



**SAPIENZA**  
UNIVERSITÀ DI ROMA

# Progetto per il corso di Metodi Quantitativi Per L'Informatica: Confronto tra le reti Darknet Yolo e Faster-RCNN

**Facoltà di Ingegneria dell'Informazione, Informatica e Statistica**  
**Dipartimento di Ingegneria Informatica, Automatica e Gestionale "Antonio**  
**Ruberti"**  
**Corso di laurea in Ingegneria Informatica e Automatica**

**Marco Costa**  
**Matricola 1691388**

# PURPOSE

The project consists in a comparison between two different types of Neural Networks on image detection. Both the networks are pre-trained on Pascal VOC 2007 Train/Val set and runs on the same hardware. They can identify 20 different categories: (Aeroplane, Bicycle, Bird, Boat, Bottle, Bus, Car, Cat, Chair, Cow, Dinning table, Dog, Horse, Motorbike, People, Potted plant, Sheep, Sofa, Train, TV/Monitor).

They will be tested on the Pascal VOC 2007 Test set (first 1000 images) and Coco 2017 Test set (first 1000 images).

# HARDWARE SPECIFICATIONS

CPU: Intel Core i7-4720HQ (2.60-3.60 GHz, 6MB Cache)

GPU: Nvidia GeForce GTX 950M (4096MB DDR3)

RAM: 16 GB (DDR3L 1600 MHz SDRAM)

STORAGE: SSD 525 GB + HDD 1000 GB

OS: Ubuntu 16.04.04 with CUDA Toolkit 8.0 and cuDNN 7.0

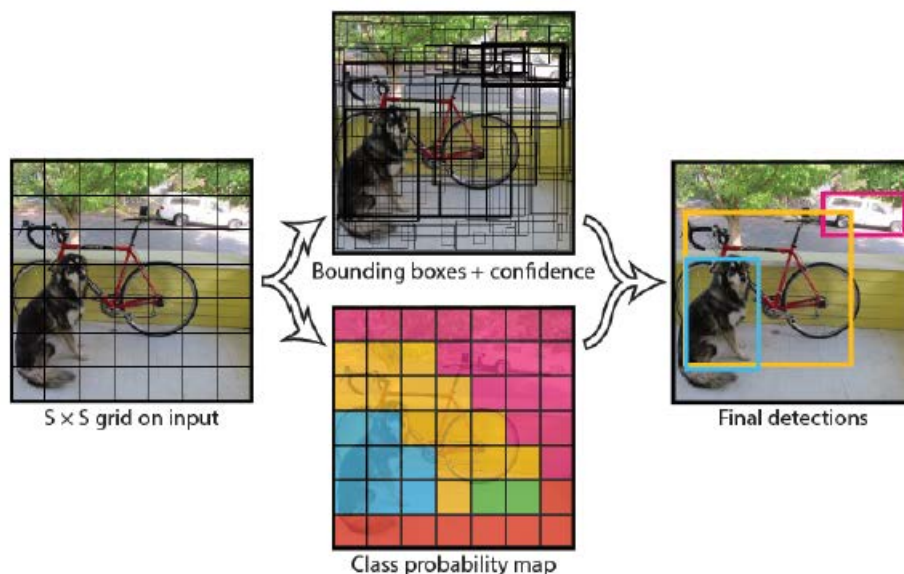
# FEATURES

## *Darknet YOLO*

YOLO (You Only Look Once) is a real-time object detection system that uses neural networks to detect objects in images. It is written in C.

### How it works:

It applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities.

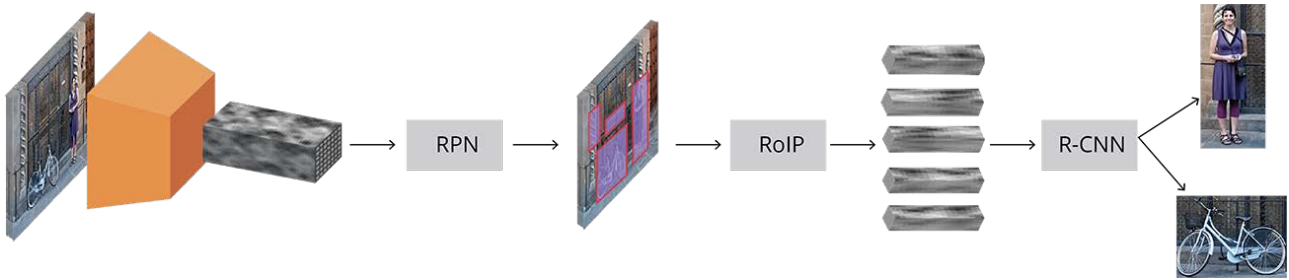


### Advantages:

- It looks at the whole image at test time so its predictions are informed by global context in the image;

- It also makes predictions with a single network evaluation unlike systems like R-CNN which require thousands for a single image;
- This makes it extremely fast, more than 1000x faster than R-CNN and 100x faster than Fast R-CNN.

YOLO detection network has 24 convolutional layers, followed by 2 fully connected layers. It alternates 1 x 1 convolutional layers to reduce the features space from the preceding layers. Initial convolutional layers extract features from the image and the fully-connected layers predict the output probabilities and coordinates of the object.



# EXPERIMENT

## Detection Threshold:

Since the output detections of the model are the probabilities of a detection being reliable, we can establish a threshold to discard predictions with low probability. This way we can adjust the level of reliability of the model and decide threshold to be more or less restrictive.

In this experiment, we use two different thresholds: 0.2 and 0.8.

## Let's start:

Running both the nets on the first 1000 images of Pascal VOC 2007 Test set we have the following situation:

Threshold \ Net	Darknet YOLO	Faster-RCNN
0.8	979 correct and 21 wrong	754 correct and 246 wrong
0.2	745 correct and 255 wrong	GPU Memory Overflow

Now we can calculate the precision:

$$\text{PRECISION} = \frac{\text{N}^\circ \text{ of correct predictions}}{\text{N}^\circ \text{ of correct predictions} + \text{N}^\circ \text{ of wrong predictions}}$$

$$\text{PRECISION}_{\text{YOLO, threshold}=0.8} = \frac{979}{979+21} = 0.979 = 97.9\%$$

$$\text{PRECISION}_{\text{Faster-RCNN, threshold}=0.8} = \frac{754}{754+246} = 0.754 = 75.4\%$$

$$\text{PRECISION}_{\text{YOLO, threshold}=0.2} = \frac{745}{745+255} = 0.745 = 74.5\%$$

$$\text{PRECISION}_{\text{Faster-RCNN, threshold}=0.2} = \frac{0}{1000} = 0.0 = 0\%$$

Analyzing the detection on the first image, using Threshold=0.8, we can see that the YOLO net detects a person with a 92% probability, while the Faster-RCNN detects a person with a 98.8% probability.

On the fourth image, however, we can see that the YOLO net only finds two cars (respectively with a probability of 85% and 88%), while the Faster-RCNN finds five cars (with a probability of 99.6%, 99.5%, car: 96.5%, car: 93.5%, car: 88.8%).

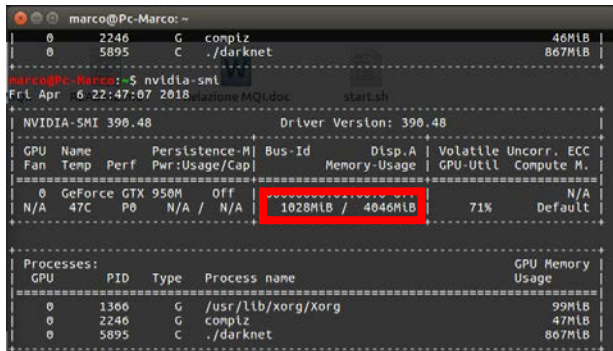
Running the nets on the first 1000 images of COCO 2017 Test set we have the same situation:

On the first image, using Threshold=0.8, we can see that the YOLO net detects a car with a 90% probability, while the Faster-RCNN doesn't detect any car.

On the second image, however, we can see that the YOLO net detects a person with a 94% probability, while the Faster-RCNN detects a person with a 98.9% probability.

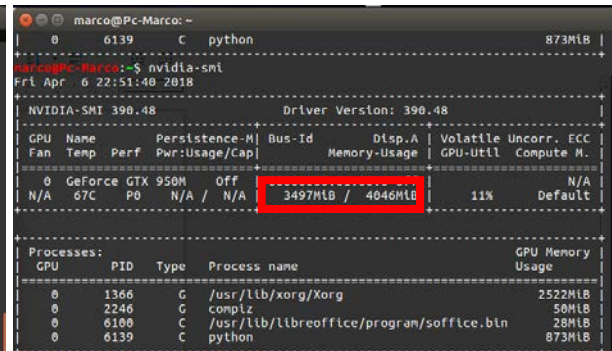
In this case, we can't check if detections are correct because there aren't any files to compare.

### Use of Hardware resources:



```
marco@Pc-Marco: ~  
0 2246 G complz 46MiB  
0 5895 C ./darknet 867MiB  
marco@Pc-Marco:~$ nvidia-smi  
Fri Apr 6 22:47:07 2018  
NVIDIA-SMI 390.48 Driver Version: 390.48  
GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC  
Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M.  
0 GeForce GTX 950M Off 1028MiB / 4046MiB 71% Default  
Processes:  
GPU PID Type Process name GPU Memory Usage  
0 1366 G /usr/lib/xorg/Xorg 99MiB  
0 2246 G complz 47MiB  
0 5895 C ./darknet 867MiB
```

YOLO use of GPU



```
marco@Pc-Marco: ~  
0 6139 C python 873MiB  
marco@Pc-Marco:~$ nvidia-smi  
Fri Apr 6 22:51:40 2018  
NVIDIA-SMI 390.48 Driver Version: 390.48  
GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC  
Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M.  
0 GeForce GTX 950M Off 3497MiB / 4046MiB 11% Default  
Processes:  
GPU PID Type Process name GPU Memory Usage  
0 1366 G /usr/lib/xorg/Xorg 2522MiB  
0 2246 G complz 50MiB  
0 6100 C /usr/lib/libreoffice/program/soffice.bin 28MiB  
0 6139 C python 873MiB
```

Faster-RCNN use of GPU

## CONCLUSIONS

Analyzing the result of this experiment, we can state that:

- The Darknet YOLO net is more precise than the Faster-RCNN;
- The first network requires less hardware resources than the second one (1028MB vs 3497MB of GPU Memory with Threshold=0.8, Faster-RCNN causes GPU Memory overflow with Threshold=0.2)
- The Faster-RCNN has a better accuracy than the Darknet YOLO net.

## REFERENCES

- **Darknet Yolo** - YOLO: Real-Time Object Detection (<https://pjreddie.com/darknet/yolo>)
- **Faster-RCNN** - ZF model (<https://github.com/rbgirshick/py-faster-rcnn>)