# BITS F464 MACHINE LEARNING

## OBJECT DETECTION USING YOLO

Group Number: ML 20

Team Members:

Jagath Kaparthi - 2019A4PS0547H

Vaibhav Mishra - 2019A3PS1350H

Avinash Gondela - 2019A8PS1357H

Akhilesh Senapati - 2019AAPS1352H

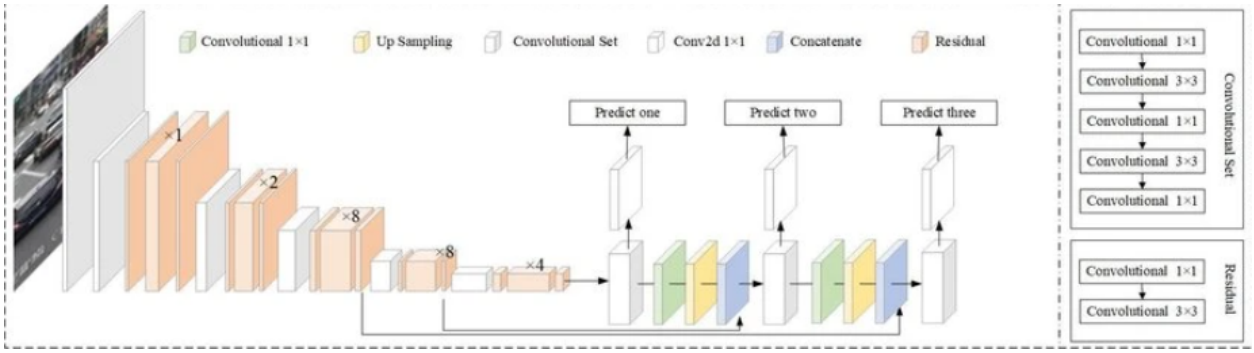
P V Sri Harsha - 2019A2PS1521H


April 2022

# Introduction

A convolution neural network is used in Yolo for object detection in our day-to-day real-time problems. Input images are processed as arrays of data and identify the common traits between them with the help of a convolutional neural network, a classifier-based system. Yolo remains advantageous with respect to other methods available since it provides the same accuracy with faster computation speed.

This model looks at the image at the global context level in order to make the prediction. In general, all neural network algorithms make a score-based region on the basis of which they make their prediction Yolo does the same by marking them into the specific classes made during the training of the data set. Here the high scoring regions are associated with a specific class.




## Working of Yolo:

Once the image has been divided into a specific grid each grid is made to compare with the predefined classes present with the help of a confidence interval set for each class. After which we perform the clustering of these grids and finally make the decision of assigning it to a specific class which helps us to make the predictions.

Various advantages of using Yolo over other algorithms:

1) Speed : This algorithm improves the speed of detection because it can predict objects in real-time.

2) High accuracy**:** YOLO is a predictive technique that provides accurate results with minimal background errors.

3) Learning capabilities**:** The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

## Dataset Used:

The dataset used for training consists of images of cars with their license plates. The main objective is to detect the license plates of these vehicles. We have taken extracted images from Amazon Web services using aws command line entered in command prompt. We have divided the dataset into a training and validation set. For testing, we took random images from google and tested the model and obtained results which were satisfactory. The training dataset consists of 1500 images and the validation set consists of 300 images (20% of training dataset size).

## Technologies Used:

We ran the model using Google Colab and used Python for the same. The model was trained using Google Colab's inbuilt GPU.

## Plan of Action:

After reading the paper, we initially implemented the paper and obtained the results after a few hours of training. Then we went on to improve the model and got two improvements where we further improved the model's accuracy. Details are given below.

# RUN 1 (Paper Implementation) :

At first we implemented the code based on the paper where we took the same type of layers and number as mentioned in the paper. After running the code we got the following result within approximately 6 hours. The results are as follows:-

**Mean average** precision (mAP)=51.41%

F1 score(**combines the precision and recall of a classifier into a single metric by taking their harmonic mean**.)=0.57

In the second image we can see that the confidence score of the block belonging to the license plate class is 0.38 approx

Results obtained after implementation -

```
 calculation mAP (mean average precision)...
 Detection layer: 82 - type = 28
 Detection layer: 94 - type = 28
 Detection layer: 106 - type = 28
300
 detections_count = 3226, unique_truth_count = 392
class_id = 0, name = license_plate, ap = 51.41%          (TP = 184, FP = 69)

 for conf_thresh = 0.25, precision = 0.73, recall = 0.47, F1-score = 0.57
 for conf_thresh = 0.25, TP = 184, FP = 69, FN = 208, average IoU = 51.26 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.514119, or 51.41 %
Total Detection Time: 7 Seconds
```

# RUN 2:

After the implementation of the paper, for the purpose of achieving better results we have introduced a few max pooling layers, which helps in pooling features and produces an output of fixed length. The size of the kernel and stride of these layers can be found below, route is nothing but a concatenation layer.

```
[maxpool]
stride=1
size=5

[route]
layers=-2

[maxpool]
stride=1
size=9

[route]
layers=-4

[maxpool]
stride=1
size=13

[route]
layers=-1,-3,-5,-6
```

Also we have increased the input resolution size(from 416*416 to 544*544) that will be passed into the model which clearly helps in improving accuracy but comes with a disadvantage of slower training. So after training our model with above changes we got slightly better results which can be seen below,

**Mean average** precision (mAP) = 54.59%

F1 score = 0.59

```
 calculation mAP (mean average precision)...
 Detection layer: 89 - type = 28
 Detection layer: 101 - type = 28
 Detection layer: 113 - type = 28
300
 detections_count = 2630, unique_truth_count = 392
class_id = 0, name = license_plate, ap = 54.59%          (TP = 186, FP = 51)

 for conf_thresh = 0.25, precision = 0.78, recall = 0.47, F1-score = 0.59
 for conf_thresh = 0.25, TP = 186, FP = 51, FN = 206, average IoU = 54.33 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.545923, or 54.59 %
Total Detection Time: 42 Seconds
```

Result after testing the trained model on a sample image from the internet gave a confidence score of 0.57.

# RUN 3:

After the above two implementations, we got motivated after getting the results before and after the improvement and thought that we could improve it further. We have added a large amount of convolutional layers with stride and padding and also adjusted the weights and dimensions after an image comes out of each layer. By doing that, we were able to get better accuracy due to the increase in the number of layers. The training time took approximately 6 hours. At the end of 6 hours, the results obtained were better than the previous two runs.

Results:
Average Loss obtained: 0.37196

Mean Average Precision obtained is 68.75%

```
 calculation mAP (mean average precision)...
 Detection layer: 139 - type = 28
 Detection layer: 150 - type = 28
 Detection layer: 161 - type = 28
3
 detections_count = 12, unique_truth_count = 4
class_id = 0, name = license_plate, ap = 68.75%          (TP = 3, FP = 1)

 for conf_thresh = 0.25, precision = 0.75, recall = 0.75, F1-score = 0.75
 for conf_thresh = 0.25, TP = 3, FP = 1, FN = 1, average IoU = 51.72 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.687500, or 68.75 %
Total Detection Time: 1 Seconds
```

After that we took a sample image from the internet and tested our detection.

# Comparisons And Conclusions

Difference of run 1 and run 2 is that there are very few max pooling layers in the convolutional neural network of run 1 and 2 with larger strides and minimal padding (which helps in speeding up the process but comes at the cost of reduced accuracy). Maximum pooling layers are layers in CNN which helps in reducing the dimensionality of the image (here dimensionality refers to the spatial size) by selection of a maximum cell value from a grid of predefined size from convolved feature map to max pooled feature map. In the run two and three all the strides (Here stride refers to the steps of the movement of the filter over the video or image to make convolved feature map, the size of the convolved Nth dimensional object is inversely proportional to the $K^{th}$ power of stride where k combined value of strides over different dimensions) are kept as one to keep as much information as possible in the convolved image.

Now the major differentiating factor that has been introduced in the run 3 on top of all run is that two features are the introduction of more convolution layers with strides and padding kept optimal so as to produce more information while convolving. By adding these features and training the data set for the same time as the other two runs we have observed a better accuracy.

In run two and three the dimensions of the image coming of the CNN filters is increased to 544*544 instead of the usual 416*416 (note that the dimension of the image must be multiple of 32 along all the axes).

Conclusions

| RUN | Mean Average Precision | F-score |
|---|---|---|
| 1 | 51.41% | 0.57 |
| 2 | 54.58% | 0.59 |
| 3 | 68.75% | 0.75 |

# Contributions of Each Team Member

| Names | Contribution |
|---|---|
| Avinash Gondela | Implemented run 3 and Improvement |
| Jagath Kaparthi | Implemented run 2 and Improvement |
| PV Sri Harsha | Literature Survey and Improvement |
| Vaibhav Mishra | Comparison between models and Improvement |
| Akhilesh Senapati | Implemented run 1 and Improvement |