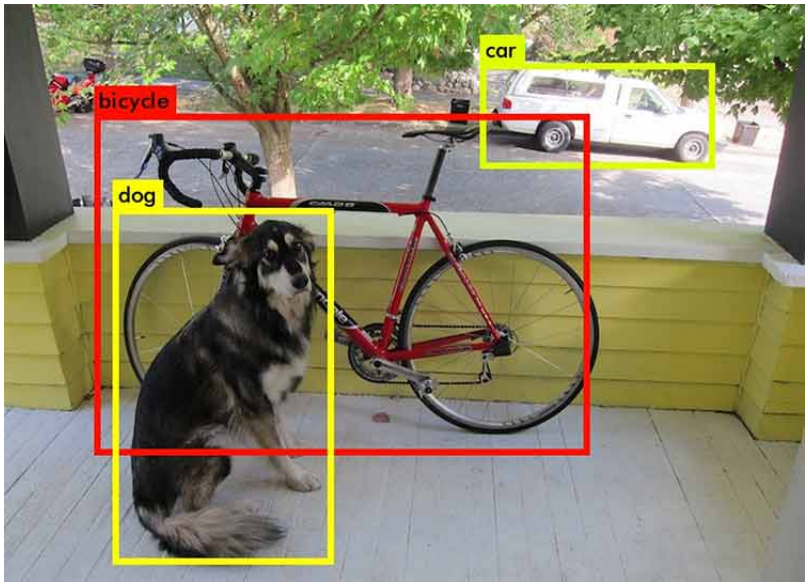
Object Detection

Introduction

An object detection algorithm is one that takes in an image, and outputs bounding boxes surrounding the objects of interest in the image. An example is shown below, a dog, bicycle, and a car.



Before you can use an object detection algorithm, you must first predefine what sort of objects (aka classes) you want to be able to detect. Next, you must train the algorithm on example images that contain the objects you want it to learn. These images must be manually labeled by people, showing where the objects are in the image. In our experience, you probably need at least a few hundred example images for each class to get decent results.

After the algorithm has been trained on the example data, you can then use it to find objects in new images that it hasn't seen before! The process of using the object detection algorithm to find objects is also known as “inference”.

Tools

We currently use the Tensorflow object detection API (henceforth abbreviated as TFODA) for both training and inference. Previously we used the Darknet framework, however we found it rather difficult to use because the code is messy and uses rather old versions of libraries. The tensorflow object detection API is developed by more people, easier to use, and kept more up to date.

To label our images, we use sloth. For more details, see (link). Sloth creates a json file that describes the labels for each image. The TFODA requires the example images and their labels to be packaged into a specific file format called a TFrecord. We have a python script for doing so in our vision_dev repository. To create a TFrecord file, you need both the json file containing the labels and the images the json file refers to. Usage is shown below:

```
./sloth_to_tf_record.py <input json file> <output directory>
```

sloth_to_tf_record.py uses the json file to find the image files, so if the script can't find images, look at the json file to see the path that it uses to locate each image. sloth_to_tfrecord.py actually generates several files which get stored in <output directory>, which must already exist (the script will not create the directory if it does not exist). The output files are:

- label_map.pbtext - defines all the image classes
- train.record - stores images and labels to be trained on
- test.record - stores images and labels for testing

You'll notice that the script actually creates two record files, train.record test.record.