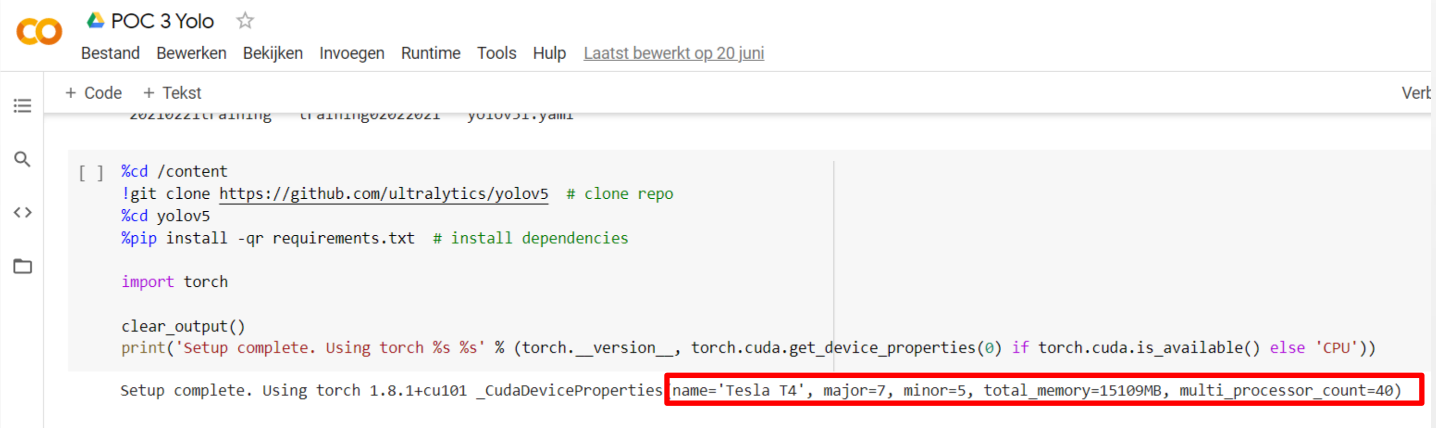
- Yolo (You Only Look Once) V5 small
- Model Summary: 283 layers, 7.074.330 parameters
- Transfer learning

Only 2 hours training

- 300 epochs

Amazing recall/precision



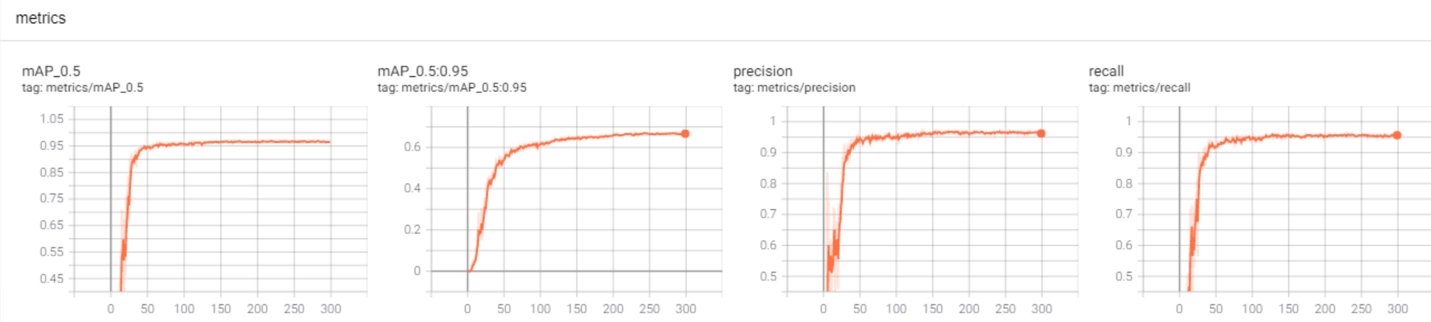


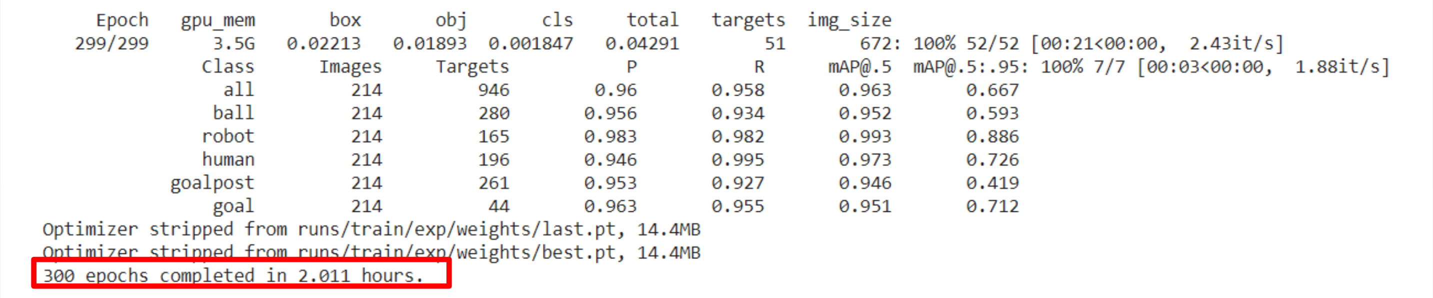
```
[ ] %cd /content
    !git clone https://github.com/ultralytics/yolov5  # clone repo
    %cd yolov5
    %pip install -qr requirements.txt  # install dependencies

    import torch

    clear_output()
    print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cuda.is_available() else 'CPU'))

    Setup complete. Using torch 1.8.1+cu101 _CudaDeviceProperties(name='Tesla T4', major=7, minor=5, total_memory=15109MB, multi_processor_count=40)
```

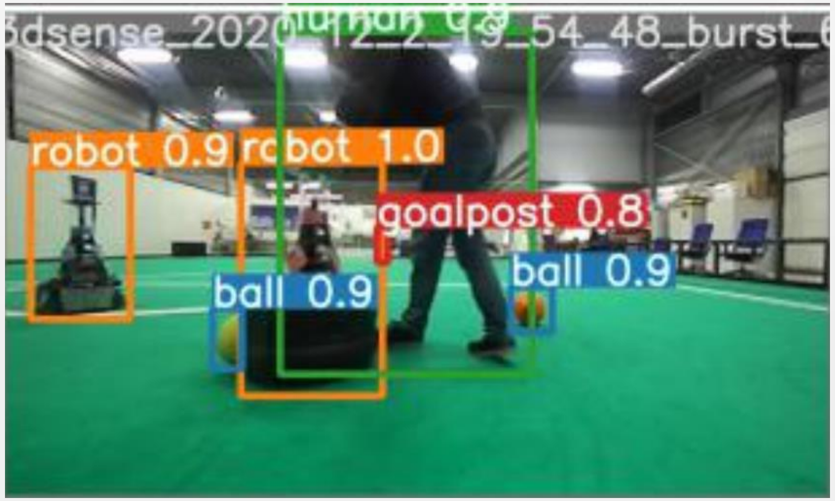




```
     Epoch   gpu_mem       box       obj       cls     total   targets  img_size
   299/299      3.5G   0.02213   0.01893  0.001847   0.04291        51       672: 100% 52/52 [00:21<00:00,  2.43it/s]
               Class    Images   Targets         P         R    mAP@.5  mAP@.5:.95: 100% 7/7 [00:03<00:00,  1.88it/s]
                 all       214       946      0.96     0.958     0.963     0.667
                ball       214       280     0.956     0.934     0.952     0.593
               robot       214       165     0.983     0.982     0.993     0.886
               human       214       196     0.946     0.995     0.973     0.726
            goalpost       214       261     0.953     0.927     0.946     0.419
                goal       214        44     0.963     0.955     0.951     0.712
Optimizer stripped from runs/train/exp/weights/last.pt, 14.4MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.4MB
300 epochs completed in 2.011 hours.
```

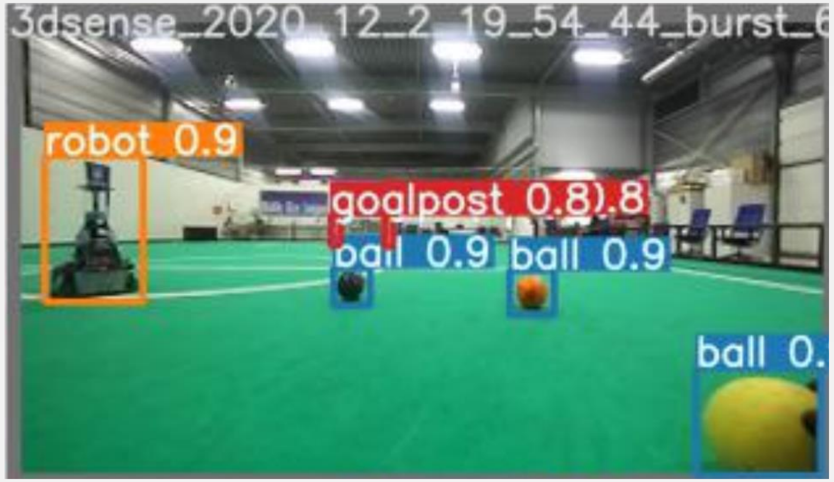
Some more results

Partially shielded ball



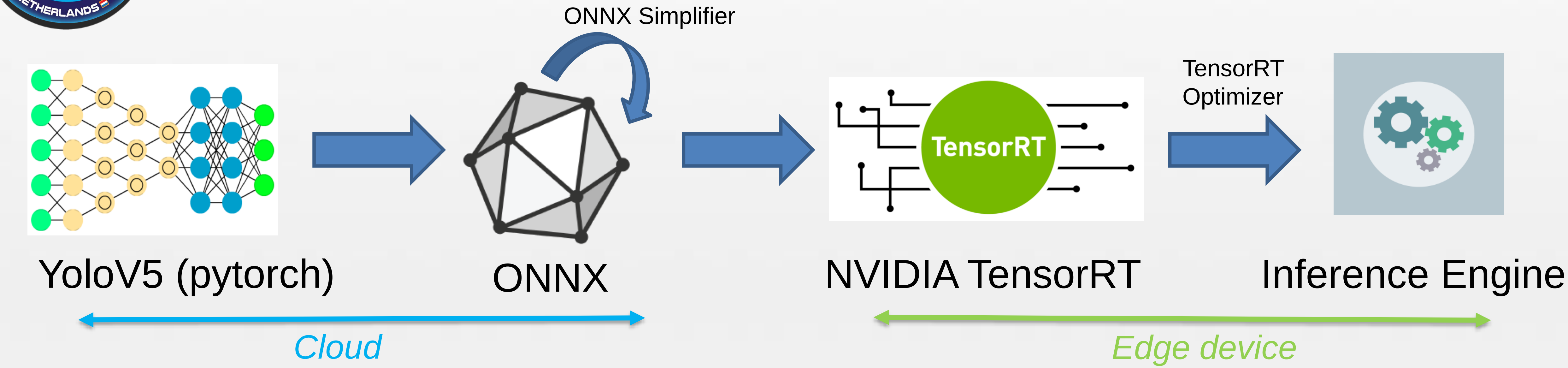
Follow the ball

Different ball colors



Different lighting conditions

Deploy model



ONNX Simplifier

TensorRT Optimizer

YoloV5 (pytorch) ONNX NVIDIA TensorRT Inference Engine

Cloud

Edge device

ONNX: Open Neural Network Exchange

- Open standard for representing machine learning networks
- Supported by all major machine learning frameworks (e.g. Tensorflow, Pytorch, MXnet)
- Provides interoperability between AI libraries

TensorRT: AI Inference Framework

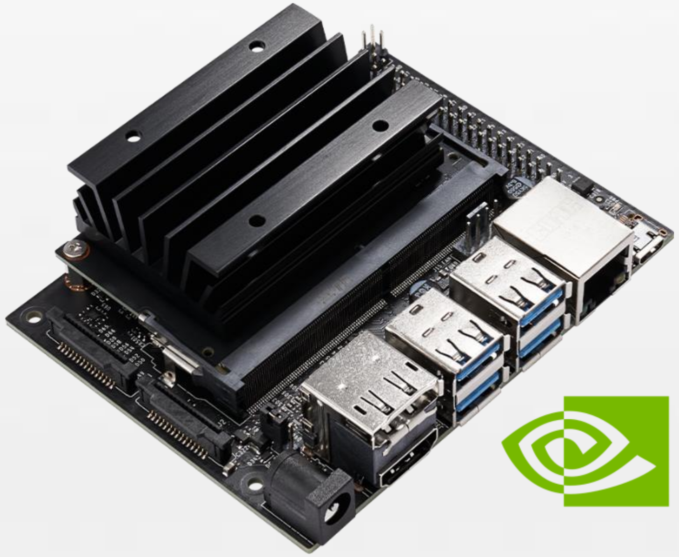
- Supported by most modern NVIDIA graphics platforms (e.g. graphics cards, Jetson)
- Takes a neural network, produces an inference engine optimized for the specific edge device
- Multiple precision levels supported (e.g. INT8, FP16, FP32)

Our code (2000 lines) will be made available on Github

Extending our robot platform with low-cost AI hardware capabilities

NVIDIA JETSON NANO

- Quad-core ARM processor, up to 4GB RAM
- Integrated GPU with 128 CUDA cores
- 5-10W power usage
- Developer kits starting at $69. VDL Robotsports used the $99 version. Third-party industrial kits available for a variety of applications



STEREOLABS ZED 2 (OPTIONAL)

- Used by VDL Robotsports for future features (depth estimation)
- Stereo depth estimation using two high-resolution cameras
- Based on NVIDIA CUDA technology
- Integrated IMU and compass
- $400

Conclusion

Low-cost fast mobile object detection for robotic soccer

Novelty

- State-of-the art A.I. @25 fps on low cost mobile HW (~ 50 ms latency)

Interest for the league

- Amazing real-time precision/recall of objects on the soccer field
- e.g. for the ball: shielded, different colors, different lighting conditions
- 2 hours training -> train on game location with new images during set-up day

Complexity

- YoloV5 training in Google cloud -> ONNX exchange standard -> tensortRT Nvidia GPU network optimization for low cost HW (countless hours, 2000 lines of code)

Relevancy for the league

- Acceleration possibility: amazing object detection for all teams
- Affordable for all
- Base on open standards: ONNX, VOC XML

Demonstrated experimental results

- practical results, from sceptical AI questions to usable object detection on soccer robots

Published results

- Check it out: https://github.com/Charelvanhoof/robocup_vdl , https://www.kaggle.com/charel/robocup-images, ONNX/TensorRT will be made available on github

Mounted on our new robot

VDL ROBOTSPORTS
NETHERLANDS