



STAATLICH  
ANERKANNTE  
HOCHSCHULE

# Data Analytics 4 - Project

## Yolo - algorithm with use case

Presented on 10.06.2020 by:

Rohit Keshav Bewoor (11011831)

Big Data and Business Analytics 2018-20 batch  
SRH Hochschule Heidelberg

# Content

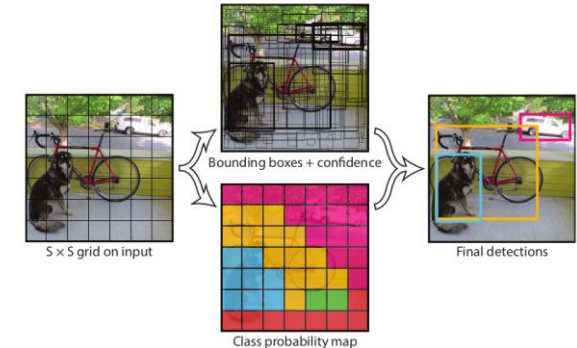
- Introduction
- How YOLO works
- Types of YOLO
- Use case
- Model Description and Visualisation
- References
- Demo
- Q&A

# Introduction

- You Only Look Once (YOLO) is an object detection algorithm
- Four versions - v1 in May 2016, v2 in December 2016, v3 in April 2018, v4 in April 2020
- Does not use region proposal based approach like the R-CNN family
- Developed using the Darknet framework
- Github Repo for this project work : [https://github.com/rbewoor/DataAnalytics4\\_Project](https://github.com/rbewoor/DataAnalytics4_Project)

# How YOLO works

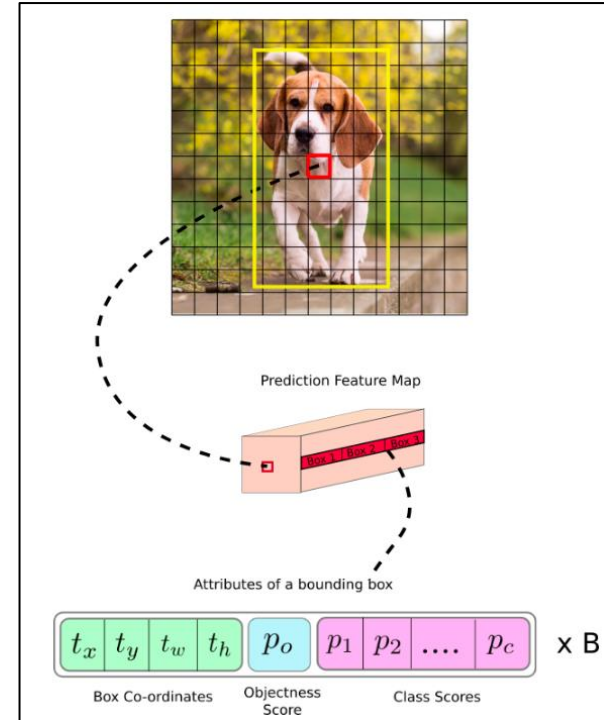
- Divides image into a grid of size  $S \times S$  where  $S$  is an integer
- Each pixel evaluated as possible center point of an object
- All detections are evaluated in one pass - very fast algorithm
- Model is trained to identify  $C$  classes of objects
- $B$  is the number of Bounding Boxes detected all over the image (without threshold consideration). Five values are output for each bounding box:
  - Two values for center coordinates
  - Two values for dimensions (height and width)
  - Confidence score
- Can handle multiple bounding boxes and aspect ratios (anchor box concept)
  - Anchor boxes are predefined boxes provided by the user to Darknet which gives the network an idea about the relative position and dimensions of the objects to be detected.
  - These are calculated using the training set Objects.



Source: YOLO v1 paper: <https://arxiv.org/abs/1506.02640>

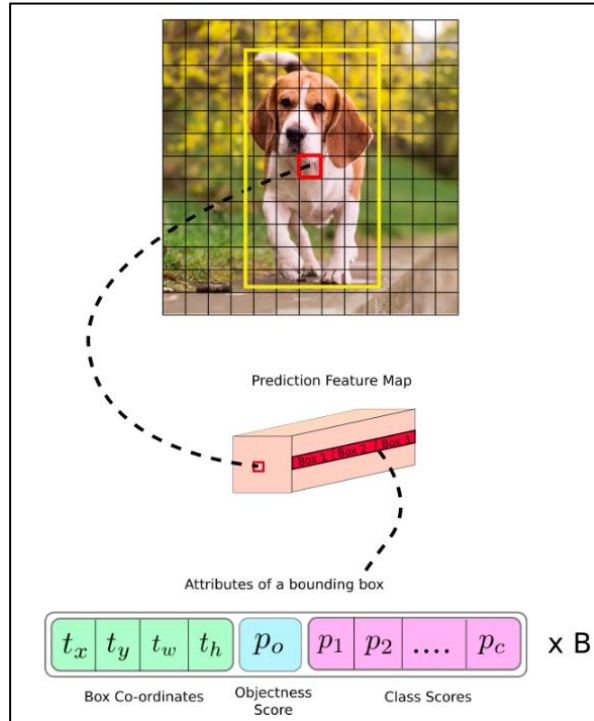
# How YOLO works

- Usually Non-max suppression used to remove redundant detections
- Total detections per image =  $(S \times S) * (B * (5 + C))$ 
  - Each bounding box has 5 + C attributes
- For example, suppose that:
  - image is divided into 13 x 13 grid (i.e.  $S = 13$ )
  - we want to detect 80 classes for COCO (i.e.  $C = 80$ )
  - 3 boxes predicted ( $B = 3$ )
  - #Detections =  $(13 \times 13) * (3 * (5 + 80)) = 13 \times 13 \times 255 = 43,095$
- Threshold value used for Confidence Score to evaluate acceptance of object detection



Source: <https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193>

# How YOLO works



- Image divided into a grid of  $S \times S$  grid-boxes
- Each grid can predict maximum  $B$  objects if it thinks center lies in the pixels covered by the grid
- Box information is regressed to give  $t_x, t_y, t_w, t_h$
- Formulae applied later to get actual bounding box center coordinates and dimensions:  $b_x, b_y, b_w, b_h$
- YOLO v3 makes predictions at 3 scales where a  $416 \times 416$  pixel input images divided into  $13 \times 13$ ,  $26 \times 26$  and  $52 \times 52$

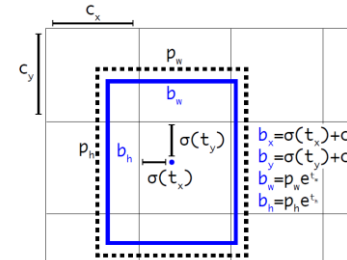


Figure 2. Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function. This figure blatantly self-plagiarized from [15].

# Types of Yolo

Model Type	YOLO v1	YOLO v2 (aka YOLO9000)	YOLOv3
Salient points	26 total (24 Conv + 2 FC)  Problem detecting small objects	30 layers (included batch norm after every Conv)  Anchor boxes introduced  No FC present  Still poor with small objects	106 layers  Detection on 3 scales to handle small to large object sizes  9 anchor boxes (3 per scale)

FC: Fully connected layer

Conv: Convolution layer

YOLOv4 is very recent and not been studied in depth

## Objective:

Use the YOLO model for desired use case and explain the architecture

## Use Case:

Present a set of new images to a pre-trained YOLO v3 model.

For each image, capture the detected **object class** and the **confidence score**.

Store information in a neo4j graph database:

- Relationship format: (i:Image)-[r:HAS]->(o:Object)
- Confidence score is a property of the “HAS” relationship
- Python script: my\_yolo3\_one\_file\_to\_detect\_them\_all\_6.py

E.g. Image123.jpg HAS the objects:

- car (score 58.98),
- person (score 98.34)
- person (score 93.23)



# Neo4j representation

Neo4j db after inserts

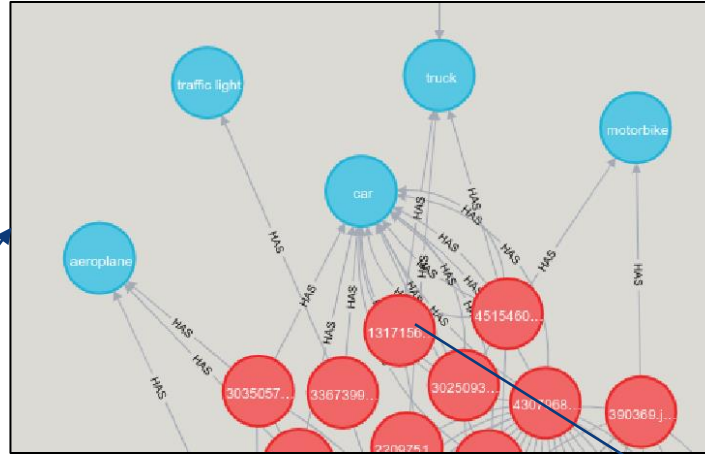
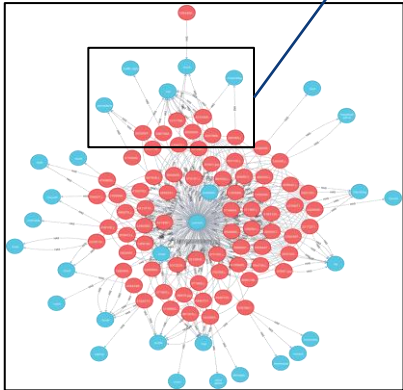


Image - HAS -> Object

Objects found in images: Traffic light, truck, motorbike, car, etc.

Many to many relationship could exist.



Detection output:  
1317156\_det.jpg

# Model description and visualisation

- Darknet-53 block diagram taken from the YOLO v3 paper:
  - maps to Layers 0 - 74 of the model built using python script `my_yolo3_one_file_to_detect_them_all_6.py`
- Model description using Keras built in functions:
  - Python script: `my_yolo3_model_stats_1.py`
  - Textual description with `model.summary` function  
Link: [https://github.com/rbewoor/DataAnalytics4\\_Project/blob/master/model\\_summary\\_1.txt](https://github.com/rbewoor/DataAnalytics4_Project/blob/master/model_summary_1.txt)
  - Visualisation with `Plot_model` function  
Link: [https://github.com/rbewoor/DataAnalytics4\\_Project/blob/master/model\\_vis\\_1.png](https://github.com/rbewoor/DataAnalytics4_Project/blob/master/model_vis_1.png)

	Type	Filters	Size	Output
1x	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
2x	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
4x	Residual			8 × 8

Source: Yolo v3 paper:  
<https://arxiv.org/abs/1804.02767>

# Model description and visualisation

- Convolution parameters
  - C: Channels
  - F: Filter/ Kernel Size
  - S: Stride
- Standard flow is convolution, then Batch Normalisation, then Leaky ReLU, repeats

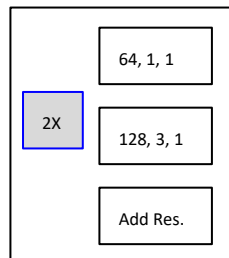
C, F, S



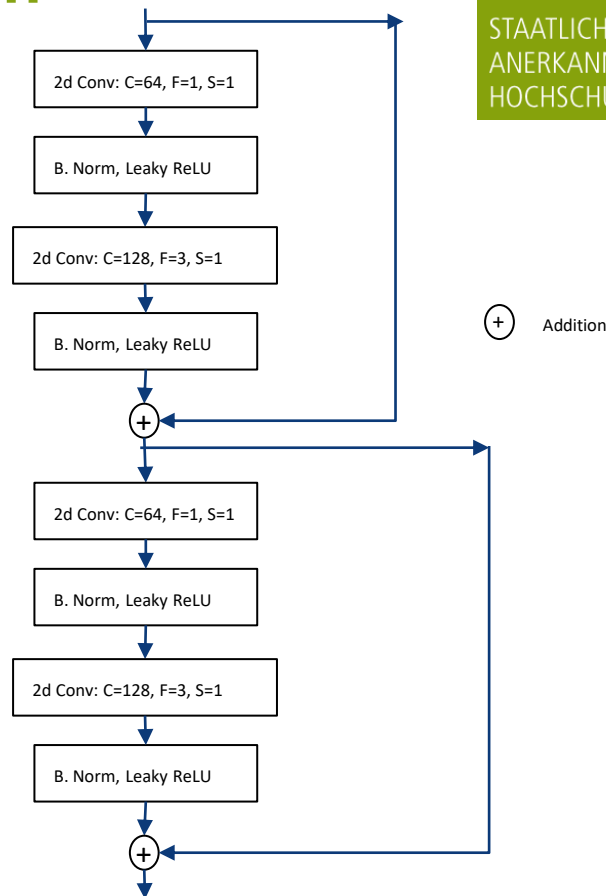
2d Conv: C filters of size F with stride S.

B. Norm

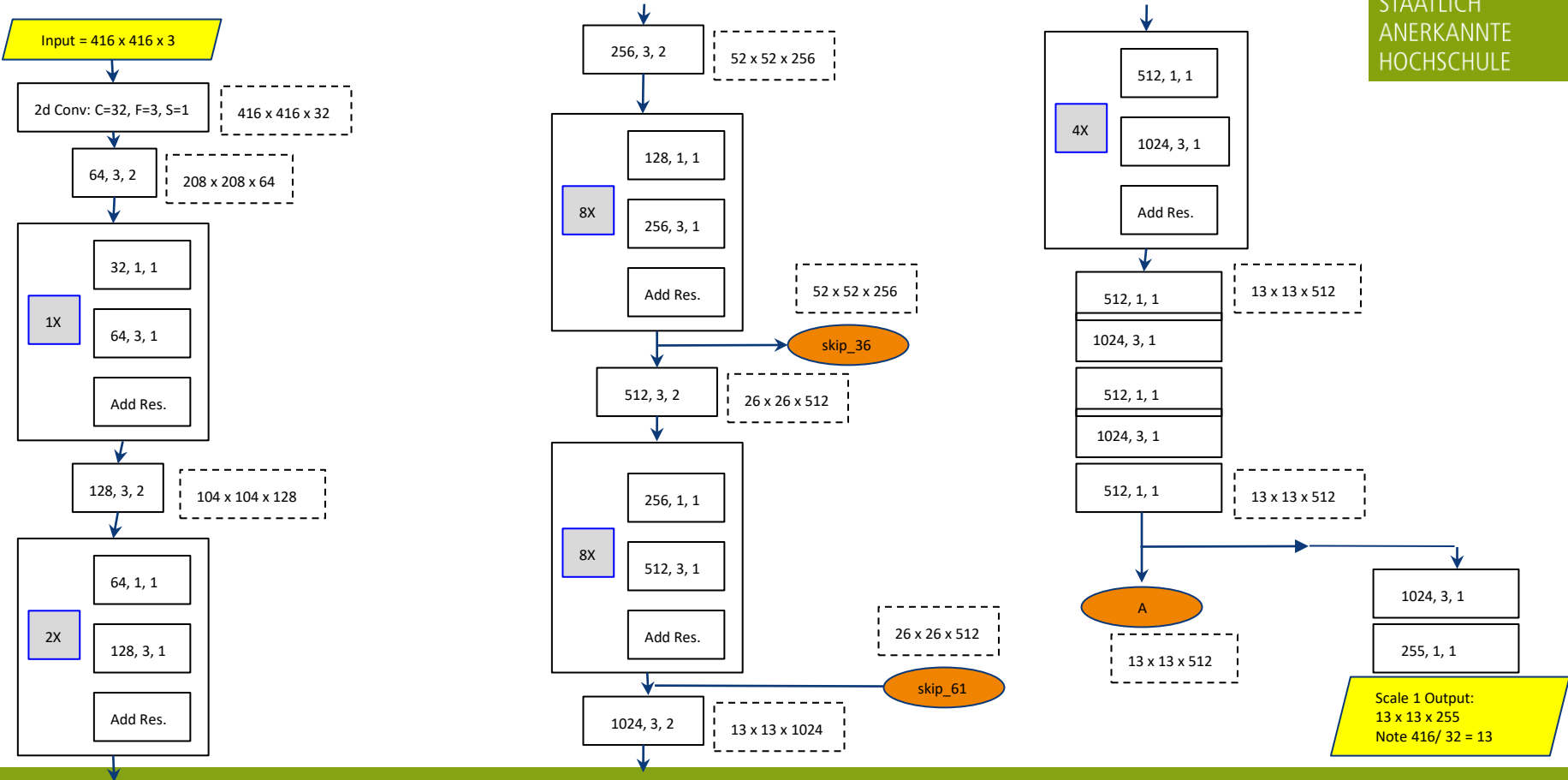
Leaky ReLU



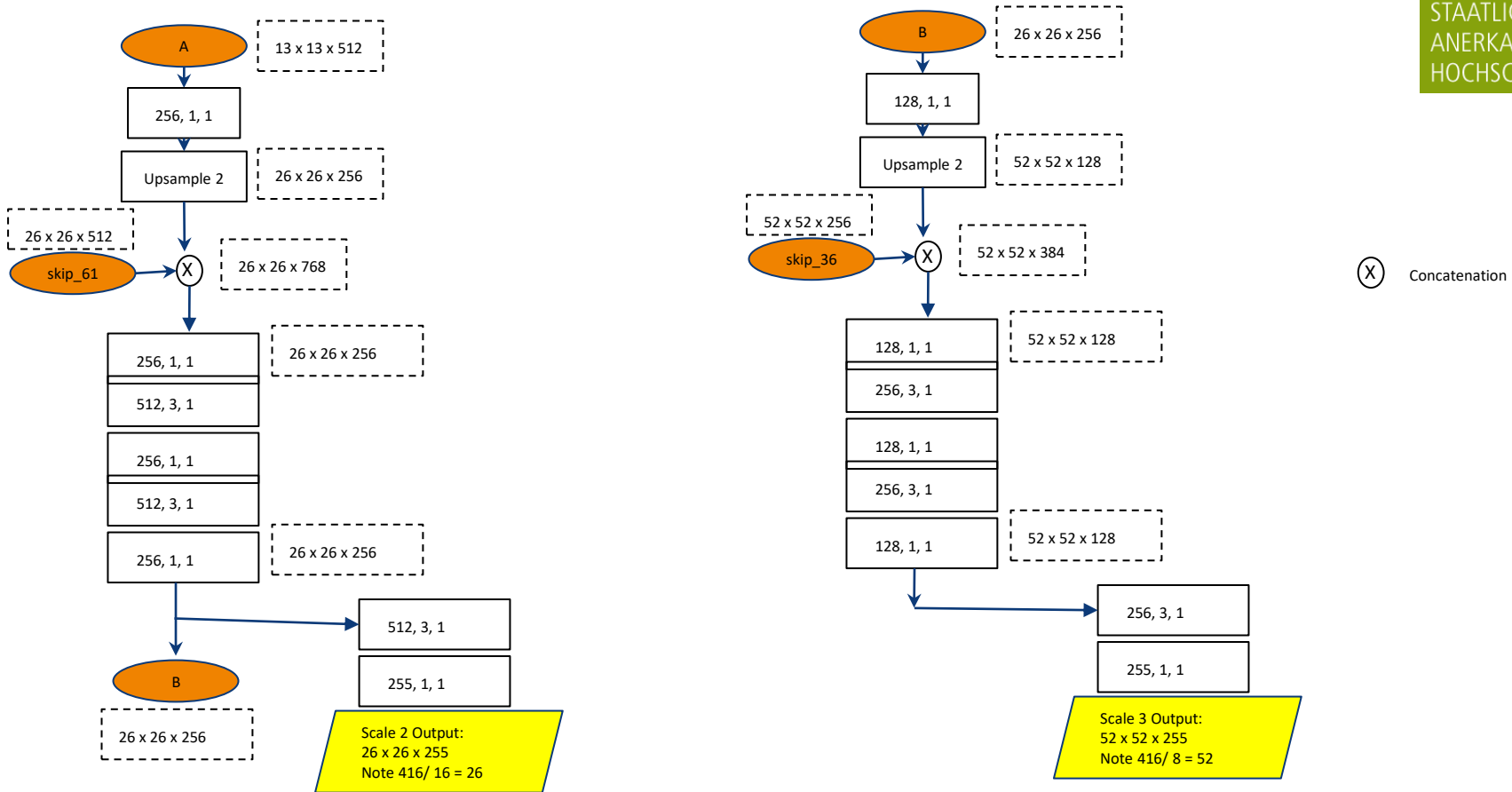
- When stride = 1, always same convolution.
- Only for stride = 2 (downsampling stages), valid convolution with single padding



# Model description and visualisation



# Model description and visualisation



# References

1. How to Perform Object Detection With YOLOv3 in Keras. <https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/>
2. Github Code <https://github.com/rbewoor/keras-yolo3> ( forked from <https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/> on 05.06.2020)
3. YOLOv3 pre-trained weights. <https://pjreddie.com/media/files/yolov3.weights>
4. J. Redmon et al. You Only Look Once: Unified, Real-Time Object Detection. 09-05-2016. <https://arxiv.org/abs/1506.02640>
5. J. Redmon et al. YOLO9000: Better, Faster, Stronger. 25-12-2016. <https://arxiv.org/abs/1612.08242>
6. J. Redmon et al. YOLOv3: An Incremental Improvement. 08-04-2018. <https://arxiv.org/abs/1804.02767>
7. How to Visualize a Deep Learning Neural Network Model in Keras. <https://machinelearningmastery.com/visualize-deep-learning-neural-network-model-keras/>
8. All About YOLO Object Detection and its 3 versions (Paper Summary and Codes). <https://medium.com/data-science-in-your-pocket/all-about-yolo-object-detection-and-its-3-versions-paper-summary-and-codes-2742d24f56e>
9. YOLO v3 theory explained, <https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193> as on 10.06.2020

# Demo and Q&A

- Thank you.