Comparative analysis of Mask R-CNN and YOLO algorithms for Object Detection

Sai Bhargav Mandavilli, *Department of Electrical and Computer Engineering, University of Florida*

***Abstract*— Object detection is the process of identifying the class of items discovered in an image. Items are discovered by locating specific regions and drawing rectangular bounding boxes around them. Researchers' efforts have ensured continuous improvement in the quality of algorithms used to solve object detection. However, there is significant study to be done on selecting the correct detection algorithm for an application. This paper mainly focusses on performing a comparative analysis between two major algorithms: Mask R-CNN and YOLO (you only look once) to find the quickest and most accurate for object detection applications. Using a common dataset, the algorithms are evaluated to quantitatively find their strengths and limitations and analyze them based on accuracy, correctness and speed. Experimental results infer that the selection of algorithms depends on the application in context. However, for a common dataset of a specific topic, YOLO proved to be faster and more accurate than Mask R-CNN.**

***Index Terms*— instance segmentation, Mask R-CNN, Object detection, you only look once**

I. Introduction

Humans can identify an object by visually observing it. The image information flows through the retina and the brain decides about the object being observed. Years of research have been dedicated to imitating this intelligence in machines to make them smarter. The human brain is a connected network of neurons (or nerve cells) which are fundamentally responsible for sensing inputs from the external environment and commanding our body to react to them. Neural networks are inspired by this behavior of the human brain and it is developed to mimic the working of biological neurons in a machine. Deep Learning is essentially a multi-layered neural network that attempts to simulate the behavior of a human brain. It contains models that train on large amounts of data to make better decisions about topics. In today's technologically advancing world, deep learning can be used in a wide range of applications. Few examples are self-driving cars, voice assistance in smart homes, TV remotes, facial recognition, health care, drug development, gaming and so many other fields [1]. With an endless list of possibilities, it is not an exaggeration to say that deep learning will improve every aspect of our lives in the near future.

Object detection is one area that has gained researchers' attention over the past few decades because of its wide applications. It is a technique that allows a computer to locate and make decisions about the objects from an image. For example, if there is a picture of a cat, we can recognize the image by looking at it because we are trained to identify cats. Similarly, object detection helps machines to do the same. It trains them with many images of cats by locating where the cat is in each image, drawing a box around it and labelling that region as 'cat'.

Neural networks can be taught the features of an image making them suitable for object detection. Computing capability and image datasets available for training has increased considerably in the last few years making neural network algorithms evident in object detection [2]. In 2014, R.Girshick et al. proposed a combination of convolutional neural networks with region proposals, called R-CNN, which gained high interest in the scientific community and paved a new path for using deep learning techniques in object detection [3]. In the following year, J. Redmon et al. published an algorithm called YOLO (You Only Look Once), which ran using only a single neural network.

In recent times, just as popular as deep learning, object detection has also been applied for advancements in many areas of science and technology. During the Covid-19 pandemic, health agencies used object detection techniques for crowd control in public areas. A research team from Japan installed cameras at strategic locations in a forest for automatically recognizing animals. They achieved an accuracy of close to 100% in recognizing the animal images by training the captured images on a R-CNN system. In another experiment, these algorithms were used in analyzing images captured by security cameras in airports to identify threats and risks. These are just a few examples of the many applications that can be served by object detection.

This brings us to the main question on how to select the right object detection algorithm. Some applications such as military or security demands the accuracy to be close to 100%. Whereas applications such as covid crowd control are a bit flexible on accuracy but expect quick results. IoT applications using camera modules monitor all kinds of environments and use these detection algorithms to process sensor data. It becomes very crucial to classify the existing algorithms and rank them on the basis of common parameters.

In this paper, we consider two commonly used object detection algorithms, Mask R-CNN and YOLO and evaluate them on the same metrics: speed, accuracy and precision. Results obtained by running the algorithms on common data sets, helps us understand the effectiveness of each algorithm and provides insights on which algorithm to use for a given application set.

II. Description

A. Using Mask R-CNN for object detection

In the previous section, we discussed convolutional neural networks and its application in object detection. R-CNN or Region-based CNN was founded for improving object detection. In this method, the entire image is divided into smaller rectangles of different sizes. These rectangles, also called region proposals, are then individually looked at in a brute-force method [4]. After dividing the image into thousands of regions, the algorithm considers each region and will create an abstraction to quantify its content. This is called a feature vector. Fig.1 shows the process of dividing an image into regions and generating the corresponding feature vector.

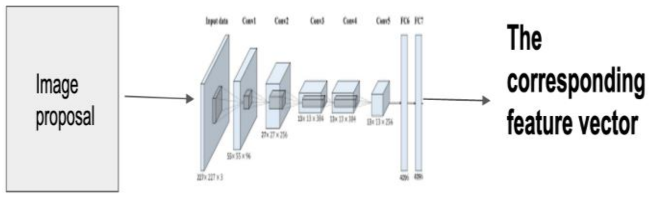


Fig. 1. Black box representation of the CNN

The vectors are then classified to generate a confidence score on the prediction. This is done to detect the class for each region. Each object class has its related SVM (support-vector machine) and we calculate the score for each SVM for all individual regions. If there are m number of object SVMs, we get m scores. After generating the scores, the algorithm uses a technique called greedy non-maximum suppression which checks the scores for all objects and rejects a region if it has a higher score when combined with another region.

Fast R-CNN was developed based on R-CNN to increase the speed of processing the regions. Proposals of the same length from the image matched in a process called ROI (Region of interest) pooling. The quickness of the algorithm is because proposals happen through the ROI layer where every region is analyzed at the same time compared to the independent processing in CNN. Faster R-CNN is the advanced version of Fast R-CNN. It improves computation speed with the help of another convolutional network called RPN (Region Proposal Network). RPN processes the input image on a backbone network and learns about objects present in the image by placing anchors between the input image and feature map. Due to this, Faster R-CNN brings down the time required for generating confidence scores for the regions by a factor of 200 and also shares the feature layers with detection stages.

Mask R-CNN is the last of the major improvements to the Neural Network techniques used for object detection. Released in 2017 by a group of Facebook AI researchers, this algorithm runs on Google TensorFlow 1.14 and below. The model was developed to overcome the pixel-to-pixel alignment problems between inputs and outputs. Along with the class label and bounding box offset used by Faster R-CNN, an additional branch was added to output the object mask [5]. It operated in two-stages. First stage is identical with Faster R-CNN (using an RPN) followed by outputting the binary mask for each region that happens parallelly with predicting the class and box offset. When training the models, each region is sampled with a multi-task loss, L = Lcls + Lbox + Lmask. Lcls represents the classification loss, Lbox and Lmask are for the bounding box and mask of a region respectively. Selecting Lmask only for the k-th mask can allow the network to generate masks for all classes without any competition.

For identifying objects in an image, Mask R-CNN performs an instance segmentation where each pixel is assigned to a label. The pixel labels are assigned between a set of categories without differentiating object instances. The individual objects are localized used bounding boxes. The following procedure (as shown in Fig.2) explains how the algorithm works for object detection:

1) RPN proposes the candidate objects confined by bounding boxes.
2) The input image is converted to grayscale and pixels are filtered by applying a threshold to obtain a binary mask. A similar mask classifier is used to generate the masks for each class.
3) Using features of previous CNN models, the image is processed to obtain the feature vectors.
4) RPN then generates a region of proposals by deploying anchors that were explained earlier.
5) Multiple bounding boxes are wrapped to a single definite box using a ROI aligned network.
6) These features are then inputted to connected layers to make a classification. The mask classifier then generates masks for all the classes without competition thereby making the process faster.

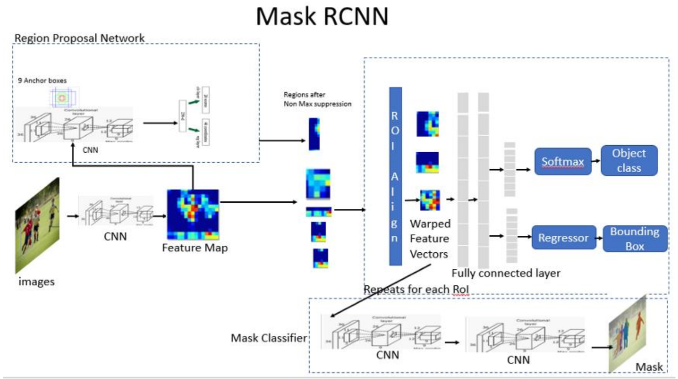


Fig. 2. Mask R-CNN process

The following section explains the usage of Mask R-CNN for object detection using a Python-based algorithm:

1) We begin with installing the necessary python libraries such as NumPy, OpenCV and Keras into the anaconda environment.
2) The available object classes are loaded as labels. The text file used in this project recognizes around a hundred commonly seen objects. Using this information, the model builds the weights for the objects.
3) The input image is then loaded and passed to the neural network. Here, the confidence score of each object is collected for all classes.
4) Then, pixel values to be used by Mask R-CNN are obtained by performing image segmentation. Then the pixel values are compared to the mask threshold to get the binary image.

5) Region of interest for the objects is obtained and is used for comparing with the mask values. Then the region is uniquely marked using a colored box and overlapped on the original image. The class of the object is added as a label to the box.

Fig.3 illustrates an example of performing Mask R-CNN on an experimental image. The results show the identified objects: a truck, dog, horse and two people with the confidence score next to each image.

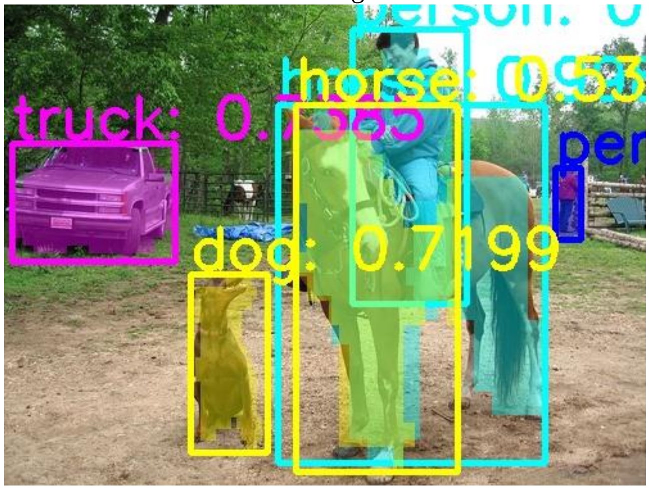


Fig. 3. Example Mask R-CNN output

B. *Using YOLO for object detection*

Traditional object detection methods employ classifiers for detection. When Joseph Redmon published, ‘You Only Look Once: Unified, Real-Time Object Detection’ [6] in 2016, it created a storm in the image processing community. The paper was cited 10,300 times since then. The algorithm solves object detection using regression on class confidence scores and bounding boxes that are spatially separated (meaning they do not overlap or have any common co-ordinates). It uses a single neural network that processes the entire image at once by identifying bounding boxes and calculating their class probabilities. The YOLO model can process up to 45 frames/images per second which is very fast compared to the CNN algorithms described in the previous section.

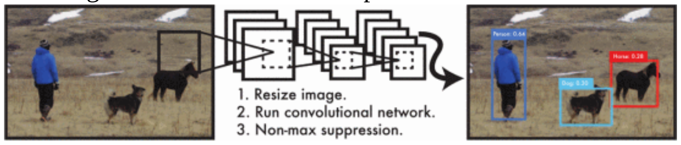


Fig. 4. YOLO detection system

Fig.4 illustrates the working of a YOLO system. Initially, any given image is resized to a fixed frame size, for example 448*448. The image is then passed through a single convolutional neural network. The network simultaneously develops the bounding boxes and generates the class probabilities for them. The scores are then matched with class thresholds obtained from weights for each class to generate a prediction. Since the algorithm treats frame detection as a regression problem, it does not require a complex pipeline making it faster to generate results. Unlike Faster R-CNN which breaks the images into regions, YOLO treats the entire image as a single region. In the former technique, even background items are treated as potential objects. Although YOLO encodes information from the entire image, it makes less than half the number of errors in identifying background objects when compared to Fast R-CNN.

The YOLO system divides an input image into a matrix of N*N dimensions. Each unit is called a grid cell. It is responsible for detecting an object. Each grid cell can generate an arbitrary number of bounding boxes and for each box inside the cell, confidence scores are generated for all classes trained to the system. This shows if the object predicted inside a bounding box of a cell is accurate or not. Confidence is given as $\Pr(\text{Object}) * \text{IOU}^{\text{truth}}$ where IOU stands for intersection over union of the predicted box and ground truth. Each bounding box comprises the axis co-ordinates of the center of the box, width and height relative to the whole image and the confidence score. Grid cells will calculate probability that tells the system whether it belongs to a certain class or not. This is done only once per grid and is applied to all bounding boxes inside the grid. When testing a trained model, we multiply the class probabilities with the confidence for each bounding box as shown in equation 1.

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}} \quad (1)$$

From this we obtain confidence scores for all bounding boxes inside a grid specific to each class trained to the system. This method is illustrated in Fig.5.

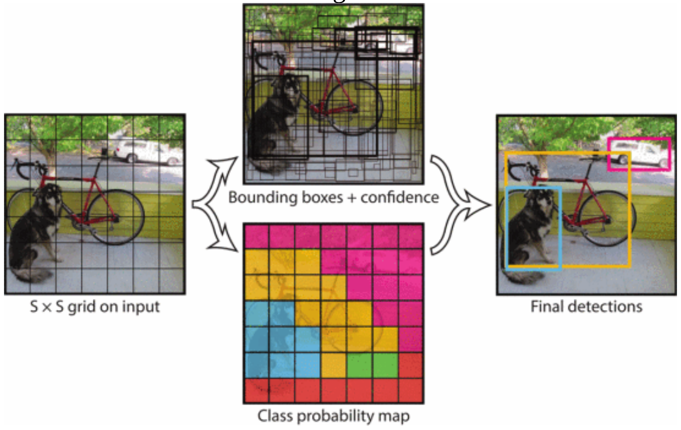
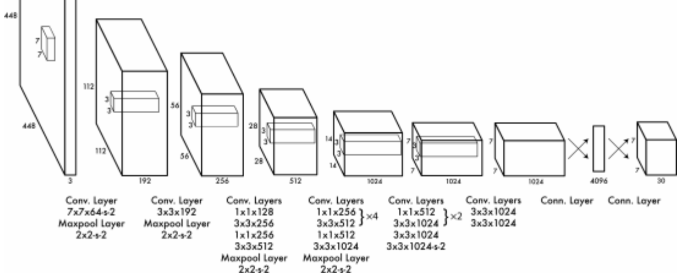


Fig. 5. YOLO system model divides the image into cells. Each cell generates class probabilities and bounding box prediction scores to detect an object.

The system processes a resized image through a neural network which has 24 layers connected to 2 full layers. The model is inspired by GoogLeNet [7]. Alternate 1*1 layers reduce the feature space of the preceding layer. These layers are called reduction layers and are followed by 3*3 convolutional layers similar to [8]. Fig.6 shows the layers of the applied network.


448
448
7
7
3
112
112
192
56
56
256
28
28
512
14
14
1024
7
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
Fig. 6. Neural network model used by YOLO. Each layer consists of 1*1 reduction layer followed by 3*3 layers and a two 2*2 layers in the end.

The following section explains the usage of YOLO for object detection using a Python-based algorithm:

1) We begin with installing the necessary python libraries such as NumPy, pandas, TensorFlow and Keras into the anaconda environment. Then we launch a Jupyter notebook from the environment to begin software development.
2) The neural network is defined by defining the convolution blocks as shown in Fig.6. We define all the layers by varying the input parameters. For example, the first layer takes image as input whereas the subsequent layers use its previous layer's convolved block as the input.
3) The class weights are loaded into the system to make the system model. These weights are used in predicting class confidence scores.
4) The previous steps prepared the system for object detection. The input image is resized and is divided into grid cells. Each cell is assigned with an array-like variable to store information.
5) Center co-ordinates, height, width and class confidence score for each cell and its bounding boxes are stored in the array data structure.
6) Define the anchors, calculate IOU for each bounding box and the probability threshold of the detected objects.
7) After applying threshold between cells to verify if a cell has higher scores when combined with another cell, correct the size of the bounding boxes to form a shape of the object.
8) Suppress the non-maximum boxes [9] and define the labels to make a class prediction for the detected object. Follow this by overlapping bounding box makers on the original image and tag it with the predicted label and confidence score.

Fig.7 illustrates applying YOLO on the same image as processed with Mask R-CNN in the previous section. We observe that YOLO has more predictions with better accuracy for this image.

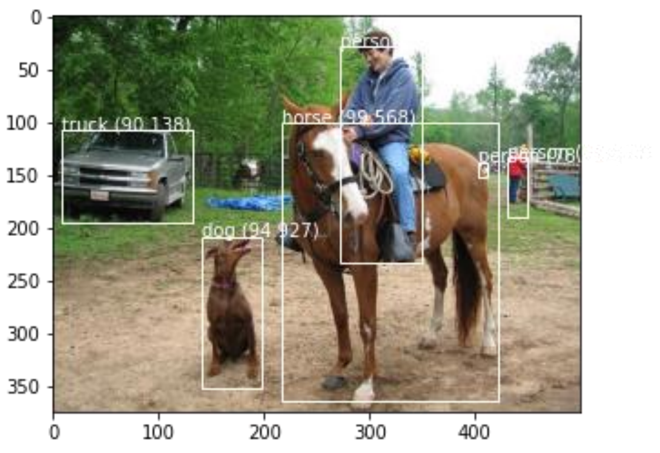


Fig. 7. Example YOLO output

III. Evaluation

Object detection is applied to solve scientific and societal problems across many domains. For example, a license plate detector uses object detection to identify the license number of a speeding car. In the agriculture sector, it is used for livestock monitoring, detecting infected crops etc. The applications are endless and if newer technologies want to interface object detection in their system, there must be some metrics to evaluate the performance of these algorithms. In general, applications demand faster and accurate results [10]. Quickness of an algorithm refers to how much time it takes from inputting the image to generating the output image with bounding boxes tagged with labels. Accuracy refers to the correctness of the prediction (measured by confidence score in percentage). Algorithms are expected to generate accurate results even when the images are distorted, blurry or have a high number of objects.

Additionally, performance of these algorithms is evaluated on two factors: classification and localization. Classification is the ability to identify and object and determine its class. Localization is to predict the location/spatial co-ordinates of the bounding boxes. Here, the predictions are compared with the ground truth co-ordinates. For each detected bounding box, IOU (Intersection Over Union) specifies how much overlap is between the prediction and the ground truth. IOU of 1 means perfect overlap. A threshold value to either discard or maintain a prediction is set based on this IOU value. We calculate the precision of an algorithm using equation 2,

$$\text{Precision} = \text{True positive} / (\text{True positive} + \text{False positive}) \quad (2)$$

where true and false positives mean whether the prediction is correct or not. Along with the precision value, the confidence score for each object is considered.

We further analyze the two algorithms based on the same parameters. To find out which of the two is faster and more accurate with their predictions, we train the models with a common data set obtained from CIFAR-10 [11]. It consists of around 60,000 images between 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck). After training, we test the models with ~25 images belonging to two classes (dog and cat). The hardware used for testing consisted of a 16GB DDR4 RAM, 512GB Solid State Drive running on Intel i7 core processor with a CPU speed of 2.6GHz. The Mask R-CNN model was tested on the community version of a python IDE developed by JetBrains called PyCharm whereas YOLO was developed on the Jupyter Notebook launched by Anaconda.

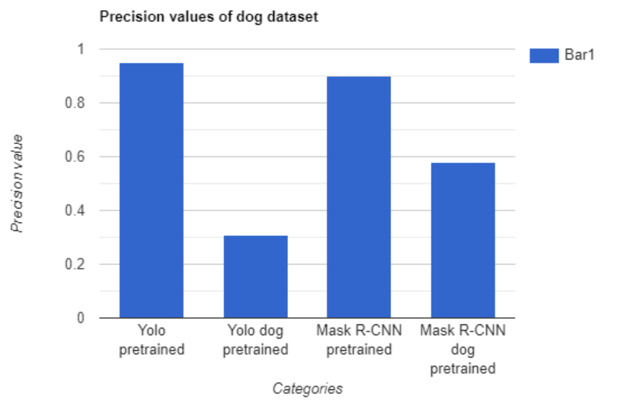


Fig. 8. Precision of generic and class specific trained models on dataset

Precision results when both the algorithms are tested on the dog images data set is displayed in Fig. 8. For both algorithms, a lower precision is reported when they are trained only on dog objects. For YOLO the precision is around 0.3 and it is close to 0.6 for Mask R-CNN. The precision scores drastically increase to over 0.95 for YOLO and to 0.9 for Mask R-CNN, which is relatively a lesser percentage increase.

One interesting example is shown in Fig. 9 when both algorithms failed to detect a dog in the foreground of the image. This is because the models were pre-trained in a generic approach and this output would not have been observed if the algorithms were trained with object specific data sets (in this case dog). This problem is called false negatives and is much evident in YOLO as compared to Mask R-CNN. Although generally trained YOLO models are known to handle farther objects better.

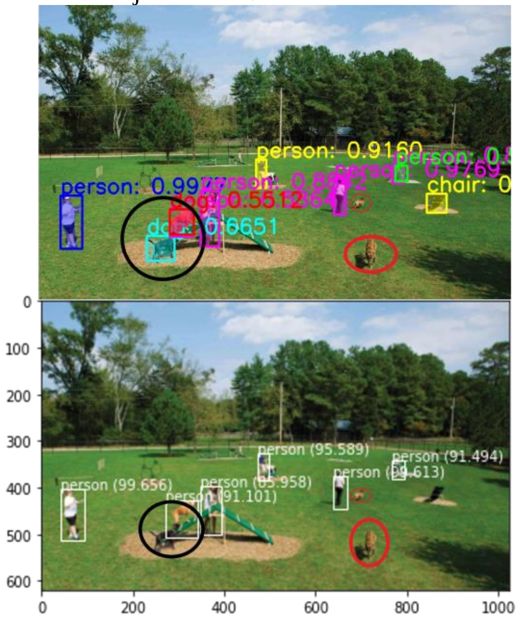


Fig. 9. Mask R-CNN (top) and YOLO (bottom). Red colored circles on the right point to the undetected dog and black colored circles on the left point the detection irregularities between both the models

The black colored circles on the left part of Fig.9 shows how Mask R-CNN is better at detecting nearby objects (here we are interested in dogs) whereas YOLO identifies the farthest person with better accuracy.

Speed in FPS (frames per second) is another factor to evaluate when deciding which algorithm has better performance. [12] performed an experiment on comparing speeds of different object detection algorithms using a COCO (Common Objects in Context) dataset. They ran models on 300*300 and 512*512 input images. Results showed that YOLO had a low of 40 and a high of 91 fps whereas Mask R-CNN had a low of 5 and high of 17 fps.

Memory footprint is critical in devices that run on limited RAM such as IoT devices or embedded systems in automobiles etc. A study in [12] also provided how much memory was consumed by object detection algorithms for processing a data set with 500 images of different classes. Multi-layered R-CNN models such as Mask R-CNN consumed almost 5GB of RAM for processing whereas a tiny version of YOLO only used 516MB of memory.

In table 1, training the models on 25 images belonging to two different classes, we collect information on the speed and precision metrics for the two methods. We additionally compare the collected data with other commonly used CNN models.

Table 1. Performance of Mask R-CNN, YOLO and others for evaluating speed and precision. Readings are based on training with 25 images from two different data sets

Detection models	Precision (%)	Speed (fps)
Mask R-CNN	76.4	5
YOLO	63.4	45
YOLO 300*300	73.7	81
YOLO 418*418	77.8	59
Faster R-CNN	70.0	0.5

Looking at the data from table 1, we can observe that YOLO and its variations offer faster processing and an acceptable accuracy but as mentioned previously, Mask R-CNN and its predecessors are suitable when processing images with smaller objects. Another metric to evaluate is the performance of the models processing blurry images. In a real-world application, we cannot expect the images to be still and the algorithms have to generate meaningful results in these conditions as well. Blur of an image is calculated using a percentage of Gaussian blur. For our experiment, we consider an image of a dog with 20% blur and evaluate it with the pre-trained models.

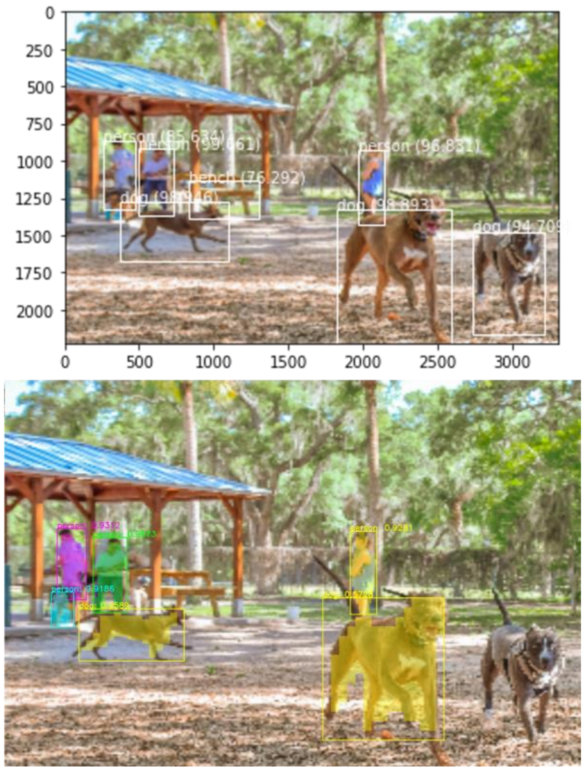


Fig. 10. Performance on blurred images by YOLO (top) and Mask R-CNN (bottom). YOLO outperforms with very accurate predictions.

Notice Fig.10 which shows the output of running a blurred image. YOLO has a clear edge in predicting very accurate results whereas Mask R-CNN fails to do so. The bottom-right dog in the image is not even detected by Mask R-CNN whereas YOLO predicts the same with an accuracy of 94.7%.

IV. Related Works

Application of deep learning algorithms in solving object detection has been studied by many researchers over the past decade. This section briefly describes a few of those works.

This paper [13] talks about studying satellite images that contained multiple objects and comparing the performance of R-CNN algorithms. It was observed that the object detection algorithms usually find it difficult to detect small-size objects, such as in satellite images. Training the models with a right data set is very important to expect favorable results since the performance of the models was observed to increase when they were trained on specific data sets rather than using a general approach. Another conclusion that was made in this paper is that a model will give better results when it is trained with a high number of images. Faster R-CNN had the highest success rate of 75.9% whereas YOLO had the lowest of all models with only 57.9% success rate when trained on a 2012 PASCAL VOC data set [14]. The paper was very successful in explaining the algorithms used for object detection. It had a clear understanding on how to evaluate the performance. The paper also contained test results that were obtained by training and testing models with satellite data to show which algorithm suits best for processing satellite images. The paper did not specify what was the experimental setup used for training and testing. It did not contain information on the speed of detection between the models and also did not specify any other information than accuracy of prediction.

Kong et al., [15] in 2016 performed a study on evaluating deep learning techniques for object detection using a model called HyperNet which obtained an accuracy score of 60% among different data sets. They studied many methods like Fast R-CNN, Faster R-CNN, RPN, Mask R-CNN and contextual modeling in object detection. They obtained accuracy values for these models between 59 and 82% for data sets covering 20 different classes.

Mittal et al [16], is another similar publication that discussed the CNN algorithms along with SSD, RetinaNet and CornerNet. The paper explains the algorithms to give a context to the user on how object detection is achieved. SSD which is abbreviated for Single Shot MultiBox Detector is an object detection method that relies on eliminating region proposals before making a prediction. It is quicker than R-CNN as it eliminates pixel-to-pixel segmentation and resizing the image. Similar to YOLO, it combines decision making into a single step. SSD is light-weight, easy to train and can be effortlessly integrated into a system. The data set used for training the models in this paper consists of low-altitude geographical images. The paper also surveys on the percentage of applications that use deep learning techniques. Faster R-CNN is used by 28.1% of object detection applications which is the highest followed by YOLO which is used 21.9%. Mask R-CNN is used by 12.5% of object detecting applications. Other commonly used are SSD, RetinaNet and CornerNet. The paper also has a dedicated section that addresses object detection for UAV datasets. The paper extensively researches on the models, providing background data covering many aspects of the models but does not provide much information on the data sets that were used in training their models which were the source of their conclusions.

UAVs (Unmanned Aerial Vehicles) offer maneuverability and can fly over remote locations to gather image data in a very short span of time. Compared to satellite imagery, drones can fly in cloudy weather and can capture more fine-grained images. They have been widely exploited in rescue operations, disaster management, in agriculture such as livestock management and crop identification. Drones are replacing traditional data collection techniques without human intervention. A major limitation of drone collected images is with blurry images, densely populated images, reflections and refractions that affect the quality of image which amplify the need for object detection algorithms. [17] conducts a review on object detection models used in UAVs. The authors state that for applications related to disaster response and recovery, where time is a critical requirement, YOLO is most commonly used among the CNN models. In forest surveillance, a project at the Shihwa Lake in Yeongjong Island used Mask R-CNN to study the population and habitat of birds because the model had very good accuracy when pre-trained with bird data sets. The paper then talks about several other applications where drones can be effectively used to capture data and which object detection model can be used for effective predictions. An example data set/evaluation would have added more trust on the author's research for the readers.

In [18], the models were trained on a COCO database to predict speed, accuracy and precision. For 2000 proposals identified in the dataset, RPN + Fast R-CNN achieved a maximum speed of 5 fps and had a mAP score of 70.4%. Although Faster R-CNN had a better score of 73.8%, it could process only 2.38 fps. Again, YOLO proved to be the fastest between the two as it achieved a processing speed of 21fps, a faster version of YOLO called Fast YOLO was able to achieve an incredible processing speed of 155 fps. For an input image resolution of 448*448, YOLO had a mAP score of 66.4%. Outperforming these models, in precision are the newer SSD algorithms which can generate an average mAP score of 75%. The speeds of these models though are not noticeably faster than YOLO. We do not like to dig deeper into SSD models since it is out-of-context for this paper. This paper is overall a good resource for classifying detection frameworks but it does not provide information on the performance of Mask R-CNN.

V. Summary

Identifying objects in images is a method of data collection. Today, the detection models are used in many applications and this trend is expected to increase in the future. In this analysis, we evaluated the algorithms against two metrics – precision and speed. The bounding boxes identified by the detector are compared with the ground truth based on overlapping percentage, also called IOU (Intersection Over Union) which is then used for calculating a confidence score among different object classes. Apart from this, we also evaluate the models based on processing speed, measured in fps and estimate memory consumption (amount of RAM allocated for the process). Data sets consisting of 25 images from two different classes were used to train the models and then helped collect information regarding their performance. To test the models, we developed code using Python programming language which used common neural network packages such as Keras and TensorFlow. Results from software testing were compiled to make the following conclusions regarding the performance of these models:

YOLO is arguably the best update to the detection frameworks. It is lightweight, fast and runs on a single neural network mask. Due to these features, the model became very popular in the industry. Although, it had some limitations. For example, with images that contained a large number of objects, YOLO failed to correctly recognize them but this was often neglected since those objects were not the focus feature. This lack of complexity was later fixed in the more recent versions of the algorithm. For images containing smaller objects, YOLO offered very low accuracy (as shown in Fig.8, there is only 25% accuracy for pre-trained dog images).

Mask R-CNN achieved a relatively higher percentage of precision, around 75%. Although there are several limitations with this model as well. Its performance drops severely when processing blurry images or objects in motion (proved in Fig.10) as compared to still objects. An improvement using a Mean Shift tracker, listed in [19] has evidently improved the performance for tracking objects in motions/blur. Also, the model only has a processing speed of 5fps making it the slowest among the other frameworks.

VI. Conclusion

This article discusses two of the latest deep learning algorithms used for object detection. Many modern-day technologies such as self-driving cars are heavily dependent on these algorithms for processing hundreds of images in milliseconds. Both the models were trained using a common dataset. It was observed that YOLO is the fastest between the two and is better at detecting blurry/in-motion objects whereas Mask R-CNN has good accuracy with still images and can perform better with images containing large numbers of objects. Although a common issue with both the algorithms is their inaccuracy in detecting nearby objects.

Each passing year introduces newer and more advanced detection algorithms as well as improvements to the existing ones. There is still a great deal of work to be done in this field only to find its use in many more applications.

References

[1] R. S. Latha, G. R. R. Sreekanth, R. C. Suganthe and R. E. Selvaraj, "A survey on the applications of Deep Neural Networks," 2021 International Conference on Computer Communication and Informatics (ICCCI), 2021, pp. 1-3, doi: 10.1109/ICCCI50826.2021.9457016.

[2] Xiao, Y., Tian, Z., Yu, J. et al. A review of object detection based on deep learning. Multimed Tools Appl 79, 23729–23791 (2020). https://doi.org/10.1007/s11042-020-08976-6.

[3] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.

[4] S. Elfouly, “R-CNN,” Medium, Nov. 19, 2020. https://medium.com/@selfouly/r-cnn-3a9beddfd55a (accessed Mar. 07, 2022).

[5] He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 2961-2969. 2017.

[6] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

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

[8] Lin, M., Chen, Q. and Yan, S., 2013. Network in network. arXiv preprint arXiv:1312.4400.

[9] J. Hosang, R. Benenson and B. Schiele, "Learning Non-maximum Suppression," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6469-6477, doi: 10.1109/CVPR.2017.685.

[10] A. Sobti, C. Arora and M. Balakrishnan, "Object Detection in Real-Time Systems: Going Beyond Precision," 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), 2018, pp. 1020-1028, doi: 10.1109/WACV.2018.00117.

[11] CIFAR-10 and CIFAR-100 datasets. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html. [Accessed: 09-Mar-2022].

[12] J. Hui, “Object detection: Speed and accuracy comparison (faster R-CNN, R-FCN, SSD, FPN, RetinaNet and...,” Medium, 26-Mar-2019. [Online]. Available: https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359. [Accessed: 09-Mar-2022].

[13] Doğan, Ferdi & Turkoglu, Ibrahim. (2021). Comparison of deep learning models in terms of multiple object detection on satellite images. Journal of Engineering Research. 10.36909/jer.12843.

[14] Pathak, A.R., Pandey, M. & Rautaray, S., 2018. Application of deep learning for object detection. Procedia computer science, 132: 1706-1717.

[15] Kong, T., Yao, A., Chen, Y. & Sun, F., 2016. Hypernet: Towards accurate region proposal generation and joint object detection, Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 845-853.

[16] Mittal, Payal & Singh, Raman & Sharma, Akashdeep. (2020). Deep learning-based object detection in low-altitude UAV datasets: A survey. Image and Vision Computing. 104. 104046. 10.1016/j.imavis.2020.104046.

[17] Ramachandran, A. and Sangaiah, A.K., 2021. A review on object detection in unmanned aerial vehicle surveillance. International Journal of Cognitive Computing in Engineering, 2, pp.215-228.

[18] Sanchez, S.A., Romero, H.J. and Morales, A.D., 2020, May. A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework. In IOP Conference Series: Materials Science and Engineering (Vol. 844, No. 1, p. 012024). IOP Publishing.

[19] D. -H. Nguyen, T. -H. Le, T. -H. Tran, H. Vu, T. -L. Le and H. -G. Doan, "Hand segmentation under different viewpoints by combination of Mask R-CNN with tracking," 2018 5th Asian Conference on Defense Technology (ACDT), 2018, pp. 14-20, doi: 10.1109/ACDT.2018.8593130.