International Journal of Computer Trends and Technology
ISSN: 2231-2803 / https://doi.org/10.14445/22312803/IJCTT-V68I6P112.

Volume 68 Issue 6, 70-76, May 2020

Original Article

Yolo Real Time Object Detection

Sujata Chaudhari[1], Nisha Malkan[2], Ayesha Momin[3], Mohan Bonde[4]

[1,2,3,4] *Final Year B. Tech IT Student, Assistant Professor, Information Technology,UMIT, SNDT Women's University, Juhu Tara Road Santacruz (W) Mumbai Maharashtra India.*

Received Date: 06 May 2020
Revised Date: 18 June 2020
Accepted Date: 19 June 2020

Abstract *- Now a day's, object detection has been the main topic in computer vision systems. With the knowledge of deep learning techniques, the accuracy of object detection has been improved. The project aims to include a modern technique for object detection with the goal of achieving high accuracy with real-time performance. An object detection system is dependent on other computer vision techniques for helping the deep learning-based approach, which leads to slow and non-desirable performance is a big challenge. In this project, we use the complete techniques of deep learning to solve the problem of object detection in an end-to-end fashion. The network is trained on the available dataset (PASCAL VOC), on which an object detection challenge is conducted annually. The following system is fast and accurate, thus helping those applications which require object detection. We present a new approach to object detection is YOLO Real Time. Earlier research on object detection repurposed classifiers to perform detection.*

Keywords *- YOLO, VOC*

I. INTRODUCTION

Humans are able to perceive the three-dimensional structure of the real world around them with apparent ease. Think of how vivid the three-dimensional precept is when you look at a book sitting on the table next to you. You can tell the shape colour of the book. Similarly, looking at a framed group pictured, you can easily count the number of people. After the concepts of deep learning, computer vision has proven to be complete for this purpose.

YOLO (You Only Look Once) is a method or way to do object detection. It is an algorithm to detect the objects present in the image. As the name suggests, it looks at the entire image only once, goes through the neural network once and detects objects.

YOLO architecture is more like FCNN (fully convolutional neural network) and passes the image (nxn) once through the F-CNN, and output is (mxm) predicted. We frame object detection as a regression problem to dimensionally separated bounding boxes and class probabilities. In one evaluation, a single neural network predicts bounding boxes and class probabilities directly from full images. Since we built a simple Convolution Neural Network for the same. In a CNN model, a number of essential features are extracted as the model is trained over the datasets. It extracts one Feature per layer and combines them at the end, which forms the fully connected layer. The training of the model occurs in this manner, and multiple test cases are implemented in the Tensor Flow environment in order to obtain accurate results. The important task in order to make computers able to understand or interact with their surroundings is to automatically recognize and locate the objects in images and videos.

Solving the problem of object detection with all the challenges has been identified as a main predecessor to solving the problem of semantic understanding of the surrounding environment. A large number of academics, as well as industry researchers, have already shown their interest in it focusing on applications, such as autonomous driving, surveillance, relief and rescue operations, deploying robots in factories, pedestrian and face detection, brand recognition, visual effects in images, digitizing texts, understanding a serial image, etc. which have object detection as the main challenge. One of the main challenges object detection must deal with is the Semantic Gap, denied by Smolders as the lack of coincidence between the information once can be extracted from visual data and its explanations by a user in given circumstances. There is, in fact, a difference of nature between raw pixel intensities contained in images and semantic information representing objects. Object detection is a natural addition to the classification problem.

II. RELATED WORK

Thomas Dean Mark A. Ruzon Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, Jay Yagnik (2013). They replace a linear convolution with an ordinal convolution by using an efficient LSH scheme. Through extensive empirical tests on DPM detectors, we have shown that

- The system performs comparably to the original DPM detectors.
- Performance degrades gracefully as the number of object classes increases.
- Up to 100,000 object classes can be simultaneously detected on a single machine in under 20 seconds.

Pablo Arbel aez1, Bharath Hariharan1, Chunhui Gu1,2, Saurabh Gupta1, Lubomir Bourdev1, 3, and Jitendra Malik1 IEEE (2012). They construct regions by multi-scale low-level hierarchical segmentation, obtaining high-quality object candidates in a simple and generic way without any midlevel information or learning. They propose a multi-class high-level region representation that integrates scanning window part detectors and global appearance cues. They propose a novel design for region-based object detectors based on class-specific region scoring, followed by pixel classification.

Ross Girshick Je- Donahue Trevor Darrell Jitendra Malik UC Berkeley (2016) Object detection performance, as measured on the canonical PASCAL VOC dataset, has been planned in the last few years. They pro- pose a simple and scalable detection algorithm that improves mean average precision (mAP) by more than 30 percent relative to the previous best result on VOC 2012, achieving an mAP of 53.3percent. Their approach combines two key insights:

- One can apply high-capacity convolutional neural networks (CNNs) to bottom-up region proposals in order to localize and segment objects.
- when labelled training data is scarce, supervised pre-training for an auxiliary task, followed by domain-specific fine-tuning, yields a significant performance boost. They combine region proposals with CNNs, which we call our method R- CNN: Regions with CNN features.

Chandrajit et al. (2016) suggested a feature-based method towards tracking the multiple moving objects in a surveillance video sequence is proposed. This proposed method is assessed quantitatively using the precision and recall accuracy metrics. Further, 756 Mukesh Tiwari and Dr Rakesh Singhai comparative evaluation with related works has been carried out to exhibit the efficacy of the proposed method.

Wei Liu1, Dragomir Anguelov2, Dumitru Erhan3, Chris- tian Szegedy3, Scott Reed4, Cheng-Yang Fu1, Alexan- der C. Berg1 (ECCV 2016). The proposed method for detecting objects in images uses a single deep neural network. Their approach, named SSD, discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. At prediction time, the network generates scores for the presence of each object category in each default box and produces adjustments to the box to better match the object shape.

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi (2018) work on object detection repurposes classifiers to perform detection. Instead, they frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network can be optimized end-to-end directly on detection performance. Our unified architecture is extremely fast.

III. OBJECT DETECTION

Object detection is a Machine Vision problem that deals with identifying and locating the object of different classes in an image. The interpretation of object localization can be made in many ways that create a bounding box around the object or segmentation. Segmentation means considering every pixel present in an image that contains the object.

Fig. 1 Object Detection Using Bounding Boxes

Here, the objects are surrounded by the rectangular boxes known as bounding boxes and labelled with the name of the objects, which are detected after training the dataset.

Fig. 2 Object Segmentation by Predicting Pixel-Level Marks

Image Segmentation in which a digital image is separated into multiple parts. This technique is used to get the information of the image at a pixel level and at a more granular level. It allows us to divide an image into meaningful parts. In this segmentation, each pixel level belongs to a particular class. In the above fig 2, classes were person, tree, fruits, etc. Any pixel belonging to any person is assigned to the same "person" class.

A. YOLO Object Detection

You Only Look Once is a real-time object detection technique. This algorithm avoids spending too much time generating region proposals.

Fig. 3 YOLO in Action

This is the image of the crowded area of the street. The classes are traffic light, person, handbag, truck, bus, car, etc. By using YOLO Object Detection Technique, we can classify, detect and localize each object present in an image. Here bounding boxes and labels of the objects are shown.

a) What are Anchor Boxes?

YOLO can work for multiple objects where they are associated with one grid cell. The grid cell is the authority for disclosing objects in the region of the image.

Detection of objects simply means predicting the class and location of an object within that region. We allow one grid cell to detect multiple objects by using anchor boxes, which actually contains the centre points of two different objects present in the case of overlap.

[Fig 4] shows the overlapping image is seen here that contains a car and a person. Since we can only have one class as an output vector of each grid cell, either the person or the car. But Anchor boxes tried to detect objects that fit into a box with the aspect ratio. It is also one of the important parameters that can tune for enhanced performance in the dataset. If anchor boxes are not tuned correctly or properly, the neural network will not know that certain little, bit or non-uniform objects exist and will not have a chance to detect them. Luckily, there are some easy steps that can be taken to make sure that we do not fall into this trap.

Fig. 4 Anchor Boxes in Action

b) What is Intersection Over Union Threshold?

Primary Bounding Box means the bounding box for an object is hand labelled and Predicted Bounding Box means the Deep Learning models predict a box around the object in the image. IoU is the ratio of Area of Intersection and Area of Union. For good prediction, IoU should be greater than 0.5. **IoU=Area of Intersection/Area of Union.**

[Fig 5] Here, we can see two coloured bounding boxes. The green colour bounding box shows Primary Boundary Box, which means we draw the rectangular box before training the data- set for detection that specifies where is the object present in an image. The red colour bounding box is the Predicted Boundary Box which means neural network models predicted the object after training the dataset.

[Fig 6] Area of Intersection means an area of overlap be- tween the primary boundary box and the predicted boundary box. Area of Union means the area enclosed by both primary boundary box and predicted boundary box. By dividing the area of intersection by area of the union gives us the Intersection over Union.

[Fig 7] In the above figure, we have included examples of poor, good and excellent Intersection over Union scores for Poor prediction IoU is 0.4034, Good prediction IoU is 0.7330 and Excellent prediction IoU is 0.9264. As we can see, predicted boundary boxes that heavily overlap with the primary boundary box have the highest IoU than those with the lowest IoU. An IoU score should be greater than 0.5 for good prediction.

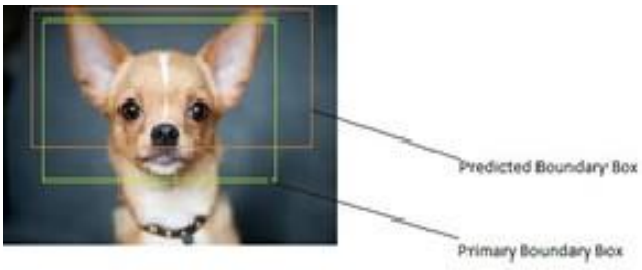

Fig. 5 Boundary Box Representation

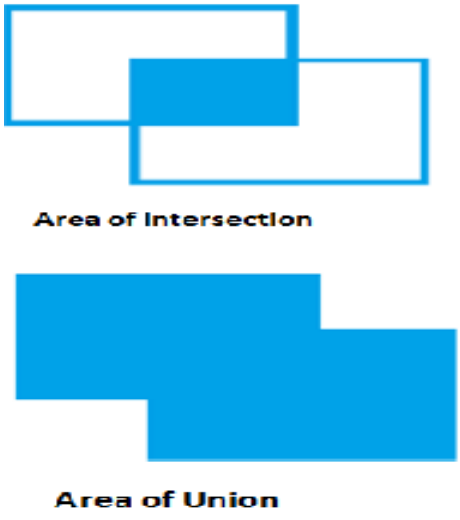

Fig. 6 Area of Intersection and Union

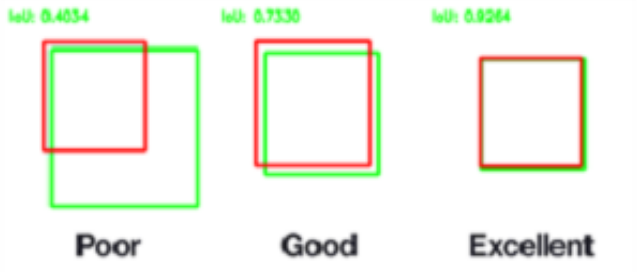

Fig. 7 An example of IoU

IV. NETWORK DESIGN

The Convolutional Neural Network is implemented and uses the Pascal VOC detection dataset to access the neural network. The starting layers of the convolution

network pull out the property from the given input, and the fully connected layer predicts the probabilities of the output and its coordinates. The Architecture is encouraged by the Google net model for object detection.

The architecture has 24 convolution layers and 2 fully connected layers. To predict the boundaries of the object, we train a fast version of YOLO. The full network is shown in Fig 8.

A. Network Design

Neural Network used for various classification problems like image, word, audio, video etc. Various type of networks is used for different purpose. For example, in Predicting the sequence of words, we use Recurrent Neural Networks more; for image classification, we use Convolution Neural networks. Before discussing the CNN (Convolution Neural Network), Let us discuss the concept of Neural Network. Neural Network has three types of layers:

a) Input Layer (First layer)

In this layer, we give the input to our model. The number of neurons in his neural network is equal to the total number of features or pixels in our data.

b) Hidden Layer (Middle Layer)

The input from the input layer is given to the middle layer. There can be one or more hidden layers that can have a number of neurons that are generally greater than the features. The output of each layer is computed by matrix multiplication of output of the previous layer with the weight of that layer and then by the addition of biases followed by the activation function, which make the network non-linear.

c) Output Layer

The output of the hidden layer is given to the logistic function like sigmoid, which convert the output of each class into the probability score of each class. The data is then given to the model, and the output of each layer is obtained; this step is called feedforward, then we calculate the error using the error function.

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrices into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout. A convolution Network is capable of capturing the spatial and temporal dependencies in the image using the filters. The architecture of the convolution neural network shows the best fitting to the object dataset due to the reduction of the number of parameters used and the re-usability of weights. A convolution neural network is a neural network that shares its parameters. It can be ex- plain using cuboids having dimensions like length, width and height. The image generally has red, green, and blue channels. The block diagram is shown in Fig 9.

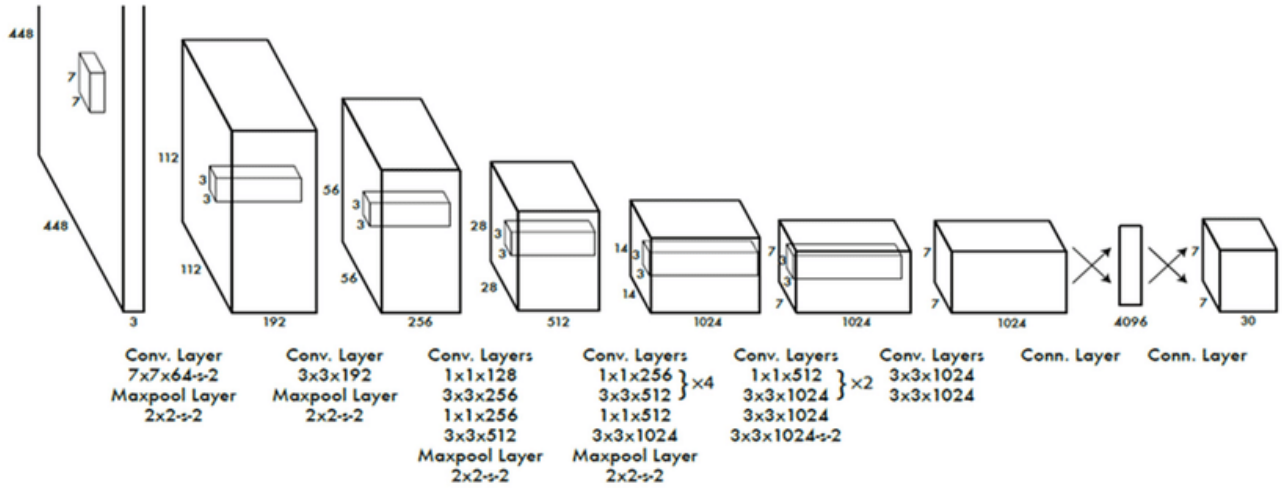

Fig. 8 Architecture of YOLO

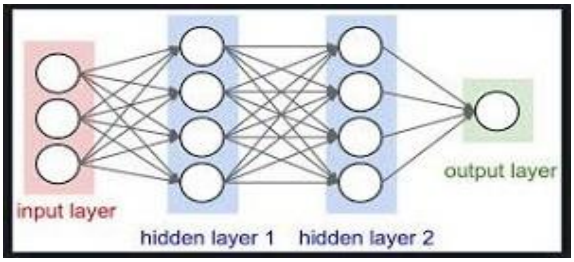

Fig. 9 Block Diagram of Neural Network

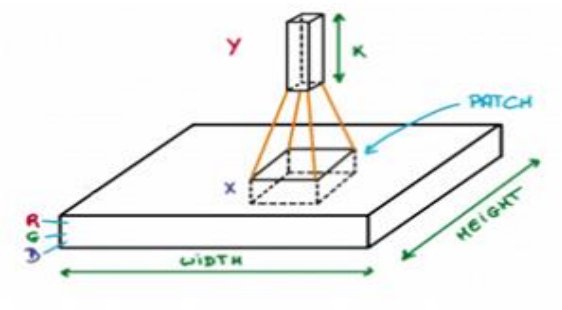

Fig. 10 Image Showing Patch

Fig 10 shows a small patch of the image, and running a neural network on it, we have k output and represent them vertically. Now move the neural network slightly across the entire image. As an output, we will get another image with a different height, width, depth. Instead of having just R, G, B channels, we have more channels but less width, height and depth. This operation is called convolution. If the size of the patch is the same as that of the image size, then it will be a regular neural network. 6cm.

B. Convolution

Convolution layers consist of a number of filters. Each filter has the same height, width and depth as that of the input layers. For example, if we have the image and have to run the convolution, an image has dimension 24x24x2. The size of filters can be axax3, where 'a' can be 3, 4, 5, etc., but the size of the filter should be small as compared to the image dimension. During the further step, we pass the filter across the input volume line by line, and we find the product between the weight of the input image and the patch from input volumes. We pass the filters to the entire image we get the 2D output for each filter. We will stack them with each other, so as a result, we will have the depth equal to the number of filters. All the filters are being learned by the network. A convolution network is built using different layers or sequences of layers, and every layer transforms one volume into another through a differentiable function. Types of layers: Example: Take an image of dimension 23*23*3 to execute it on a convolution neural network

a) *Input layer*

This is the first layer in which we give the raw input of an image. In his example, height, weight and depth are 23, 23 and 3, respectively.

b) *Convolution Layer*

This layer is doing a dot product between filters and input image and give the output. For example, we are using 10 filters for this layer, so we will get an output is 23*23*10.

c) *Activation Function Layer*

In this Layer, the output of a convolution is passing through the activation function section wise. The activation function is activated or not by adding bias or weight to it. If it is correct, then we are getting the result of the activation function as 23*23*10, which will remain the same as the convolution layer.

d) *Pool Layer*

In this layer, we reduce the size of an image which make a decision fast and also prevents overfitting. There are two types of polling 1) Max Pooling 2) Average polling. Max pooling is a sample-based discretionary process. The objective is to downsample an input representation, reducing its conditionally and allowing for assumptions to be made about features contained in the sub-regions binned. This is done in part to help overfitting by providing an abstracted form of representation. As well, it reduces the computational cost by reducing the number of parameters. [Fig 11]

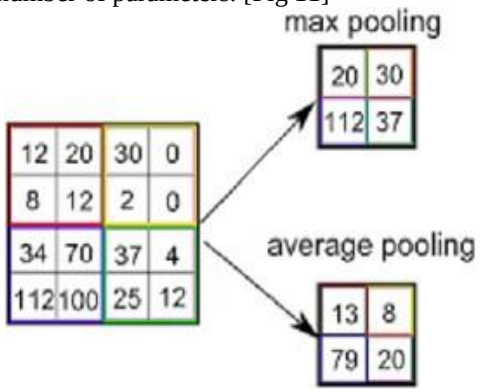

Fig. 11 Pooling Layer

e) Fully Connected Layer

This layer takes the output of the max-pooling layer as an input and gives the output as a 1D array of size equal to the number of classes. [Fig 12] shows the fully-connected layer function.

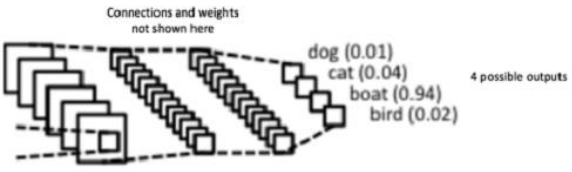

Fig. 12 Fully Connected Layer

C. Convolution Layer – The Kernel

The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the colour yellow. We have selected K as a 3x3x1matrix. Kernel/Filter, K= [1 0 1, 0 1 0, 1 0 1]. The Kernel shifts 9

times because of Stride length = 1, every time performing a matrix multiplication operation between K and the portion P of the image over which the Kernel is hovering. The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed

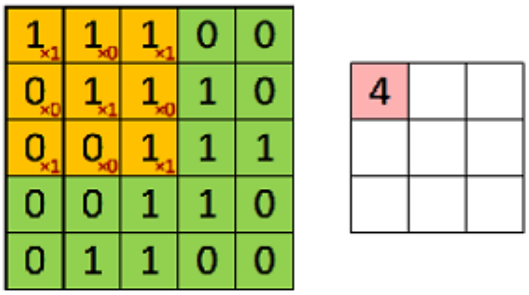

Fig. 13 Convoluting a 5x5x1 Image with a 3x3x1 Kernel to get a 3X3X1 Convolved Feature

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between K and I in the stack ([K1, I1]; [K2, I2]; [K3, I3]), and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.

V. IMPLEMENTATION

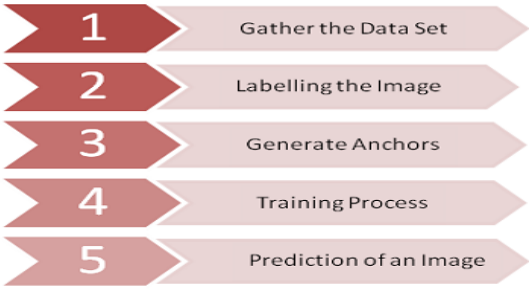

Fig. 14 Implementation Process Flow

A. *Gather the Data*

We collected images for object detection from various online sources. For training, we are considering 23 classes like a cat, dog, bag, bottle, sofa, bike, aeroplane, car, chair, flower, fruit, kangaroo, raccoon, knife, mobile, telephone, table, person and many more.

B. *Labelling the Image*

We labelled every image by creating the boundary box or rectangle box around the object in the image. We used a software called labelling. When we save these images, it is stored as an XML file. **Steps:**

- Download the LabelImg software from the internet.
- Open the software and select the folder for labelling.

C. *Generating Anchors*

We generate the anchors for the training set by using the following command on the command prompt:

python gen-anchors.py -c config.json

D. *Training the Data Set*

We trained the datasets by using thefollowing command:

python train.py -c config.json

At the end of this process, the output will write the weights of the best model to full-yolo-cat.h5. As soon as the validation set is not improved in 3 consecutive epochs, the training process is stopped.

E. *Result*

We get the output of the trained weights by using the following command:

python predict.py -c config.json -w full-yolo-cat.h5 -i "C: Nisha Malkan keras-yolo2-master-image43.jpg"

It gives the image with a detected bounding box around the object with its label.

```
Loading pre-trained weights in full_yolo_bed.h5
Epoch 1/103 - 61003s - loss: 10.0293 - val_loss: 10.0508
Epoch 00001: val_loss improved from inf to 10.05075, saving model to full_yolo_bed
Epoch 2/103 - 84635s - loss: 10.0514 - val_loss: 10.0446
Epoch 00002: val_loss improved from 10.05075 to 10.04456, saving model to
full_yolo_bed.h5
Epoch 3/103 - 36942s - loss: 10.0265 - val_loss: 10.0458
Epoch 00003: val_loss did not improve from 10.04456
Epoch 4/103 - 67493s - loss: 0.8447 - val_loss: 0.8727
Epoch 00004: val_loss improved from 10.04456 to 0.87268, saving model to
full_yolo_bed.h5
Epoch 5/103 - 27184s - loss: 0.3108 - val_loss: 0.9325
Epoch 00005: val_loss did not improve from 0.87268
Epoch 6/103 - 27249s - loss: 0.2428 - val_loss: 0.8887
Epoch 00006: val_loss did not improve from 0.87268
Epoch 7/103 - 27301s - loss: 0.2174 - val_loss: 0.8601
Epoch 00007: val_loss improved from 0.87268 to 0.86008, saving model to full_yolo
Epoch 8/103 - 27264s - loss: 0.1999 - val_loss: 0.8509
Epoch 00008: val_loss improved from 0.86008 to 0.85086, saving model to full_yolo
Epoch 9/103 - 60071s - loss: 0.1890 - val_loss: 0.9079
Epoch 00009: val_loss did not improve from 0.85086
Epoch 10/103 - 27781s - loss: 0.1629 - val_loss: 0.8781
Epoch 00010: val_loss did not improve from 0.85086
Epoch 00010: early stopping
```

Fig. 15 Iterations

After training the models, we get an output that shows the class name and probability of an image. It also shows the loss function after completion of the training process.

The learning curves shows how accuracy and loss vary during training. Accuracy is a measure of how accurate our image prediction is compared to original

data. Loss is an error made in training or validating data. As we train the model, again and again, the loss will be decreased. During each training, increased the epoch value to well train the model.

Classes	Training 1	Training 2	Training 3	Training 4	Training 5	Training 6	Training 7	Training 8	Training 9
Airplane	0	0.833	1	1	1	1	1	0	1
Bag	1	1	1	1	1	1	1	1	0.8889
Bed	1	0.96	1	1	1	1	1	1	1
Bike	1	1	1	1	1	1	1	1	1
Book	0	0	0	0.75	0.52	0.45	1	1	0.2465
Bus	1	0.78	1	0.8	1	1	1	1	1
Car	0.49	0.33	0.9	0.62	1	1	0.92	0.933	1
Cat	0.86	0.75	0.889	0.78	0.94	0.94	0.96	1	0.9924
Chair	0.28	0.5536	0.875	0.88	0.94	1	1	0.97	0.8516
Dog	0.9058	0.4881	0.973	0.5556	0.9264	0.9413	0.9736	0.9769	0.7857
Flower	0	0.16	0.125	0.37	0.33	0.6	0.97	0.83	0.1429
Fruit	1	1	1	0.5	1	1	1	1	0.8
Kangaroo	1	0.33	0	0.733	0	1	0.76	1	0.8571
Kitchen	0	0.75	0	0.33	0.8	1	0	0.86	1
Mobile	1	0.3	1	0.44	1	1	1	1	0.9167
Person	0	0.8	0.5833	0.73	0.5	0.9	0.33	1	0.8218
Raccoon	0.25	0.5	1	0.75	0	1	1	1	1
Saucepan	0	0.5	1	0.87	1	0	1	1	1
Sofa	0	0.76	1	0.833	1	0	1	1	0.9762
Table	1	0.77	1	1	0	0.8	1	1	0.8333
Telephone	1	1	1	1	1	1	1	1	1
Tree	0	1	0	0.6667	0	0.33	0.432	0.48	0.5143
Truck	0	0.02	0.125	0.66	0.33	0.071	0.4	1	0.7143
mAP	**0.4906**	**0.6276**	**0.7843**	**0.7334**	**0.7475**	**0.8218**	**0.8249**	**0.8827**	**0.8409**

Fig. 16 Comparison of Different Training Results

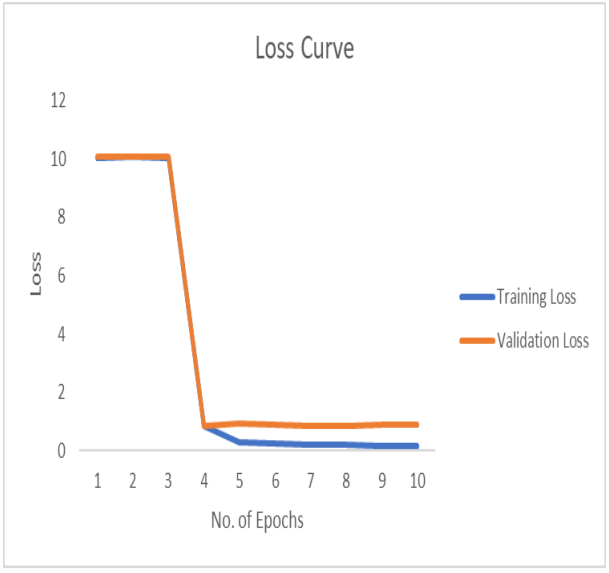

Fig. 17 Graph Showing Training and Validation Loss

We trained different classes after each training, and the prediction rate is different from that shown in the table.

VI. CONCLUSION AND FUTURE SCOPE

This project presents a comprehensive review of convolutional neural network architecture. For the applications in the computer vision domain, the report mainly explains how the advancements of CNN based schemes have made it most suitable for images. During each training, the validation loss and training loss decrease and mAP increases. Some images are not identified during 1 and 2 training, but in further training, that image is trained and shows better results. The accuracy of an image during train 1 to train 9 continuously increases. In the future, one can work on various other aspects of this project like.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (2016).

[2] L. Han, Object Detection Module Based on Implementation of Java and OpenCV, Journal of Computer Applications, 28(3) (2008) 773-775.

[3] P.Hemalatha, C.K.Hemantha Lakshmi and Dr.S.A.K.Jilani, Real time Image Processing based Robotic Arm Control Standalone System using Raspberry pi SSRG International Journal of Electronics and Communication Engineering 2(8) (2015).

[4] Figure 2f from Irimia R, Gottschling M Taxonomic Revision of Rochefor- tia Sw. (Ehretiaceae, Boraginales). Data Journal 4(2016) e7720. https://doi.org/10.3897/BDJ.4.e7720.

[5] GitHub-experiencor/Keras-yolo2: Easy Training on Custom dataset. Various Backends (MobileNet and SqueezeNet) Supported. A YOLO Demo to Detect Raccoon run Entirely in Browser is Accessible at https://git.io/vF7vI (not on windows)Retrieved from https://github.com/experiencor/keras-yolo2 YOLO Object Detection with OpenCV and Python. Retrieved from https://www.arunponnusa-object-detection-opencv-python.html

[6] Sagar Badgujar, Amol Mahalpure, Priyanka Satam, Dipalee Thakar, Prof. Swati jaiswal, Real time number plate recognition and tracking vehicle system SSRG International Journal of Computer Science and Engineering 2(12) (2015).

[7] Retrieved from https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithmsBiodiversity-36d53571365e

[8] Retrieved from https://heartbeat.fritz.ai/gentle-guide-on-how-yolo-object-localization-works-with-keras-part-2-65fe59ac12d