**State-Of-The-Art Object Detection Models**

Cagri Yoruk

Boston University

**Abstract**

Introduction to state-of-the art object detection models and comparisons between the similar and different models. This report doesn’t involve every state-of-the-art model. The models selected are generally updated versions of their previous model. The models we are going to consider for this report are Yolo, Faster R-CNN, Mask R-CNN and Rotated Mask R-CNN. The report includes comparisons of mAP(mean Average Precision), speed and conceptual differences between models.

**1. Introduction**

Recent improvements in object detection models are influenced by the understanding of different concepts such as region proposal networks. Also, we have lots of new different datasets now, compared to back then. We are going to look at 4 models, and the conceptual similarities and differences between them. I decided to select 3 R-CNN models to compare and show the results of newer R-CNN models.

**2. Frameworks**

We are going to analyze and take a look at the models considerably high level. This includes bounding box prediction, class prediction and the system design.

**2.1. YOLO**

You only look once (YOLO) is a real-time object detection system. On a Pascal Titan X it processes images at 30 FPS and has a mAP of 57.9% on COCO dataset. Comparison to other models it is fast and accurate. Even though there are other models on par like Focal Loss, YOLO is about 4 times faster, Moreover, it is easy to trade-off between speed and accuracy simply by changing the model size without need to train the model again.

Their system predicts the bounding boxes using dimension clusters as anchor boxes. The network predicts 4 coordinates for each bounding box. They predict the objectness score of each bounding box with logistic regression.

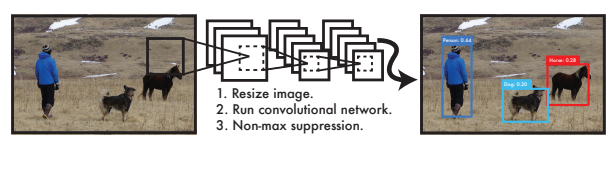


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model’s confidence. [1]

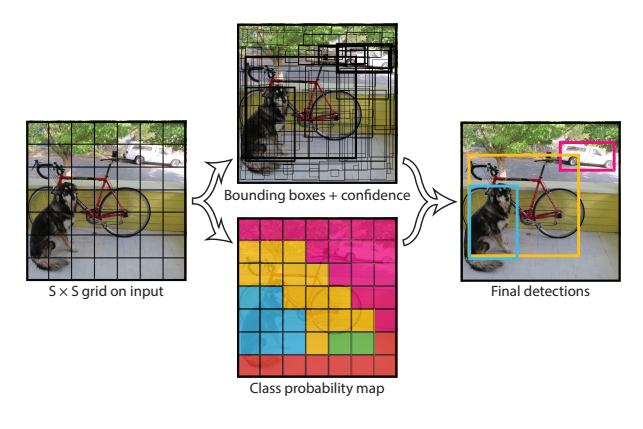


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. [1]

The system predicts the classes of the bounding box with multilabel classification. They do not use a softmax which is a normalized exponential function that distributes the probability. Instead, they use independent logistic classifiers. This is because for more complex datasets, there may be overlapping labels, and using softmax appoints the assumption that each bounding box has exactly one class, which is often not the case. For this purpose, they use multilabel approach to better model the data.

Also take feature maps from earlier in the network and merge it with the up sampled features using concatenation. [1]

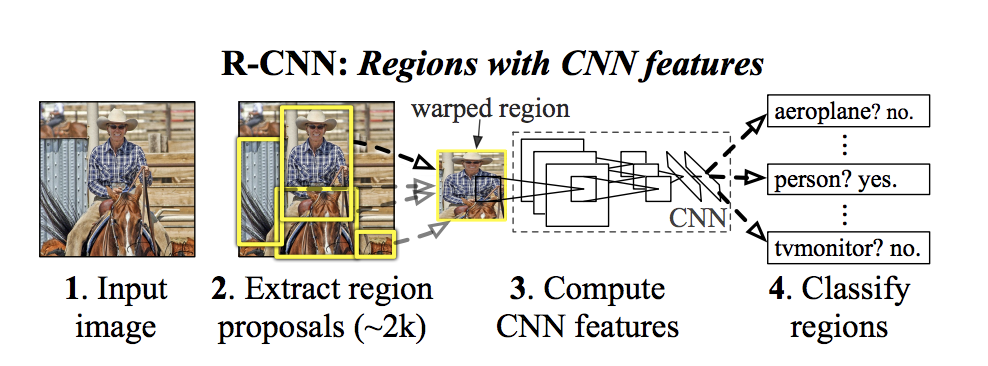


Figure 3: R-CNN architecture. [2]

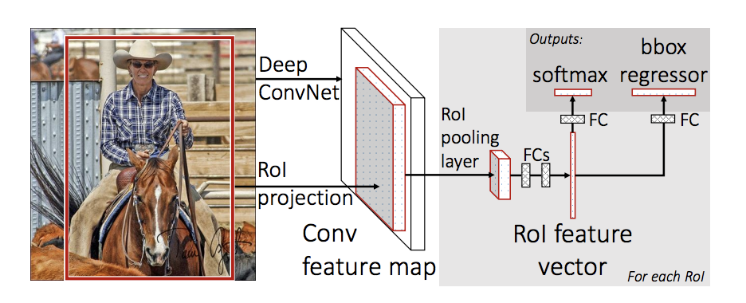


Figure 4: Fast R-CNN architecture. [3]

With this they get more meaningful semantic information and finer-grained information from the earlier feature map.

YOLOv3 predicts boxes at 3 different scales. Their system uses a similar concept to feature pyramid networks. From the base feature extractor, they add several convolutional layers. Then they take the map from 2 layers previous and up sample it by 2. They

For their feature extractor, they use a new network called darknet-53 which has 53 convolutional layers. This network is a hybrid of their earlier network from YOLOv2 and darknet-19. Compared to earlier networks it is more powerful and faster. They also achieved the highest measured floating-point operations per second. This means the network structure better utilizes the usage of GPU’s, making more efficient to evaluate and thus faster.

In terms of COCO’s mean AP metric, we can see that YOLOv3 is on par with other models. If we must look at the downside, with the new multi-scale predictions, it has comparatively worse performance on medium and big object detections. [7]

**2.2. Faster R-CNN**

Since I didn’t choose to analyze the R-CNN model, (Region Based Convolutional Neural Network) we can take a quick introduction to it and work our way toward Faster R-CNN.

The purpose of R-CNN’s is to solve the problem with bounding box problems. Given a certain image, we want to be able to draw bounding boxes over all the objects. The process can be considered as 2 steps. The region proposal and the classification step.

For an input image, selective search performs the function of generating 2000 different regions that have the possibility to contain an object. This is called extracting region proposals. After this function, the proposals are warped into an image size that can be fed to the trained convolutional neural network, which is AlexNet in this case, that extracts a feature vector for each region. Then this vector is fed to a linear Support Vector Machines to classify the image. Beside this there is also a step to increase the precision of the bounding box with prediction offset values. [2] [5]

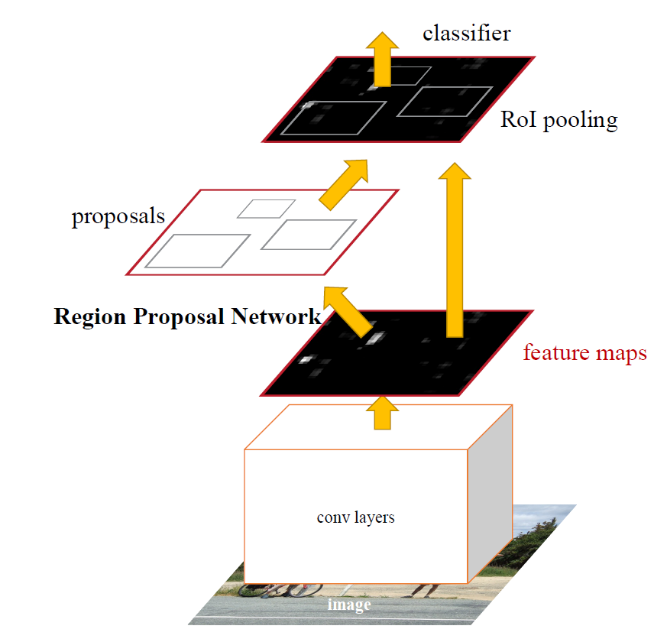


Figure 5: Faster R-CNN is a single, unified network

for object detection. The RPN module serves as the

‘attention’ of this unified network. [4]

Even though it may sound good, but there were some side effects of this new model. The network takes a huge amount of time because of the classifying 2000 region proposals per image. Also, it wasn’t good to work in real-time since it takes 47 seconds to test each image. Because of these consequences there was a need for a new model and Fast R-CNN came up.

The same author of the previous paper solved some of the drawbacks to build a faster model and it was called Fast R-CNN. The difference in this algorithm was,

instead of feeding the extracted region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. We fed the convolutional feature map to RoI (Region of Interest) pooling layer that returns fixed size squares as the output, so that they can fed to fully connected layers. From the RoI feature vector, there is a softmax function layer to predict the class of the proposed region and the offset values for the bounding boxes. [3]

Both of R-CNN and Fast R-CNN uses selective search which is a slow and time-consuming process that affects the network adversarial.

Faster R-CNN is an object detection system composed of two models. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector that uses the proposed regions. The entire system is a single, unified network for object detection (Figure 5).

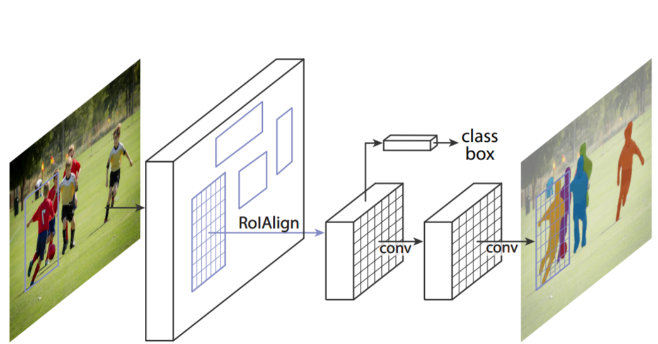


Figure 6: Mask R-CNN network for instance segmentation [8]

Using neural network with ‘attention’ mechanisms, the RPN module tells the Fast R-CNN module where to look enabling easier learning and higher quality. Like Fast R-CNN, the image is provided as an output to a convolutional neural network that provides a convolutional feature map. [4] [6]

**2.3. Mask R-CNN**

Mask R-CNN is conceptually very similar to Faster R-CNN. Faster R-CNN has two outputs for each candidate object, a class label and a bounding box offset. Mask R-CNN has one more branch that is distinct from the other branches. This third branch outputs the object mask on each Region of Interest (RoI), in parallel with other existing branches (Figure 6.) The mask branch is a small FCN applied to each RoI, predicting a segmentation mask in a pixel to pixel manner.

Mask R-CNN adopts the same two-stage procedure, with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. This contrasts with most recent systems, where classification depends on mask predictions. Our approach follows the spirit of Fast R-CNN that applies bounding-box classification and regression in parallel.

A mask encodes an input object’s spatial layout. Thus, unlike class labels or box offsets that are inevitably collapsed into short output vectors by fully-connected (fc) layers, extracting the spatial structure of masks can be addressed naturally by the pixel-to-pixel correspondence provided by convolutions.This pixel-to-pixel behavior requires our RoI features, which themselves are small feature maps, to be well aligned to faithfully preserve the explicit per-pixel spatial correspondence. This was the motivation to develop RoIAlign layer that plays a key role in mask prediction.

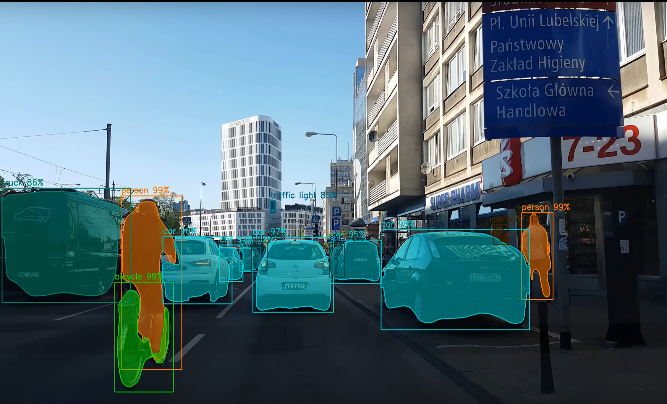


Figure 7: Mask R-CNN Implementation [9]

RoIPool is a standard operation for extracting a small feature map (e.g., 7×7) from each RoI. RoIPool first quantizes a floating-number RoI to the discrete granularity of the feature map, this quantized RoI is then subdivided into spatial bins which are themselves quantized, and finally feature values covered by each bin are aggregated.

Quantization is performed on a continuous coordinate x by computing [x/16], where 16 is a feature map stride and [·] is rounding; likewise, quantization is performed when dividing into bins (e.g., 7×7). These quantizations introduce misalignments between the RoI and the extracted features. While this may not impact classification, which is robust to small translations, it has a large negative effect on predicting pixel-accurate masks.

To address this, we propose an RoIAlign layer that removes the harsh quantization of RoIPool, properly aligning the extracted features with the input. Our proposed change is simple: we avoid any quantization of the RoI boundaries 3 or bins (i.e., we use x/16 instead of [x/16]). We use bilinear interpolation [22] to compute the exact values of the input features at four regularly sampled locations in each RoI bin and aggregate the result. [9]

**2.4. Rotated Mask R-CNN**

Rotated Mask R-CNN is an implementation of Mask R-CNN. Due to bounding box ambiguity, Mask R-CNN fails in relatively dense scenes with objects of the same class, particularly if those objects have high bounding box overlap. In these scenes, **both recall (due to NMS) and precision (foreground instance class ambiguity)**are affected.

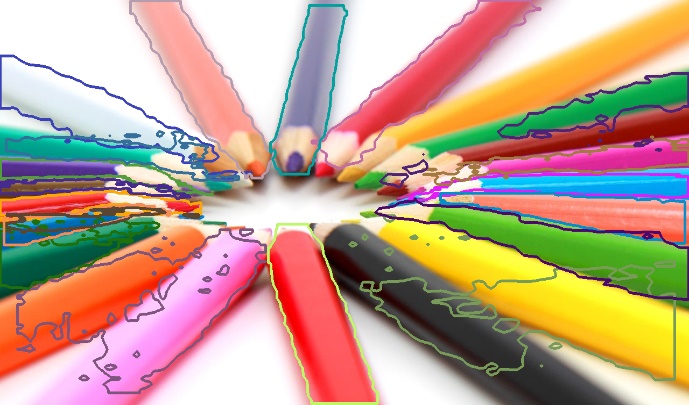


Figure 8: Mask-RCNN failing bounding box [10]

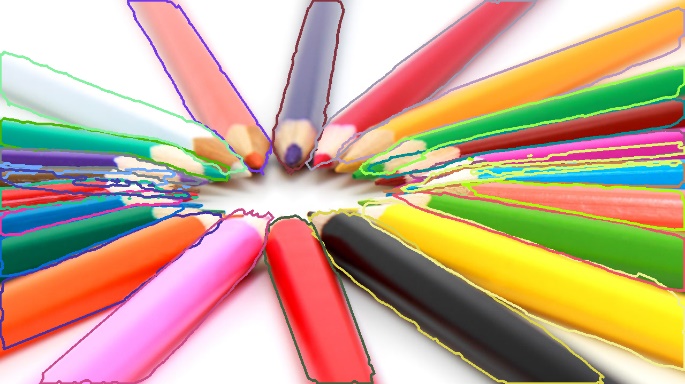


Figure 9: Rotated Mask R-CNN implementation [10]

MaskRCNN takes a bounding box input to output a single foreground (instance) segmentation per class. The hidden assumption here (as is common in many detection networks) is that a good bounding box contains just one object in that class. This is not the case for dense scenes like the pencil image above.

Unfortunately, such scenes are underrepresented in the most popular instance segmentation datasets - MSCOCO, Pascal VOC, Cityscapes. Yet they are not uncommon in many real-world applications e.g. robotics/logistics, household objects. As a result, the author of the project released **a simple, small dataset called PPC - Pens, Pencils, Chopsticks** and show the significant difference between Mask R-CNN and Rotated Mask R-CNN in such scenes. **Rotated Mask R-CNN**resolves some of these issues by adopting a rotated bounding box representation. [10]

**3. Recommendations**

We looked at different object detection frameworks, I think the most convenient ones are YOLO and Rotated Mask R-CNN. Since Rotated Mask R-CNN is a derived version from Mask R-CNN and Faster R-CNN, I think there are some improvements with edge cases such like household objects and dense scenes with objects. Because of these facts I would like to build on top of Rotated Mask R-CNN instead of different versions of R-CNN. As for YOLO, I think this is the best model that is implementable for real-time. The mAP is similar with the best models out there, but it is couple times faster. If we look at the fact that it works on 30FPS, I think it is convenient to use.

**4. Conclusions**

We looked at different object detection models. Compared them by their architecture, system performance and the implementation styles. More detailed explanation and implementations can be found in references. We can clearly see that in different times there are needs for different edge cases. Most of the models were derived from each other by solving these edge cases. I think the most challenging problem is real-time since autonomous systems make decision based on real-time. The object detection model should be fast to keep up with real-time encounters. Because of these factors I think speed is one of the important aspects of these models.

**References**

[1] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition (CVPR), 2017

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[3] R. Girshick, “Fast R-CNN,” in IEEE International Conference on Computer Vision (ICCV), 2015.

[4] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[5] Rohith Gandhi, R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>

[6] Umer Farooq, from R-CNN to Mask R-CNN <https://medium.com/@umerfarooq_26378/from-r-cnn-to-mask-r-cnn-d6367b196cfd>

[7] J. Redmon and A. Farhadi YOLOv3: An Incremental Improvement 2019.

[8] Kaiming He, Georgia Gkioxari, Piotr Dollar, Ross Girshick - “Mask R-CNN”.

[9] Karol Majek <https://github.com/karolmajek/Mask_RCNN>

[10] Shijie Looi – Rotated Mask R-CNN

<https://github.com/mrlooi/rotated_maskrcnn>