YOLOv4: Research Review

[Original Paper: https://arxiv.org/pdf/2004.10934.pdf]

Abstract

You Only Look Once (YOLO) is popular and industry standard framework for object classification and detection in real-time. With the recent development of YOLOv4, it made it more powerful tool. The newer version is faster and accurate for object detection. In YOLOv4, the authors have used new features such as Cross mini-Batch Normalization (CmBN), Weighted-Residual-Connections (WRC), Self-adversarial-training (SAT), Mish activation, Mosaic data augmentation, Cross-Stage-Partial-connections (CSP), CmBN, DropBlock regularization and CIoU loss, and incorporates some of them to reach state-of-the-art results. The study provides evidence that YOLOv4 is faster and more accurate than all available alternative detectors.

In this paper, first I will be briefly introducing YOLO in general and YOLOv4. It is made of up many components so I will be explaining each component of its architecture. The authors have conducted experiments which I will be summarizing and comparing the results with other state of art models and with its own benchmarks. In addition, I have conducted my experiment to test the performance and accuracy. Finally, I will also advocate some of the techniques that can be used to improve accuracy and make it more efficient.

Acronyms and Abbreviations

Abbreviation	Explanation
ASPP	Atrous Spatial Pyramid Pooling
BoF	Bag of Freebies
BFLOP	Billion Floating Point Operations
CGBN	Cross-GPU Batch Normalization
CIoU	Complete Intersection over Union
CmBN	Cross mini-Batch Normalization
CNN	Convolutional Neural Network
CSP	Cross-Stage-Partial-connections
DIoU-NMS	Distance Intersection over Union – Non Maximum Suppression
FPN	Feature Pyramid Network
GPU	Graphical Processing Unit
IoU	Intersection Over Union
MiWRC	Multi-input Weighted Residual Connections
MS COCO	Microsoft Common Objects in Context
NAS_FPN	Neural Architecture Search - Feature Pyramid Network

PAN	Path Aggregation Network
PANet	Path Aggregation Network
R-CNN	Region-based CNN
RFB	Receptive Field Block
RNN	Recurrent Neural Network
SAT	Self Adversarial Training
SAM	Spatial Attention Module
SPP	Spatial Pyramid Pooling
WRC	Weighted-Residual-Connections
YOLO	You Only Look Once
YOLOv4	You Only Look Once version 4

Introduction

You Only Look Once (YOLO), is an object detection model that is the new state of art in object detection, based on Convolutional Neural Network (CNN) architecture. The goal of YOLO is to create grid of many cells by splitting the image and then for each cell predict the probability of having an object using bounding boxes. It outputs the coordinates of bounding boxes and probability classes. Mainly two types of object detection model are there in deep learning two-stage detectors and one stage detectors (Cohen, 2020). YOLO belongs to the group of one-stage detectors. The main idea of such group is to only look at the image once.

YOLOv4 is recently introduced with “Optimal Speed and Accuracy of Object Detection” which is faster and accurate than YOLOv3. The older version has been very popular among the industries for object detection as it is quicker than any other frameworks that are currently available. However, with the development of YOLOv4, it has improved again with accuracy and speed. The incredibly fast speed is the biggest advantage of using YOLO as it can process 45 frames per second (Sharma, 2018).

It is designed to provide production systems with fast operating speed of an object detector and parallel computations optimization rather than the low computation volume theoretical indicator (BFLOP) (Bochkovskiy, et al., 2020). The authors also believe that this framework can be easily trained and used for someone using a conventional GPU to train and test the model can achieve real-time, high quality and convincing object detection results.

Architecture

The original YOLO (Redmon, et al., 2016) was simple and single neural network which predicts the bounding boxes and class probabilities directly from the full images. Due to this simplicity, it can be optimized. Its architecture was inspired by the GoogLeNet mofel for image classification.

The YOLO architecture had 24 convolutional layers followed by 2 fully connected layers (Redmon, et al., 2016). Instead of the inception modules used by GoogLeNet, they used 1x1 reduction layers followed by 3x3 convolutional layers.

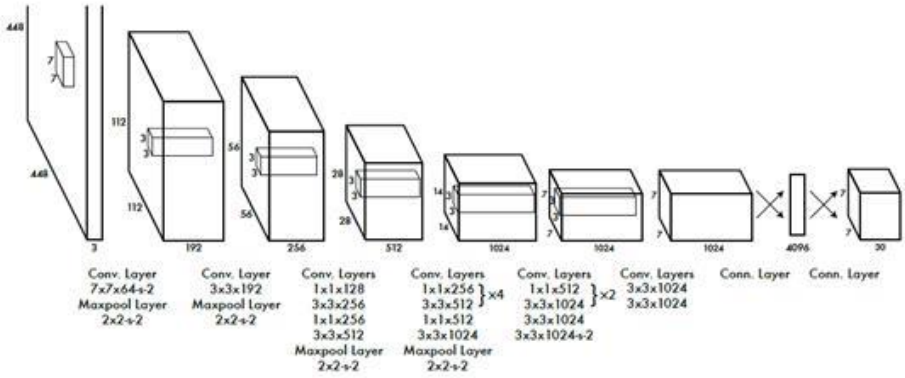


Figure 1 Architecture of original YOLO (Redmon, et al., 2016)

The YOLOv2 made a number of iterative improvements including BatchNorm, higher resolution and anchor boxes on top of original YOLO.

The YOLOv3 added an objectness score to bounding box prediction, connections to the backbone network layers, and predictions at three separate levels of granularity on the top of previous version to improve performance on smaller objects (Redmon, et al., 2018).

There are several changes compared to previous versions of YOLO for new YOLOv4. The authors have considered CSPResNext50, CSPDarknet53 and EfficientNet-B3 for the backbone which help to alleviate the vanishing gradient problem as it is difficult to back propagate loss signals through a very deep network. These are used to bolster feature propagation and encourage the network to reuse features which can reduce the number of network parameters in addition. YOLOv4 uses PANet for the feature aggregation of the network and adds a SPP bock after CSPDarknet53 to increase the receptive field and separate out the most important features from the backbone.

However, there are several changes in YOLOv4, it still re-uses some components from previous versions. For the detection step, YOLOv4 still uses same YOLO head as YOLOv3 for detection with the anchor-based detection steps and three levels of detection granularity.

YOLOv4 has also added Bag of Freebies (BoF) and Bag of Specials (BoS) to improve performance of the network without adding to inference time in production. Most of the BoF includes data augmentation. To get the most performance out of the model, data augmentation is important in computer vision and widely used. This is used to expand the training data and expose the model to semantic situations in the new version of YOLOv4. The Bag of Specials significantly increases the performance. The DIoU NMS is used to separate out predicted bounding boxes for network to predict multiple bounding boxes in a single object which could be useful to pick best out efficiently. Also, it uses DropBlock regularization technique which forces network to learn features.

YOLOv4 belong to the family of one-stage it is composed of several parts which include:

- Input
- Backbones
- Neck
- Heads
- Dense Prediction
- Sparse Prediction (two-stage)

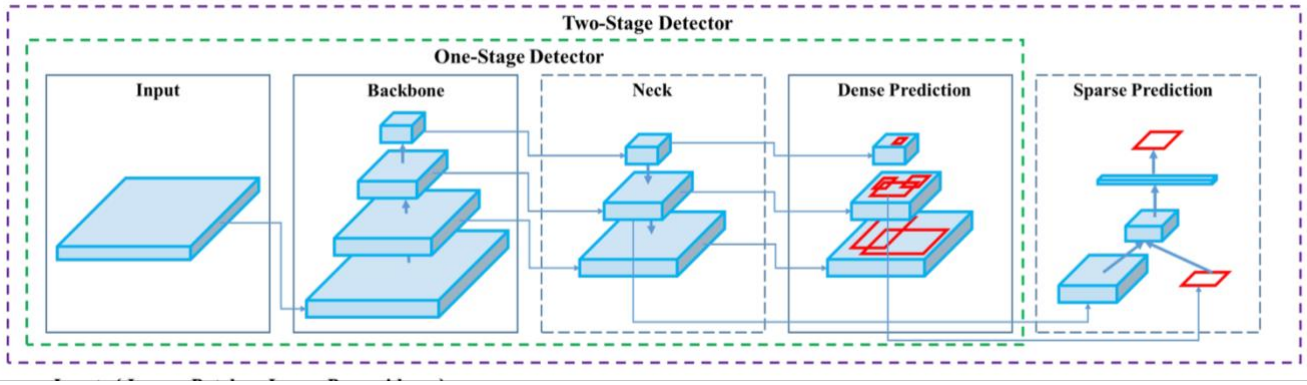


Figure 1 Architecture of YOLOv4

Firstly, YOLO framework takes an image as an **input** then it divides the input into grids such as (3x3) grid. Image classification and localization are applied to each grid and then its bounding boxes is predicted with their corresponding class probabilities for object if any object is detected in the input image.

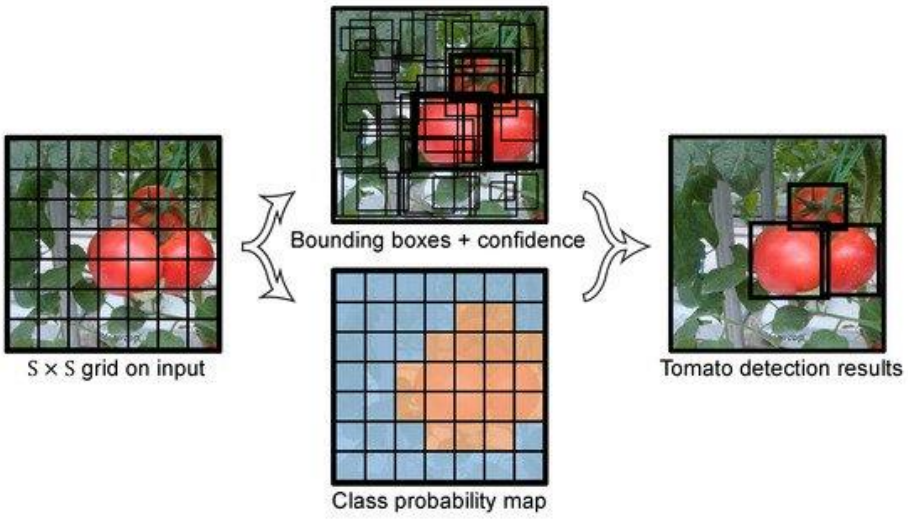


Figure 2 YOLO model detection (Liu, et al., 2020)

Backbone is the feature extraction architecture such as Tiny YOLO or Darknet53. The difference between such architectures is the backbone that is being used. Tiny YOLO consists of only 9 convolutional layers which makes it less precise, but it becomes faster and enhanced for small devices such as mobile and embedded projects (Cohen, 2020). On the other side, Darknet53 has 53 convolutional layers whose accuracy is higher than Tiny YOLO, but it is

slower. There are multiple versions of YOLO depending on the backbone that is being used to develop. In YOLOv4, backbones can be VGG16, ResNet-50, SpineNet, EfficientNet-B0/B7, CSPResNeXt50 and CSPDarknet53. YOLOv3 used to use Darkenet53 as a backbone, however, in YOLOv4, it has been changed to CSPDarknet53. CSP is Cross-Stage-Partial connections which helps to separate the current layer into 2 parts where one part will go through block of convolutions and the other won't go through them. This can help us to aggregate the
results.

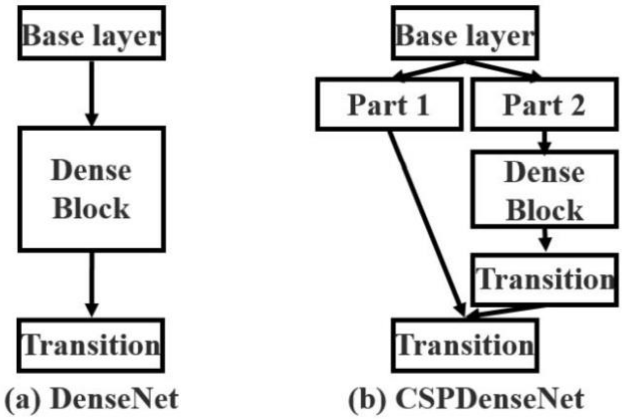


Figure 3 Example DensNet and CSPDenseNet (Cohen, 2020)

Neck is other vital part in YOLO4 architecture. The goal of it is to increase additional layers between the backbone and the head (dense prediction block). Within the neck, we have additional blocks and path-aggregation blocks. Additional blocks include layers such as SPP, ASPP RFB and SAM (Bochkovskiy, et al., 2020). Similarly, path-aggregation blocks include FPN, PAN, NAS_FPN and Fully connected FPN to name few (Bochkovskiy, et al., 2020). Feature Pyramid Network (FPN) is popular technique but it is not used in YOLOv4. Next technique is Path Aggregation Network (PANet) which is used for aggregating information to get higher accuracy in this version of YOLO. Similarly, Spatial Attention Module (SAM) is attention mechanism commonly used in Recurrent Neural Network (RNN) which focuses on specific part of the input. Spatial Pyramid Pooling (SPP) is used in R-CNN networks and other algorithms. YOLOv4 uses modified Path Aggregation Network (PANet), modified Spatial Attention Module (SAM) and Spatial Pyramid Pooling (SPP) (Cohen, 2020) for neck component to add extra layers.

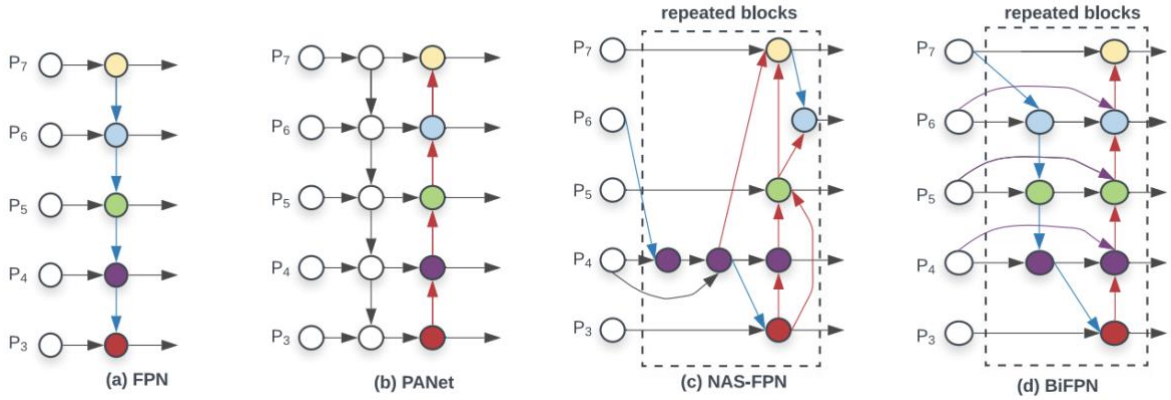


Figure 4 Feature network design (Pang, et al., 2020)

Another component is **head** which is used to detect bounding boxes and classify the object inside each box. This is same process as in YOLOv3 where network detects the boxes

coördinates x, y, height and width. In addition, it also detects the confidence score for each class. It includes Dense Prediction (one-stage) and Sparse Prediction (two-stage).

Similarly, the authors have implemented techniques to improve the accuracy of the model while training and in post-processing which are called bag-of-freebies and bag-of-specials (Bochkovskiy, et al., 2020). According to the authors, the purpose of it is to increase the variability of input images so that the designed object detection model has higher robustness to the images obtained from different environments. The approaches such as data augmentation is used to achieve the variability in the input images.

According to the paper, here's the list of techniques used:

Bag of Freebies for the backbone: CutMix, MixUp and Mosaic data augmentation, DropBlock regularization, Class label smoothing.

Bag of Freebies for the detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, eliminate grid sensitivity, cosine annealing scheduler, self-adversarial training, using multiple anchors for a single ground-truth sample, optimizing hyperparameters, random training shapes.

CutMix is an image data augmentation strategy which replaces the removed regions with a patch from another image instead of just removing pixels as in Cutout. MixUp generates new samples by linear interpolation of multiple samples and their labels. This technique can generalize better than the traditional Empirical Risk Minimization (ERM) method.

DropBlock is a structured form of dropout that is used to regularize in CNN. In DropBlock units, feature map of contiguous region is dropped together which discards features in a correlated area. Label smoothing is a regularization technique which adds noise to the labels (Papers With Code, 2020).

Bag of Specials is another technique which helps to increase the accuracy of object detection. In addition, it enhances the certain features in a model such as enlargement of receptive field (SPP, ASPP and RBF) (Bochkovskiy, et al., 2020). Similarly, it introduces attention mechanism and strengths feature integration capability. The post-processing methods is used for screening model prediction results.

According to the paper, here's the list of techniques used:

Bag of Specials for backbone: Mish activation, cross-stage partial connections (CSP), multi-input weighted residual connections (MiWRC)

Bag of Specials for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

According to the authors, the YOLOv4 consists of

Backbone: CSPDarknet53,

Neck: Spatial Pyramid Pooling additional module, PANet path-aggregation, and

Head: YOLOv3

Experiment

The authors experimented to investigate the influence of different training improvement techniques on accuracy of the classifier on ImageNet and MS COCO datasets.

ImageNet image classification had 8,000,000 of training steps with batch size or 128 and mini batch of 32. The 0.1 was set as an initial learning rate with momentum of 0.9 and weight decay of 0.005. In addition, they also verified that the MixUp, CutMix, Mosaic, Blurring data augmentation and label smoothing regularization methods (Bochkovskiy, et al., 2020). Moreover, they also compared the different kind of activation functions such as LReLu, Swis and Mish which were trained on 1080 Ti and 2080 Ti GPU (Bochkovskiy, et al., 2020).

MS COCO object detection had 500,500 training steps. The initial adopted learning rate was 0.01 and multiply with a factor of 0.1 at the 400,000 steps and 450,000 steps. The momentum was set to 0.9 and weight decay was set to 0.0005. They also proved threshold for assigning ground truth 0.213, and loss normalizer 0.07 for genetic algorithm experiments. Large number of BoF is confirmed, including grid sensitivity elimination, mosaic data augmentation, Intersection over Union (IoU) threshold, genetic algorithm, DropBlock, class label smoothing, cross mini-batch normalization, cosine annealing scheduler, dynamic mini-batch size, self-adversarial training, Optimized Anchors, different kind of Intersection over Union (IoU) losses (Bochkovskiy, et al., 2020). For entire experiments, they use only one GPU for training. The techniques such as syncBN that optimizes multiple GPUs were not used.

My experiment includes testing 10 images to detect objects in the image. I will be using Google Colab.

Dataset Description:

The images are randomly picked from the dataset “Open Images 2019 – Object Detection” (Google Research, 2019) that Google Research published on Kaggle for competition. The competition prize was $25,000. There are 99,999 test images in the dataset. However, I randomly picked only 10 images to experiment the YOLOv4 speed and accuracy. The dataset can be found at https://www.kaggle.com/c/open-images-2019-object-detection/data.

Result

During the experiments, introducing features such as CutMix, MixUp, Bluring and Mosaic data augmentation improved the classifier’s accuracy. Using their Bag of Freebies backbone for

classifier training they used CutMix, MixUp, Bluring and Mosaic data augmentation and Class label smoothing. Moreover, they used Swish and Mish activation as a complementary option. The following table is the result.

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.9%	94.0%
✓							77.2%	**94.0%**
	✓						**78.0%**	**94.3%**
		✓					**78.1%**	**94.5%**
			✓				77.5%	93.8%
				✓			**78.1%**	**94.4%**
					✓		64.5%	86.0%
						✓	**78.9%**	**94.5%**
	✓	✓		✓			**78.5%**	**94.8%**
	✓	✓		✓		✓	**79.8%**	**95.2%**

Figure 5 BoF and Mish influence on the CSPResNeXt-50 classifier accuracy. (Bochkovskiy, et al., 2020)

Here is other list of Results shown in the paper:

- Even though classification accuracy of CSPResNeXt50 models trained with different features was higher compared to CSPDarknet53 models, the CSPDarknet53 model showed better accuracy for object detection.
- Increase in model's classification accuracy while using Bag of Freebies and Mish for the CSPResNeXt50 classifier training, but supplementary application of these pre-trained weightings for detector training reduced the detector accuracy.
- The CSPDarknet53 model proved a greater capability to rise the detector accuracy owing to numerous enhancements.

Since different methods use GPUs of diverse architectures for inference time verification, they ran YOLOv4 on frequently adopted GPUs of Pascal, Maxwell, and Volta architectures, and matched them with other state-of-the-art methods.

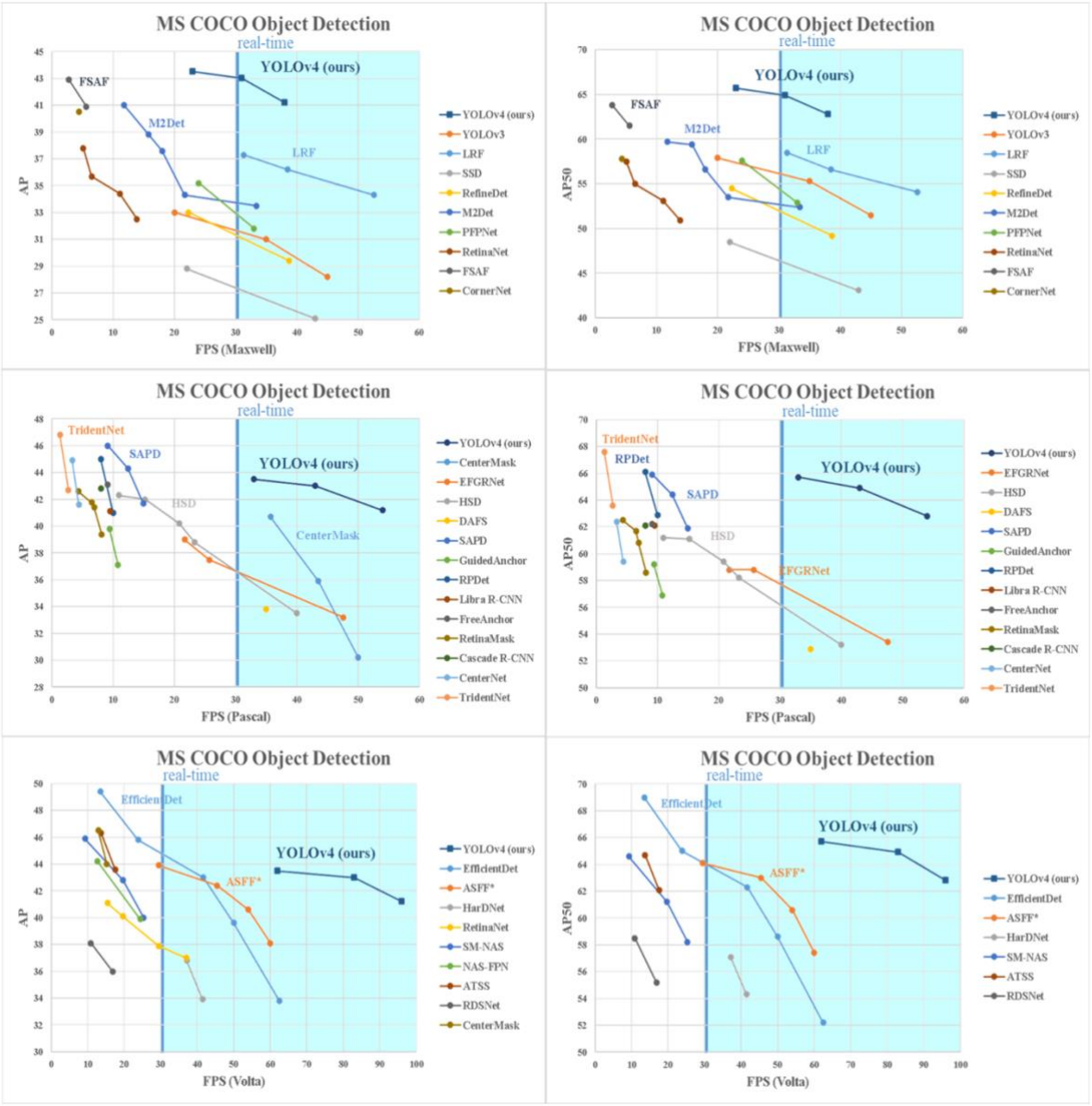


Figure 6 Speed and accuracy comparison of different object detectors. (Bochkovskiy, et al., 2020)

My result:

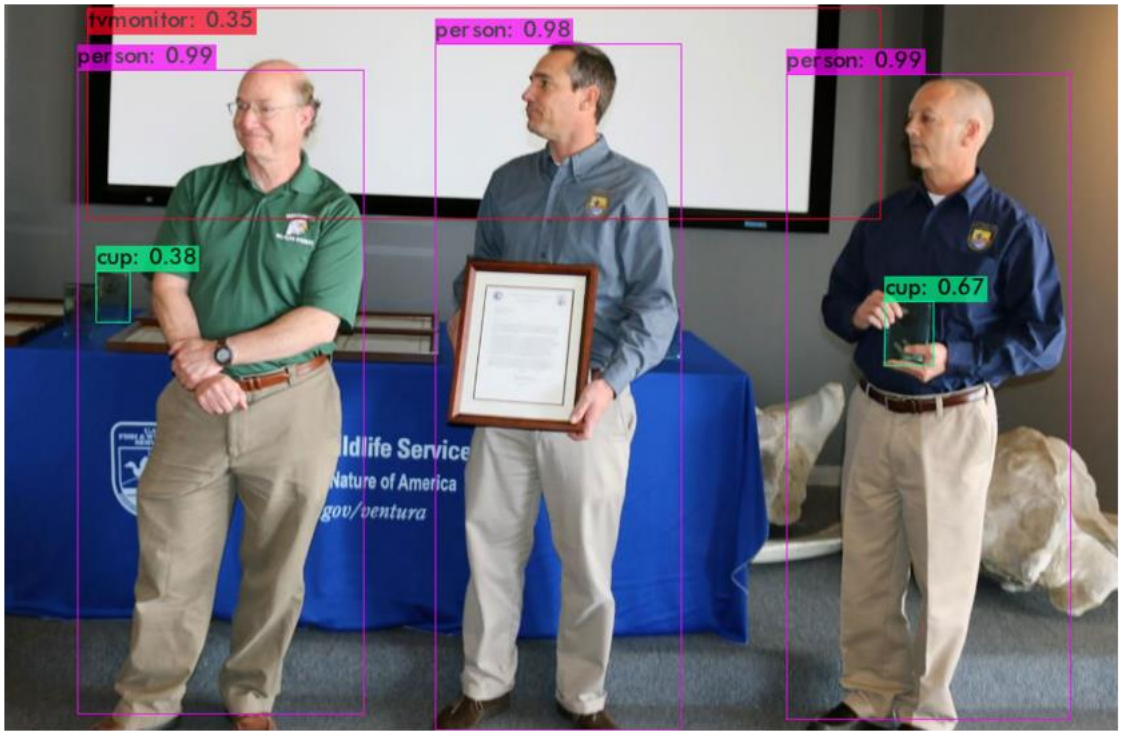
Image 1:

Most of the objects are identified in the image. However, two of the handbags are identified but one bag is not identified.

Image 2:

Not all people in the image are identified. Mostly people in the foreground are identified. This could be because of lighting as well as there is light in the room.

Image 3:

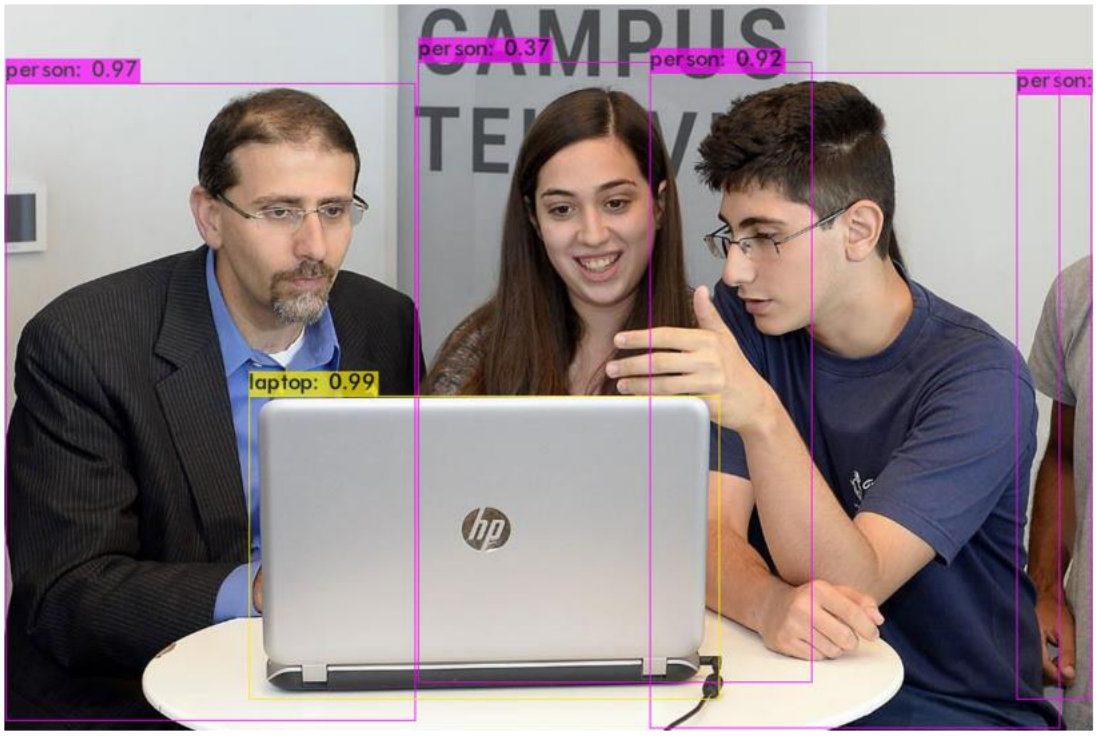


People are detected with high confidence in the image, but trophy is classified as cup. Similarly, projector in the room is classified as tv monitor.

Image 4:

Some of the soldiers in the image are identified but some are not identified specially the ones in the far back of the image. One of the aeroplane is detected but second one is not detected. Horse is detected in the image but there is no horse in the image.

Image 5:

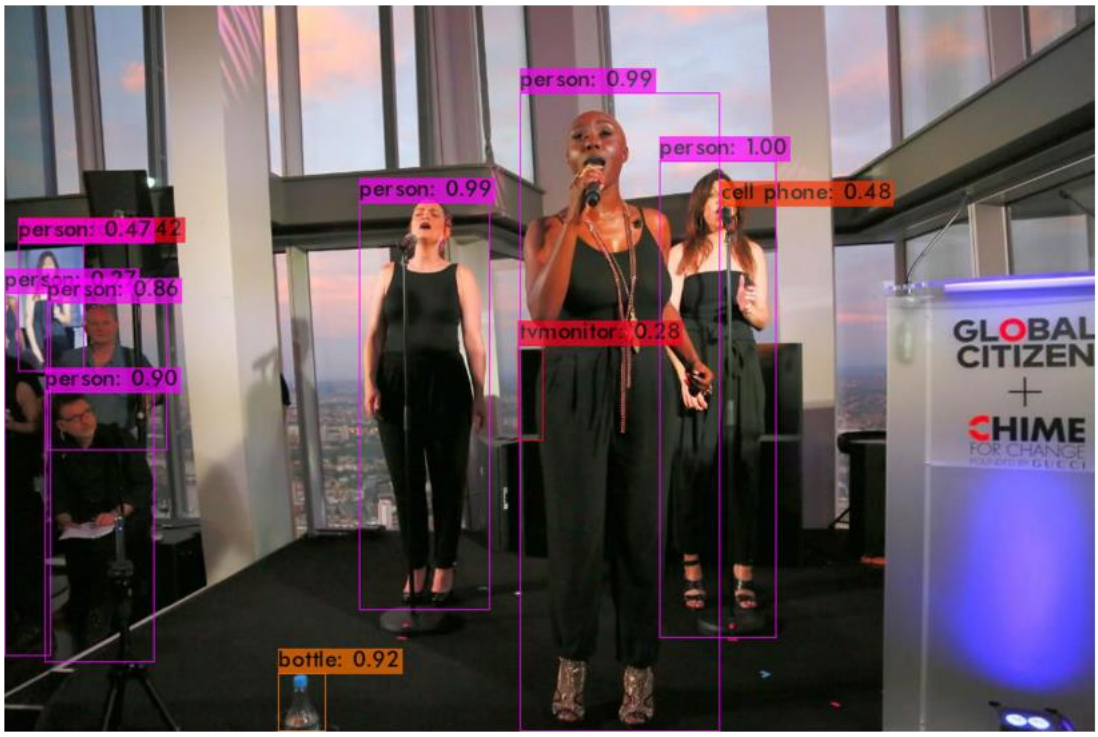


All objects in the images are correctly identified.

Image 6:

Foreground people are detected. However, people in the background are not detected. Similarly, speaker is identified as suitcase in the image. Standing person is 99% confidence of being a person that the person dancing with head which is just 34%. The Dining table is detected but it seems like there is no dining table in the image.

Image 7:



All persons in the image are identified. The bottle on the bottom is also identified even though it is not shown entirely in the image.

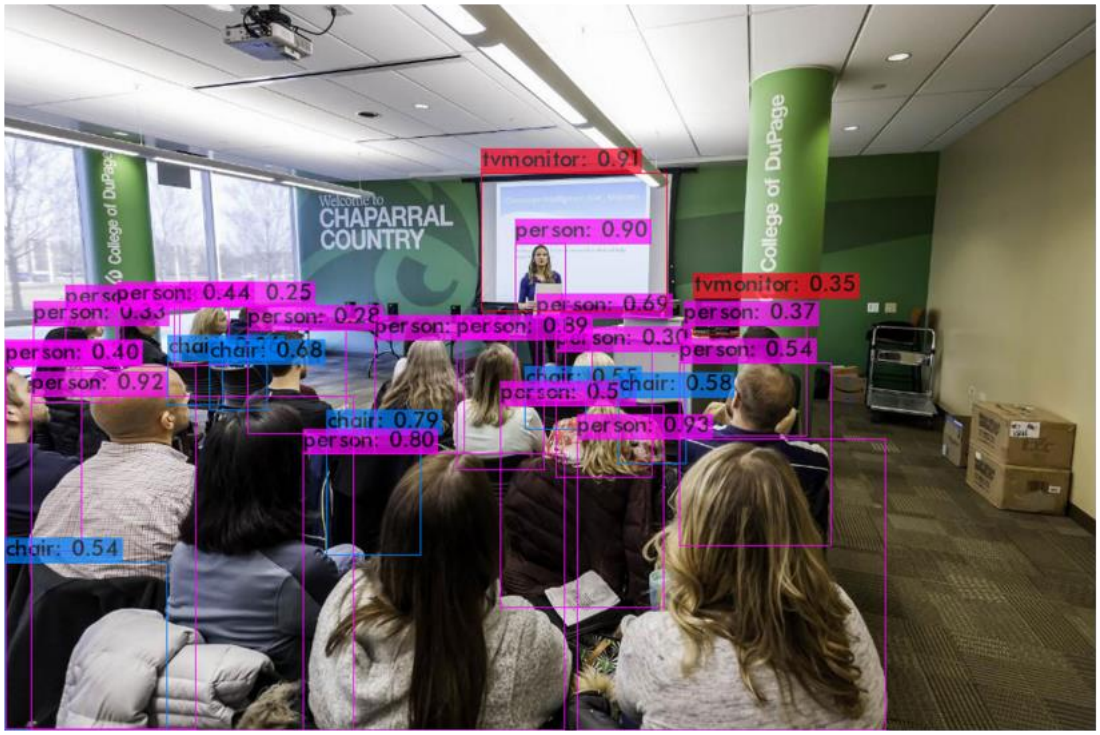
Image 8:

Most of the objects are identified in this image. The spoon in the table is also detected which shows that it can detect small objects in the image as well.

Image 9:

Even though there are houses and building in the image, YOLOv4 could not detect the objects.

Image 10:



Most of the objects in the image are classified correctly. However, there seems to have two tv monitor in the room but there is only one projector in the room.

Computational time to detect images:

Image 1	Image 2	Image 3	Image 4	Image 5	Image 6	Image 7	Image 8	Image 9	Image 10
166.19	164.65	164.97	165.03	161.86	134.87	135.31	165.73	137.45	159.01

The above table shows the time taken to detect the object in the images in milli seconds (ms). The objects were detected very quickly.

After experimenting YOLOv4 with 10 images, it is fast and accurate. Even though it did not detect all objects in the image, and some were classified incorrectly, it was fast and accurate for the pretrained model. The custom training with huge data will significantly improve the accuracy. According to the literature evidence base, it is faster and more accurate than all available alternative detectors.

Conclusion

It is clear from the research that YOLOv4 is powerful object detection model which offers state of the art detector that is faster and more accurate. Although there are several alternative detectors, it is far better than other object detectors in terms of speed and accuracy. They have used many of features to improve the accuracy for both the classifier and the detector. They

have verified many of features and selected for use such of them for improving the accuracy of both the classifier and the detector. The authors believe that those features can be used as best-practice for future studies and developments. After experimenting with 10 images, I found that most of the objects were detected and correctly classified in the image. However, not all the images were identified in the image and identified correctly. The object detection using Google Colab GPU did not take long detect objects in the images.

Although YOLOv4 is fast and accurate, there is always room for improvement. As in my experiment not all objects were identified correctly or could not find at all. Even though YOLOv4 uses many data augmentation techniques such as CutOut, MixUp and CutMix, augmentation methods like random cropping with constraints, expansion, horizontal flip, resize with random interpolation and color jittering including brightness, hue, saturation and contrast could improve the model's performance during training (Singh, 2020). The Google Brain team (Pang, et al., 2020) has recently introduced EfficientNets which achieves better efficiency than previous commonly used backbones so if YOLOv4 uses this as backbone, it could be more efficient. In addition, YOLOv4 currently uses batch normalization or cross-GPU Batch Normalization (CGBN) but according to the (Pang, et al., 2020), fast normalized fusion can be used to increase efficiency and accuracy. This technique applies Relu activation function after each weight(w1) and epsilon (e=0.000001) which is a small value to avoid numerical instability. The normalized weight is between 0 and 1 so it makes more efficient. Current research supports the use of proposed improvements as discussed above; however, a continuation of current research with consistent and strengthened methodologies will help justify its improvement in the algorithm.

References

Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M., 2020. *YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv 2020. arXiv preprint arXiv:2004.10934.*

Cohen, J., 2020. *Introduction to YOLOv4: Research review.* [Online]
Available at: https://heartbeat.fritz.ai/introduction-to-yolov4-research-review-5b6b4bd5f255
[Accessed 10 August 2020].

Google Research, 2019. *Open Images 2019 - Object Detection.* [Online]
Available at: https://www.kaggle.com/c/open-images-2019-object-detection/overview/evaluation
[Accessed 18 August 2020].

Liu, G., Nouaze, J. C., Philippe, M. L. T. & Kim, J. H., 2020. *YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3.* Sensors, 20(7), p.2145.

Tan, M., Pang, R. and Le, Q.V., 2020. *Efficientdet: Scalable and efficient object detection*. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 10781-10790).

Papers With Code, 2020. [Online]
Available at: https://paperswithcode.com/method/yolov4
[Accessed 11 August 2020].

Redmon, J. and Farhadi, A., 2018. *Yolov3: An incremental improvement*. arXiv preprint arXiv:1804.02767.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. *You only look once: Unified, real-time object detection*. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

Sharma, P., 2018. *A Practical Guide to Object Detection using the Popular YOLO Framework – Part III.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/
[Accessed 11 August 2020].

Singh, R. K., 2020. *How to Improve YOLOv3.* [Online]
Available at: https://blog.paperspace.com/improving-yolo/
[Accessed 15 August 2020].