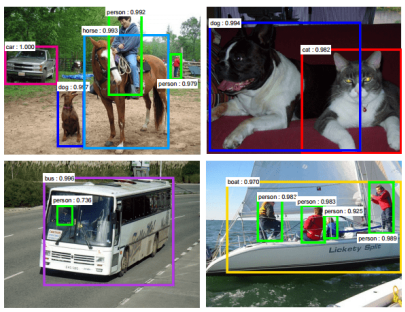
OBJECT DETECTION

Deep Learning + Computer Vision

Definition

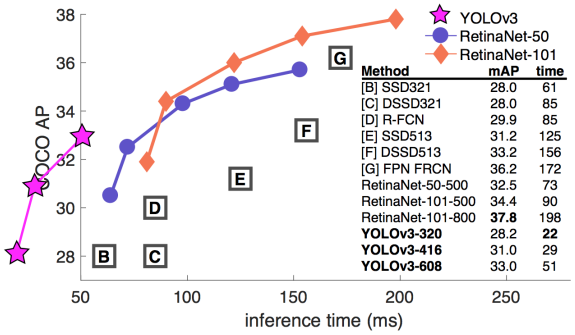
- **Object detection** consists in determining the location of objects in an image, as well as classifying those objects.
  - **Possible Applications:**
    - Security and Surveillaince
      - face detection, license plate detection
    - Tracking of objets in videos
      - people and vehicles
    - Automated vehicle systems
    - Machine inspections
    - Image retrieval systems
    - Robotics



source: https://sigmoidal.io/dl-computer-vision-beyond-classification/

https://buzztowns.com/wp-content/uploads/2019/09/bounding-box1.gif

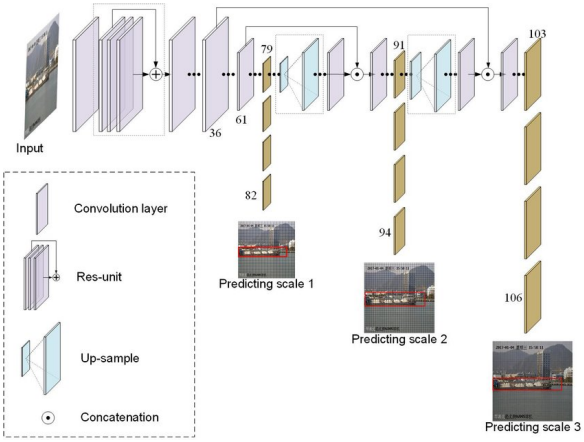
Deep models (https://pjreddie.com/media/files/papers/YOLOv3.pdf)



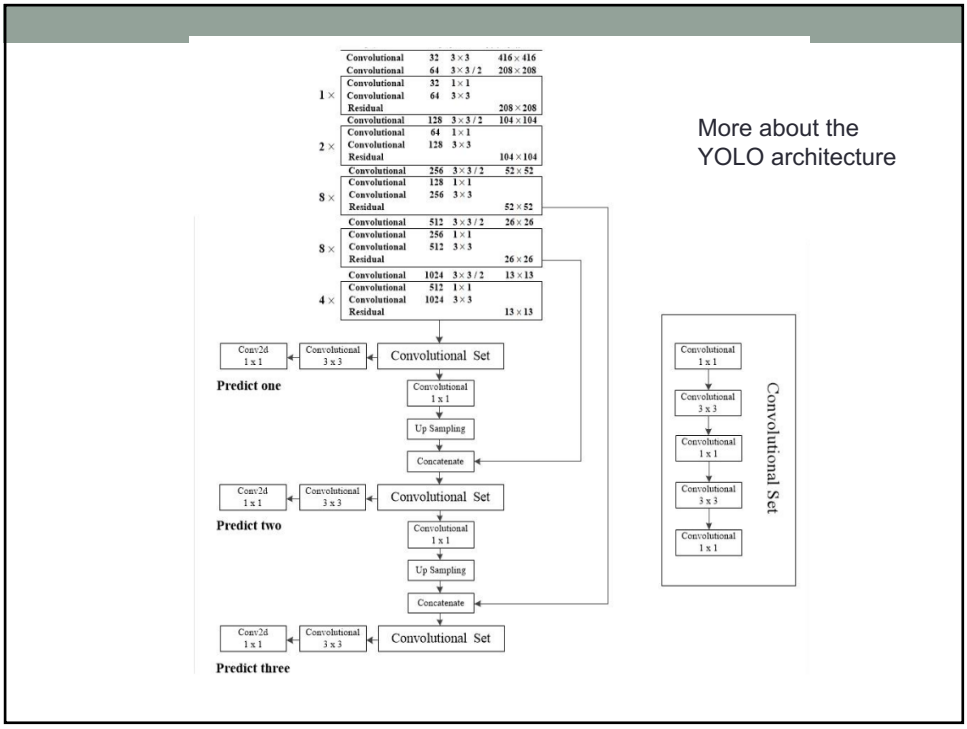
Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

SSD - Single Shot Detector
DSSD - Deconvolutional Single Shot Detector
R-FCN – Region-based Fully Convolutional Neural Network
FPN FRCN – Feature Pyramid Network (Fast Region-based Convolutional Network)
RetinaNet
YOLOv3 – You Only Look Once

Yolo v3 Architecture (adapted from https://www.researchgate.net/profile/Wen_Liu121)



- 106 layers
- Object detection at 3 different scales



Yolo - loss function

Loss Function

Localization

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Bounding box detection

Confidence

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

Confidence that an object is in the box

Classification

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Object class prediction

Yolo v3 prediction (adapted from reference https://arxiv.org/pdf/1904.04620.pdf)

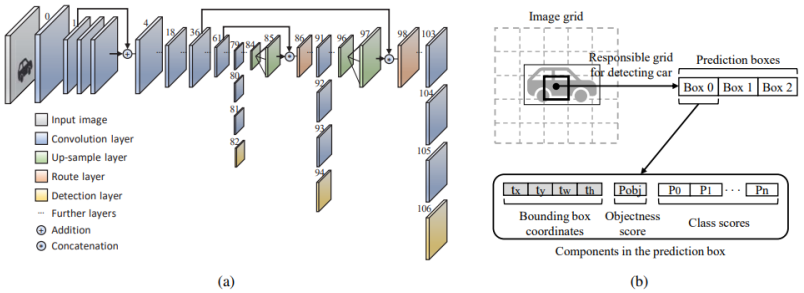
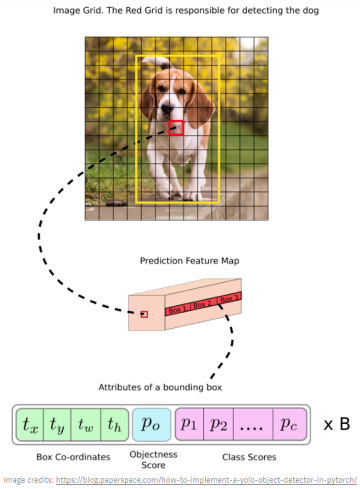


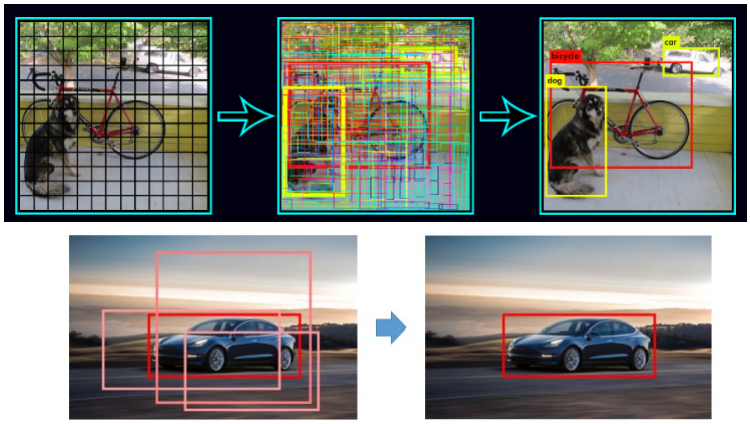
Figure 1: (a) Network architecture of YOLOv3 and (b) attributes of its prediction feature map.

Yolo output



- The shape of the detection kernel is **1 x 1 x (B x (5 + C)).**
- Here B is the number of bounding boxes a cell on the feature map can predict, "5" is for the 4 bounding box attributes and one object confidence, and C is the number of classes.
- In YOLO v3 trained on COCO, B = 3 and C = 80, so the kernel size is 1 x 1 x 255.

Non Maximum Suppression (Filtering the Yolo Output)



source: https://pjreddie.com/darknet/yolov2/

Yolo v3 Layers

- 5 types of layers
  - convolutional layers
    - Parameters:
      - batch_normalize=1
      - filters=64
      - size=3
      - stride=1
      - pad=1
      - activation=leaky
  - shorcut layers: output of this layer is obtained by **adding** feature maps from the previous and the 3rd layer backwards from the *shortcut* layer.
    - Parameters
      - from=-3
      - activation=linear

Yolo v3 Layers

- Upsample layers: upsamples the feature map in the previous layer by a factor of stride using bilinear upsampling.
  - Parameters:
    - stride=2
- Route layers: the layer will output feature map from the n^{th} layer backwards from the *Route* layer.
  - Parameters
    - layers = -4
    - layers = -1, 55 (in this case, it concatenates previous layer with the layer 55)

Yolo v3 Layers

- Yolo layer: layer for detection of objects
  - Parameters:
    - mask = 0,1,2
    - anchors = 10,13, 16,30, 33,23, 30,61, 62,45, 59,119, 116,90, 156,198, 373,326
    - classes=80
    - num=9
    - jitter=.3
    - ignore_thresh = .5
    - truth_thresh = 1
    - random=1

Yolo v3 Layers

- Net
  - Parameters
    # Testing
  - batch=1
  - subdivisions=1
    # Training
  - batch=64
  - subdivisions=16
  - width= 416
  - height = 416
  - channels=3
  - momentum=0.9
  - decay=0.0005
  - angle=0
  - saturation = 1.5
  - exposure = 1.5
  - hue=.1

Exemplo: Training a tinny YOLO for a specific problem Tinny YOLO has only 23 layers

- Problem: detection of guns
- Step 1:
  - Preparing the datasets for training and testing:
    - Format: *<object-class> <x_center> <y_center> <width> <height>*
    - *Tools:* https://github.com/puzzledqs/BBox-Label-Tool

Exemplo

- *0 0.534560 0.543860 0.598743 0.631579*

Exemplo: Training a tinny YOLO for a specific problem

- Step 2: Download darknet, since YOLOv3 was create on Darknet
  - download the framework “darknet.zip” (available in the Blackboard)
- Step 3: Change the makefile
  - ***GPU*** = 1 ,***CUDNN=1*** and ***OPENCV=1***
- *Step 4:* Download darknet pre-trained model (weights)
  - https://pjreddie.com/media/files/darknet53.conv.74
- Step 5: Modify configuration file (*yolov3.cfg) save as (yolov3_tinny.cfg)*
  - *Number of classes de 80 para 1*
  - *Number of filters of the last layer before [yolo layers]: de 255 para 18*
    - *filters = (classes + 5)*3*
- Step 6: Split train/validation datasets
  - Done: see files train.txt and test.txt

Exemplo: Training a tinny YOLO for a specific problem

- Step 7: Modify the file “yolo.names”
  - This file must have the class names, in our case only one name:
    - Gun
- Step 8: Modify the file “yolo.data”

  classes= 1

  train = data/train.txt
  valid = data/val.txt
  names = data/yolo.names
  backup = backup
- Step 9: Verify the files
  - **images** folder in directory **darknet\data** contains 1000 images
  - **labels** folder in directory **darknet\data** contains 1000 .txt file
  - **train.txt, val.txt, yolo.data, yolo.names** in directory **darknet\data**
  - **yolov3_tiny.cfg** in directory **darknet\cfg**

Exemplo: Training a tinny YOLO for a specific problem

- Step 10: download the Nvidia Cuda compiler driver
  - !nvcc --version
- Step 11: compile the model
  - %cd '/content/drive/My Drive/darknet'
  - !make
  - !chmod +x '/content/drive/My Drive/darknet'
- Step 12: Install the tool to convert files from Dos to Unix
  - !sudo apt install dos2unix
- Step 3: Convert the following files from Dos to Unix
  - !dos2unix ./data/train.txt
  - !dos2unix ./data/val.txt
  - !dos2unix ./data/yolo.data
  - !dos2unix ./data/yolo.names
  - !dos2unix ./cfg/yolov3_custom_train.cfg
  - !dos2unix ./cfg/yolov3_tiny.cfg
  - !dos2unix ./cfg/yolov3_tinyy.cfg

Exemplo: Training a tinny YOLO for a specific problem

- Step 14: Train the model
  - %cd '/content/drive/My Drive/darknet'
  - !chmod +x darknet
  - !./darknet detector train data/yolo.data cfg/yolov3_tinyy.cfg darknet53.conv.7
- Step 15: Testing the model
  - Uses the Python script (Google Colab)