Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

A survey on objection detection: Using YOLO Version 3

Kaushik Das[a]*, Arun Baruah[b]

[a]Department of CSE, Dibrugarh University, Dibrugarh-786004,India
[b]Department of Mathematics, Dibrugarh University,Dibruarh-786004,India

Abstract

This paper focuses on study of approaches for efficient object detection techniques in the field of computer vision. The improvement in the accuracy had been studied based on the deep learning techniques. It aims to develop an online tool for a better study about the object detection techniques as a platform to review the object detection in both online and offline images. A popular state of art object detection algorithm called YOLOv3 which is trained on COCO and ImageNet. The result obtained in the developed online tool has shown a positive result with a possibility in detecting the most common objects available in this real world.

Peer-review under responsibility of the scientific committee of the Third International Conference on Computing and Network Communications (CoCoNet'19)

Keywords: Artificial Neural Networks; Convolutional Neural Networks; Residual Network; You Only Look Once.

1. Introduction

People are effectively ready to distinguish what sort of articles are there before us, where they are, and how they cooperate. It does not take one moment to distinguish the kind of article present before us. So with the assistance of Convolutional neural systems, if calculations could be exact and quick enough for picture handling, the PCs would probably drive autos without particular sensors, and assistive gadgets would most likely pass on on-going scene data to clients. In like manner, it would be genuine Artificial Intelligent (AI) if these calculations could finish Deep Learning (DL) errands with high productive and fantastic execution like individuals do. In this manner, acknowledgment: arrangement, confinement and item discovery are a sequential of the center undertakings of picture handling and the key difficulties are: exactness, speed, cost and intricacy.

Later methodologies like R-CNN use district proposition strategies to initially produce potential bounding boxes in a picture and afterward run a classifier on these proposed boxes. After grouping, present handling is utilized on refine the anchor box, take out copy discoveries, and rescore the case dependent on different articles in the scene [13]. These mind boggling pipelines are moderate and difficult to streamline in light of the fact that every individual segment must be prepared independently. We reframe object identification as a solitary relapse issue, straight from picture pixels to anchor box facilitates and class probabilities. Utilizing our framework, You Just Look Once (YOLO) at a picture to foresee what articles are available and where

* Corresponding author. Tel.: +91-9864675792; fax: 0373-2370774.
E-mail address: kd.duiet@dibru.ac.in

Peer-review under responsibility of the scientific committee of the Third International Conference on Computing and Network Communications (CoCoNet'19)

they are. YOLO is refreshingly straightforward: as in Figure 1. YOLO prepares on full pictures and straightforwardly improves location execution. This brought together model has a few advantages over conventional strategies for item location. In the first place, YOLO is amazingly quick. Since we outline recognition as a relapse issue we needn't bother with a mind boggling pipeline. We just run our neural system on another picture at test one time to anticipate recognitions. Quick R-CNN, a top identification technique [14], botches foundation fixes in a picture for articles since it can't see the bigger setting. YOLO makes not exactly a large portion of the quantity of foundation mistakes contrasted with Fast R-CNN. YOLO learns generalizable portrayals of articles. At the point when prepared on regular pictures and tried on fine art, YOLO beats top discovery strategies like DPM and R-CNN by a wide edge. Since YOLO is exceptionally generalizable, it is less inclined to separate when connected to new spaces.

1.1. Artificial Neural Networks

ANNs are made out of different hubs, which mirror natural neurons of human mind. The neurons are associated by connections and they communicate with one another. The hubs can take input information and perform straightforward tasks on the information. The consequence of these activities is passed to different neurons. The yield at every hub is called its initiation or hub esteem. Each connection is related with weight. ANNs are equipped for realizing, which happens by changing weight esteems.

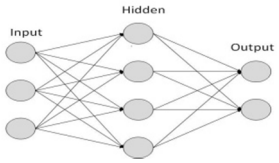

Figure 1: Source: www.tutorialspoint.com/ANN

1.2. Covolutional Neural Networks

Convolutional Neural Networksare fundamentally the same as customary Neural Networks. They are comprised of neurons that have learnable loads and biases [13]. Every neuron gets a few sources of info, plays out a spot item and alternatively tails it with a non-linearity. The entire system still communicates a solitary differentiable score work: from the crude picture pixels toward one side to class scores at the other. Despite everything they have a misfortune work (for example SVM/Softmax) on the last (completely associated) layer and every one of the tips/traps we created for learning standard Neural Networks still apply[12].

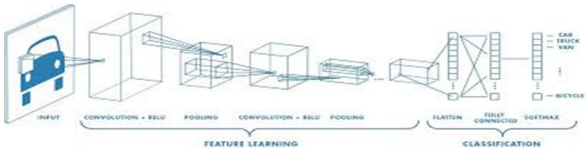

Figure_2-Source:“A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way", 2019

1.3. Residual Network

ResNet or a Residual Network [11] is a neural system engineering which takes care of the issue of disappearing gradients[12] as basic as could be expected under the circumstances. On the off chance that there is inconvenience sending the angle signal in reverse, we furnish the system with an alternate way at each layer to get things going all the more easily.

In a customary system the enactment at a layer is characterized as follows [14]:

$$y = f(x)$$

where f(x) is our convolution, grid duplication, or group standardization, and so forth. At the point when the sign is sent in reverse, the inclination constantly should go through f(x), which can raise ruckus because of the nonlinearities which are included. Rather, at each layer the ResNet executes:

$$y = f(x) + x$$

The "+ x" toward the end is the alternate way. It enables the inclination to pass in reverse straightforwardly. By stacking these layers, the slope could hypothetically "skip" over all the middle of the road layers and achieve the base without being decreased.

While this is the instinct, the genuine usage is somewhat more unpredictable. In the most recent manifestation of ResNets, f(x) + x takes the structure:

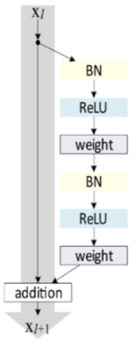

Figure 3: Source:https://chatbotslife.com/resnets-highwaynets-and-densenets-oh-my- 9bb15918ee32

ResNet Unit architecture. BN stands for Batch Normalization. Weight can refer to fully-connected or Convolutional layer.
With Tensorflow we can execute a system made out of these Residual units. To at long last work on the article dtection, we have to experience a few works of others that will be useful for our work. Before really jump into ResNet how about we experience how a basic CNN (Convolutional Neural Network) works. If there should arise an occurrence of an ordinary CNN we pass our info x through conv-ReLu-conv arrangement and we get our yield F(x)[1].

Be that as it may, if there should be an occurrence of a ResNet how about we perceive how it functions:

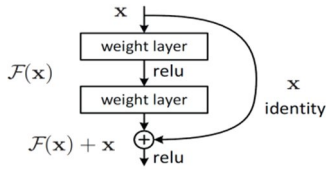

Figure 4: Residual learning: a building block.

From the above diagram we can clearly understand that our input x go through conv-relu-conv layer first but, in addition to that we add our input x to the final output, let's call it h(x) = f(x) + x and that is called skip connection and by doing so ResNet was able to optimize the error rate.
Since, from the hypothetical piece of CNN we came to realize that the more profound the neural system goes precision of the CNN increments. However, as a general rule it isn't valid. In the time of 2015 Microsoft research group has acquainted another thought with make an a lot further system called ResNet[1] or lingering learning and, in [1] they imagined with the assistance of a chart that on the off chance that we increment the quantity of layers in a neural system the preparation mistake just as test blunder increments.

1.4. You Only Look Once (YOLO)

You Only Look Once (YOLO) [2] - is the most progressive framework for item recognition continuously. In Pascal Titan X it forms the pictures at 30 outlines for each second and has a guide at 57.9% COCO test-dev.

YOLO [2] - a model for the discovery of items and is equipped for working continuously. It has 3 renditions are accessible up until this point.

YOLO [2] utilizes a solitary CNN arrange for order, and for the area of an article by methods for cradle squares.

YOLO [3] predicts progressively prohibitive squares on the lattice cell.

Yield of every framework cell:

The yield tensor of every framework cell contains '5 + class number' values in it where '5 = 1 certainty score esteem + 2 esteem for the middle co-ordinate + 2 esteem for tallness and width of the bounding box.

How about we consider a circumstance where we have '3 * 3' framework cell and 3 class along these lines yield y would be of shape '3 * 3 * 8'.
Likewise, YOLO v3 uses anchor boxes for foreseeing more than one article allotted to a similar network cell. On the off chance that we consider 2 grapple boxes for our above characterized model. At that point our yield y would be of shape '3 * 3 * 2 * 8'. Here worth 2 speaks to the quantity of anchor boxes. Since for YOLO v3 it utilizes 9 grapple boxes in this manner the worth would be 9.

General Formula:

$$(N * N) * [num\_anchors * (5 + num\_anchors)]$$

where, N = brace cell shape.

Non-Max Suppression:

Non-Max suppression is a calculation utilized for tidying up when different lattice cell predicts a similar article.

I. Discard all anchor boxes with certainty score $\leqslant$ 0.6 (for example $\leqslant$ 60%).
II. Pick the crate with biggest certainty score yield as a forecast.
III. Discard any residual box with IoU$\geqslant$ 0.5(i.e. $\geqslant$50%).

Since YOLO and YOLO v2 had some issue to identify little items and thus YOLOv3[4] utilizes three screening layer initial one having '13 * 13' lattice cell giving us a yield highlight guide of size '13 * 13 *255' and again one screening layer having '26 * 26' framework cell gives us a component guide of size '26 * 26 * 255' and in conclusion a screening layer having '52 * 52' gives us an element guide of size '52 * 52 * 255' and it's the last yield of the network(the 106th layer).

YOLO can just recognize objects that have a place with the classes that are available in the informational index utilized for system training [5]. These loads were gotten from the set up COCO system learning information, and in this manner we can distinguish 80 classes of items.

YOLO [6] results are promising. The mind boggling informational index to recognize VOC Pascal, YOLO conceivable to accomplish a mean normal exactness, or mapping, of 63.4 (100) during the utilization of 45 outlines for every second. For examination, the further developed model R-CNN VGG 16 quicker achieves MAP 73.2, however just works with up to 7 outlines for every second, which is multiple times diminishes productivity.

Detection Framework	mAP	FPS
Faster RCNN - VGG16	73.2	7
Faster RCNN - ResNet	76.4	5
YOLO	63.4	45
SSD 500	76.8	19
YOLO v2 (416x416 image size)	76.8	67
YOLO v2 (480x480 image size)	77.8	59

Figure 5: Source: https://blog.statsbot.co/real-time-object-detection-yolo-cd348527b9b7

In this way, the fundamental selling point for YOLO is its guarantee of good execution in article recognition at ongoing rates. That permits its utilization in frameworks, for example, robots, self-driving vehicles, and automatons, where being time basic are absolutely critical.

1.5. Yolo V3

At 320 × 320 YOLO v3[8] keeps running in 22 ms at 28.2 mAP, as precise as SSD yet multiple times quicker. When we take a gander at the old .5 IOU mAP discovery metric YOLOv3[8] is very great. It accomplishes 57.9 AP50 in 51 ms on a Titan X, contrasted with 57.5 AP50 in 198 ms by RetinaNet, comparative execution however 3.8× quicker.

Correlation with Other Detectors:

YOLO v3 is very quick and exact. In mAP estimated at .5 IOU YOLOv3 is keeping pace with Focal Loss yet about 4x quicker. In addition, you can without much of a stretch trade-off among speed and exactness just by changing the size of the model, no retraining required!

The following is the most noteworthy and least FPS. However, the outcome beneath can be very one-sided specifically they are estimated at various mAP.

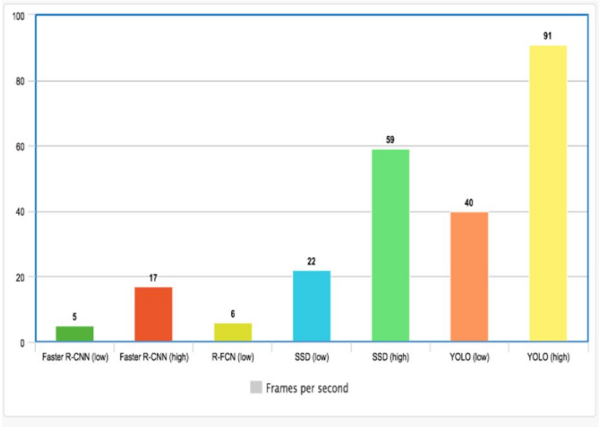

Figure 6: Source: https://pjreddie.com/darknet/yolo[7].

1.6. Prediction of each grid cell

We should consider a circumstance where we are attempting to foresee three classes utilizing a 3 * 3 framework cell at that point, for every matrix cells YOLO predicts the accompanying.

Y=[Pc bx by bhbw c1 c2 c3]

where, Pc = Confidence Score or Confidence esteem; bx =x co-ordinate of the inside point; by = y co-ordinate of the inside point; bh= stature of the anchor box; bw= width of the anchor box; c1 = class 1; c2 = class 2; c3 = class 3. Pc is one when an a network cell contaions an item and it is zero when a lattice cell doesn't contain an article. At the point when confidense score is zero we couldn't care less about the remainder of the qualities. bx, by, bh, bw are the bounding boxes of the item. These worth gives the co-ordinates of the bounding box of the article. bx and by are the inside co-ordinates and bh and bw are the stature and width of the bounding box. Lastly we have the quantity of classes. c1, c2, c3 ..... cn .

1.7. Necessity of anchor-boxes

In the past area, it is said that each cell organize in charge of the forecast of the item. However, imagine a scenario where the cell system is in charge of estimating a few articles. Next, the idea of grapple box becomes an integral factor. On the off chance that there is just one sort of prohibitive field, YOLO won't most likely anticipate more than one article related with that specific cell of the system. When all is said in done, what really YOLO predicts the article related with a particular cell of the system, which predicts the points of confinement related with said cell arrange. Along these lines, the grapple box - it is just a prohibitive boxes all things considered and different parts of the race. What's more, they are commonly good with an article. Presently, in the event that we consider the past circumstance, where a phone system was utilized and 3 * 3 Class 3, and on the off chance that we consider n tying down boxes in a cell of the lattice, at that point y is an impression of the size of 3 * 3 * 8 * n.

Subsequently, the general equation for computing elements of the zise card:

(N * N) * [number_of_anchors * (5 + number_of_classes)]

where, n * n - the size of the lattices. Furthermore, 5 here in light of the fact that each securing field 5 explicit qualities that have, 4 point of confinement arranges certainty score + 1 = 5

Before we talk about what's happening in YOLOv3 we should see the engineering underneath:

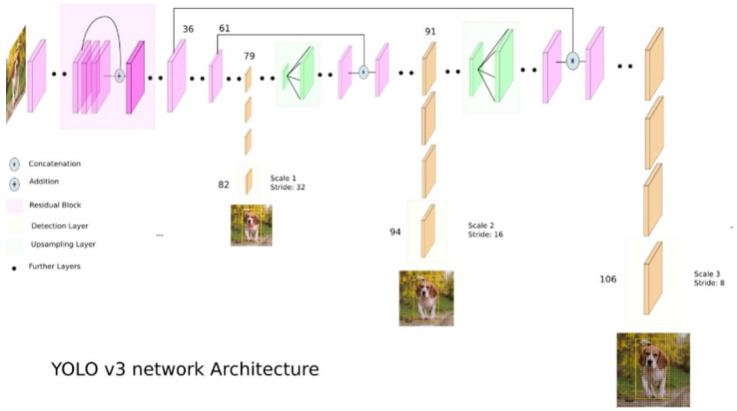

Figure 7: Source: https://pjreddie.com/darknet/yolo.

1.8. Prediction of anchor boxes in YOLOv2 and later

YOLO v3 utilizes 9 anchor boxes per lattice cell and they utilizes K-mean clustering to produce 9 grapple boxes per network cell.
In the prior subjects we have just clarified that what YOLO predicts as a yield y; there we came to realize that bx, by, bw, bh are the x and y co-ordinates and with and high estimations of the inside point and anchor box individually. However, how these qualities are get anticipated.
It may make sense to foresee the width and the tallness of the bounding box, however by and by, that prompts insecure slopes during preparing. Rather the greater part of the cutting edge objects indicator including Faster R-CNN uses balances.

What are counterbalances? Balance is only what amount we should move the anticipated bounding box so as to get the ideal anchor box.
At that point, these changes are connected to the anchor boxes to get the forecast. YOLOv3 has three anchor boxes, in this way it predicts three anchor boxes per cell.
YOLO v2 and YOLO v3 utilize these following recipes to compute the estimations of co-ordinate of the bounding boxes.

$$bx=\sigma(tx)+cx;\ by=\sigma(ty)+cy;\ bw=pwetw;\ bh=pheth$$

Here tx, ty, tw, th are what the system yields c and c are the upper left co-ordinates of the lattice. pw and ph are stays measurements for the container. What's more, we have passed our inside co-ordinates expectation through a sigmoid capacity in this way we get yield somewhere in the range of 0 and 1.

Following diagram is the visualization of what we have explained:

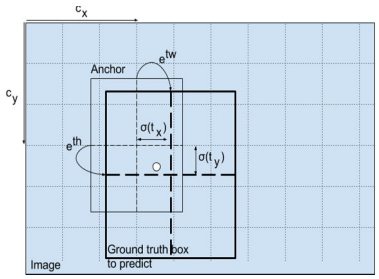

Figure 8: Source: https://pjreddie.com/darknet/yolo.

The resultant forecasts, bw and bh are standardized by the stature and the width of the picture. In this way, if the expectations bw and bh for the case containing a pooch are (0.3 , 0.8) at that point the genuine width and stature on a 13 * 13 highlight guide is (13 * 0.3, 13 * 0.8).

1.9. The total number of bounding box that YOLOv3 can predict

As we as a whole realize that YOLOv3 predicts at 3 unique scales; they are 13 * 13, 26 * 26 and 52 * 52. Hence all out number of bounding box that YOLOv3 can foresee is:
(13 * 13 * 3) + (26 * 26 * 3) + (52 * 52 * 3) = 10,647 bounding boxes; that implies all out 10,647 forecasts.

1.10. Class confidence in YOLO v3

To figure class certainty score softmax is utilized in YOLO and YOLOv2 yet in YOLOv3 they rather utilized typical sigmoid capacity.

1.11. YOLO loss function

The loss function $\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2$ that YOLOv2 use resembles the accompanying one:

λcoord ∑ i=0S2 ∑ j=0B1objij[(xi−x^i)2+(yi−y^i)2]
+λcoord∑i=0S2∑j=0B1objij[(wi−−√−w^i−−√)2+(hi−−√−h^i−−√)2]
+∑i=0S2∑j=0B1objij(Ci−C^i)2
+λnoobj∑i=0S2∑j=0B1noobjij(Ci−C^i)2
+∑i=0S21obji∑c∈classes(pi(c)−p^i(c))2

How about we get this:
Order misfortune: If an article is recognized, the grouping misfortune at every cell is the squared blunder of the class restrictive probabilities for each class is:
Confinement misfortune: The restriction misfortune estimates the blunder in the anticipated limit box areas and sizes. We just tally the case in charge of distinguishing the article.
Restriction misfortune is spoken to by the accompanying piece of the capacity:
To put more emphasis on the bounding box accuracy, loss is multiplied by the λ_{coord}, and by default the value of λ_{coord} is 5.

Confidence loss: If an object is detected in the box, the confidence loss is:

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

1_{ij}^{obj}= 1 if the j^{th}boundary box in cell i is responsible for detecting the object, otherwise 0.

If no object is detected in the box, the confidence loss is:

where 1_{ij}^{noobj} is the complement of 1_{ij}^{obj}.
and $\lambda_{noobj} = 0.5$

2. Implementation

Our technique is partitioned into two sections:

Section 1: Model advancement
Section 2: API advancement

Presently, Step-1: Model advancement: The first YOLO v3 was composed utilizing DarkNet structure. In any case, here we have sent this calculation in Tensor Flow with Python.

2.1. Required dependencies

1. TensorFlow (GPU adaptation liked) 2. NumPy 3. Pad/PIL 4. IPython

2.2. Batch Norm and Fixed Padding

We will characterize a capacity for group standard. Since we realize that model uses clump standards that are same as Resents. What's more, YOLO v3 utilizes convolution with fixed cushioning.

2.3. Features Extraction: Darknet-53

For highlight extraction YOLOv3 utilizes Darknet-53, a neural system pre-prepared on ImageNet. Darknet-53 has lingering associations. We will expel the last three layers of Darknet-53 since we just need the highlights. The last three layers of Darknet-53 comprises of Averagepool, completely associated and SoftMax layers.

2.4. Convolution Layers

Since we definitely realize that YOLOv3 is a Fully Convolutional Network (FCN). In this manner, clearly YOLO v3 has huge quantities of convolution layers. For us it is useful to assemble them in a solitary capacity.

2.5. Detection Layers

YOLO v3 contains three location layers and in every three identification layers distinguish at various scales. For every cell in the element map, identification layer predicts 'num_anchors * (5 + num_class)' values utilizing '1 * 1' convolution. YOLO v3 utilizes 3 grapple boxes for every identification layers. Therefor 3 stays for every discovery layers and 3 identification layers produce '3 * 3 = 9'anchor boxes altogether. Also, for every 3 anchor boxes it predicts 4 directions of the crate (bx,by, bw, bh), certainty score(pc)and class probabilities(c1, c2, c3… … . cn).

2.6. Up sample Layers

Since we have to connect with the remaining yields of the Darknet-53 preceding applying recognition on an alternate scale we will utilize the example of the component guide utilizing closest neighbour introduction.

2.7. Non-max suppression

Since the model will create a great deal of boxes, so we will dispose of all undesirable boxes with low certainty score (pc). Likewise, we will maintain a strategic distance from different boxes comparing for one article. We will dispose of all cases with high cover utilizing non-max suppression for each class.

2.8. Final Model

At long last, we will characterize the model class by utilizing every one of the layers that are going to use in the YOLOV3 model.

2.9. Converting weights to TensorFlowFormat

Since we have just conveyed the YOLO v3[1] model in TensorFlow so now its opportunity to stack the official loads. What we will do now is that, we will repeat through the first weight record and bit by bit make 'tf.assign' tasks.

2.10. Running the model and mapping weights to the TensionFlow model

To start with, we will stack the first weight document and guide it to the TensorFlow model and we at that point pass an example picture to create yield of it.

Step-2: API advancement:

Since, goal of our venture is to advancement of an online framework to perceive protests in a scene picture, in this way we have to send an API that can deal with every one of the undertakings performed by our model sent in Step-1.

2.11. Required dependencies

I. Flask II. OS III. Requests IV. Subprocess

3. Conclusion and Future Scope

Object detection is the primary ultimate ability for most computer and robot vision system. Even though much progress has been noticed in the recent days, and some pre-existing methodologies are now a part of multiple electronics (e.g., auto-focus in smart phones is a one kind of face detection) which are consumer-based or have been associated in assistant driving methodologies, we are very distant from having performance, in particular in terms of open-world learning, in similar to a human being. It is a matter to be noted that object detection has not been performed much in many domains where it could be of much assistance. As robots which are mobile or are able to move on instructions or command, and in general automatic machines, are beginning to be more broadly deployed (e.g. drones., quad-copters and soon service robots), the requirement of object detection systems is definitely and is gaining more popularity and more importance.

References

[1] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing and Sun Jian.(2016)"Deep Residual Learning for Image Recognition" *2016 IEEE Conferenceon Computer Vision and Pattern Recognition (CVPR).*

[2] Redmon, Joseph, Divvala, Santosh, Girshick, Ross and Farhadi, Ali"You Only Look Once: Unified, Real-Time Object Detection"2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[3] Redmon, Joseph and Farhadi, Ali."YOLO9000: Better, Faster, Stronger" 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[4] Redmon, Joseph and Farhadi, Ali. YOLOv3: An Incremental Improvement. University of Washington, 8 Apr 2018.

[5] International Journal of Innovative Research in Computer and Communication Engineering (IJSR)

[6] HuiJonathan,"Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3)" by

[7] REDMON, J. YOLO: Real-Time Object Detection

[8] Redmon, Joseph and Farhadi, Ali. YOLOv3: An Incremental Improvement. University of Washington, 8 Apr 2018.

[9] Deng, Jia, Dong, Wei, Socher, Richard, Li, Jia-Li, and Fei-Fei, Li. "ImageNet: A Large-Scale Hierarchical Image Database"2009 IEEE Conference on Computer Vision and Pattern Recognition, 200.

[10] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., &Jackel, L. D. (1989). "Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551".

[11] LeCun, Y., Bottou, L., Bengio, Y., &Haffner, P. (1998). "Gradient-based learning applied to document recognition"Proceedings of the IEEE, 86(11), 2278-2324.

[12] Simonyan, K., &Zisserman, A. (2014)."Very deep convolutional networks for large-scale image recognition.arXiv preprint arXiv:1409.1556"

[13] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D&Rabinovich, A. (2015)." Going deeper with convolutions". In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[14] He,K.,Zhang,X.,Ren,S.,&Sun,J.(2016). "Deep residual learning for image recognition".In Proceedings of the IEEE conference on computer vision & pattern recognition (pp. 770-778).

[15] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., &Ramanan, D. (2010). "Object detection with discriminatively trained part-based models". IEEE transactions on pattern analysis and machine intelligence, 32(9), 1627-1645.

[16] Girshick, R. B., Felzenszwalb, P. F., &McAllester, D. (2012). Discriminatively trained deformable part models, release 5.

[17] Hosang, J., Benenson, R., Dollár, P., &Schiele, B. (2016). "What makes for effective detection proposals?".IEEE transactions on pattern analysis and machine intelligence, 38(4), 814-830.

[18] Redmon, J., &Farhadi, A. (2016). YOLO9000: better, faster, stronger. arXiv preprint arXiv:1612.08242.

[19] Sharif Razavian, A., Azizpour, H., Sullivan, J., &Carlsson, S. (2014)."CNN features off-the-shelf: an astounding baseline for recognition". In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 806-813).

[20] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[21] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

[22] Girshick, R. (2015). "Fast r-cnn". In Proceedings of the IEEE international conference on computer vision.