Object Detection: FasterRCNN and SSD for Custom Object Detection

David Hughes
Cal Poly Pomona
3801 W Temple Ave, Pomona, CA 91768
dmhughes@cpp.edu

Abstract

Object Detection has gone through a renaissance in recent years with proposed methods like FasterRCNN and SSD. Models created from these methods have greatly increased accuracy and speed over past models with tradeoffs between popular state-of-the-art methods in terms of performance and accuracy. However, the common machine-learning issue of a limited dataset still exists in Object Detection making it difficult to handle Custom Object Detection. To identify a new object, image labels must be drawn by hand, and the model must be retrained on the new images. The goal of this project is both to learn about the details of FasterRCNN and SSD, as well as investigate automated methods for Custom Object Detection. Finally, an implementation of these ideas will be demonstrated on detectors for webcams and mobile phones on toy applications.

1. Introduction

Object detection is a task that can be easily performed by humans. We look out of a car and easily identify various signs, pedestrians, road condition, etc. However, as a task for a computer, it requires computer vision and image processing to discover the semantic meaning behind the rectangle of pixels that is a typical image. Object detection was once performed as a segmentation task, where images were partitioned by pixels in hopes of revealing the objects within the image. This method is insufficient for correctly identifying objects. The strategy shifted to finding probable segmentations; algorithms declare that within a certain bounding box, there might be an object. These probable segmentations are called object proposals or proposals.

However, like many other machine-learning and deep-learning tasks, the models are limited by the available training dataset. The Object Detection model can only identify objects that it was trained for; if a new object is added to the model then the model must be retrained with a sufficient set of images and associated image labels (rectangular bounding boxes in the XML file format). The process to create these labels is often done by hand, which makes it easy to see the tedious nature of this process. The goal is to create a pipeline of tasks that takes a desired object, generates data, and trains a model for Custom Object Detection.

The deep learning methods that will be reviewed in this project are FasterRCNN (2015) and SSD (2016).

2. Review of Literature

FasterRCNN is an evolution of FastRCNN, which came from RCNN (Region-based Convolutional Neural Network). Each iteration of the method seeks to fix a major bottleneck in the process in order to greatly decrease the necessary computation time. FasterRCNN introduces a Region Proposal Network (RPN) that shares features and computation time with the detection network in order to fix the bottleneck of generating proposals. The previously used method of finding proposals for FastRCNN was the Selective Search algorithm; this algorithm is a type of region proposal algorithm that is based on computing hierarchical grouping of similar regions based on color, texture, size and shape compatibility [1] Thus the cost of generating proposals consumes similar time to the detection network itself. The authors of FasterRCNN noticed that the convolutional (conv) feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals [6]. Therefore, they created a unified network with shared feature between the tasks that waives nearly all computational burdens of SS at test-time [6]. One problem with FasterRCNN is that it still requires many passes through one image to extract all of the objects.

Single Shot MultiBox Detector (SSD) is a method that breaks away from the standard approach in FasterRCNN of hypothesiz[ing] bounding boxes, resample[ing] pixels or features for each box, and apply[ing] a high-quality classifier [4]. SSD predicts category scores and box offsets for a fixed set of bounding boxes, which, along with various other improvements, allowed the method to produce high detection accuracy within a single-shot system.

Based on the COCO mAP (mean average precision) evaluation metric, the differences between the two algorithms become clear. For example, with the same COCO trained model and the same inception_v2 model: faster-RCNN has a speed of 58 ms and a COCO mAP of 28 and SSD has a speed of 42 ms and a COCO mAP of 24 [5]. It can be seen that with the modern solutions, there is a trade-off between precision and time. FasterRCNN is slower but more accurate, while SSD is faster but less accurate. Therefore, the application of Object Detection must be considered when choosing a popular, pre-trained model.

3. Objectives

The objective of this project is to learn; I aim to learn about the FasterRCNN and SSD object detection methods in order to be able to continue this project for the near future. I hope to gain a sufficient understanding and complete enough work in order to turn this project into a Masters project/thesis in the year following the completion of this semester. Additionally, I will familiarize myself with the automated data preparation procedure described in [3].

4. Tentative Technical Approach

The basic idea of the approach that I will use is to slowly familiarize myself with each part of the project then rapidly complete the necessary deliverables. First, I plan to grasp a simplistic understanding of the theory (by technical papers and online resources). Then, I will follow a tutorial in order to run a pre-trained version of the methods. Third, I will understand the code for both FasterRCNN and SSD, while reviewing the technical resources for both methods in order to gain a greater understanding of the theory and implementation. At this point, I hope to have enough familiarity in order to complete the deliverables for this project. Simultaneously, I will review of code created [3] and be able to implement these tasks myself.

5. Timeline

As this is a medium/hard project, I am targeting a Student Presentation that occurs after Spring Break (Weeks 12-16). However, I plan to finish as early as I can for each deliverable.

- Week 11 Complete Technical Presentation Slides
- Week 12 (between 12-16) Present Technical Presentation
- Week 13 Presentation Poster/Assignment
- Week 14 Finish Term Paper

6. Deliverables

The first deliverables are the common set: presentation slides, research poster, code, Github README assignment, and term paper. The unique service will be detectors for webcams and mobile phones with a toy application of fruit and safety hat detection.

7. Resources

Currently, I do not have an Nvidia GPU in my laptop or desktop, so I am trying to leverage other resources for this project. I will attempt leverage the Google Colaboratory environment, because it seems to fulfill the requirements I need; at the time of this submission, I am able to run the test-cases for the Object Detection API [7], so I am ready to continue working in Colab. It appears, with the help of various available online tutorials [2], I will be able to use Colab to build and train the model and deploy the model to browser/mobile. Meanwhile, I will find a way to use a webcam (perhaps training the model on colab, then running the model on the CPU version of Tensorflow on my laptop).

Additionally, for the service I will need to use Android Studio or Expo (for React Native) to build a mobile application depending on the details of Tensorflow.js and TensorFlowLite.

References

[1] V. S. Chandel. Selective search for object detection (c++/python), Sep 2017.

[2] EdjeElectronics. Edjeelectronics/tensorflow-object-detection-api-tutorial-train-multiple-objects-windows-10, Sep 2018.

[3] P.-Y. Gao, M. Farraj, and Y.-L. Tsou. Automated data preparation for custom object detection.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[5] pkulzc, V. Rathod, and N. Wu. Tensorflow detection model zoo, Dec 2018.

[6] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[7] Tensorflow. Tensorflow object detection api, Sep 2018.