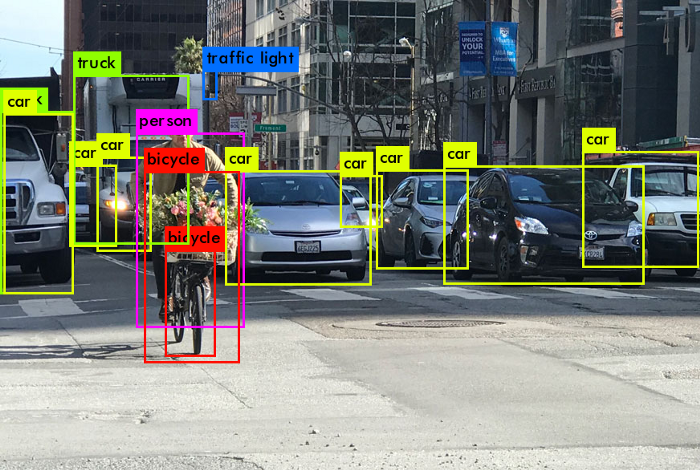
**YOLOv4: Optimal Speed and Accuracy of Object Detection**

-Saumya Arora



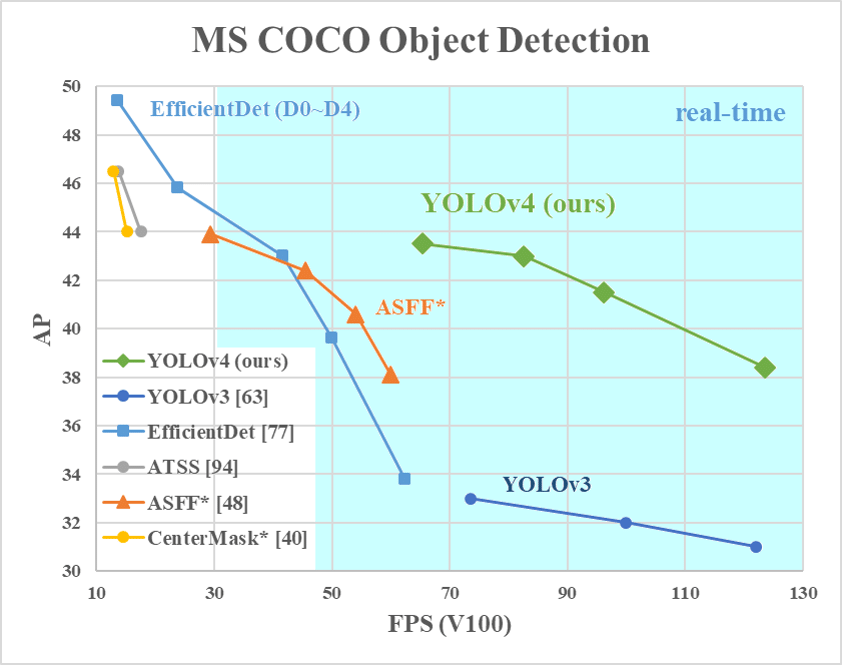
**Abstract**

There are a huge number of features that are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross Mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss, and combine some of them to achieve state-of-the-art results: 43.5% AP (65.7% AP50) for the MS COCO dataset at a realtime speed of ∼65 FPS on Tesla V10.

**Table of contents**

1. **INTRODUCTION**
2. **RELATED WORK**
3. **Object Detection Model**
4. **Parts of Object Detector**
5. **Bag of Freebies**
6. **Bag of Specials**
7. **METHODOLOGY**
8. **Selection of Architecture**
9. **Selection of BoF and BoS**
10. **Additional Improvements**
11. **YOLOv4**
12. **Uses of YOLOv4**
13. **ADVANTAGES**
14. **DISADVANTAGES**
15. **CONCLUSION**
16. **Introduction**

The majority of CNN-based object detectors are largely applicable only for recommendation systems**. For example, searching for free parking spaces via urban video cameras is executed by slow accurate models, whereas car collision warning is related to fast inaccurate models.** Improving the real-time object detector accuracy enables using them not only for hint generating recommendation systems, but also for stand-alone process management and human input reduction. **Real-time object detector operation on conventional Graphics Processing Units (GPU) allows their mass usage at an affordable price. The most accurate modern neural networks do not operate in real time and require large number of GPUs for training with a large mini-batch-size.** We address such problems through creating a CNN that operates in real-time on a conventional GPU, and for which training requires only one conventional GPU.

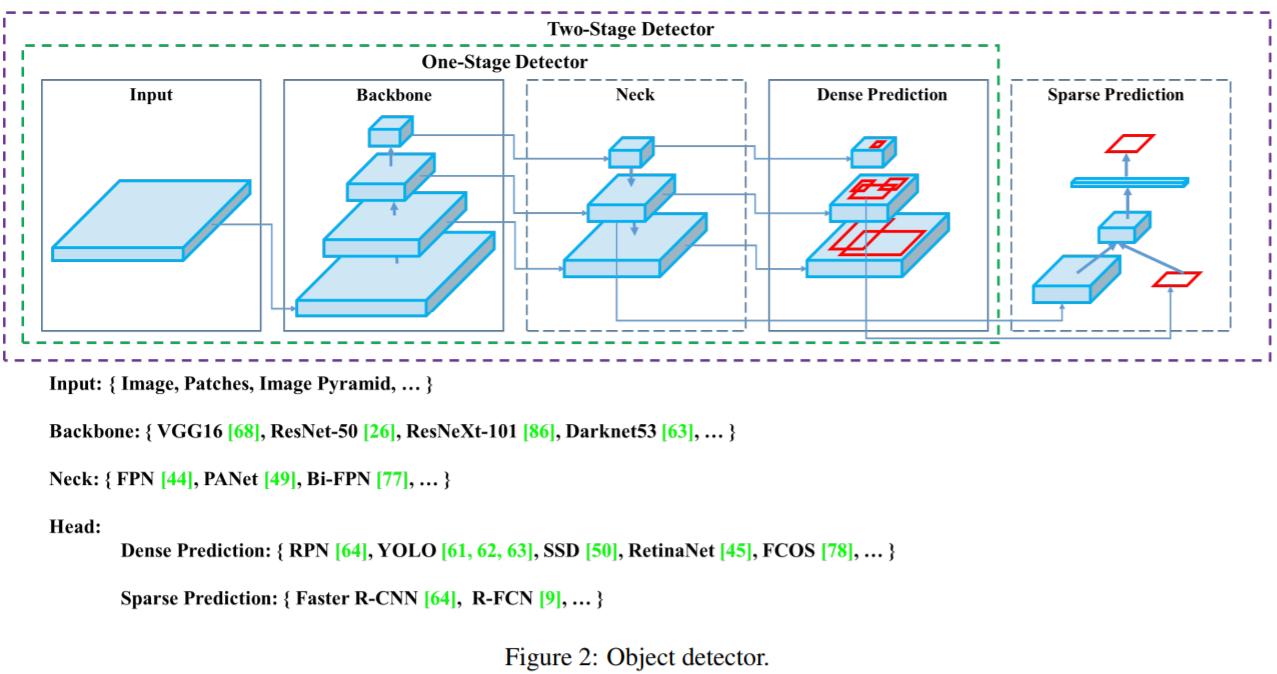
**Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than efficient with comparable performance. Improves YOLOv3’s AP and FPS by 10% and 12%, respectively.**

The main goal of this work is designing a fast-operating speed of an object detector in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator. For example, anyone who uses a conventional GPU to train and test can achieve real-time, high quality, and convincing object detection results, as the YOLOv4 results shown in Figure 1. Our contributions are summarized as follows:

**1.**We develop an efficient and powerful object detection model. It makes everyone can use a 1080 Ti or 2080 Ti GPU to train a super-fast and accurate object detector.

**2.**We verify the influence of state-of-the-art Bag-of Freebies and Bag-of-Special’s methods of object detection during the detector training.

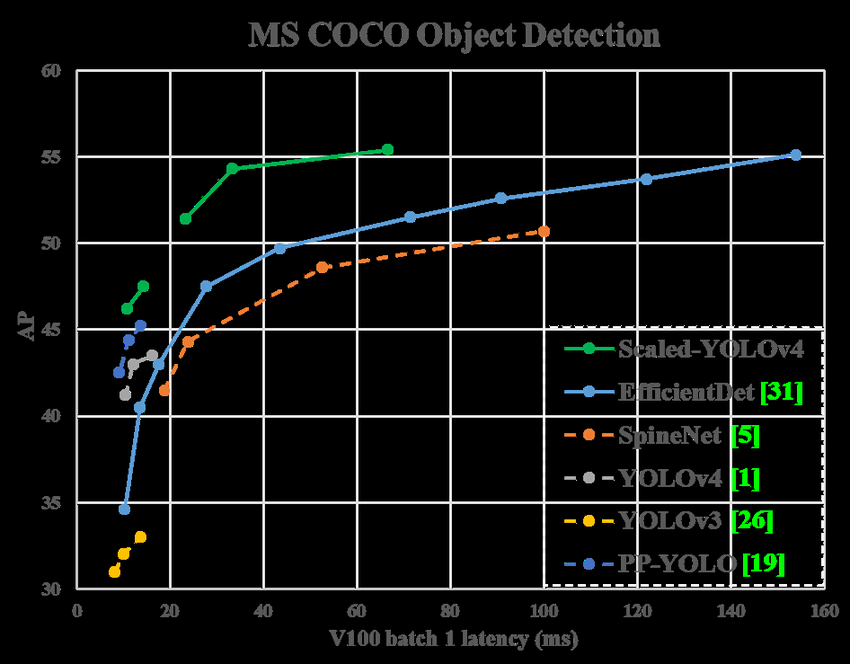
**3.**We modify state-of-the-art methods and make them more efficient and suitable for single GPU training, including CBN [89], PAN [49], SAM [85], etc



1. **Related work**
2. **Object Detection Models**

A modern detector is usually composed of **two parts, a backbone which is pre-trained on ImageNet and a head** which is used to predict classes and bounding boxes of objects. For those detectors running on GPU platform, their backbone could be VGG [68], ResNet [26], ResNeXt [86], or DenseNet [30]. For those detectors running on CPU platform, their backbone could be Squeeze Net [31], Mobile Net [28, 66, 27, 74], or Shuffle Net [97, 53]. As to the head part, it is usually categorized into two kinds, i.e., one-stage object detector and two-stage object detector. The most represent two-stage object detector is the R-CNN [19] series, including fast R-CNN [18], faster R-CNN [64], R-FCN [9], and Libra R-CNN [58]. It is also possible to make a two-stage object detector an anchor-free object detector, such as RepPoints [87]. As for one-stage object detector, the most representative models are YOLO [61, 62, 63], SSD [50], and RetinaNet [45]. In recent years, anchor-free one-stage object detectors are developed. The detectors of this sort are CenterNet [13], CornerNet [37, 38], FCOS [78], etc. Object detectors developed in recent years often insert some layers between backbone and head, and these layers are usually used to collect feature maps from different stages. We can call it the neck of an object detector. Usually, a neck is composed of several bottom-up paths and several top-down paths. Networks equipped with this mechanism include Feature Pyramid Network (FPN) [44], Path Aggregation Network (PAN) [49], BiFPN [77], and NAS-FPN [17]. In addition to the above models, some researchers put their emphasis on directly building a new backbone (DetNet [43], DetNAS [7]) or a new whole model (SpineNet [12], HitDetector [20]) for object detection.

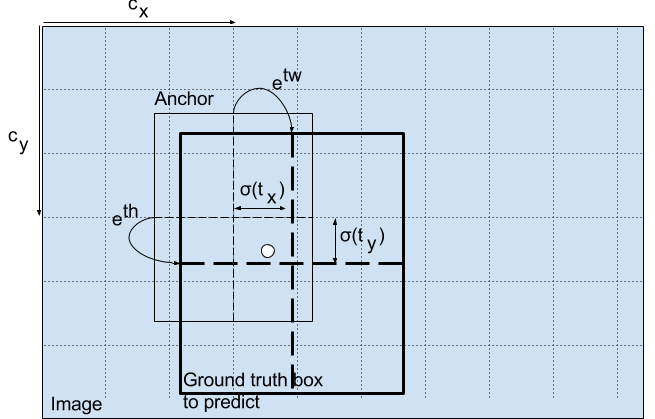
1. **Parts of Object Detector**
   1. **Input:** Image, Patches, Image Pyramid
   2. **Backbones:** VGG16 [68], ResNet-50 [26], SpineNet [12], EfficientNet-B0/B7 [75], CSPResNeXt50 [81], CSPDarknet53 [81]
   3. **Neck:**
2. **Additional blocks:** SPP [25], ASPP [5], RFB [47], SAM [85]
3. **Path-aggregation blocks:** FPN [44], PAN [49], NAS-FPN [17], Fully-connected FPN, BiFPN [77], ASFF [48], SFAM [98]
   1. **Heads:**
4. **Dense Prediction (one-stage):** RPN [64], SSD [50], YOLO [61], RetinaNet [45] (anchor based) ◦ CornerNet [37], CenterNet [13], MatrixNet [60], FCOS [78] (anchor free)
5. **Sparse Prediction (two-stage):** Faster R-CNN [64], R-FCN [9], Mask RCNN [23] (anchor based) ◦ RepPoints [87] (anchor free)

**Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. The dashed line means only latency of model inference, while the solid line includes model inference and post-processing.**

1. **Bag of Freebies**

Usually, a conventional object detector is trained offline. **Therefore, researchers always like to take this advantage and develop better training methods which can make the object detector receive better accuracy without increasing the inference cost. We call these methods that only change the training strategy or only increase the training cost as “bag of freebies.”** The purpose of data augmentation is to increase the variability of the input images, so that the designed object detection model has higher robustness to the images obtained from different environments.

**For examples,** photometric distortions and geometric distortions are two commonly used data augmentation method and they definitely benefit the object detection task. In dealing with photometric distortion, we adjust the brightness, contrast, hue, saturation, and noise of an image. For geometric distortion, we add random scaling, cropping, flipping, and rotating.

**The data augmentation methods mentioned above are all pixel-wise adjustments, and all original pixel information in the adjusted area is retained. In addition, some researchers engaged in data augmentation put their emphasis on simulating object occlusion issues.** They have achieved good results in image classification and object detection. For example, random erase [100] and Cut-out [11] can randomly select the rectangle region in an image and fill in a random or complementary value of zero. As for hide-and-seek [69] and grid mask [6], they randomly or evenly select multiple rectangle regions in an image and replace them to all zeros. If similar concepts are applied to feature maps, there are Dropout [71], Drop Connect [80], and DropBlock [16] methods. In addition, some researchers have proposed the methods of using multiple images together to perform data augmentation. For example, MixUp [92] uses two images to multiply and superimpose with different coefficient ratios, and then adjusts the label with these superimposed ratios. As for CutMix [91], it is to cover the cropped image to rectangle region of other images, and adjusts the label according to the size of the mix area. In addition to the above-mentioned methods, style transfer GAN [15] is also used for data augmentation, and such usage can effectively reduce the texture bias learned by CNN.



In dealing with the problem of semantic distribution bias, a very important issue is that there is a problem of **data imbalance between different classes**, and this problem is **often solved by hard negative example mining** [72] or online hard example mining [67] in two-stage object detector.

**But the example mining method is not applicable to one-stage object detector,** because this kind of detector belongs to the dense prediction architecture. Therefore **Lin et al.** [45] proposed **focal loss to deal with the problem of data imbalance existing between various classes.** Another very important issue is that it is **difficult to express the relationship of the degree of association between different categories with the one-hot hard representation.**



In order to make this issue processed better, some researchers recently **proposed IoU loss** [90], which puts the coverage of **predicted BBox area and ground truth BBox area into consideration.** The IoU loss computing process **will trigger the calculation of the four coordinate points of the BBox by executing IoU with the ground truth, and then connecting the generated results into a whole code.** Because IoU is a scale invariant representation. Recently, some researchers have continued to improve IoU loss.

**For example**, GIoU loss [65] is to include the shape and orientation of object in addition to the coverage area. They proposed to find the smallest area BBox that can simultaneously cover the predicted BBox and ground truth BBox, and use this BBox as the denominator to replace the denominator originally used in IoU loss. As for DIoU loss [99], it additionally considers the distance of the centre of an object, and CIoU loss [99], on the other hand simultaneously considers the overlapping area, the distance between centre points, and the aspect ratio. CIoU can achieve better convergence speed and accuracy on the BBox regression problem.

1. **Bag of Specials**

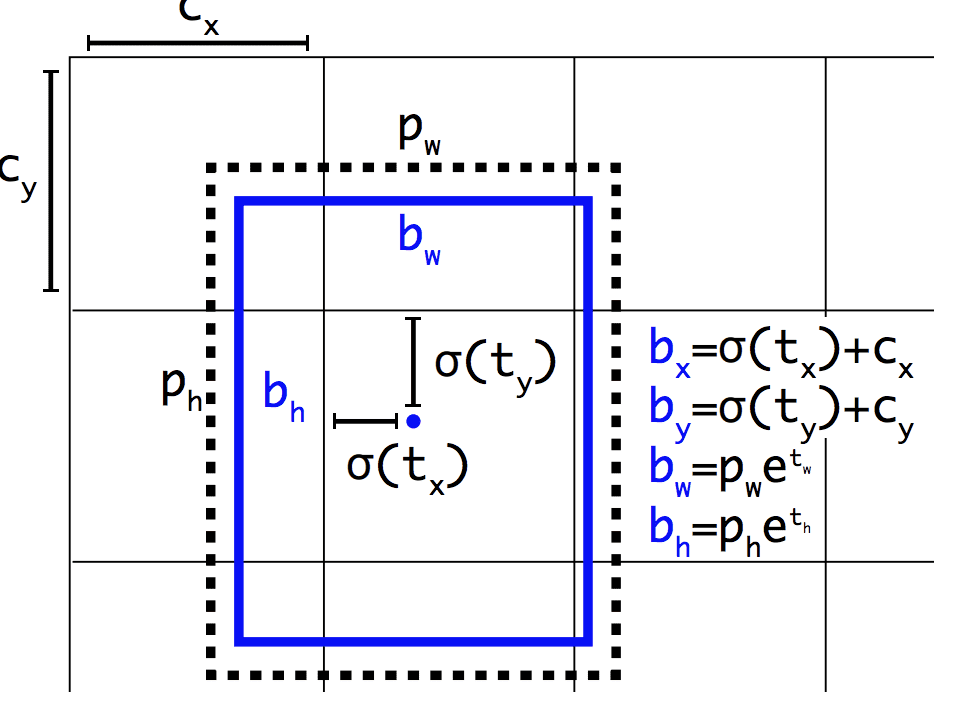
**For those plugin modules and post-processing methods that only increase the inference cost by a small amount but can significantly improve the accuracy of object detection, we call them “bag of specials”.** Generally speaking, these plugin modules are for enhancing certain attributes in a model, such as enlarging receptive field, introducing attention mechanism, or strengthening feature integration capability, etc., and post-processing is a method for screening model prediction results.

**Common modules that can be used to enhance receptive field are SPP [25], ASPP [5], and RFB [47].** The SPP module was originated from Spatial Pyramid Matching (SPM) [39], and SPMs original method was to split feature map into several **d × d equal blocks, where d can be {1, 2, 3, ...},** thus forming spatial pyramid, and then extracting bag-of-word features. SPP integrates SPM into CNN and use max-pooling operation instead of bag-of-word operation. Since the SPP module proposed by **he et al.** [25] will output one dimensional feature vector, it is infeasible to be applied in Fully Convolutional Network (FCN). Thus, in the design of YOLOv3 [63], Redmon and Farhadi improve SPP module to the concatenation of max-pooling outputs with kernel **size k × k, where k = {1, 5, 9, 13}, and stride equals to 1.**

Under this design, a relatively large k × k maxpooling effectively increase the receptive field of backbone feature. After adding the improved version of SPP module, YOLOv3-608 upgrades AP50 by 2.7% on the MS COCO object detection task at the cost of 0.5% extra computation**.**

**The difference in operation between ASPP [5] module and improved SPP module is mainly from the original k×k kernel size, max-pooling of stride equals to 1 to several 3 × 3 kernel size, dilated ratio equals to k, and stride equals to 1 in dilated convolution operation.** RFB module is to use several dilated convolutions of k×k kernel, dilated ratio equals to k, and stride equals to 1 to obtain a more comprehensive spatial coverage than ASPP. RFB [47] only costs 7% extra inference time to increase the AP50 of SSD on MS COCO by 5.7%.

The attention module that is often used in object detection is mainly divided into channel-wise attention and pointwise attention, and the representatives of these **two attention models are Squeeze-and-Excitation (SE) [29] and Spatial Attention Module (SAM) [85], respectively.** Although SE module can improve the power of ResNet50 in the ImageNet image classification task 1% top-1 accuracy at the cost of only increasing the computational effort by 2%, but on a GPU usually it will increase the inference time by about 10%, so it is more appropriate to be used in mobile devices. But for SAM, it only needs to pay 0.1% extra calculation and it can improve ResNet50-SE 0.5% top-1 accuracy on the ImageNet image classification task. Best of all, it does not affect the speed of inference on the GPU at all.

**Since multi-scale prediction methods such as FPN have become popular, many lightweight modules that integrate different feature pyramid have been proposed. The modules of this sort include SFAM [98], ASFF [48], and BiFPN [77]. The main idea of SFAM is to use SE module to execute channel wise level re-weighting on multi-scale concatenated feature maps.** **As for ASFF, it uses SoftMax as point-wise level reweighting and then adds feature maps of different scales. In BiFPN, the multi-input weighted residual connections is proposed to execute scale-wise level re-weighting, and then add feature maps of different scales.**

**Example of the Representation Chosen when Predicting Bounding Box Position and Shape Taken from: YOLO9000: Better, Faster, Stronger**

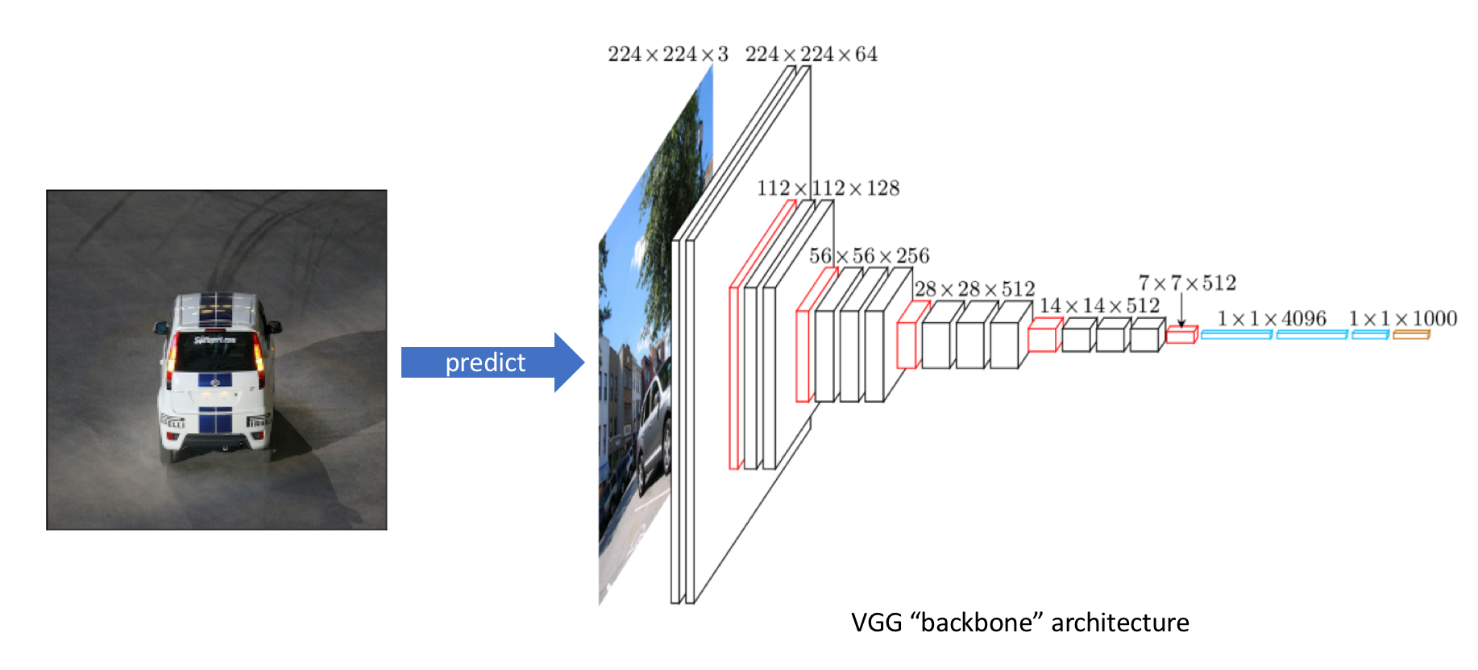
**In 2010, Nair and Hinton [56] propose ReLU to substantially solve the gradient vanish problem which is frequently encountered in traditional tanh and sigmoid activation function. Subsequently, LReLU [54], PReLU [24], ReLU6 [28], Scaled Exponential Linear Unit (SELU) [35], Swish [59], hard-Swish [27], and Mish [55], etc., which are also used to solve the gradient vanish problem, have been proposed.** The main purpose of LReLU and PReLU is to solve the problem that the gradient of ReLU is zero when the output is less than zero. As for ReLU6 and hard-Swish, they are specially designed for quantization networks. For self-normalizing a neural network, the SELU activation function is proposed to satisfy the goal. One thing to be noted is that both Swish and Mish are continuously differentiable activation function.

**The post-processing method commonly used in deep learning-based object detection is NMS, which can be used to filter those BBoxes that badly predict the same object, and only retain the candidate BBoxes with higher response.** The way NMS tries to improve is consistent with the method of optimizing an objective function. The original method proposed by NMS does not consider the context information, so **Girshick et al.** [19] added classification confidence score in R-CNN as a reference, and according to the order of confidence score, greedy NMS was performed in the order of high score to low score. As for soft NMS [1], it considers the problem that the occlusion of an object may cause the degradation of confidence score in greedy NMS with IoU score. **The DIoU NMS [99] developers’ way of thinking is to add the information of the centre point distance to the BBox screening process on the basis of soft NMS. It is worth mentioning that, since none of above postprocessing methods directly refer to the captured image features, post-processing is no longer required in the subsequent development of an anchor-free method.**

1. **METHODOLOGY**

The basic aim is fast operating speed of neural network, in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator (BFLOP). We present two options of real-time neural networks:

1. **For GPU -** we use a small number of groups (1 - 8) in convolutional layers: CSPResNeXt50 / CSPDarknet53
2. **For VPU** **-** we use grouped-convolution, but we refrain from using Squeeze-and-excitement (SE) blocks - specifically this includes the following models: EfficientNet-lite / MixNet [76] / GhostNet [21] / MobileNetV
3. **Selection of Architecture**



**For instance, our numerous studies demonstrate that the CSPResNext50 is considerably better compared to CSPDarknet53 in terms of object classification on the ILSVRC2012 (ImageNet) dataset [10].** However, conversely, the CSPDarknet53 is better compared to CSPResNext50 in terms of detecting objects on the MS COCO dataset [46]. **The next objective is to select additional blocks for increasing the receptive field and the best method of parameter aggregation from different backbone levels for different detector levels: e.g., FPN, PAN, ASFF, BiFPN.** A reference model which is optimal for classification is not always optimal for a detector. In contrast to the classifier, the detector requires the following:

1. **Higher input network size (resolution) –** for detecting multiple small-sized objects.
2. **More layers –** for a higher receptive field to cover the increased size of input network.
3. **More parameters –** for greater capacity of a model to detect multiple objects of different sizes in a single image.

The influence of the receptive field with different sizes is summarized as follows:

1. **Up to the object size -** allows viewing the entire object.
2. **Up to network size -** allows viewing the context around the object.
3. **Exceeding the network size -** increases the number of connections between the image point and the final activation.

We add the SPP block over the CSPDarknet53, since it **significantly increases the receptive field, separates out the most significant context features and causes almost no re**

**duction of the network operation speed.** We use PANet as the method of parameter aggregation from different backbone levels for different detector levels, instead of the FPN used in YOLOv3. Finally, we choose CSPDarknet53 backbone, SPP additional module, PANet path-aggregation neck, and YOLOv3 (anchor based) head as the architecture of YOLOv4. In the future we plan to expand significantly the content of Bag of Freebies (BoF) for the detector, which theoretically can address some problems and increase the detector accuracy, and sequentially check the influence of each feature in an experimental fashion. We do not use Cross-GPU Batch Normalization (CGBN or SyncBN) or expensive specialized devices. This allows anyone to reproduce our state-of-the-art outcomes on a conventional graphic processor e.g., GTX 1080Ti or RTX 2080Ti.

1. **Selection of BoF and BoS**

For improving the object detection training, a CNN usually uses the following:

1. **Activations:** ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish
2. **Bounding box regression loss:** MSE, IoU, GIoU, CIoU, DIoU
3. **Data augmentation:** CutOut, MixUp, CutMix
4. **Regularization method:** DropOut, DropPath [36], Spatial DropOut [79], or DropBlock
5. **Normalization of the network activations by their mean and variance:** Batch Normalization (BN) [32], Cross-GPU Batch Normalization (CGBN or SyncBN) [93], Filter Response Normalization (FRN) [70], or Cross-Iteration Batch Normalization (CBN) [89]
6. **Skip-connections:** Residual connections, Weighted residual connections, multi-input weighted residual connections, or Cross stage partial connections (CSP).

As for training activation function**, since PReLU and SELU are more difficult to train, and ReLU6 is specifically designed for quantization network, we therefore remove the above activation functions from the candidate list.** In the method of regularization, the people who published DropBlock have compared their method with other methods in detail, and their regularization method has won a lot. Therefore, we did not hesitate to choose DropBlock as our regularization method. As for the selection of normalization method, since we focus on a training strategy that uses only one GPU, syncBN is not considered.

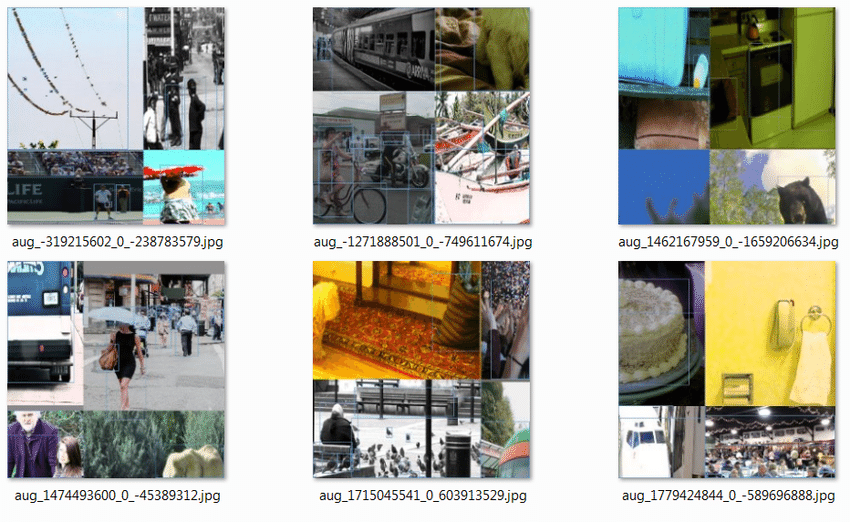
1. **Additional Improvements**

In order to make the designed detector more suitable for training on single GPU, we made additional design and improvement as follows:

We introduce a new method of data augmentation Mosaic, and Self-Adversarial Training (SAT)

We select optimal hyper-parameters while applying genetic algorithms

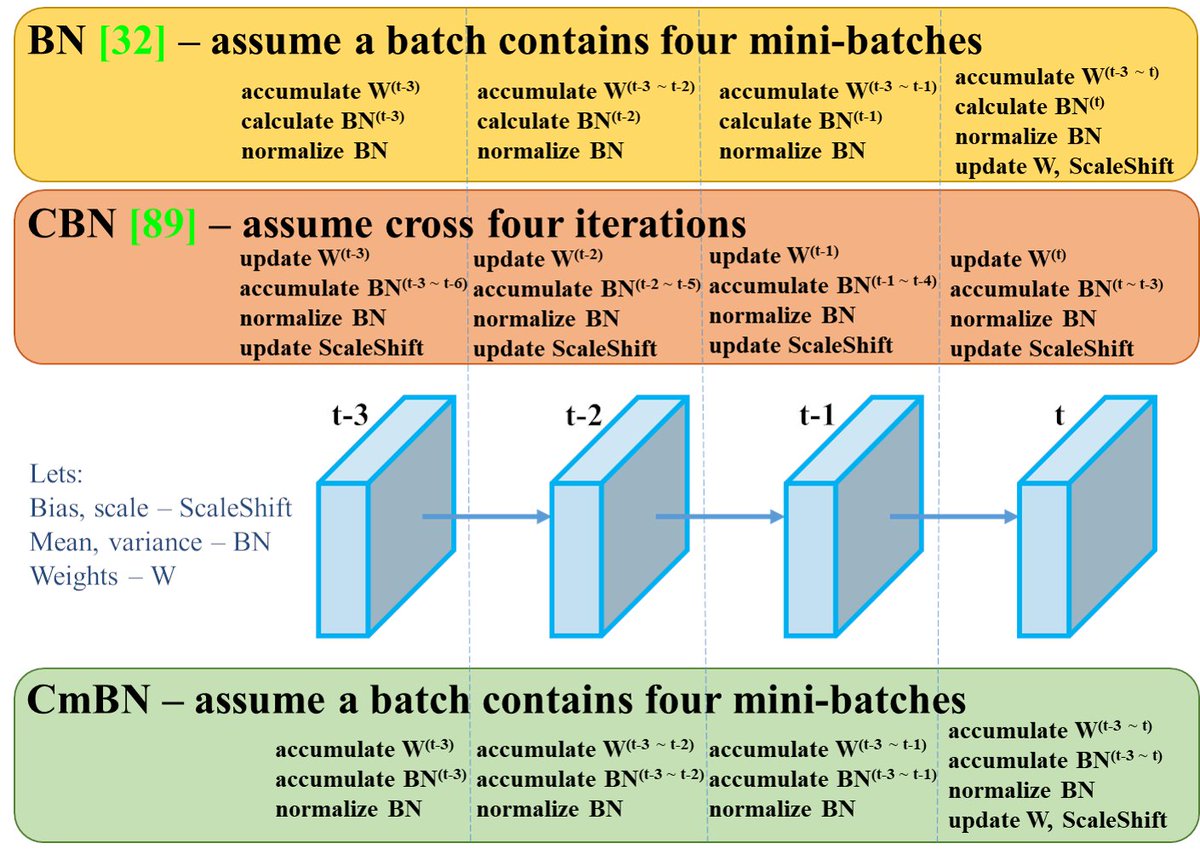
We modify some existing methods to make our design suitable for efficient training and detection - modified SAM, modified PAN, and Cross Mini-Batch Normalization (CmBN)



**Figure 3: Mosaic represents a new method of data augmentation.**

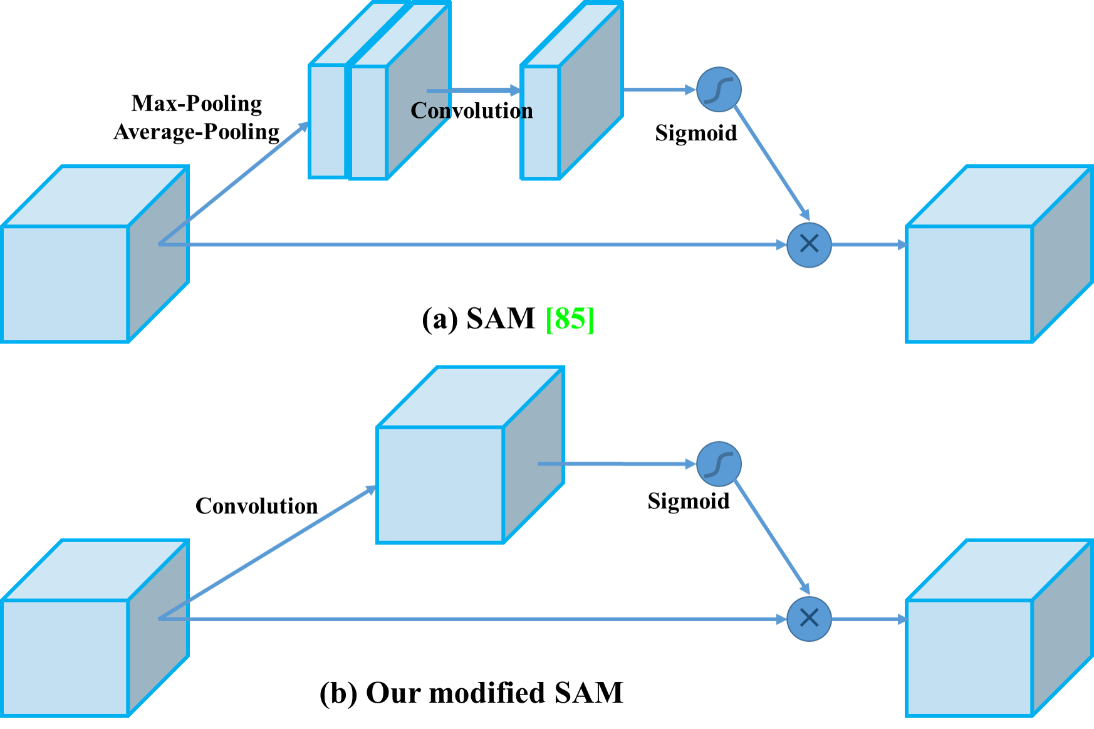
**Mosaic represents a new data augmentation method that mixes 4 training images. Thus 4 different contexts are mixed, while CutMix mixes only 2 input images.** This allows detection of objects outside their normal context. In addition, batch normalization calculates activation statistics from 4 different images on each layer. This significantly reduces the need for a large mini-batch size.

**Self-Adversarial Training (SAT) also represents a new data augmentation technique that operates in 2 forward backward stages.** In the 1st stage the neural network alters the original image instead of the network weights. In this way the neural network executes an adversarial attack on itself, altering the original image to create the deception that there is no desired object on the image. In the 2nd stage, the neural network is trained to detect an object on this modified image in the normal way.



**Figure 4: Cross Mini-Batch Normalization.**

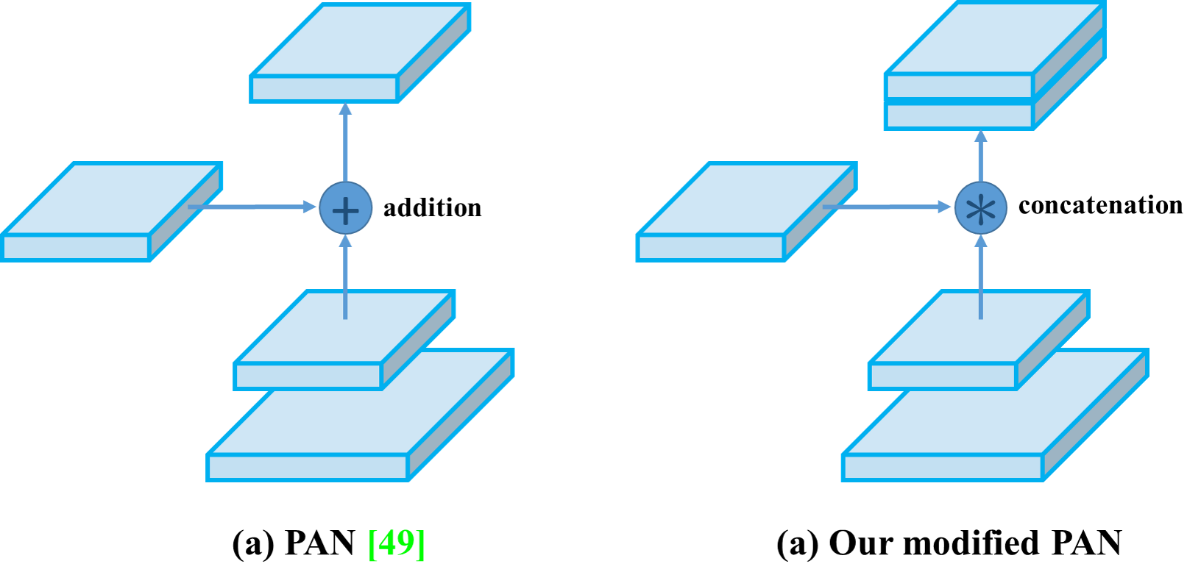
CmBN represents a CBN modified version, as shown in Figure 4, defined as Cross Mini-Batch Normalization (CmBN). This collects statistics only between mini-batches within a single batch. We modify SAM from spatial-wise attention to pointwise attention, and replace shortcut connection of PAN to concatenation, as shown in Figure 5 and Figure 6, respectively



**Figure 5:**

**Modified**

**SAM**



**Figure 6: Modified PAN**

1. **YOLOv4**

In this section, we shall elaborate the details of YOLOv4. It consists of:

1. **Backbone:** CSPDarknet53 [81]
2. **Neck:** SPP [25], PAN [49]
3. **Head:** YOLOv3 [63]
   1. **Uses of YOLOv4:**
4. **Bag of Freebies (BoF) for backbone:** CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing
5. **Bag of Specials (BoS) for backbone:** Mish activation, Cross-stage partial connections (CSP), multi input weighted residual connections (MiWRC)
6. **Bag of Freebies (BoF) for detector:** CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, eliminate grid sensitivity, using multiple anchors for a single ground truth, Cosine annealing scheduler [52], Optimal hyperparameters, Random training shapes
7. **Bag of Specials (BoS) for detector:** Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS.
8. **Advantages**
9. It’s incredibly fast and can process 45 frames per second to 150 frames per second.
10. YOLO also understands generalized object representation.
11. The network is able to generalize the image better.
12. **Disadvantages**
13. Comparatively low recall and more localization error compared to Faster R\_CNN.
14. Struggles to detect close objects because each grid can propose only 2 bounding boxes.
15. Struggles to detect small objects.
16. **Conclusion**

We offer a state-of-the-art detector which is faster (FPS) and more accurate (MS COCO AP50...95 and AP50) than all available alternative detectors. The detector described can be trained and used on a conventional GPU with 8-16 GB-VRAM this makes its broad use possible. The original concept of one-stage anchor-based detectors has proven its viability. We have verified a large number of features, and selected for use such of them for improving the accuracy of both the classifier and the detector. These features can be used as best-practice for future studies and developments.

**REFRENCE AND LINKS**

**MOST OF THE CONTENT IS FROM THE RESEARCH PAPER WHCH HAVE BEEN PROVIDED TO US.**

**[ ] THESE BOXES CONTAINS THE LINK NUMBER, WHICH HAVE BEEN PROVIDED IN THE RESEARCH PAPER**

**LINKS OTHER THAN RESEARCH PAPER:**

<https://www.google.com/search?q=Figure+1:+Comparison+of+the+proposed+YOLOv4+and+other+state-of-the-art+object+detectors.+YOLOv4+runs+twice+faster+than+efficient+with+comparable+performance.+Improves+YOLOv3%E2%80%99s+AP+and+FPS+by+10%25+and+12%25,+respectively&rlz=1C1JZAP_enIN871IN871&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiy56bjnLnxAhUOzTgGHYgECF0Q_AUoAnoECAEQBQ&biw=1536&bih=754&dpr=1.25#imgrc=MCtWVsw30NF7mM>

<https://www.google.com/search?q=yolo+prediction&rlz=1C1JZAP_enIN871IN871&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjjqqXDuLnxAhV573MBHRFMC1YQ_AUoA3oECAEQBg&biw=1536&bih=754#imgrc=-iUfGHNlOy6sdM>

<https://www.google.com/search?q=yolo+prediction&rlz=1C1JZAP_enIN871IN871&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjjqqXDuLnxAhV573MBHRFMC1YQ_AUoA3oECAEQBg&biw=1536&bih=754#imgrc=-iUfGHNlOy6sdM> <https://www.google.com/search?q=problem+faced&tbm=isch&ved=2ahUKEwik-8XEuLnxAhXHs0sFHQJ5A1kQ2-cCegQIABAA&oq=problem+faced&gs_lcp=CgNpbWcQAzICCAAyAggAMgIIADICCAAyAggAMgIIADICCAAyAggAMgIIADICCAA6BAgAEBg6BAgAEEM6BQgAELEDOggIABCxAxCDAToHCAAQsQMQQzoGCAAQChAYULDhPFiYkj1gm5Y9aAFwAHgFgAHdBIgBrymSAQwwLjQuMTAuMi4xLjKYAQCgAQGqAQtnd3Mtd2l6LWltZ7ABAMABAQ&sclient=img&ei=a0nZYOSZMsfnrtoPgvKNyAU&bih=754&biw=1536&rlz=1C1JZAP_enIN871IN871#imgrc=kLn0-68KOFTyHM&imgdii=JBO4g2uYblWYXM>

<https://www.google.com/search?q=solution&tbm=isch&ved=2ahUKEwik-8XEuLnxAhXHs0sFHQJ5A1kQ2-cCegQIABAA&oq=solution&gs_lcp=CgNpbWcQAzIFCAAQsQMyBQgAELEDMgIIADICCAAyAggAMgIIADICCAAyAggAMgIIADICCAA6BAgAEBg6CAgAELEDEIMBOgQIABBDUPu4TViH6k1g9e5NaAJwAHgEgAGJA4gBthySAQgwLjEuMTIuMpgBAKABAaoBC2d3cy13aXotaW1nsAEAwAEB&sclient=img&ei=a0nZYOSZMsfnrtoPgvKNyAU&bih=754&biw=1536&rlz=1C1JZAP_enIN871IN871#imgrc=9Dc-4F0X8-tlKM>

<https://www.researchgate.net/figure/Parameters-of-neural-networks-for-image-classification_tbl1_340883401>

<https://www.google.com/search?q=advantages+of+yolo+predictor&rlz=1C1JZAP_enIN871IN871&oq=advantages+of+yolo+predictor&aqs=chrome..69i57j33i10i160.8915j0j15&sourceid=chrome&ie=UTF-8>

<https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/>

**GITHUB**

<https://github.com/saumyaarora80/YOLO-Predictor>