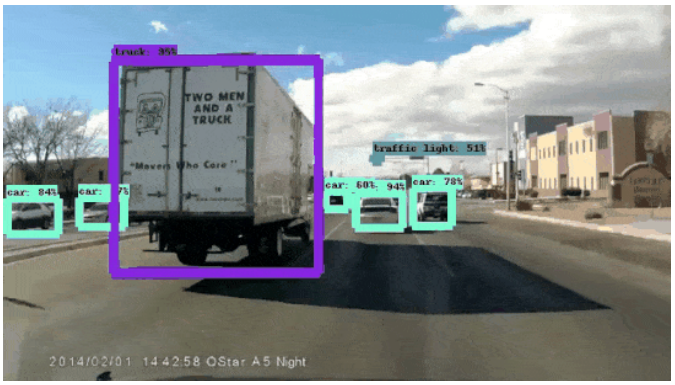
Priya Dwivedi Follow
Passionate about using machine learning, deep learning and AI
Jul 12 · 4 min read

Is Google Tensorflow Object Detection API the easiest way to implement image recognition?

Doing cool things with data!

There are many different ways to do image recognition. Google recently released a new Tensorflow Object Detection API to give computer vision everywhere a boost. Any offering from Google is not to be taken lightly, and so I decided to try my hands on this new API and use it on videos from you tube :) See the result below:



Object Detection from Tensorflow API

You can find the full code on my Github repo

So what was the experience like? First lets understand the API.

Understanding the API

The API has been trained on the COCO dataset (Common Objects in Context). This is a dataset of 300k images of 90 most commonly found objects. Examples of objects includes:

Some of the object categories in COCO datset

The API provides 5 different models that provide a trade off between speed of execution and the accuracy in placing bounding boxes. See table below:

Model name	Speed	COCO mAP	Outputs
ssd_mobilenet_v1_coco	fast	21	Boxes
ssd_inception_v2_coco	fast	24	Boxes
rfcn_resnet101_coco	medium	30	Boxes
faster_rcnn_resnet101_coco	medium	32	Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37	Boxes

Here mAP (mean average precision) is the product of precision and recall on detecting bounding boxes. It's a good combined measure for how sensitive the network is to objects of interest and how well it avoids false alarms. The higher the mAP score, the more accurate the network is but that comes at the cost of execution speed.

You can get more information about these models at this link

Using the API

I decided to try the most light weight model (ssd_mobilenet). The main steps were:

> Download the frozen model (.pb—protobuf) and load it into memory
>
> Use the built in helper code to load labels, categories, visualization tools etc.
>
> Open a new session and run the model on an image

Overall a fairly simple set of steps. The API documentation also provides a handy Jupyter notebook that walks through the main steps.

The model had pretty good performance on the sample image (see below):

Running on Videos

Next I decided to try this API on some **videos.** To do this, I used the Python

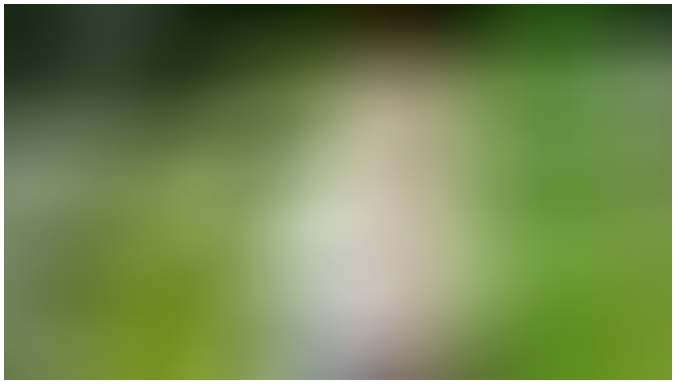
moviepy library. The main steps are:

- Use the VideoFileClip function to extract images from the video
- The fl_image function is an awesome function that can take an image and replace it with a modified image. I used this to run object detection on every image extracted from the video
- Finally all the modified clip images were combined into a new video

This code takes a bit of time to run (~ 1 minute) for a 3–4 second clip. But since we are using a frozen model loaded to memory, all of this can be done on a computer without a GPU.

I was very impressed! With just a little bit of code, you can detect and draw bounding boxes on a good number of commonly found objects with decent accuracy.

There were cases where I felt that the performance could be better. See example below. The birds are not detected at all in this video.



Next Steps

Couple of additional ideas for further exploration of this API

- Try the more accurate but high overhead models and see how much of a difference they make
- Find out ways of speeding up the API, so it can be used for real time object detection on a mobile device
- Google also provides the ability to use these models for transfer learning i.e load the frozen models and add another output layer with different image categories

Give me a ❤ if you liked this post:) Hope you pull the code and try it yourself.

Other writings: https://medium.com/@priya.dwivedi/

PS: I live in Toronto and I am looking to switch career into deep learning. If

you like my post and can connect me to anyone, I will be grateful :). My email is priya.toronto3@gmail.com

References:

Google Tensorflow Object Detection Github

COCO dataset