RPN for faster RCNN

Sheng Zhong

March 12, 2019

1 Overview

The paper's discussion of their cls and reg layers in the RPN were confusing so I will explain what each component does. [1] Firstly, the paper proposes a method to do object recognition, the second most difficult computer vision task shown in Figure 1. The sequential increase in difficulty comes from each task needing to do the previous task while providing additional information.

The RPN divides its input (shared conv layer output) into overlapping regions called *anchors*. For each anchor the RPN outputs its class (cls) and bounding box refinement (reg). Class is either object (foreground) or not object (background). The bounding box refinement is a shift in (x,y,width,height) to better fit the anchor to the object. Anchors with the highest object scores (cls) are passed to the RCNN for classification.

2 Review

Overall the RPN is solid, and is used without much modification in its extension Mask RCNN to perform instance segmentation [2], however there are a couple of points I question and others where I have suggestions.

The first is their motivation for creating the RPN. They acknowledged that current region proposal methods are slow because they are implemented in CPU, but then they claim it's not worth implementing it in GPU because it would not share computation. The ultimate goal of this paper is to speed up object detection, not to share computation. So what they could have done better is to include a comparison of RPN against an existing proposal method implemented in GPU. It would even be understandable if they didn't do this (because it's a lot of work) and mention it in the discussion but they ignored this comparison.

Another question is why they rescaled the images to 600 pixel on the shorter if the typical PASCAL image is 500x375. Did scaling down not work because they lost information? Working on a smaller image will drastically reduce runtime since the number of pixels scales $O(s^2)$ to side length s. This is also not explained in their earlier paper on Fast RCNN [3] where

[1] pictures and explanation source from https://engineering.matterport.com/splash-of-color-instance-segmentation-with-mask-r-cnn-and-tensorflow-7c761e238b46

[2] https://arxiv.org/pdf/1703.06870.pdf

[3] https://arxiv.org/pdf/1504.08083.pdf

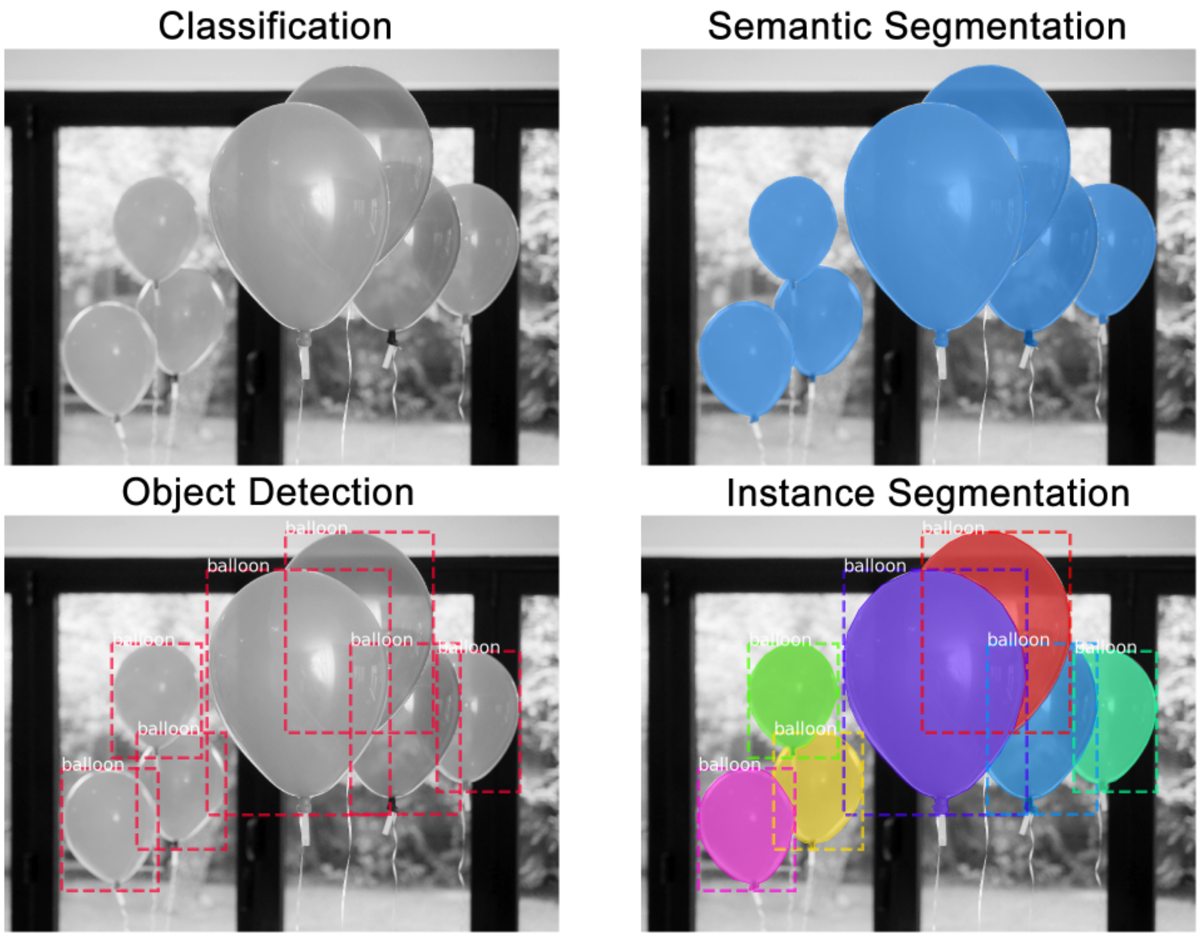

Figure 1: Computer vision tasks. In order of difficulty, the tasks are classification, semantic segmentation, object detection, and instance segmentation.

they do explain why it's not any greater (to fit the image into their GPU during fine-tuning), but not why it's not smaller. Their justification is that their concern is not to optimize s for each model. It would be better to show how performance changes when s is made smaller since it is critical to their run time.

A point of improvement is in their weight initialization for cls and reg layers. They currently initialize it using a normal distribution with 0 mean and fixed standard deviation 0.01. To avoid vanishing or exploding gradients, they should initialize instead with standard deviation that is a function of the filter kernel dimensions, as mentioned in [4] (surprisingly these guys are also from Micro$oft research).

[4] https://arxiv.org/pdf/1502.01852.pdf and http://www.telesens.co/2018/04/09/initializing-weights-for-the-convolutional-and-fully-connected-layers/