International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 05 | May - 2022 Impact Factor: 7.185 ISSN: 2582-3930

YOLO: A REAL TIME OBJECT DETECTION ALGORITHM

MRS.G.ANUSHA BHUVANESHWARI M.E

Assistant Professor, Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Tamil Nadu,

Leela S(AC18UCS054), Nivitha Jose X C(AC18UCS073), Mariya M(AC18UCS059)
UG Scholar, Department of Computer Science and Engineering, Adhiyamaan College of Engineering, Tamil Nadu, India.

ABSTRACT

Visual impairments have become one of the most predominant problems for the last few decades. To keep doing their daily tasks, vision-impaired people usually seek help from others. Hence, over thousands of papers have been published on these subjects that propose a variety of computer vision products and services by developing new electronic aids for the blind.

This paper describes CPU Based YOLO, a real time object detection model to run on non-GPU computers that may facilitate the users of low configuration computer. There are a lot of well improved algorithms for object detection such as YOLO, Faster R-CNN, Fast R-CNN, R-CNN, Mask R-CNN, R-FCN, SSD, Retina Net etc. YOLO is a Deep
Neural Network algorithm for object detection which is most fast and accurate than most other algorithms. YOLO is designed for GPU based computers which should have above 12GB Graphics Card. In our model, we optimize YOLO with OpenCV such a way that real time object detection can be possible on CPU based Computers. Our model detects object from video in 10.12 – 16.29 FPS and with 80-99% confidence on several Non –GPU computers. CPU Based YOLO achieves 31.05% map.

Keywords - Object Detection, real time, YOLO, CPU, Deep Learning

INTRODUCTION

Computer vision is associate knowledge base field that has been gaining enormous amounts of traction within the recent years and self-driving cars have taken centre stage. Another integral part of computer vision is real time object detection. Real time object detection aids in cause estimation, vehicle detection, traffic control, CCTV monitoring etc. For this reason, day to day progress in technology is necessary so that the ability to give vision in computer is possible. So, low-cost technology and learning tools are in requirement [1], [2], [3]. The techniques which are currently used for computer vision are more costly than previous because of developing larger and deeper networks to gain higher accuracy [4], [5],[6]. Techniques that are entirely computer based mostly need vast quantity of GPU power and even they aren't always real time, making them inappropriate for day-to-day applications. The methods such as YOLO (You Only Look Once) [7], Faster R-CNN [8], Fast R-CNN [9] etc. have affluently obtained an efficient, fast and accurate model with high mean average precision (MAP); But, their frames per second (FPS) on non-GPU computers drive them unsuitable for real time application. In this paper, we are proposing a model named "CPU Based YOLO" to run YOLO [7] algorithm on non-GPU computer with OpenCV [10]. Our model is able to execute videos from external source or from webcam with minimum 10.12FPS, 80-99% confidence and with 31.05% MAP that is suitable for real time application within low cost and less effort.

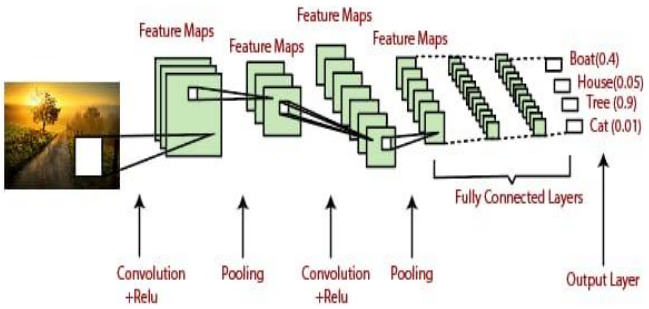


Fig. 1. Convolutional Neural Networks design

DEEP LEARNING

Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower-level features. Automatically learning features at multiple levels of abstraction allow a system to learn complex functions mapping the input to the output directly from data, without depending completely on human-crafted features. Deep learning algorithm seek to exploit the unknown structure in the input distribution in order to discover good representations, often at multiple levels, with higher-level learned features defined in terms of lower-level features. The hierarchy of concepts allows the computer to learn complicated concepts by building them out of simpler ones. If we draw a graph showing how these concepts are built on top of each other, the graph is deep, with many layers. For this reason, we call this approach to AI deep learning. Deep Learning excels on problem domains where the inputs (and even outputs) are analog. Meaning, they are not a few quantities in a tabular format but instead are images of pixel data, documents of text data or files of audio data. Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction.

OpenCV

OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state of the art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies. Along with well-established companies like Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota that employ the library, there are many startups such as Applied Minds, Video Surf and Zeitera that make extensive use of OpenCV. OpenCV's deployed uses span the range from stitching street view images together, detecting intrusions in surveillance video in Israel, monitoring mine equipment in China, helping robots navigate and pick up objects at Willow Garage, detection of swimming pool drowning accidents in Europe, running interactive art in Spain and New York, checking runways for debris in Turkey, inspecting labels on products in factories around the world on to rapid face detection in Japan.

TENSORFLOW

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. Itis used for both research and production at Google, TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). Tensor Flow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

ALGORITHM

RCNN - Regional Based Convolutional neural network

Convolutional Neural Network (CNN) operates from a mathematical perspective and is a regularized variant of a class of feedforward artificial network (ANN) known as multilayer perceptron's that generally means fully connected networks in which every neuron in a layer is connected to all neurons in the further layers Regularization applies to objective functions in ill-posed optimization problems and adds on the information in order to solve an ill-posed problem or to prevent overfitting

Now, we'll see how CNN trains and predicts in the abstract level so, When it comes to programming a CNN, it usually takes an order 3 tensor as input with shape (no. of mages) x (image width) x(image depth) that sequentially goes through a series of processing like convolutional layer, a pooling layer, a normalization layer, a fully connected layer, a loss layer etc that makes abstracted images to a feature map, with the shape (no. of images) x (feature map width) x (feature map Channels) [7,16]. Here, tensors are just higher-order matrices and below we have given layer by layer running of CNN in a forward pass:

Here, tensors are just higher-order matrices and below we have given layer by layer running of CNN in a forward pass:

$$x^1 \rightarrow w^1 \rightarrow x^2 \rightarrow \ldots \rightarrow x^{L-1} \rightarrow w^{L-1} \rightarrow x^L \rightarrow w^L \rightarrow z^L$$

Where usually an image (order 3 tensor). It goes through the processing in the first layer, denotes parameters involved in the first layer's processing collectively as a tensor. The output of the first layer which also acts as the input to the second layer processing and the same follows till all layers in the CNN have been finished, which outputs. To make a probability mass function, we can set the processing in the (L-1) th layer as a SoftMax transformation of (cf. the distance metric and data transformation notes) last layer is the loss layer. Let us suppose here t that is the corresponding target (ground-truth) value for the input then a cost or loss function can be used to measure the discrepancy between the CNN prediction and the target t, for which a simple loss function can be given as follows:

$$\frac{p}{1-p} = b^{\beta_0+\beta_1 x_1+\beta_2 x_2}$$

$$p = \frac{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}}{b^{\beta_0+\beta_1 x_1+\beta_2 x_2}+1} = \frac{1}{1+b^{-(\beta_0+\beta_1 x_1+\beta_2 x_2)}}$$

By the above formula when are fixed, either the probability that Z=1 for a given observation or the log-odds that Z=1 for a given observation can be easily computed. In a logistic model [17,18], the main use-case is to be given the probability p that Z=1 and an observation (x1, x20).

RELATED WORK

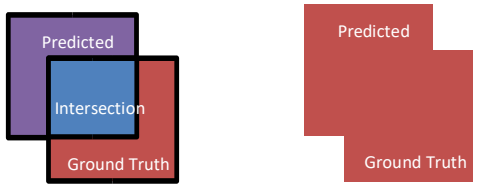
The evaluation in computer vision with Deep Learning has been established and accomplished with time, primarily over one particular well-known algorithm named Convolutional Neural Networks (CNN). Fig. 1 shows the architecture of CNN. It has a convolution layer where a filter is convolved with various parts of the input to generate the output. The use of a convolution layer permits for relative patterns to be drawn from associate input. Moreover, a convolution layer tends to own less weight that require to be learned than a completely connected layer as filters don't need assigned weight from each input to each output [1], [2].

A. YOLO

To generate a single step procedure including object detection and classification You Only Look Once (YOLO) [7] was introduced. YOLO is successful to gain 45 FPS and further a lite version named Tiny-YOLO, achieves around 244 FPS (Tiny-YOLOv2 [11]) on GPU computers. Why YOLO is differed from other existing networks? The answer is: at the same time bounding box predictions and class predictions can be done by YOLO. First, the input image is sliced up into $S \times S$ grids [7]. Second, B bounding boxes are included in each grid cell, each with a score of confidence. The probability of an object exists in each bounding box is known as confidence and is defined as:

$$\mathbf{C = Pr\ (object) * IOU}^{\mathbf{truthpred}}$$

Where, IOU is the abbreviation of intersection over union which describes a measure (a fraction between 0 and 1) of overlap between two bounding boxes. Intersection is the overlapped area between the predicted bounding box and ground truth (Object), and union is the total area of predicted and ground truth boxes as drawn in Fig. 2. For accurate detection, the IOU should be near to 1, so that the predicted bounding box lies closer to the ground truth. IOU can be expressed as:



(a) Area of Overlap (b) Area of Union

Fig. 2. Intersection Over Union

At the same time, while generating bounding boxes, each grid cell predicts C conditional class probability of the object. The class-specific probability for each grid cell is:

$$\Pr(\text{Class}_i \mid \text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truthpred}} = \Pr(\text{Class}) * \text{IOU}^{\text{truthpred}} \quad (3)$$

YOLO utilizes the equation below for calculating loss function and finally improve confidence [7]:

Loss Function:

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]$$
$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$
$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left(C_i-\hat{C}_i\right)^2$$
$$+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}\left(C_i-\hat{C}_i\right)^2$$
$$+\lambda_{noobj}\sum_{i=0}^{S^2}1_{i}^{obj}\sum_{c\in classes}\left(p_i(c)-\hat{p}_i(c)\right)^2. \quad (4)$$

$$avgPrecision=\sum_{k=1}^{n}P(k)\Delta r(k) \quad (5)$$

$P(k)$ refers to the precision at threshold k and $\Delta r(k)$ refers to the fluctuate in recall [7].

CPU BASED YOLO ARCHITECTURE

Our aim is to build the model (CPU Based YOLO) for CPU Based real time object detection. Developers of YOLO used Darknet [16] framework for running the algorithm. We run YOLOv3 with Dark Flow [17] and Open CV [10] as only they are favorable for CPU Based YOLO. Our aim was to develop the model in Windows operating system but we found that Dark Flow installation in windows is really a complex task. However, we installed it and run YOLOv3 [12] on it, but the result was terrible. The FPS was too low and the starting time was too much. Then we start to do the task on Open CV. We input a video though Open CV and found a good FPS in the output. The architecture is shown in

SETUP

We loaded YOLOv3 and Dataset (COCO) [14] through the framework Open CV Procedure of loading program.

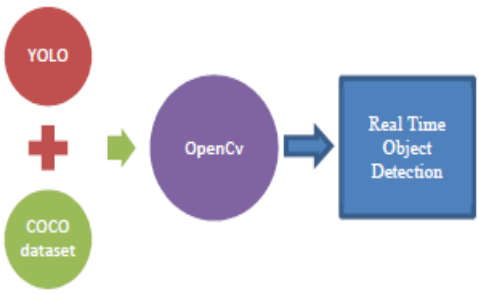


Fig. 3. Architecture of CPU Based YOLO

PROPOSED SYSTEM

This project tries to detect the object and transform that object into the audio form and inform blind person about those objects. Our system consists of a box which has a portable camera and a system which will process that image. Images are captured with a portable camera device with real-time image recognition by using object. This project tries to detect the object and transform that object into the audio form and inform blind person about those objects.

Our system consists of a box which has a portable camera and a system which will process that image. Images are captured with a portable camera device with real-time image recognition by using object detection models. After detecting an object that information is translate into audio. It takes an image as an input goes through Mask-RCNN procession which generates proposals regions of that object from a given input image which is taken from camera.

It takes an image as an input goes through Mask-RCNN procession which generates proposals regions of that object from a given input image which is taken from camera. After detecting an object that information is translate into audio.

FUNCTION OF YOLO FRAMEWORK

YOLO algorithm works using the following three techniques:

1. RESIDUAL BLOCK

First, the image is divided into various grids. Each grid has a dimension of S x S. The following image shows how an input image is divided into grids.

Fig. 4. SxS DIMENSION GRIDS

In the image above, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

2. BOUNDING BOX

A bounding box is an outline that highlights an object in an image. Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)

Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.

Bounding box center (bx, by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.

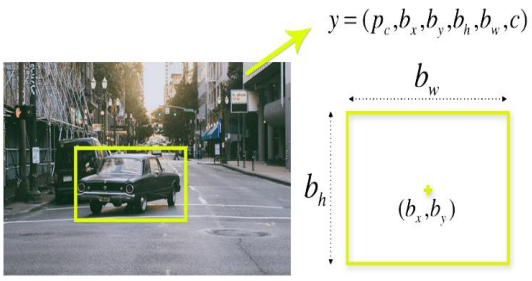


Fig. 5. BOUNDING BOX

YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

3. INTERSECTION OVER UNION (IOU)

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

The following image provides a simple example of how IOU works.

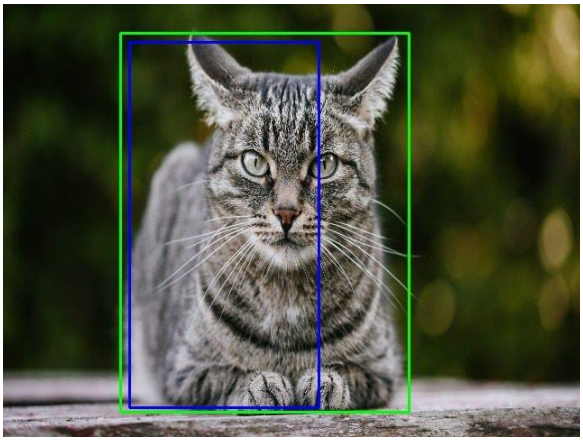
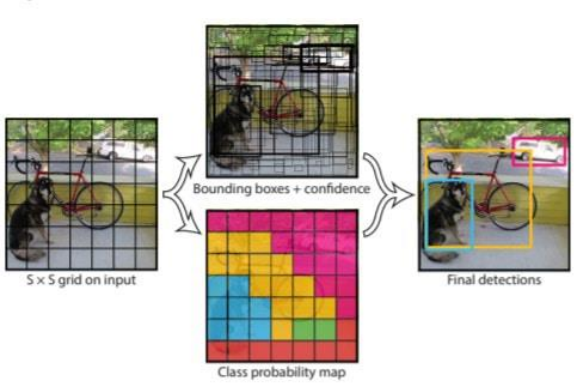


Fig. 6. IOU

In the image above, there are two bounding boxes, one in green and the other one in blue. The blue box is the predicted box while the green box is the real box. YOLO ensures that the two bounding boxes are equal.

Combination of the three techniques

The following image shows how the three techniques are applied to produce the final detection results.



First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object.

For example, we can notice at least three classes of objects: a car, a dog, and a bicycle. All the predictions are made simultaneously using a single convolutional neural network. Intersection over union ensures that the predicted bounding boxes are equal to the real boxes of the objects. This

phenomenon eliminates unnecessary bounding boxes that do not meet the characteristics of the objects (like height and width). The final detection will consist of unique bounding boxes that fit the objects perfectly.

For example, the car is surrounded by the pink bounding box while the bicycle is surrounded by the yellow bounding box. The dog has been highlighted using the blue bounding box.

CONCLUSION & FUTURE ENHANCEMENTS

The Project entitled "Object Detection and Translation for Bind People Using Deep Learning" has been developed and this satisfies all proposed requirements. The system is highly usable and user friendly. All the system objectives have been met. All these phases of development are done according to methodologies. The application will execute successfully by fulfilling the objectives of the project. Further extensions to this application can be made as required with minor modifications.

Discovering of object is a very time exhausting process to draw large quantities of bounding boxes manually. To release this burden, semantic prior unsupervised object discovery multiple instances learning and deep neural network prediction can be integrated to make the best use of image-level supervision to cast object category tags to similar object regions and improve object limits. Furthermore, this model is loaded into android, and objects are detected in mobile camera, and those object names are spelled out by the voice assistant API in the app which is helpful for blind people to navigate oneself.

REFERNCES

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks, vol. 61, pp. 85–117, 2015. 1

[2] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in Advances in neural information processing systems, 2013, pp. 2553–2561. 1

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587. 1

[4] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 2818–2826. 1

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." In AAAI, vol. 4, 2017, p. 12. 1

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. 1

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE with the OpenCV Library, O'Reilly Media Inc. Sebastopol. 1, 2

[8] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," CoRR, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506. 01497. 1

[9] R. B. Girshick, "Fast R-CNN," CoRR, vol. abs/1504.08083, 2015. /abs/1504.08083. 1

[10] G. Bradski, A. Kaehler, 2008, Learning OpenCV – Computer Vision Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," arXiv preprint, 2017. 2

[12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018. 2

[13] M. Everingham, L. Van Gool, C. Kl. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. JJCV, 2010. 2

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár. Microsoft COCO: Common Objects in Context. In ECCV. 2014. 2, 3, 4

[15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal
Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan,
Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Largescale machine learning on heterogeneous distributed systems, 2016, arXiv:1603.04467. 2

[16] J. Redmon. Darknet: Opensource neural networks in c. http://pjreddie.com/darknet/, 2013–2016. [Accessed January 06, 2020]. 2

[17] Darkflow, 2019. https://github.com/thtrieu/darkflow. [Accessed
December 25, 2019]. 2

[18] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models.
IEEE Transactions on Pattern Analysis and Machine Intelligence, (9):1627–1645, Sept 2010.4

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer, 2016. 4

[20] C. Severance, "Discovering JavaScript Object Notation," Computer, vol. 45, pp. 6–8, 2012.4