Real-Time Object Detection Using YOLO: A Review

Preprint · May 2021

DOI: 10.13140/RG.2.2.24367.66723

2 authors:

Upulie Handalage
Sri Lanka Institute of Information Technology
5 PUBLICATIONS **10** CITATIONS

SEE PROFILE

Lakshini Kuganandamurthy
Sri Lanka Institute of Information Technology
2 PUBLICATIONS **8** CITATIONS

SEE PROFILE

Real-Time Object Detection using YOLO: A review

Upulie H.D.I
IT18107074
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
ireshaupulie@gmail.com

Lakshini Kuganandamurthy
IT17073592
Sri Lanka Institute of Information Technology
Malabe, Sri Lanka
lakkuga@gmail.com

***Abstract*—With the availability of enormous amounts of data and the need to computerize visual-based systems, research on object detection has been the focus for the past decade. This need has been accelerated with the increasing computational power and Convolutional Neural Network (CNN) advancements since 2012. With various CNN network architectures available, the You Only Look Once (YOLO) network is popular due to its many reasons, mainly its speed of identification applicable in real-time object identification. Followed by a general introduction of the background and CNN, this paper wishes to review the innovative, yet comparatively simple approach YOLO takes at object detection.**

Keywords—YOLO, CNN, object detection, image classification

I. Introduction

Although the human eye is capable of instantly and precisely identifying a given visual, including its content, location, and visuals close by interacting with it, the human made, computer vision-enabled systems are relatively low in accuracy and speed. Any advancements leading to improvements in efficiency and performance in this field could pave paths to creating more intelligent systems, much like humans. These advancements, in turn, would ease human life through systems such as assistive technologies that allow humans to complete tasks with little to no conscious thought. For instance, driving a car equipped with a computer vision-enabled assistive technology could predict and notify a driving crash prior to the incident, even if the driver is not conscious of their actions. Therefore, real-time object detection has become a highly required subject in continuing the automation or replacement of human tasks. Computer vision and object detection are prominent fields under machine learning and are eventually expected to aid unlocking the potential general-responsive robotic systems.

With the current technological advancements, creating openness and attainability of data to and from everyone connected to it has become an easy task. Most human lives revolved around mainstream personal computers (PCs), and smartphones have made this process even more accessible. Along with this process, the expansion of information and images available on the internet/cloud has become to the point of millions per day. Usage of computerized systems to utilize this information and make necessary recognitions and processes is vital due to humans' impracticality performing the same iterative tasks. The initial step of most such processes may include recognizing a specific object or area on an image. Due to the unpredictability of the availability, location, size, or shape of an item in each image, the recognition process is inconceivably hard to be performed through a traditional programmed computer algorithm. Factors such as the complexity of the foundation, light intensities too contribute to this.

Different strategies have been proposed to solve the problem of object identification throughout the years. These techniques focus on the solution through multiple stages. Namely, these core stages include recognition, classification, localization, and object detection. Along with the technological progression over the years, these techniques have been facing challenges such as output accuracy, resource cost, processing speed and complexity issues. With the invention of the first Convolutional Neural Network (CNN) algorithm in the 1990s inspired by the Neocognitron by Yann LeCun et al. [1] and significant inventions like AlexNet [2], which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 (thus later referred to as ImageNet) CNN algorithms have been capable of providing solutions for the object detection problem in various approaches. With the purpose of improving accuracy and speed of recognition, optimization focused algorithms such as VGGNet [3], GoogLeNet [4] and Deep Residual Learning (ResNet) [5] have been invented over the years.

Although these algorithms improved over time, window selection or identifying multiple objects from a single image was still an issue. To bring solutions to this issue, algorithms with region proposals, crop/warp features, SVM classifications and bounding box regression such as Regions with CNN (R-CNN) were introduced. Although R-CNN was comparatively high in accuracy with the previous inventions, its high usage of space and time later led to the invention of Spatial Pyramid Pooling Network (SPPNet) [6]. Despite SPPNet's speed, to reduce the similar drawbacks it shared with R-CNN; Fast R-CNN was introduced. Although Fast R-CNN could reach real-time speeds using very deep networks, it held a computational bottleneck. Later Faster R-CNN, an algorithm based on ResNet, was introduced. Due to Faster R-CNN not yet capable of surpassing state of the art detection systems, YOLO was introduced. This paper reviews the dominating real-time object detection algorithm You Only Look Once (YOLO).

Consisting of layers in the basic CNN architecture and YOLO networks, each layer's characteristics and the two versions of YOLO; YOLO-V1 and YOLO-V2 would be reviewed under this paper. The strengths and weaknesses of YOLO would be exposed, finally being followed by a summarized conclusion.

II. Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) could be taken as a subcategory under Deep Neural Networks specifically invented for image processing and object detection. CNN algorithms can be utilized without requiring an enormous amount of predefined substantial parameters for the provided image. This ease at training a model and the vast amount of

information available through the internet has made CNN algorithms possible. The mechanism CNN algorithms follow to express and extract features of the input data is entirely mathematical. This mechanism involves a weight sharing process that recognizes and identifies information that holds similar features. This process enables networks to analyze high data dimensions to achieve the final output of excellent classification in the end. One of the apparent obstacles in moving forward with getting better results using CNN models is the processing capabilities of available hardware and the scope of parameters in datasets.

The invention of the CNN [7] in 1998 with LeNet and its bloom in 2012 with AlexNet was at the error rate of 15.3% followed by ZF-net. The inventions of GoogLeNe and VGGNet has made the error rate lower over time. An exceptional milestone in this timeline was when ResNet surpassed the error rate of 3.6%, which was lower than that of the human eye (5.1%) in 2015, proving that deep learning models could surpass human capabilities.

A. *Structure of CNN*

A typical CNN is structured with multiple layers: an input layer, a convolutional layer, an active layer, a pooling layer, a fully connected layer and finally, an output layer. Some types of CNN models might include other layers for different purposes too. Figure 1 shows the basic structure of a CNN architecture.

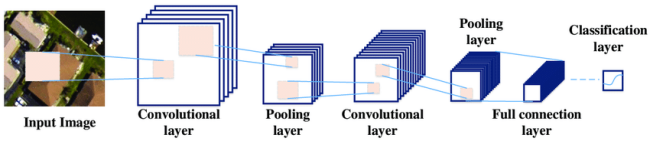

Figure 1: The typical CNN structure with seven layers

Source:https://www.researchgate.net/publication/340102110_Hierarchical_Multi-View_Semi-Supervised_Learning_for_Very_High-Resolution_Remote_Sensing_Image_Classification

This multi-layered architecture is diverse in layers and uses forward pass and error backpropagation calculations to achieve the target's proficiency. Training this architecture to become a model is a directed procedure that requires a collection of imagery data and their labels. Eventually, at the end of the training process, the most suitable weights would be calculated to be used at the testing phase. These layers, as mentioned above, could be further explained as follows.

1) *Input Layer*

The input layer is used to initialize the input image data and make all the available dimensions zero-centered. This layer is also responsible for normalizing the scale of all input data to a range within 0 and 1, which would help in accelerating the speed of converging. This normalization is also helpful in reducing redundancy by whitening the data. Principal Component Analysis (PCA) is done to degrade and decorate the available dimensions of the extracted data while focusing on key dimensions.[8]

2) *Convolutional Layer*

As the layer, which is why CNN received its name, the convolutional layer is the most critical layer in a CNN structure. Comprised of multiple element maps and many neurons inside them, each of these neurons is created to untangle nearby qualities of various positions in the previous layer [9]. Many nearby associations and many mutual attributes use a filter called CONV kernel, which slides on the original image inputted to it. The CONV kernel calculates the image's component portrayal by multiplying and adding the values of each pixel of the local correlated data within it before being added to the convolutional result. This so-called rule of convolution enables the features of the image to be extracted using the CONV kernel. The reason for filtering the various parts of an image with the same CONV kernel is that this refers to shared weights. This usage of shared weights enables neutral cells with the same features to be recognized and classified into the same object type. Parameters such as kernel size, depth, stride, zero-padding, and filter quantity can be inputted onto this.

3) *Active Layer*

The active layer is the layer used to solve the problem of the vanishing gradient due to underfitting. This underfitting, nonlinear problem is caused by the previous convolutional layer. One of the active layer functions such as Sigmoid, Tanh, the rectified Linear Unit (ReLu), the exponential Linear Unit (ELU), Leaky ELU, or Maxout could be used in solving underfitting, following their usage [10]. Considering the converging speed, ReLu function has been the most popular although Sigmoid and Tanh functions are still commonly used due to their simplicity and efficiency.

4) *Pooling Layer*

The pooling layer's job is to efficiently reduce the dimensions of the results sent from the convolutional layer. This is achieved by joining the neurons' outcome at one layer into a single neuron in the following layer, thus diminishing the elements of the component maps and incrementing the strength of selected extractions. Pooling layers are usually situated between two convolutional layers and can be categorized into three distinct types based on their width: general pooling, overlapping pooling and Spatial Pyramid Pooling (SPP). A pooling layer is called a general pooling layer when its width is mainly equal to its stride. General pooling's activities include max pooling and normal pooling. When the most extreme incentives from each neuron group from the previous layer are utilized, it is called max pooling. When it is done for the normal incentives, it is referred to as normal pooling. Overlapping pooling is when the width is longer than the stride. Therefore, abnormal state attributes from the input layer can be extracted and acquired by structuring a few convolutional layers along with a final pooling layer.

5) *Fully Connected Layer*

Often the last layer before the output layer, the fully connected layer transmits data to the output layer while being the completely associated layer amongst the CNN layers. By utilizing each neuron in the past layer and interfacing them to each neuron on its own, it simplifies and speeds up the data calculation process. It being a completely associated layer saves no spatial data and is constantly trailed by a yield layer.

6) Other Layers

Apart from the different layers used in structuring a CNN model mentioned above, some CNN models need additional layers to achieve the expected output. Layers such as dropout layers, regression layers come under this. Dropout layers are often used to solve overfitting by avoiding majorly subjective weights by updating weights of the neural cell knot with a certain probability (which is decided by the stochastic policy). Whereas, regression layer is used to classify features using a method such as logistic regression (LR), Bayesian Linear Regression (BLR) and Gaussian Processes for Regression (GPR). The output of a regression layer is the probabilities of all the possible object types.

III. Types of object detection algorithms

Algorithms available for object detection can be divided into two categories: classification-based algorithms and regression-based algorithms.

1) Classification based algorithms

Classification based algorithms are implemented in two stages. The initial stage is the selection of region that is of interest (RoI) in the image. Then these regions are classified with the use of a convolutional neural network. This approach of performing one stage prior to the other can be slow due to the need to run the prediction algorithms on each region selected in the first stage. Few common examples for this type of algorithms are the Retina Net, Region-based CNN (RCNN), the Fast-RCNN, Faster R-CNN and Mask-RCNN (which is known to be a state-of-art under regional-based CNN algorithms).

2) Regression-based algorithms

Regression-based algorithms are implemented so that instead of selecting and singling out regions of interest in an image, they predict classes and their relevant bounding boxes for the whole image in one run through the model. Since frame detection is treated as a regression problem, a complex pipeline is not necessary for regression-based algorithms. Famous examples of this type of algorithms are the Single Shot Multibox Detector (SSD) and YOLO algorithms. Due to the simultaneousness of the detection and its nature of high speed (achieved with a tradeoff with accuracy), these are commonly used for real-time object detection. The detection and understanding of the more popular YOLO algorithms require an initial establishment of what will be predicted before the models are used. The prediction would result in a bounding box (specifying the Object's location) along with a class that has the highest probability amongst the established set of classes.

IV. You Only Look Once (YOLO) algorithm

YOLO is a novel approach to detect multiple objects present in an image in real-time while drawing bounding boxes around them. It passes the image through the CNN algorithm only once to get the output, thus the name. Although comparatively similar to R-CNN, YOLO practically runs a lot faster than Faster R-CNN because of its simpler architecture. Unlike Faster R-CNN, YOLO can classify and perform bounding box regression at the same time. With YOLO, the class label containing objects, their location can be predicted in one glance. Entirely deviating from the typical CNN pipeline, YOLO treats object detection as a regression problem by spatially separating bounding boxes and their related class probabilities, which are predicted using a single neural network. This process of performing both bounding box prediction and class probability calculations is a unified network architecture that YOLO initially introduced.

YOLO algorithm extends GoogLeNet equations to be used as their base forwarding transport computation, assumably the reason behind the speed and accuracy of YOLO's real-time object detection. In comparison with R-CNN architectures, unlike running a classifier on a potential bounding box, then reevaluating probability scores, YOLO predicts bounding boxes and class probability for those bounding boxes simultaneously. This optimizes the YOLO algorithm and is one of the significant reasons why YOLO is so fast and less likely to have errors to be utilizable for real-time object predictions.

YOLO's architecture is similar to a typical convolutional neural network inspired by the GoogLeNet model for image classification. The network's initial layer first extracts the image's features, and the fully connected layers predict the output probabilities and coordinates. With 24 convolutional layers, two fully connected layers, 1x1 reduction layers and 3x3 convolutional layers, the full YOLO network model created [12].

A. Unified Detection of YOLO

YOLO is introduced as a unified algorithm as separate components merge into a single neural network as the final pipeline. For each bounding box to be predicted parallelly, the features of the entire image are globally reasoned. YOLO is designed in such a way that it does its own end-to-end training in real-time while keeping high-level average precision. To achieve unified detection, YOLO first separates the input image into a S X S size grids. If the Object's center is being placed into the grid cell; the grid cell tries object detection on itself. Thus, every grid cell tries to estimate a bounding box and their confidence scores across all classes trained to predict. The predicted confidence scores will reflect how confident it is to provide each label and bounding box to each object. Formally the confidence scores are defined as **Pr (Object) x IOUtruthpred**. If an object has been found inside the cell, this confidence score will be equal to the intersection over union (IOU) between the ground truth and the predicted box. If not, the confidence score would be equal to zero. The unified detection outputs each confidence score to have five parameters: **w, y, w, h,** and **confidence**. The **(x, y)** coordinates represent the center of the box with respect to the grid cell's boundaries. As mentioned above, if the box's center does not fall inside the grid cell, then the cell is not responsible for its prediction. With each coordinate being normalized to be contained inside the range of 0 and 1, the estimated Object's height and width are calculated with respect to the entire image. According to Mauricio Menegaz in his article [11] the prediction is of a few steps.

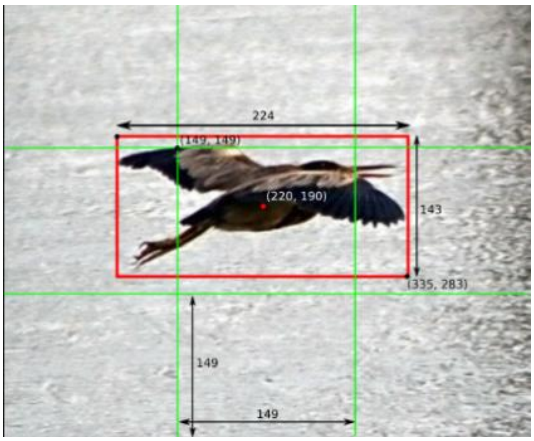

Figure 2: Example of how to coordinate parameters are calculated in a 448 X 448 image with S = 3
Source: https://hackernoon.com/understanding-yolo-f5a74bbc7967

Figure 2 depicts how the x coordinate of (220-149)/149 is normalized as 0.48. And y coordinate of (190-149)/149 is normalized as 0.28. The width (w) of 224 is calculated as 224/448 = 0.50 with respect to the entire image. And height (h) of 143 is calculated as 143/448 = 0.32 with respect to the entire image.

The confidence score predicts the IOU among the prediction box and the ground truth box along with these parameters. This confidence score reflects the presence or absence of an object of any class inside the bounding box. Along with these calculations, every grid cell having an object also estimates the conditional class probabilities, given as **Pr (Class(i) | Object)**. This probability is conditioned on the grid cell containing one object. Therefore, if no object is present on the grid cell, the loss function will not penalize it for a wrong class prediction. Since the network will only predict one set of class probabilities per cell regardless of the number of boxes (**B**), the total number of class probabilities could be taken as **S X S X C**. It is said that at the time of testing when confidence scores for each box is individually calculated, the conditional class probabilities and the individual box confidence predictions are multiplied as; **Pr (Class (i) | Object) X Pr (Object) X IOUtruthpred = Pr (Class(i)) X IOUtruthpred**. The confidence scores for each box reflect the class's possibility being shown inside the box and how exactly the Object fits the estimated box.

B. How YOLO Algorithm works?

YOLO algorithm is an algorithm based on regression. It predicts class probabilities of the object and bounding boxes specifying the object's location, for the entire image. The bounding boxes of the object are described as: bx, by, the x, y coordinates represent the center of the box relative to the bounds of the grid cell. The bw, bh as the width and height are predicted relative to the whole image and the value c is representing the class of the object. YOLO takes the image as input and divides it into S x S grids (3 x 3). Then, image classification and object localization techniques are applied to each grid of the image and each grid is given a label. The YOLO algorithm then checks every grid for an object and identifies its label and bounding boxes. The label of a grid that does not have an object is indicated as zero. Every labelled grid is defined as S.S having 8 values. The 8 values namely are pc, bx, by, bw, bh, c1, c2, c3. Pc shows if a particular grid has an object or not. If an object is available, the pc is assigned 1 else 0. bx, by, bh, bw are bounding box parameters of a grid and are only defined if a proper object is available in that grid. c1, c2, c3 are classes. If the object is a car, then the value of c1, c2, c3 are 0,1,0 respectively [11].

Figure 3: Example image with 3x3 grids.
Source: https://jespublication.com/upload/2020-110682.pdf

In the example grid, a proper object cannot be identified from the first grid. Therefore, pc value is 0 and bounding box parameters need not be assigned as there is no defined object. Class probability cannot be identified as there is no proper object (Figure 4). The 6th grid has a proper object and therefore pc value is assigned 1 and bounding boxes for the object are bx, by, bw and bh. Since the object is a car, the classes for the grid are 0,1,0 (Figure 5) [11].

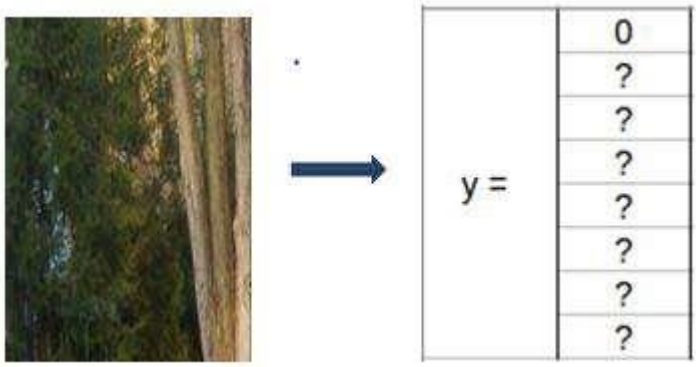

Figure 4:Bounding box and class values for grid 1
Source: https://jespublication.com/upload/2020-110682.pdf

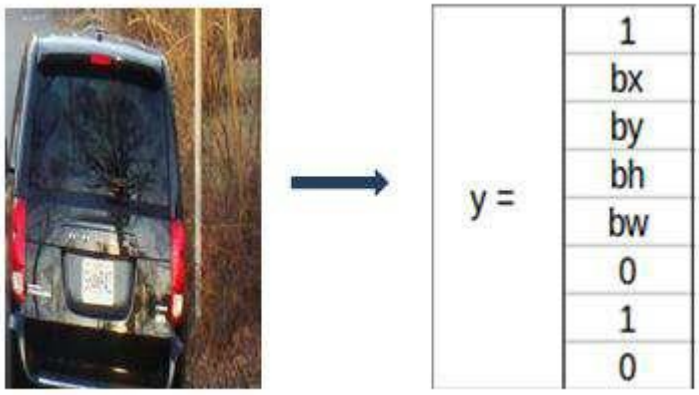

Figure 5:Bounding box and class values for grid 6
Source: https://jespublication.com/upload/2020-110682.pdf

The matrix is defined as S x S x 8, where S x S represents the entire grid size, image gets divided into and 8 indicates the total count of pc, bx, by, bw, bh, c1, c2, c3 values. Bounding boxes differ for each grid depending on the position of objects in the relative grid. If more than two grids have the same object, then the grid cell that has the center of the object is used to detect that object. For a precise identification of the object, two methods can be used; 1. Intersection over Union (IOU) 2. Non-Max Suppression [11]. In IOU, actual and estimated bounding box values are used and the IOU of both values are computed using the following formulae.

$$\text{IOU} = \frac{\text{Intersection Area}}{\text{Union Area}}$$

It is better if the computed IOU is greater than a threshold value (an assumed value for increasing the accuracy of the detected object.) 0.5 [11].

In Non-Max Suppression, the next method, high possibility boxes are used and the boxes with high IOU values are suppressed [11]. This process is followed many times until a box is considered as the bounding box for the object. Each grid cell also predicts 'c' conditional class probabilities for the object in that grid. These probabilities are conditioned on the grid cell containing an object. Only one set of class probabilities is predicted for a grid cell, regardless of the number of bounding boxes for that grid cell. [12]

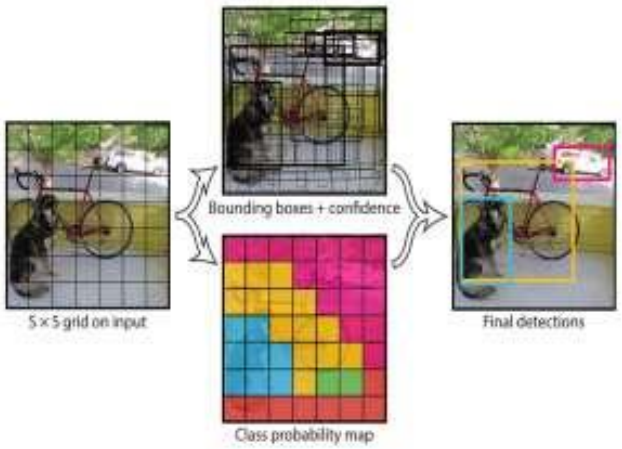

Figure 6: Complete process of Object detection by YOLO
Source: https://jespublication.com/upload/2020-110682.pdf

V. YOLO Versions

- YOLOv1

Base YOLO is also called YOLO Version 1[10]. It detects the object basing it as a regression problem. A single convolutional network predicts multiple bounding boxes and class probabilities for all the grid cells simultaneously. The input image is divided into S x S grids. If the center of a proper object falls into a grid cell, that grid cell will be considered in detecting that object. Each grid cell predicts N bounding boxes and confidence scores for the boxes and c class probabilities. At test time, the class probabilities and the individual box confidence predictions are multiplied resulting in class-specific confidence scores for each predicted box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object (Figure 6). These predictions are encoded as an $S \times S \times (N * 5 + c)$ tensor. The width(bw) and the height(bh) are predicted relative to the whole image. And that is why YOLOv1 uses Nx5 for calculating tensor. [10]

$$\textbf{Pr(Class i |Object)} * \textbf{Pr(Object)} * \textbf{IOU pred} = \textbf{Pr(Class i)} * \textbf{IOU pred.}$$

These confidence scores reflect how confident the model is in ensuring that the predicted box contains an object and how accurate the model thinks the box around the different objects is. Confidence score is defined as:

$$\textbf{Pr(Object) * IOUtruthpred}$$

YOLOv1's network has 24 convolutional layers as opposed to YOLOv2, which has 19 layers [10]. For evaluating YOLO model on the PASCAL VOC detection dataset, these values are used: S=7, therefore a 7x7 grid. N=2, number of bounding boxes. The PASCAL VOC dataset has 20 labelled classes so c=20. Therefore YOLOv1's final prediction is a 7x7x(5x2+20)=7x7x30 tensor. Here only 98 bounding boxes per image is used [10], [12]

- YOLOv2

YOLO Version 2 is an improved version of the existing YOLO algorithm. The speed of detection performance remains same while the mAP value increased compared to YOLOv1's Map value of 63.4. New multi-scale training method can be used to run the YOLOv2 run at various sizes offering improvements in accuracy and speed in prediction.

YOLOv2 adds a list of significant solutions to increase mAP. Batch Normalization preprocesses the input data. High Resolution Classifier from YOLOv1's 224x224 to 448x448 raises the mAP by 4%. Its neural network has 19 convolutional layers compared to the YOLOv1 which has 24. YOLOv2 adopts convolutional with anchor boxes and increases each grid cell's resolution from YOLOv1's 7x7 to 13x13. It also has only one bounding box for each grid cell. Finally, YOLOv2 adds a pass-through layer to get the extracted features from the former layer and combine them with the original final output features, so that the ability of detecting the small object would be enhanced. In this mean, YOLOv2 raises the mAP by 1% [10].

VI. Strengths and Weaknesses of YOLO

YOLO is the state-of-art real-time object detection algorithm that surpasses the previous CNN detection speed limits while maintaining a good balance between speed and accuracy. YOLOv2, the latest version of YOLO achieving a mean Average Precision (mAP) rate of 76.8 at 67 Frames per Second (FPS) and 78.6 mAP rate at 76 FPS, outperforms regional-based algorithms such as Faster R-CNN in both speed and accuracy. Another great strength in YOLO is its global reasoning skills that encode the contextual information about the whole image rather than a specific region. Along with these global reasoning skills, the ability to predict false positives in the background increases, improving the algorithm's reasoning skills as a whole. Lastly, with YOLO's

ability to learn basic representations of the labelled objects, it has outperformed other detection methods, including (Deformable Part Model) DPM and R-CNN when generalizing natural images amongst images like artwork. Due to YOLO's generalizability and applicability in new domains and unexpected outputs, it is considered one of the best object detection algorithms in the domain.

Although YOLO has many unique strengths, it also has weaknesses. One of the notable weaknesses of YOLO is its spatial constraints on bounding boxes. These spatial constraints are held due to each cell being able to predict only two boxes and one class. It limits the number of predictable objects nearby to each other in groups (such as the recognition of a flock of birds, a basket of similar fruits). Due to only being taught through input data, YOLO also has a weakness in generalizing objects in unusual or new aspect ratios. However, this could be considered as more of a general problem in the object detection domain. Since the model only uses relatively coarse features for prediction, the architecture has a few down sampling layers from the input images, which can be mentioned as a general weakness of YOLO. Compared with an ideal algorithm, YOLO's loss function (which approximates the detection performance) treats errors with the same loss despite its object boxes' size. This is not advantageous since a small error in a small box is not equivalent to a small error in a large box, which has a more significant effect on the Intersection over Union (IOU), giving out incorrect localizations in the end. Therefore, these could be taken as some of the areas YOLO algorithm could improve upon.

VII. Conclusion and future scope

This paper reviews the fundamental structure of CNN algorithms and an overview of YOLO's real-time object detection algorithm. CNN architecture models can remove highlights and discover objects in each given image. When properly used, CNN models can solve deformity identification, instructive/ learning application creation etc. When in comparison with other CNN algorithms, YOLO has many advantages in practice. Being a unified object detection model that is simple to construct and train in correspondence with its simple loss-function, YOLO can train the entire model in parallel. The second major version of YOLO, YOLOv2, provides the state-of-art, best tradeoff between speed and accuracy for object detection. YOLO is also better at generalizing Object representation compared with other object detection models and can be recommended for real-time object detection as the state-of-art algorithm in object detection. With these marks, it is acknowledgeable that the field of object detection has an expanding, great future ahead.

Acknowledgement

The authors would like to thank Dr Dharshana Kasthurirathna of the Faculty of Computing, SLIIT, to write this review paper as an assignment. The expertise of everyone who improved this study in numerous ways is also gratefully appreciated.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Lerning Applied to Document Recognition," *proc. IEEE*, 1998, [Online]. Available: http://ieeexplore.ieee.org/document/726791/#full-text-section.

[2] T. F. Gonzalez, "Handbook of approximation algorithms and metaheuristics," *Handb. Approx. Algorithms Metaheuristics*, pp. 1–1432, 2007, doi: 10.1201/9781420010749.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[4] C. Szegedy *et al.*, "Going Deeper with Convolutions," 2015, doi: 10.1002/jctb.4820.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," doi: 10.1002/chin.200650130.

[6] J. Hosang, R. Benenson, P. Dollar, and B. Schiele, "What Makes for Effective Detection Proposals?," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 4, pp. 814–830, 2016, doi: 10.1109/TPAMI.2015.2465908.

[7] A. S. R. H. A. J. S. S. Carlsson, "CNN Features off-the-shelf: an Astounding Baseline for Recognition," *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Work.*, vol. 7389, pp. 806–813, 2014, doi: 10.1117/12.827526.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," [Online]. Available: https://www.cv-foundation.org//openaccess/content_cvpr_workshops_2014/W15/papers/Razavian_CNN_Features_Off-the-Shelf_2014_CVPR_paper.pdf.

[9] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," *2017 IEEE 2nd Int. Conf. Big Data Anal. ICBDA 2017*, pp. 721–724, 2017, doi: 10.1109/ICBDA.2017.8078730.

[10] J. Du, "Understanding of Object Detection Based on CNN Family and YOLO," *J. Phys. Conf. Ser.*, vol. 1004, no. 1, 2018, doi: 10.1088/1742-6596/1004/1/012029.

[11] Mauricio Menegaz, "Understanding YOLO – Hacker Noon," *Hackernoon*. 2018.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2016, doi: 10.1109/CVPR.2016.91.