Fast R-CNN

Girshick

1 Summary

Fast R-CNN employs a deep convolutional architecture to perform object detection (classification and localization of multiple objects in an image). It first runs the image through a fully convolutional architecture to produce a convolutional feature map. Then, *for each externally-generated region proposal*, fast R-CNN uses RoI pooling to extract a fixed-length feature vector for the proposed region in the convolutional feature map, and sends this through a series of fully connected layers with two output heads – one for classification and one for bounding box regression.

I am mostly interested in this paper for the architecture and the RoI pooling.

2 Introduction

This work builds on that of R-CNN and SPPnet, citing increased speed, increased mAP detection quality, and single-stage training as improvements over those approaches.

3 Architecture

As input, the network takes an image and a set of object (region) proposals. As output, the network provides *for each region proposal* a softmax probability estimate over all K classes (plus a catch-all background class) and an (x, y, h, w) bounding box for all K classes.

3.1 RoI Pooling

The point of RoI pooling is to extract features of *fixed length* for a proposed region in a convolutional feature map. In the end, it's just max pooling specialized for producing a fixed-length output. A RoI pooling layer takes as input a RoI ("a rectangular window into a convolutional feature map") along with hyperparameters H and W representing the spatial output dimensions. It then divides the RoI window into an $H \times W$ grid of sub-windows and max pools the values in each sub-window. This happens independently in each feature channel, so the number of channels remains the same.

3.2 Multi-Task Loss

Each output head has a loss; the overall loss is the sum of each of the two losses i.e.

$$\underbrace{-\log p_u}_{\text{classification loss}} + \underbrace{\lambda[u \geq 1] \sum_{i \in \{x,y,w,h\}} L_1(t_i^u - v_i)}_{\text{localization loss}}$$

where the classification loss is simply the log loss for the true class u and the localization loss is a sum of robust L_1 losses between predicted (t_i^u) and actual (v_i) bounding box coordinates.

$[u \geq 1]$ is an indicator function that is 1 when $u \geq 1$, as we only care about the bounding box that is predicted for the correct class.

λ is a hyperparameter which controls the balance between classification and localization losses. In practice, the losses have been equally weighted, i.e. $\lambda = 1$ for all of the author's experiments.

References

[1] Ross Girshick. Fast R-CNN. *arXiv preprint arXiv:1611.08050 (2017).*