Pandas Images Detection by Faster R-CNN

Group 5: adversatial attack

Author: Eafan Yang(yifany13), Tian Sun(tiansun2)

***Abstract*—Object Detection is one of the fundamental problems of computer vision. In this report, we use Faster R-CNN to detect pandas in images. After several generations of improvements, this Faster R-CNN has been significantly improved in speed and accuracy compared to the original RNN. So, in this work, we will show you how do we apply Faster R-CNN to our objective detection with an ideal result.**

Keywords—Faster R-CNN, object detection

I. Introduction

Object Detection is one of the fundamental problems of computer vision. It forms the basis of many other downstream computer vision tasks, for example, instance segmentation, image captioning, object tracking, and more. As a very basic application of Machine Learning to real world, we choose this topic to help us enhance the knowledge we learn from this class. The main challenging for us is that we don't have much deep learning background of both theory and practice.

So in this report, we will show you how we use Faster R-CNN to complete our pandas detection mission. From a practical point of view, with the help of effective panda detection algorithms, breeders can effectively monitor the living conditions of pandas, ensure the health of pandas, and conduct further research on their living habits.

The input of the network are photos with at least one panda and the output will be same photos with pandas framed by a red box.

II. Related Work

The R-CNN[2] method trains CNNs end-to-end to classify the proposal regions into object categories or background. R-CNN mainly plays as a classifier, and it does not predict object bounds (except for refining by bounding box regression). Its accuracy depends on the performance of the region proposal module. Several papers have proposed ways of using deep networks for predicting object bounding boxes.

In the Over-Feat method[3], a fully-connected layer is trained to predict the box coordinates for the localization task that assumes a single object. The fully-connected layer is then turned into a convolutional layer for detecting multiple class specific objects.

The Multi-Box methods[4][5] generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes, generalizing the "single-box" fashion of Over-Feat. These class-agnostic boxes are used as proposals for R-CNN.

The Multi-Box proposal network is applied on a single image crop or multiple large image crops, in contrast to our fully convolutional scheme. Multi-Box does not share features between the proposal and detection networks. We discuss Over--Feat and Multi-Box in more depth later in context with our method. Concurrent with our work, the Dee-Mask method is developed for learning segmentation proposals. Shared computation of convolutions has been attracting increasing attention for efficient, yet accurate, visual recognition. The Over-Feat paper computes convolutional features from an image pyramid for classification, localization, and detection. Adaptively-sized pooling (SPP) on shared convolutional feature maps is developed for efficient region-based object detection and semantic segmentation. Fast R-CNN enables end-to-end detector training on shared convolutional features and shows compelling accuracy and speed.

III. Data

The data we use for training is collected from "ImageNet Large Scale Visual Recognition Challenge 2017"[11] proposed by ImageNet. It is composed with images in JPEG with annotation for bounding boxes in xml. The resolution for image is smaller than 500 x 500 and the xml files are in PASCAL VOC format.

The original data with 1000 categories is large in sizes. We pick only one single species of animal, panda, for research. There are 1300 for panda with only 501 corresponding files. To guarantee that each picture we train could have a correct bounding area. We delete pictures without given annotations and keep only 501 images with 501 xml files.

To decode the bounding boxes in xml files, we write python script to merge all .xml files into an array with columns "filename," "width," "height," "class," "xmin," "ymin," "xmax," "ymax." This is helpful to draw boxes on original pictures.

Among total 501 images we split into 400 training with 101 for validation.

Figure 1: Original picture for "Man with panda" on the left and Annotated picture with bounding box on the right.

	filename	width	height	class	xmin	ymin	xmax	ymax
0	n02510455_100053.JPEG	333	500	n02510455	7	257	226	470
1	n02510455_1010.JPEG	500	336	n02510455	157	53	399	254
2	n02510455_1016.JPEG	500	375	n02510455	129	26	406	298
3	n02510455_102216.JPEG	500	332	n02510455	104	45	480	331
4	n02510455_10292.JPEG	500	375	n02510455	68	2	430	319

Figure 2: First 5 rows for "panda_label.csv" file with 8 columns.

IV. Faster R-CNN

Our object detection algorithm, Faster R-CNN, has two networks: region proposal network (RPN) and an object detection network. Different from R-CNN and Fast R-CNN using selective search algorithm, Faster R-CNN uses the technique of "anchor boxes" to localize objects and predict them, which is much faster in generating region proposals. Hence more computation will be shared with the object detection network. In section A, we briefly talk about how "anchor boxes" works and in section B, we discuss the region proposal network.

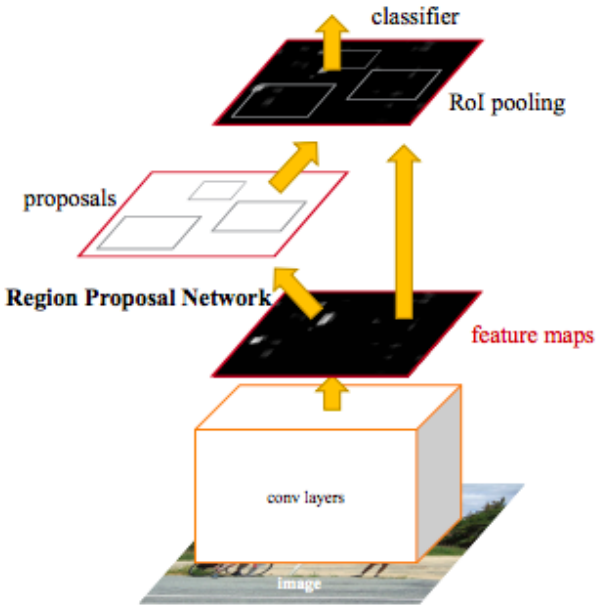

Figure 3: Faster R-CNN process with Region Proposal Network, Roi pooling, and Classifier.

A. *Anchor Boxes*

Anchor box plays a significant role in Faster R-CNN. For a conventional feature map, at each sliding-window location, we simultaneously predict multiple region proposals, where the number of maximum possible proposals for each location is denoted as *k*.[1] And we have two layers generalized from the intermediate layer, a box-regression layer (*reg*) and a box-classification layer (*cls*). The *reg* layer has 4*k* outputs encoding the coordinates of *k* boxes, and the *cls* layer outputs 2*k* scores that estimate probability of object or not object for each proposal. The *k* proposals are parameterized relative to *k* reference boxes, which we call anchors. By default, there are $k = 9$ anchors at a position of an image, 3 scales 128x128, 256x256, 512x512 with three height width ratios 1:1, 1:2, and 2:1 respectively. [10]

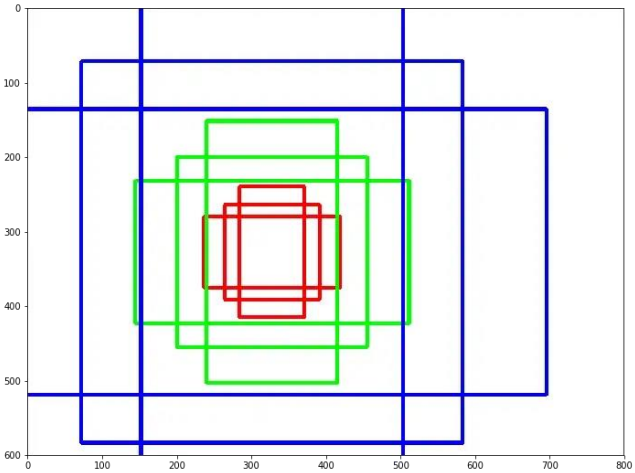

Figure 4: 3x3 Anchor boxes for a single position.

B. *Region Proposal Networks*

A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a bunch of boxes/proposals that sends to a classifier and regressor to evaluate the occurrence of objects. It predicts the possibility of an anchor being background or foreground and refine the anchor. We model this with a fully convolutional network. We want to label the anchors having the higher overlaps with ground-truth boxes as foreground, the ones with lower overlaps as background. Next, we want to know what features of the anchors are. For instance, a 600x800 image shrinks 16 times to a 39x51 feature map after applying CNNs. Every position in the feature map has 9 anchors, and every anchor has two possible labels (background, foreground). If we make the depth of the feature map as 18 (9 anchors x 2 labels), we will make every anchor have a vector with two values (normal called logit) representing foreground and background. If we feed the logit into a SoftMax/logistic regression activation function, it will predict the labels.[10]

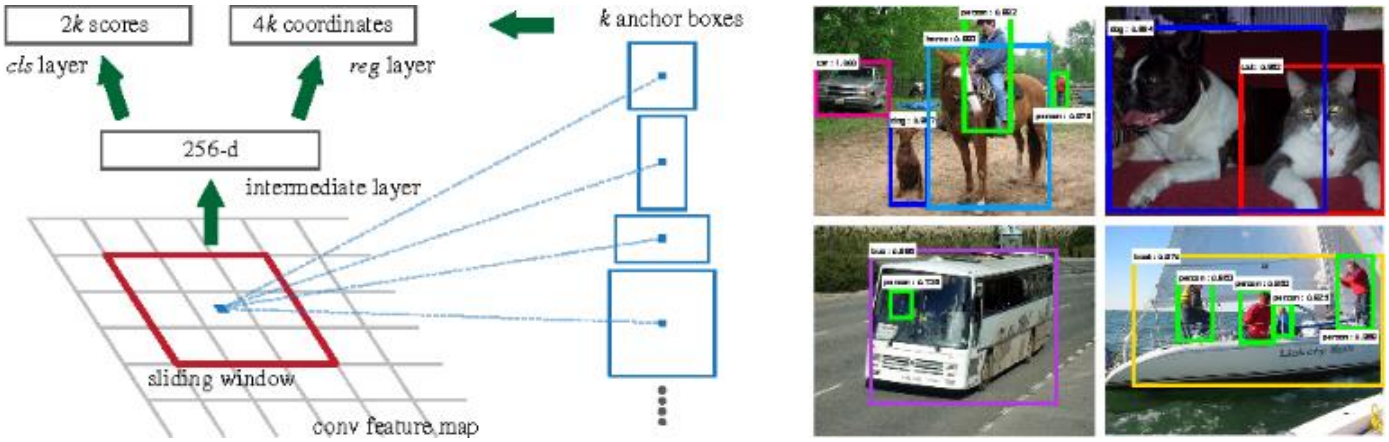

Figure 5: Region Proposal Network (RPN) on the left and Example detections using RPN proposals on PASCALVOC 2007 test on the right.

C. *Loss Function*[1]

Our loss function for an image is defined as:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum L_{cls}(p_i, {p_i}^*) + \lambda \frac{1}{N_{reg}}\sum {p_i}^* L_{reg}(t_i, {t_i}^*).$$

Here, i is the index for anchor and p_i is the probability for anchor i. Note that p_i^* is 1 if the anchor is positive and 0 otherwise. t_i is a vector representing the 4 parameterized coordinates of the bounding box, which will discuss later and t_i^* is the vector associated with positive anchors. The classification loss L_{cls} is log loss over two classes. For regression loss, we use the robust loss function (smooth L_1) in which:

$$smooth_{L_1}(x) = \begin{cases} 0.5x^2, & if\ |x| < 1; \\ |x| - 0.5, & otherwise. \end{cases}$$

Note that the two terms are normalized by N_{cls} and N_{reg} and weighted by a balancing parameter λ. In our current implementation N_{cls} = 256 and N_{reg} ~ 2,400. Hence by default we set λ = 10 to roughly equally weighted both *cls* and *reg*.

We adopt 4 coordinates for bounding box regression:

$$t_x = \frac{x - x_a}{w_a}, \quad t_y = \frac{y - y_a}{h_a},$$

$$t_w = \log\left(\frac{w}{w_a}\right), \quad t_h = \log\left(\frac{h}{h_a}\right),$$

$$t_x^* = \frac{x^* - x_a}{w_a}, \quad t_y^* = \frac{y^* - y_a}{h_a},$$

$$t_w^* = \log\left(\frac{w^*}{w_a}\right), \quad t_h^* = \log\left(\frac{h^*}{h_a}\right),$$

Where *x*, *y*, *w*, and h denote the center coordinates with its width and height for the anchor box. *x*, x_a, and x^* are for the predicted box, anchor box, and ground-truth box respectively.

D. *Alternative Approaches*

Besides of this Faster R-CNN model we use, there are other approaches as well. As discussed before, RPN is first used in the Fast R-CNN model. Although it uses the same RPN algorithm, it uses selective search to find region which takes many time. In an order way, R-CNN is also theoretically achievable. Without using RPN for regions it uses selective search and propose more than 2000 region in one image, which will take much more time to train.

V. *RESULT*

After 10 epochs of training, our model gives Map 0.755.

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.755
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.982
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.840
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.559
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.761
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.739
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.794
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.794
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.600
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.800
```

Figure 6: Average Precision, Average Recall for Model.

And for test of model with testing data, we get the output that has a score over 0.8.

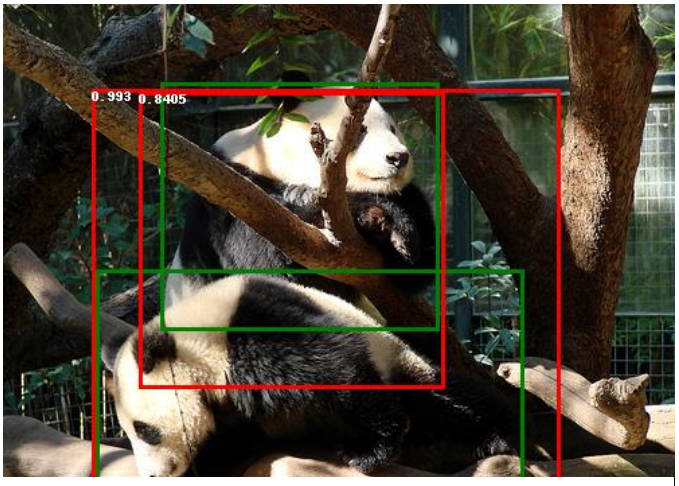

Figure 7: Two prediction with 0.993 and 0.8405 for test.

V. Conclusion

As we have present Faster R-CNN models based on RPNs, our model enables a real-time deep learning-based object detection system. The RPN improves the time cost of generating region proposals comparing to selective search.

As working in a group, we learn how to make plans ahead. How efficient it is to do single parts ahead and merge them together. The final presentation draws a full stop for our project, but not our friendship.

Note that we focus on a single animal for object detection. The result is as expected but not useful in real life. In the future we will try adding more categories inside so that we could detect multiple objects in an image.

References

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks".

[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] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in International Conference on Learning Representations (ICLR), 2014.

[4] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[5] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," arXiv:1412.1441 (v1), 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in European Conference on Computer Vision (ECCV), 2014.

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

[8] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," International Journal of Computer Vision (IJCV), 2013.

[9] M. Beckers, "Building your own object detector — PyTorch vs TensorFlow and how to even get started?" *Towards Data Science* (blog), 25.Apr. 2020.

[10] H, Gao, "Faster R-CNN Explained," *Medium* (blog), 27.Sep, 20

[11] Kaggle, "ImageNet Object Localization Challenge," Kaggle (code), 2019. https://www.kaggle.com/c/imagenet-object-localization-challenge

[12] T, Sun, "tiansun3 / panda_dataset" *Github* (code), 16 .Dec. 2022. https://github.com/tiansun3/panda_dataset