Multiple Object Recognition with Focusing and Blurring

Holly Chiang
Stanford University
hchiang1@stanford.edu

Yifan Ge
Stanford University
gyifan@stanford.edu

Connie Wu
Stanford University
wuconnie@stanford.edu

Abstract

In this project we use CNNs to identify significant objects in a scene for applications in photo or video editing. We tackle this problem with two different approaches: faster R-CNN and YOLO. Faster R-CNN is used to more accurately identify objects in a scene and then classify those regions to label object categories. YOLO is used to identify and classify videos that need to be processed in real time, trading off accuracy for speed. We then use those labels to determine the significance weighting of the objects of the image to isolate people or objects of greatest importance, so that we can apply image processing techniques to make those objects stand out. Faster R-CNN and YOLO are both trained on the VOC 2007 dataset.

1. Introduction

Locating and identifying multiple items in an image is something that is still difficult for machines to accomplish. However, significant work has been made in the last few years on object detection with convolutional neural networks(CNNs). In this paper, we apply CNNs to object detection in video, with the aim of creating a detector that can identify the most important objects in a scene for possible applications in photography or video editing.

The input to our detector is an unprocessed photo or video. Depending on the application we then use either faster R-CNN or YOLO to detect and classify the objects in each image so that we can then rank each object in terms of importance and apply image processing filters to make the most important objects stand out. The output of our detector is the processed photo or video.

As an instance of the utility of such a detector: If there is a video of a sporting event, the detector will be able to locate the moving athletes so that we can apply video processing techniques to reduce the distraction from changing background scenery. Another instance is if you have a photo a target person of interest in front of a famous landmark but there are too many tourists in the background, our detector will be able to determine that the person and the landmark

Figure 1: Example of Bokeh (2) vs Blurring (3)

are the most significant objects in the picture, and apply photography techniques to such as bokeh (Seen in Figure 1) or blur to reduce the background noise. Bokeh with focus on multiple objects, in particular, is very difficult to achieve in the real world because cameras can only have one depth of view for focusing. Therefore, if we can identify the important objects' bounding boxes, we can theoretically focus and blur multiple objects with a bokeh effect that is impossible to do otherwise.

To accomplish our goal, we have to address several challenges. The first challenge we face is in training a model capable of accurately locating and classifying a wide range of object classes. The second challenge is creating a detector capable of processing videos in real time. We describe how we address these challenges in the following paper.

2. Related Work

To allow our detector to work for a variety of different image or video applications, we look at two different detection algorithms: YOLO and Faster R-CNN. YOLO divides the image into a grid and within each grid cell predicts bounding boxes and their confidence scores as well as conditional class probability predictions for each cell. As each bounding box can only predict one class, YOLO suffers from errors when there are many small objects inside one bounding box [1].

Faster R-CNNs work better than previous generations of R-CNNs due to the addition of a Region Proposal Net-

work (RPN) that reduces the bottleneck on region proposal time by identifying regions and their respective scores at the same time. The detection is done using Fast R-CNN with the RPN. For the VGG-16 model, Faster R-CNN has a frame rate of 5fps, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. [2, 3, 4].

The two algorithms have a trade off between speed and accuracy so each algorithm has specific applications it is better suited for. YOLO can process images with mean average precision (mAP) of 63.4 at 45 FPS on the PASCAL VOC 2007 dataset, which is fast enough to apply to real time video processing applications. On the other hand, Faster RCNN achieves a higher accuracy of 73.2 mAP, but at only 7 FPS [1].

We looked to previous work on ranking the significance of objects in an image. One paper used Mechanical Turk to human annotate the relative significance of objects in an image [5]. Other papers used text labeled images: one paper used the UIUC Pascal Sentence data set (UIUC), which has sentence descriptions for each image [6], while another work made use of Mechanical Turk to add text labels to image datasets [7]. While these approaches were promising, Mechanical Turk was beyond the capability and funding of this project, while using descriptors to determine the relative importance of objects requires additional NLP techniques to correlate the text description to image objects.

3. Methods

To approach this problem, we divided it into four major steps listed below.

1. Object detection: This step involves using Convolutional Neural Networks to detect the objects of interests inside images. We apply two different models: YOLO and Faster R-CNN, depending on the specific application.

2. Significance ranking: With the observed objects from previous step, we developed a ranking scheme to assign importance to each object. This gives us the flexibility to choose the number of objects we would like to focus on.

3. Focusing and blurring: This step applies image processing techniques to focus on the most important objects and blur out the rest of the image with either vignette, blur, or bokeh, using the identified object bounding boxes.

4. Optimization for video rendering: The final step is to enable the program to process video frames. To avoid large fluctuations of the bounding box sizes in the rendering process, we needed to make sure the completed program is efficient enough to provide reasonable frame rates, which we achieved with YOLO.

Faster R-CNN vs. YOLO		
Metrics	Faster R-CNN	YOLO
FPS	7	45
mAP	73.2	63.4
Number of bboxes	RPN	98

Table 1: There is a tradeoff between speed and accuracy between the two methods

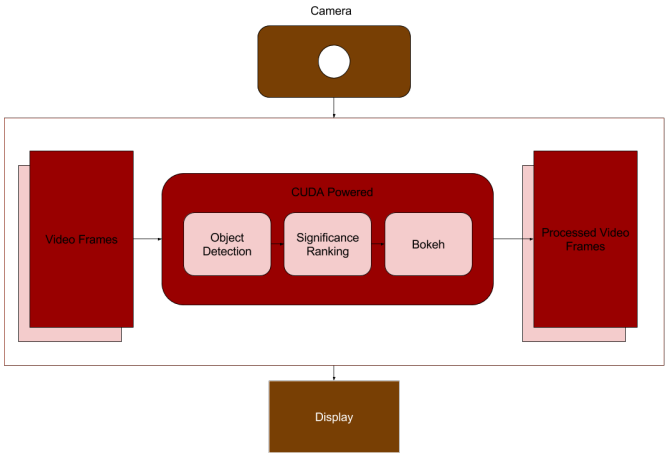


Figure 2: System Diagram

As object detection is the key and the first step in our application, we started by investigating the two proposed algorithms: YOLO and Faster R-CNN.

3.1. YOLO

YOLO is an improved model over Faster R-CNN, with respect to faster processing speed. On a Titan X GPU, The base network runs at 45 frames per second with no batch processing and a fast version runs at more than 150 frames per second. [1]. This makes it possible to process video streams in real-time with less than 25 milliseconds of latency.

3.1.1 Design and Architecture

YOLO uses unified detection which unites the separate components of object detection into a single neural network. This network uses features from the entire image to predict each bounding box. This design differentiates YOLO from

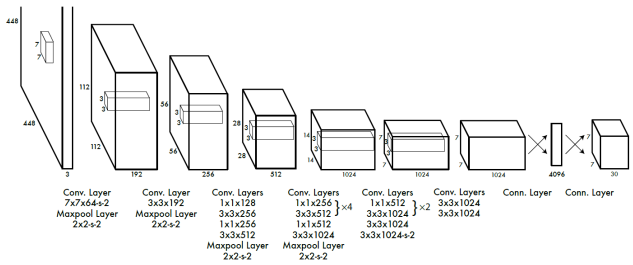


Figure 3: YOLO Architecture Diagram [1]

other methods and enables end-to-end training with real-time speeds while maintaining high average precision. [1] Unified detection system uses S x S grid cells and predicts B bounding boxes for each cells. Each bounding boxes produces 5 predictions: x, y, w, h, and confidence. The (x, y) is the coordinates of the center of box relative to the bounds of the grid cell. (w, h) is the width and height of the bounding box. The confidence score is calculated by the probability of the object multiplied by the intersection of the union of ground truth and prediction labels. Then each grid cell produce the conditional class probabilities. These probabilities are used at test time to produce the final predictions. The final class-specific confidence scores are calculated by multiplying the conditional class probabilities with the individual box confidence predictions.

The neural network architectural design includes two main components: feature extraction and prediction. The feature extraction consists of 24 convolutional layers to obtain the final 7x7x1024 activation map. This image information is passed to prediction section that has 2 fully connected layers. The architecture diagram is shown in Figure 3. The probability and coordinates are produced by the last FC layer in the 7 x 7 x 30 tensor of predictions.

3.1.2 Implementation

YOLO is implemented on Darknet, an open source neural network framework written in C and CUDA. This implementation of YOLO takes advantage of GPU computing power to process video streaming at real-time. Darknet platform includes CUDA kernel implementations of most of the layers: activation layer, average pool layer, convolutional layer, crop layer, and deconvolutional layer. The video screening subroutine is also implemented in CUDA kernel. The program launches two threads at run-time: fetch thread and detect thread. Each thread runs in parallel and converge after each image. This effectively double buffers the input frames. To speed up the processing, Darknet has its own image class to do the image manipulation. OpenCV is used as a user interface wrapper to obtain the images, display, and save the images.

In our implementation, we took advantage of Darknet platform by creating our own processing CUDA kernel. The CUDA kernel is implemented based on the original video screaming demo code. We used double frame buffering by launching two threads: fetch thread, detect and bokeh thread. This implementation achieved around 20 frame per second.

3.2. Faster R-CNN

The second approach we tried is Faster R-CNN, which achieves higher accuracy benchmarks than YOLO, but performs at fewer frames-per-second. A huge bottleneck in object detection is the regional proposal process. A set of bounding boxes needs to be computed so that the model can compute image scores of the image in the bounding box to determine the object class.

3.2.1 Object Detection

In Faster R-CNN, it computes the object scores in a region proposal and thresholds the object detection. In our implementation we varied the threshold from 0.4-0.8 and found it performed best around 0.7.

3.2.2 Region Proposals

In R-CNN for object detection, typically we can do region proposals in two ways: selective search and with a region proposal network (RPN). In selective search, regions are proposed based on color, size, fill, and texture. It has a high bounding box recall of around 0.98-0.99 but slows down the overall speed because it operates independently from the network for object detection and has to be run at test time, whereas the run-time of the RPN at test-time is just 10 milliseconds [3].

In the RPN, the convolutional neural network for object detection is shared with the regional proposal task based on low-dimensional convolutional features, this is why Faster R-CNN is better than Fast R-CNN because it reduces the time caused by the region proposal bottleneck. The region proposal method can be visualized in Figure 4. Therefore for our implementation we enabled the RPN.

First to generate region proposals, the RPN slides a small network over the convolutional feature map output by the last shared convolutional layer in squares to create the low-dimensional features [3]. Then, the convolutional neural network for object detection is shared with the regional proposal task based on low-dimensional convolutional features, this is why Faster R-CNN is better than Fast R-CNN because it reduces the time caused by the region proposal bottleneck. The region proposal method can be visualized in Figure 4. Therefore for our implementation we enabled the RPN.

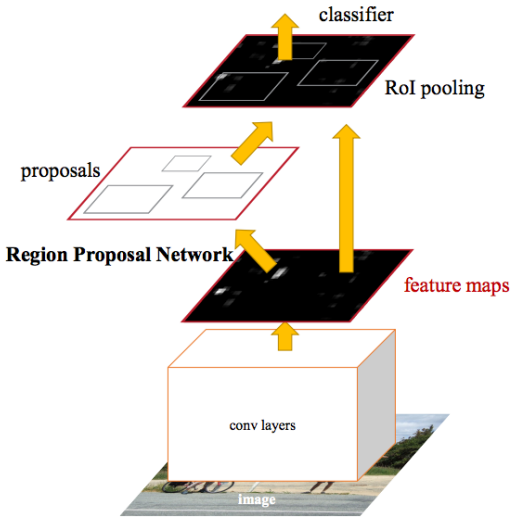


Figure 4: Regional Proposal Network on Features

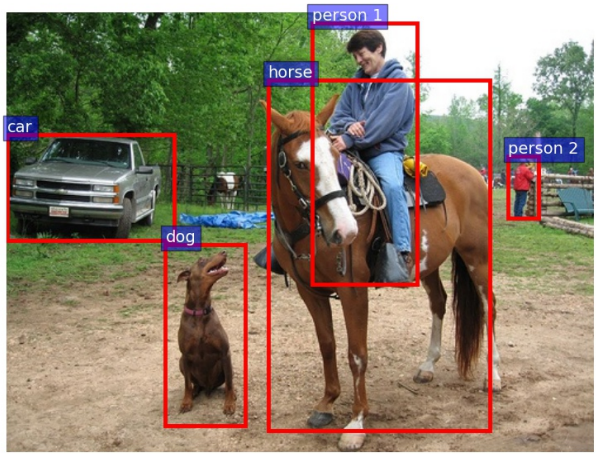


Figure 5: Image for detecting significance ranking

3.3. Significance Ranking

Ranking the importance of objects in images is somewhat subjective and past studies that aimed to rank objects used human annotation through Mechanical Turk to generate training and test datasets. As we did not have access to such datasets, we chose the best performing features noted in previous papers, and heuristically determined the relative weights for each feature. Features that had some of the highest correlation to importance prediction were area percentage and distance to center. The area percentage of an object was calculated as the fraction of its area to the total detected object area. Distance to center was calculated as an Euclidean distance from the center of a detected object to the center of the image, normalized by the greatest such distance.

Area percentage:

$$\frac{A_i}{\sum_{j \in objects} A_j}$$

Distance from center:

$$d^{(i)2} = (x_c - x_i)^2 + (y_c - y_i)^2$$

$$d^{(i)}_{norm} = \frac{d^{(i)}}{\max_j(d^{(j)})}$$

In Figure 5 we were able to get rather accurate ranking of the importance of each detected object, as shown in Table 2. We are able to detect the Horse, Person 1, and the Dog as the first, second, and third most important objects respectively. Between the three objects the significance ranking may vary based on the subjective opinion of the viewer, but the given ranking is one that most people will agree is reasonable.

3.4. Image Processing

We implemented three different image processing techniques to be used on an image or video. Using the bounding boxes detected by either faster R-CNN or YOLO we applied either vignette-ing, blurring, or introducing bokeh to the background of the image. Vignette was implemented using a dark mask while blurring was implemented as a gaussian filter. To simulate the bokeh effect we applied a gaussian filter followed by randomly selecting pixels to enlarge into circles, followed by another gaussian layer. For all three filters we used gaussian filters to soften the edges between the unfiltered objects and the background.

One complication to applying filters to the images is dealing with multiple detected objects. We prevent filter overlaps by keeping track of the object regions that should not be processed.

The major difficulty with applying filters is differentiating the object from its background. We tried multiple approaches such as segmenting by color or by edges, but we did not get good results, and those approaches had a hard time dealing with object overlaps. We ended up going with a simple oval shape using the bounding box dimensions, which generalizes well across all object classes.

4. Dataset and Features

For YOLO and Faster R-CNN we were able to use both the VOC 2008 and ILSVRC2013 datasets. For VOC we were able to use the pre-trained models. But for ILSVRC2013, we obtained the detection data which included 19,812 images. We used 15,848 images for training and 3,964 images for a five-fold validation on the Imagenet dataset[8].

Object	Area Percentage	Distance from Center	Importance Value	Importance Ranking
Car	0.126	0.980	0.145	4
Dog	0.103	0.631	0.472	3
Horse	0.560	0.363	1.197	1
Person 1	0.201	0.459	0.743	2
Person 2	0.011	1	0.011	5

Table 2: Results from significance ranking for Figure 5

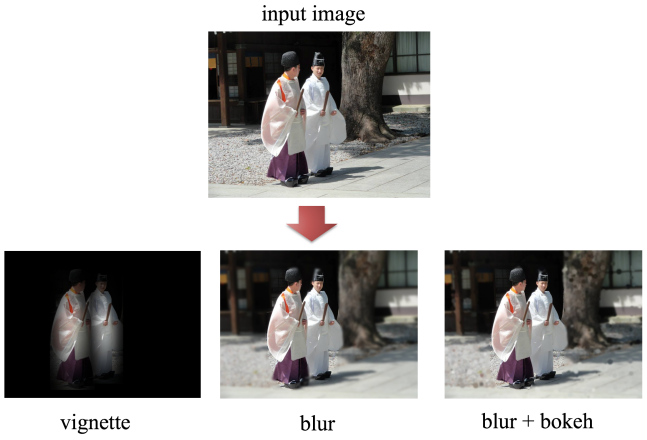


Figure 6: Bokeh Generation Process

4.1. Training and Testing of Faster R-CNN

We implemented Faster R-CNN trained on the VOC 2007 dataset in CPU mode with the VGG-16 network with Caffe and Python which had a very slow performance at around 5 seconds per image. Then on GPU mode we were able to get the timing down significantly.

Since VOC can only recognize 20 classes, we also trained Faster R-CNN on Imagenet with the VGG-16 network over 200 classes. The weights of the VGG network on Imagenet were pre-trained. But to adjust the pipeline for training with Faster R-CNN, we constructed a training database (imdb) with labeling from the synset labels.

4.2. Training and Testing of YOLO

Similarly, YOLO was also limited by 20 classes from VOC. To improve the performance, we tuned ILSVRC2013 to train YOLO. The training was done with a CUDA implementation and batch size of 64 images. We also increased filter numbers to enlarge the model to accommodate the larger number of classes. During training, each batch requires around 5-6 seconds to train. We tuned the dropout rate and filter sizes in the model and evaluated the hyperparameters set by validating on the 4000th iteration weight. However, the weights did not perform as well as expected.

5. Results

5.1. Experiments

5.1.1 Video Output Results from YOLO

For each video we processed the frames in real-time using YOLO. Below in Figure 7 we can see the output of the object detection and the subsequent bokeh effect.

5.1.2 Comparison of Faster R-CNN and YOLO

From our experiments we noticed that the bounding boxes can vary quite a bit for the same image in Faster R-CNN versus YOLO. Surprisingly, sometimes the bounding box on YOLO is more accurate than that of R-CNN. This could be because of the score thresholds of the algorithms. In YOLO, we set the score threshold at 0.2 whereas in Faster R-CNN the score threshold was 0.7. This means that unfamiliar parts of objects will not be included in the bounding box of the object detection. For example in a picture of a cat, YOLO acknowledge the entire cat, but Faster R-CNN only captured the cat's face because it had a higher class score confidence. We provide an example here of a woman and a dog in Figure 8. Faster R-CNN can detect the woman, dog, and a plant behind. Whereas YOLO can only recognize the woman. With regards to significance we see a clear trade-off here. In YOLO, we are able to get rid of the plant but also lose the dog. But keeps keeps the most important object based on class score.

5.1.3 Speed Benchmarking

In testing Faster R-CNN, it was found that there was substantial difference in performance depending on the computing device. Even between the NVIDIA GRID K520 and NVIDIA GTX 980 Ti GPUs there was still nearly a 0.4 second difference. The specifications can be seen in Table 3.

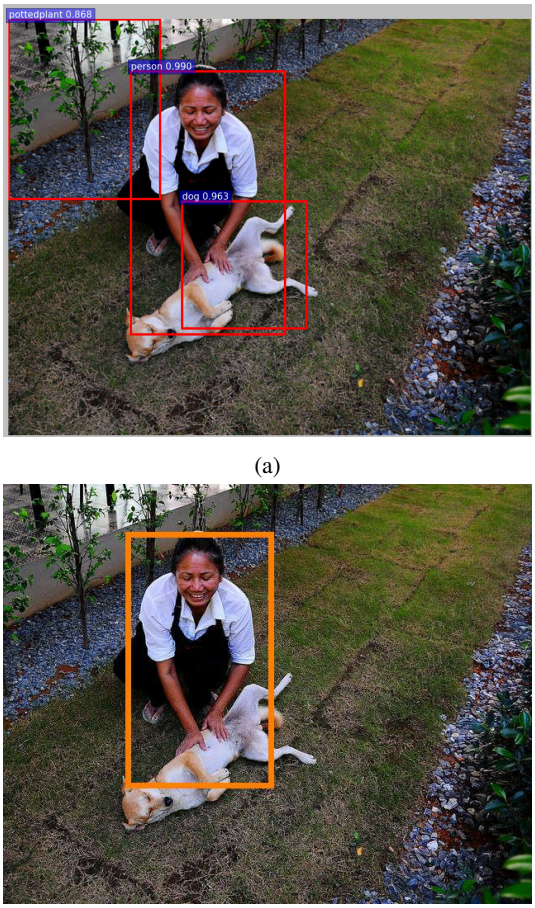
Figure 7: Sample Video Processing

Benchmarking on Hardware		
Device	# of RP	Time
2.2 GHz Intel Core i7	300	5.352s
NVIDIA GRID K520	300	0.586s
NVIDIA GTX 980 Ti	300	0.133s

Table 3: We aimed to achieve the highest speed possible for flexibility in live video processing

5.2. Analysis

For post-processing on videos where speed is not crucial, we used Faster R-CNN because from the literature review, it has higher accuracy. However, for live streaming of video we were able to confirm that using YOLO allowed for faster processing time and as a result allowed us to playback frames in real time as they were being capture through a web camera, for example.



(a)

(b)

Figure 8: Bounding Box Results for sample image. We can see that Faster R-CNN is able to detect more objects in the image. a) Faster R-CNN b) YOLO

Moreover, in YOLO there are generally fewer bounding boxes found than that of Faster R-CNN, which can help us eliminate some of the non-significant picture subjects, and as a result further reduce our processing time.

6. Conclusion

After using Faster R-CNN and YOLO, our image processing module applies image processing techniques such as the bokeh effect around the detected objects. Based on the inherited advantages from Faster R-CNN and YOLO, we implemented two pipelines to process pre-recorded videos and real-time videos, respectively. In Faster R-CNN implementation, we were able to detect objects more consistently and produce the video with less fluctuation. On the other hand, the YOLO implementation produced real-time

processed video at 20 frames per second, at slightly lower accuracy.

Using the detected object bounding boxes, our image processing module creates a near bokeh effect around the detected objects. When multiple objects are detected, our heuristic ranking algorithm determines the top objects to apply bokeh.

One area we can improve on in the future is implementing better image segmentation. One option is to use customized templates for each object class. Semantic or instance segmentation. For example, for each region's highest object class score, we can create a template of the average of images in that class, then use edge detection or color variation in the template to create the approximate boundary for objects in that class. However, that method would require overcoming deviations from the average object outline in such an image. Another option is to implement semantic or instance segmentation, to more accurately and adaptively segment the objects from the background. This may require additional optimization to be able to apply it to video processing in real time.

Last but not least, we could also improve on the number of detection classes. Although we have started training Faster R-CNN and YOLO near the end of project, some fine tuning on the hyper parameters are still needed to achieve a better detection result.

References

[1] Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." arXiv preprint arXiv:1506.02640 (2015).

[2] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[3] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[4] Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." Advances in Neural Information Processing Systems. 2015.

[5] Kong, Yan, et al. "Measuring and Predicting Visual Importance of Similar Objects." (2016).

[6] Berg, Alexander C., et al. "Understanding and predicting importance in images." Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012.

[7] Hwang, Sung Ju, and Kristen Grauman. "Learning the relative importance of objects from tagged images for retrieval and cross-modal search." International Journal of Computer Vision 100.2 (2012): 134-153.

[8] Deng, Jia, et al. "Imagenet: A large-scale hierarchical image database." Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. IEEE, 2009.

[9] Gidaris, Spyros, and Nikos Komodakis. "Object Detection via a Multi-Region and Semantic Segmentation-Aware CNN Model." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[10] Liang, Xiaodan, et al. "Proposal-free network for instance-level object segmentation." arXiv preprint arXiv:1509.02636 (2015).