Object Detection based on Convolutional Neural Network

Shijian Tang
Department of Electrical Engineering
Stanford University
sjtang@stanford.edu

Ye Yuan
Department of Computer Science
Stanford University
yy0222@cs.stanford.edu

Abstract

In this paper, we develop a new approach for detecting multiple objects from images based on convolutional neural networks (CNNs). In our model, we first adopt the edge box algorithm to generate region proposals from edge maps for each image, and perform forward passing of all the proposals through a fine-tuned CaffeNet model. Then we get the CNNs score for each proposal by extracting the output of softmax which is the last layer of CNN. Next, the proposals are merged for each class independently by the greedy non-maximum suppression (NMS) algorithm. At last, we evaluate the mean average precision (mAP) for each class. The mAP of our model on PASCAL 2007 test dataset is 37.38%. We also discuss how to further improve the performance based on our model in this paper.

1. Introduction

Convolutional neural networks (CNNs) has been widely used in visual recognition from 2012 [1] due to its high capability in correctly classifying images. In [1], the authors show an extremely improvement on the accuracy of image classification in ImageNet Large Scale Visual Recognition Challenge (ILSVRC). And CNNs become the most preferable choice for solving image classification challenges. Besides image classification, researchers have extend the application of CNNs to several other tasks in visual recognition such as localization [2], segmentation [3], generating sentences from image [4], as well as object detection [5].

In our project, we mainly focus on the task of object detection which has tremendous application in our daily life. The goal of object detection is recognise multiple objects in a single image, not only to return the confidence of the class for each object, but also predict the corresponding bounding boxes.

Among most of the works in object detection, region CNNs (rCNN) [5] is the most remarkable one that combines selective search [6], CNNs, support vector machines (SVM) and bounding box regression together to provide a high performance in object detection.

In this paper, we will provide an alternative approach of object detection by reducing the complexity of the rCNN. First, we adopt edge box [7], a recent published algorithm to generate region proposals, instead of selective search used in rCNN. [8] shows that even though the mean average precision (mAP) between edge boxes and selective search are almost the same, edge boxes runs much faster than selective search. Secondly, we remove all of the class specific SVMs, and directly use the output of softmax in the last layer of CNN as our score. To compensate the possible performance degrade raised by removing SVMs, we carefully design our training data to fine-tune CNN. Fig. 1 is an overview of our model.

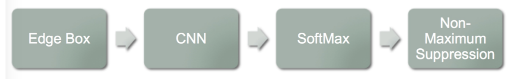


Figure 1: The overview of our object detection system. 1). Edge box algorithm generates proposals. 2). A fine-tuned CNN with object classes plus one background class to extract the CNN features for each proposal. 3) Employ the softmax to give the confidence (score) of all the classes for each proposal (which is typically the last layer of CNN). 4). For each class, independently use the non-maximum suppression (NMS) to greedily merge the overlapped proposals.

In addition to the previous model, we also develop a traditional model as our baseline. In this model, we adopt sliding window to generate proposals, and use histogram orientation gradient (HOG) features to describe such proposals. Then we use our trained linear SVM to score each proposal.

The rest of this paper can be organized as follows. In section 2, we briefly summarize the previous work in object detection and introduce the state-of-art approaches. In section 3, we will cover more details of the technical approaches in our model including the general framework, theoretical background as well as performance evaluation metrics. In section 4, we will present our experiment results and briefly discuss the results. Section 5 will conclude our work and discuss the method to further improve the performance of our model.

2. Background

Object detection becomes an attractive topic in visual recognition area in the last decade. In the early stage, people tend to design features from raw images to improve the performance of the detection. Among these features, SIFT [9] and HOG [10] features are the most successful ones. These features combined with SVMs have successfully detect the pedestrians from images. However, when these models are applied into multiple classes and object detection in a single image, the performance is not as our expected.

Instead of using linear SVMs, some other works try to use ensemble SVMs [11] as well as latent SVMs [12] based on HOG feature description. But the performance of object detection still barely improves.

People switch to focus on CNNs since [1] shows a significant improvement of classification accuracy by employing a deep CNNs. Unlike classification, the detection task also requires us to localize the object by specifying a bounding edge box. Therefore, we can not directly use CNNs in object detection without solving the localization problem. [2] tries to treat the localization problem as a regression task. But the performance has minimal improvement. Then other researchers [13] adopt the sliding window combined with CNNs as one possible solution. However, due to the exhaustive nature of sliding window with high computational complexity, this method can not be practically implemented. Even though sliding window approach can not work in practice, this method provides an idea that is solving localization by classifying region proposals of the images.

Instead of sliding window, several algorithms are also capable to generate proposals efficiently, such as objectness [14], selective search [6], category-independent object proposals [15], constrained parametric min-cuts (CPMC) [16] and edge [7]. In the rCNN paper [5], the authors choose selective search as proposal generation algorithm due to its fast computational time. In this paper, we will choose a new published algorithm named edge instead of selective search for object detection.

Other people also develop weakly supervised learning [17] and deformable CNNs [18] to detect objects last year.

3. Approach

In this section, we will cover the details of our object detection model.

3.1. Proposal generation

In this paper we will use the edge boxes as our proposal generation algorithm. The basic idea of edge boxes is that this algorithm generates and scores the proposal based on the edge map of the image. Specifically, in the first step we should generate an edge map with a structured edge detector where each pixel contains a magnitude and orientation information of the edge.

Then we group the edges together by a greedy algorithm where the sum of orientation of all the edges in the group is less than $pi/2$. After that, we will compute the affinity between two edge groups which is important for the score computation.

Next, for a given bounding box, we will compute the score of the bounding box based on the edge groups entirely inside the box.

At last, we use the non-maximal suppression to merge the bounding box to get the proposals.

Compared with the selective search which has been chosen in the rCNN, [8] shows that for VOC 2007 dataset, the mAP of edge boxes is 31.8$\%$ which is slightly larger than selective search with mAP as 31.7$\%$. One advantage of edge boxes is that the runtime is much faster than the majority of the proposal generation schemes. For edge boxes, the average runtime is 0.3 seconds while for selective search as 10 seconds.Therefore, the edge boxes decreases the time complexity without degrading the performance. Hence, we choose the edge boxes as the proposal generation algorithm.

3.2. Training procedure

In this paragraph, we will describe preparing our training data and fine-tuning caffe model. As we mentioned in section 1, for compensating the removal SVMs from the rCNN, we will carefully design our training data, especially for the background data.

In the rCNN model, the authors train both CNN and class specific SVMs using different training data set. For CNN, they choose the bounding box with IoU (intersection over union) larger than 0.5 as positive data, others as negative (background) data. For SVMs, they use the ground-truth as positive data to improve localization precision, IoU less than 0.3 as negative data and ignore all the rest of cases.

In our approach, all the training data are extracted from the raw images. As we eliminate the SVMs, we did not choose IoU as 0.5 to split positive and negative (also known as background) data for CNN, because that would decrease the performance of localization. Our schemes are as follows. For the detection problem, during the test time, the number of background proposals is typically much larger than that of the positive proposals. Hence, intuitively, we need more background data compared with positive data for training. So, we collect background data four times larger than the positive data. Note that we will not add more background data, since it will make the training dataset imbalance and, therefore, harder for a classifier to classify it.

Then we divide the background data into four folders 1,2,3,4. For folder 1, the IoU with ground truth are between 0.5 and 0.7, for folder 2, the IoU with ground truth are between 0.3 and 0.5, for folders 3 and 4, the IoU with ground truth are less than 0.3. In the case of positive data, we randomly extract a region from raw image, if the IoU with ground truth is larger than 0.7, it is a positive data with the same class label as the ground truth. The last step is shuffling all the data. By designing the training data in this way, we can suffer a precise localization without class specific SVMs.

The next step is fine-tuning the pre-trained CNN models using the generated training data as we described above. We choose the CaffeNet model as our pre-trained model. This model is a replicate of AlexNet, having 5 convolutional layers and pre-trained on the ILSVRC2012 dataset. We change the output number of last layers to 21 (for VOC 2007 dataset, 20 object classes plus 1 background class). We will describe the process and results for tuning the CaffeNet model.

3.3. Testing procedure

In the testing time, we firstly generate regional proposals by edge boxes, and then perform the forward pass for all of the proposals through the fine-tuned CNN. Note that since the input size of CaffeNet model is fixed as 227× 227 pixels, the proposals with different shapes are resized to the required shape before forward pass.

After that, we will extract the output of the softmax as a 21-element vector for each proposal, with each entry in the vector represent the confidence of the corresponding proposal in each class.

The we employ the non-maximum suppression (NMS) algorithm to ignore the redundant proposals. The basic idea of this algorithm is that sort the proposals by the confidence (also known as score), and then ignore the proposals overlaping with a higher-scored proposal. The threshold of the overlap is typically defined as the IoU between two proposals. Note that the IoU threshold will affect the performance of our detector, which should be tuned carefully to achieve the best performance.

To evaluate the performance of the detection, we use the mean average precision (mAP). The mAP equals to the integral over the precision-recall curve $p(r)$.

$$mAP = \int_0^1 p(r)dr \tag{1}$$

To determine the precision-recall curve, we need to compute the true positive and false positive value of our prediction first. We use the IoU (intersection of union) to determine the successful detection by

$$IoU = \frac{A_{pred} \bigcap A_{gt}}{A_{pred} \bigcup A_{gt}} \tag{2}$$

where A_{pred} and A_{gt} are the areas included in the predicted and ground truth bounding box, respectively. Then we designate a threshold for IoU, for example 0.5, if the IoU exceeds the threshold, the detection marked as correct detection. Multiple detections of the same object are considered as one correct detection and with others as false detections. After we get the true positive and false positive values, we can compute the precise-recall curve, and then evaluate the mAP. Details about the error analysis can be found in [19].

3.4. Baseline model

As the baseline, one approach of generating proposal we experimented is using features. [10]The process of localization and classification are separated. Firstly, HOG features, a binary-classification SVM and Non-Maximum Suppression are used for localisation, and then result boxes from the previous process will be classified by CNN.

When in the training process of the SVM, comparable numbers of object and background boxes are used. HOG features of such boxes are passed into the SVM to train the weights. SVM will determine if one box contains objects or only background.

In the testing process. HOGs of sliding windows of threes shapes and three scales are passed into the trained SVM to obtain scores on each label. Then, NMS is used to take away bounding boxes that overlap others with higher scores. The proposals remained after this process will be the localization of the objects. These region will then passed into the CNN for classification. The mAP we obtained based on this model is 22.6%.

For more detail of this model, please have a look as our CS 231A final project report.

4. Experiment Results

4.1. Dataset description

We adopt the VOC 2007 dataset in our project, which is a popular dataset for classification, detection and segmentation [20]. This dataset contains 5011 images for the task of detection and classification. All the images are divided into training set and validation set. For the training dataset we have 2501 images, and 2510 images for validation.

The objects in the dataset can be classified into 20 classes. Each image contains more than one object which are not necessarily in the same class. The total number of objects in this dataset is 12608, with 6301 in training dataset and 6307 in validation dataset. On the average, there will be 2.51 objects per image. Therefore, this dataset is an appropriate choice for detection problem. In addition, the objects can be also described by the regular objects and difficult objects. The difficult objects are typically not clearly visible such as occluded by other objects. These objects marked as difficulty can not be simply classified or detected without additional information such as the view of the image. This is beyond the topic of this project. Therefore, in our project we ignore the difficult objects as most other works did.

As we mentioned in the previous section, we have preprocessed the training data as well as validation data for fine-tuning CNN. We have generated 30120 training data with 6024 positive images labeled by 20 classes, and 24096 background images. Also, we get 30232 validation data with 6064 positive images and 24168 background images. Fig. 2 illustrates the samples of resized training data with positive and negative labels.

For the test data, the VOC 2007 has released the whole test dataset with 4952 images and the corresponding ground truth bounding box for each object. There is no object labeled as "difficult" in the test dataset.

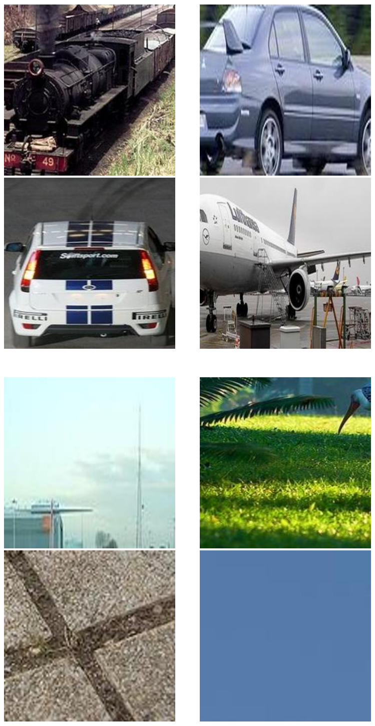
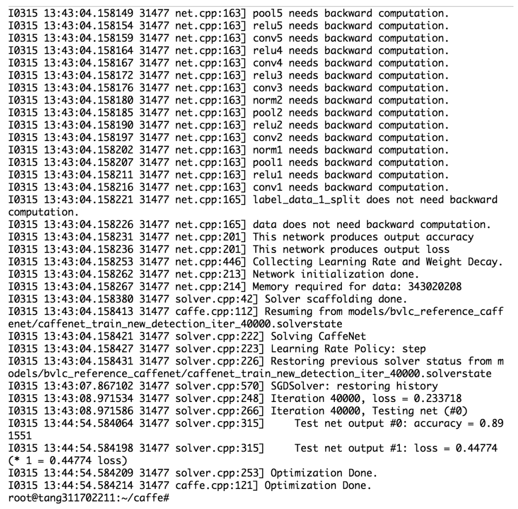


Figure 2: The samples from training data with all images resized into 256×256 pixels. The first two rows are positive image, and the second two rows are negative images.

4.2. Fine-tuning Caffe

The goal of the training process is to fine-tune the pretrained CaffeNet model on our dataset. Our strategy is that first we have frozen all the layers except for the last layer (softmax layer), then use a relatively aggressive learning rate to train the last layer from scratch. This procedure is equivalent to use the 4096 dimension CNN features to train a softmax classifier. We initialize the learning rate as 0.001 and decrease the learning rate by a factor of 0.9 after every 2000 iterations. After the 10000 iterations, the validation accuracy is 0.780875, with loss as 0.758219.

Then for the next step of fine tuning, we release all the layers, and train them together. Here, we choose a relatively small learning rate initialized as 5e-6, and decrease the learning rate by a factor of 0.9 after every 2000 iterations. After 30000 (40000 in total) iterations, we can get the the validation accuracy as 89%. And the loss is 0.4474, as Fig. 3 shows.



```
I0315 13:43:04.158149 31477 net.cpp:163] pool5 needs backward computation.
I0315 13:43:04.158154 31477 net.cpp:163] relu5 needs backward computation.
I0315 13:43:04.158159 31477 net.cpp:163] conv5 needs backward computation.
I0315 13:43:04.158164 31477 net.cpp:163] relu4 needs backward computation.
I0315 13:43:04.158167 31477 net.cpp:163] conv4 needs backward computation.
I0315 13:43:04.158172 31477 net.cpp:163] relu3 needs backward computation.
I0315 13:43:04.158176 31477 net.cpp:163] conv3 needs backward computation.
I0315 13:43:04.158180 31477 net.cpp:163] norm2 needs backward computation.
I0315 13:43:04.158185 31477 net.cpp:163] pool2 needs backward computation.
I0315 13:43:04.158190 31477 net.cpp:163] relu2 needs backward computation.
I0315 13:43:04.158197 31477 net.cpp:163] conv2 needs backward computation.
I0315 13:43:04.158202 31477 net.cpp:163] norm1 needs backward computation.
I0315 13:43:04.158207 31477 net.cpp:163] pool1 needs backward computation.
I0315 13:43:04.158211 31477 net.cpp:163] relu1 needs backward computation.
I0315 13:43:04.158216 31477 net.cpp:163] conv1 needs backward computation.
I0315 13:43:04.158221 31477 net.cpp:165] label_data_1_split does not need backward
computation.
I0315 13:43:04.158226 31477 net.cpp:165] data does not need backward computation.
I0315 13:43:04.158231 31477 net.cpp:201] This network produces output accuracy
I0315 13:43:04.158236 31477 net.cpp:201] This network produces output loss
I0315 13:43:04.158253 31477 net.cpp:446] Collecting Learning Rate and Weight Decay.
I0315 13:43:04.158262 31477 net.cpp:213] Network initialization done.
I0315 13:43:04.158267 31477 net.cpp:214] Memory required for data: 343020208
I0315 13:43:04.158380 31477 solver.cpp:42] Solver scaffolding done.
I0315 13:43:04.158413 31477 caffe.cpp:112] Resuming from models/bvlc_reference_caff
enet/caffenet_train_new_detection_iter_40000.solverstate
I0315 13:43:04.158421 31477 solver.cpp:222] Solving CaffeNet
I0315 13:43:04.158427 31477 solver.cpp:223] Learning Rate Policy: step
I0315 13:43:04.158431 31477 solver.cpp:226] Restoring previous solver status from m
odels/bvlc_reference_caffenet/caffenet_train_new_detection_iter_40000.solverstate
I0315 13:43:07.867102 31477 solver.cpp:570] SGDSolver: restoring history
I0315 13:43:08.971534 31477 solver.cpp:248] Iteration 40000, loss = 0.233718
I0315 13:43:08.971586 31477 solver.cpp:266] Iteration 40000, Testing net (#0)
I0315 13:44:54.584064 31477 solver.cpp:315]     Test net output #0: accuracy = 0.89
1551
I0315 13:44:54.584198 31477 solver.cpp:315]     Test net output #1: loss = 0.44774
(* 1 = 0.44774 loss)
I0315 13:44:54.584209 31477 solver.cpp:253] Optimization Done.
I0315 13:44:54.584214 31477 caffe.cpp:121] Optimization Done.
root@tang311702211:~/caffe#
```

Figure 3: The accuracy and loss for the fine-tuned CaffeNet for VOC 2007 object detection task.

4.3. Testing results

In the testing procedure, we first generate proposals for each test image by edge boxes. The average run time for each image is around 0.3 seconds. However, for each image the number of proposals ranges from 2000 up to 6000. We use the terminal GPU+Caffe instance, the forward pass for each proposal takes roughly 54 ms per proposal and around 1.8 to 5.4 minutes per image (including the time of load and save files). Based on the analysis above, we can conclude that for the worst case the total runtime for pass all the test images is roughly 445.68 hours. Hence, for this course project, we need to reduce the number of proposals per image heuristically.

We find that VOC 2007 test dataset, most of the objects are large. Therefore, we can remove the proposals with tiny area which is generally impossible to be good candidates

VOC 2007	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	mAP
IoU=0.1	0.4188	0.5458	0.2947	0.2402	0.1807	0.4377	0.5187	0.4964	0.1719	0.2815	
IoU=0.2	0.4271	0.5485	0.3014	0.2436	0.1939	0.4427	0.5259	0.5007	0.1771	0.2856	
IoU=0.3	0.4340	0.5536	0.3041	0.2529	0.1943	0.4542	0.5291	0.5116	0.1779	0.2797	
IoU=0.4	0.4378	0.5555	0.3023	0.2541	0.2045	0.4551	0.5261	0.5163	0.1792	0.2829	
IoU=0.5	0.4316	0.5493	0.2987	0.2563	0.2022	0.4506	0.5229	0.5098	0.1774	0.2701	
Esemble	0.4378	0.5555	0.3041	0.2563	0.2045	0.4551	0.5291	0.5163	0.1792	0.2856	
VOC 2007	diningtable	dog	horse	motorbike	person	pottedplant	sheep	sofa	train	tvmonitor	mAP
IoU=0.1	0.2620	0.4795	0.3796	0.4258	0.4347	0.1955	0.2006	0.2908	0.5455	0.3773	0.3589
IoU=0.2	0.2690	0.4889	0.3831	0.4364	0.4620	0.2032	0.2025	0.3008	0.5541	0.3843	0.3665
IoU=0.3	0.2703	0.4945	0.3842	0.4368	0.4757	0.2048	0.2134	0.3063	0.5538	0.3930	0.3712
IoU=0.4	0.2711	0.4872	0.3789	0.4467	0.4831	0.2053	0.2143	0.3085	0.5507	0.3835	**0.3722**
IoU=0.5	0.2684	0.4798	0.3672	0.4460	0.4865	0.2031	0.2129	0.3035	0.5455	0.3775	0.3680
Ensemble	0.2711	0.4945	0.3842	0.4467	0.4865	0.2053	0.2143	0.3085	0.5541	0.3930	**0.3738**

Table 1: The mAP performance of our model. Each column in the table corresponding to the average precision for each class, and the mAP is the mean of the average precision for all the class. The IoU denotes the overlap threshold in NMS. When IoU=0.4, we can achieve the best mAP as 0.3722. After ensemble the results from various IoUs together, we get 0.3738 mAP.

for object bounding boxes. We also further test that if we cut down the proposals with area smaller than 2000 square pixels (which equals to 44.7×44.7 pixels image), the total number of proposals are reduced to around 2000 per image, i.e., almost half of the proposals generated by edge boxes are tiny. Therefore, considering most of the object are large in test dataset, to speed up our calculation, we can safely remove the proposals with its area smaller than 2000 square pixels without degrading the performance too much.

Then we pass all the proposals to the CNN and get the softmax scores for each proposal. Next, for each class, we perform the NMS, to eliminate the overlapped proposal. At last we compute the mAP. Note that we have find that the mAP depends on the the threshold of IoU in NMS. Therefore, we have tuning the IoU from 0.1 to 0.5, and then find that IoU = 0.4 yields the best mAP performance as 0.3722. The mAP can be further improved by ensemblly choose the maximum average precision for each class compared among all the IoU values, we can improve the mAP to 0.3738. The results are summarized in Table 1. The precise-recall curve for all of the 20 classes in the case of IoU = 0.4 are listed in the Appendix 1. And several example detections in Appdendix 2.

4.4. Performance analysis and further improvement

From the above results, we can find that in the following classes, the performance of our model degrades heavily. These classes are bottle, chair, pottedplant and sheep. The average precision for those classes are around 20%. After checking the test data set carefully, we have found that the objects belong to these classes are relatively small. This is why the performance is not as well as we expected. We have choose the two images in the test dataset to illustrate this phenomenon in Fig. 4a. We can observe that, in the above figure, the two sofas are tiny and the detector is not able to recognize them. It mislabeled a person and a bottle. On the other hand, the large chair is detected with high confidence.

To increase the performance, recall that to save time, we have heuristically discard the proposals with area less than 2000 square pixels. This may induce the poor performance in detecting tiny objects.

Besides, the pre-trained CNN model we adopt here is not deep enough. If we switch to the deeper network such as VGG, the confidence as the output of CNN will be better. Additionally, the data augmentation could be another choice to improve the performance.

At last, to improve the accuracy of localization, we can further adopt the regressor as [5] does. Also, we can add more ground truth data as our training data, therefore, the network will prefer the bounding box close to the ground truth.

5. Conclusion

Overall in this project, we have learned hands on experience in working with CNN such as debugging network, transfer learning and working with Caffe. We also adopt the CNNs to solve the detection problem and try to improve the exist model such as rCNN.

In this paper, we provide a new model for object detection based on CNN. In this model, we use the edge boxes algorithm to generate proposals, and use a fine-tuned the CaffeNet model to generate the score for each proposals.

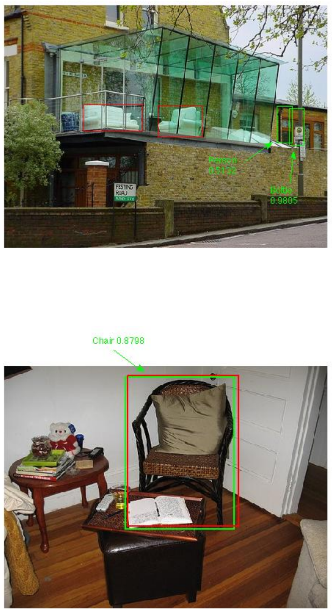


Figure 4: The red box is the ground truth bounding box, and green one is generated from our model with number as the confidence. The detector will sometimes fail to detect tiny objects.

Then, we merge the proposals by NMS.

Our model achieves the 0.3738 mAP on VOC 2007 dataset. To further improve this model beyond the scope of this project, we will use all the proposals generated from the edge boxes rather than throw the tiny proposals as we do in this paper. Furthermore, we will change a more deeper network to increase the accuracy of classification as well as to add the ground truth bounding boxes into the training data to improve the localization accuracy.

6. Appendix

6.1. Precise-recall curve for IoU=0.4

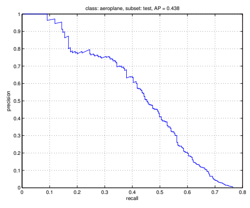


Figure 5: aeroplane

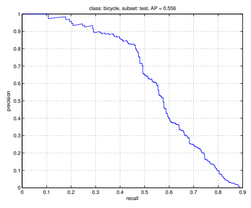


Figure 6: bicycle

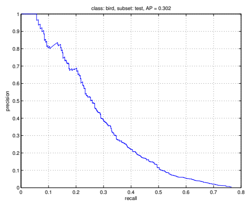


Figure 7: bird

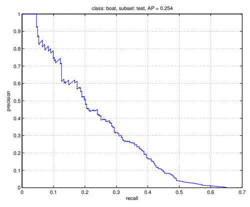


Figure 8: boat

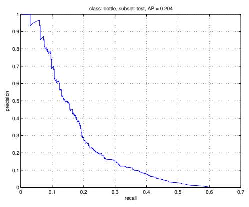


Figure 9: bottle

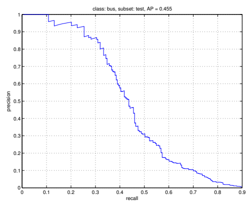


Figure 10: bus

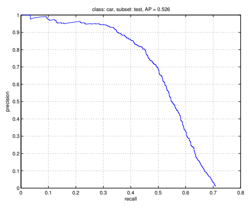


Figure 11: car

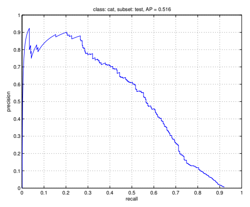


Figure 12: cat

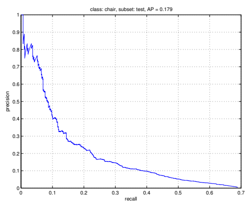


Figure 13: chair

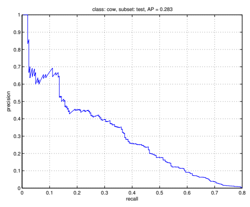


Figure 14: cow

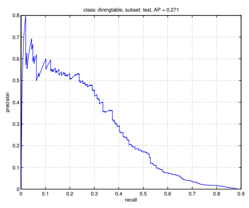


Figure 15: diningtable

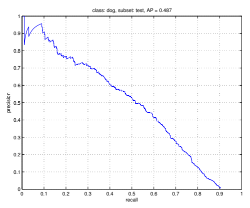


Figure 16: dog

6.2. Example detections

Here are several example detections. The red boxes are the detected bounding box with the class and confidence associate with each box.

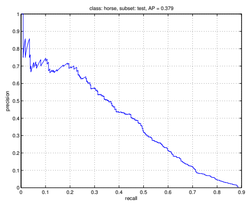


Figure 17: horse

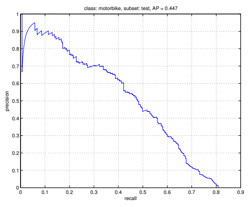


Figure 18: motorbike

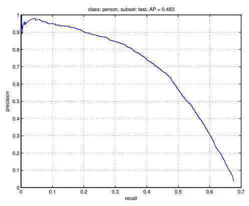


Figure 19: person

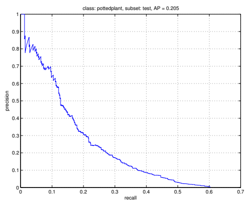


Figure 20: pottedplant

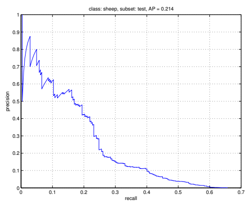


Figure 21: sheep

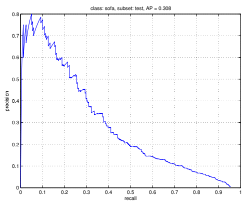


Figure 22: sofa

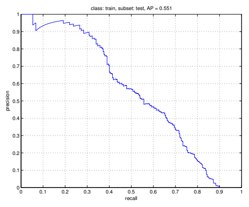


Figure 23: train

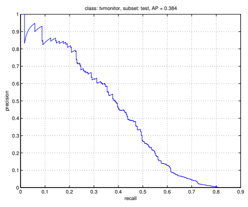
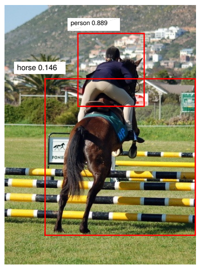
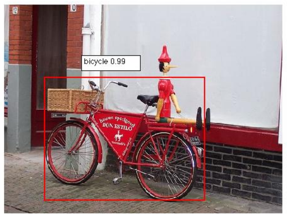
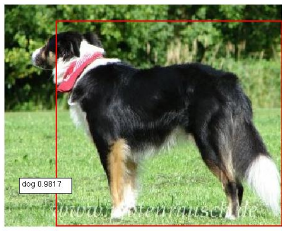
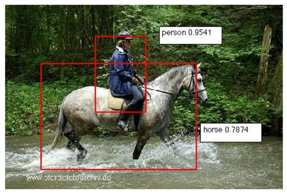


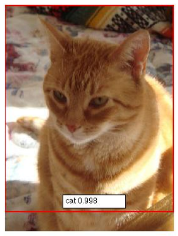
Figure 24: tvmonitor

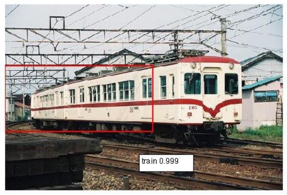


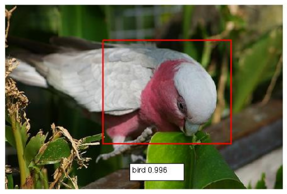


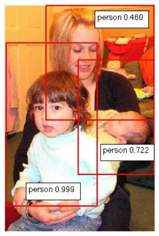


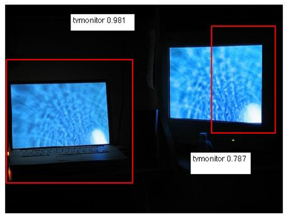


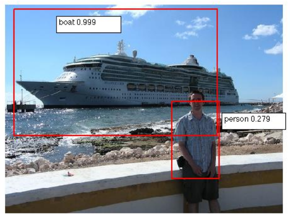












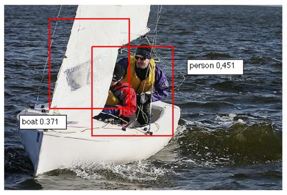
References

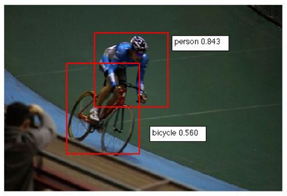
[1] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012. 1, 3, 4, 7

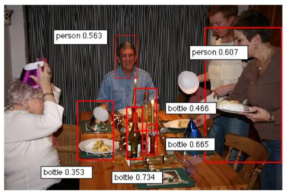
[2] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In NIPS, 2013. 2

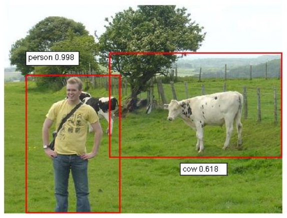
[3] L. Jonathan, S. Evan, D. Trevor, Fully convolutional networks for semantic segmentation, to appear in CVPR 2015.

[4] K. Andrej, Li Fei-Fei, Deep visual-semantic alignments for generating image descriptions.









arXiv:1412.2306.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. in Proc. CVPR, 2014.

[6] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, Selective search for object recognition. in Proc. IJCV, 2013.

[7] C. Lawrence Zitnick and Piotr Dollar, Edge Boxes: Locating Object Proposals from Edges. in ECCV 2014.

[8] H. Jan, B. Rodrigo, S. Bernt How good are detection proposals, really? arXiv:1406.6962.

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 2004. 1

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005. 1

[11] M. Tomasz, G. Abhinav and E. Alexei. Ensemble of exemplar-SVMs for object detection and beyond. in Proc. ICCV 2011.

[12] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. TPAMI, 2010. 2, 4, 7, 12

[13] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun OverFeat: integrated recognition, localization and detection using convolutional networks. arXiv:1312.6229, 2014.

[14] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. TPAMI, 2012. 2

[15] I. Endres and D. Hoiem. Category independent object proposals. In ECCV, 2010. 3

[16] J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. TPAMI, 2012. 2, 3

[17] C. Ramazan, V. Jakob and S. Cordelia. Weakly supervised object localization with multi-fold multiple instance learning. arXiv:1503.00949 2015.

[18] O. Wanli, L. Ping and Z. Xingyu et al. DeepID-Net: multi-stage and deformable deep convolutional neural networks for object detection. arXiv:1409.3505 2015

[19] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In ECCV. 2012

[20] M. Everingham and J. Winn, The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Development Kit. 2007