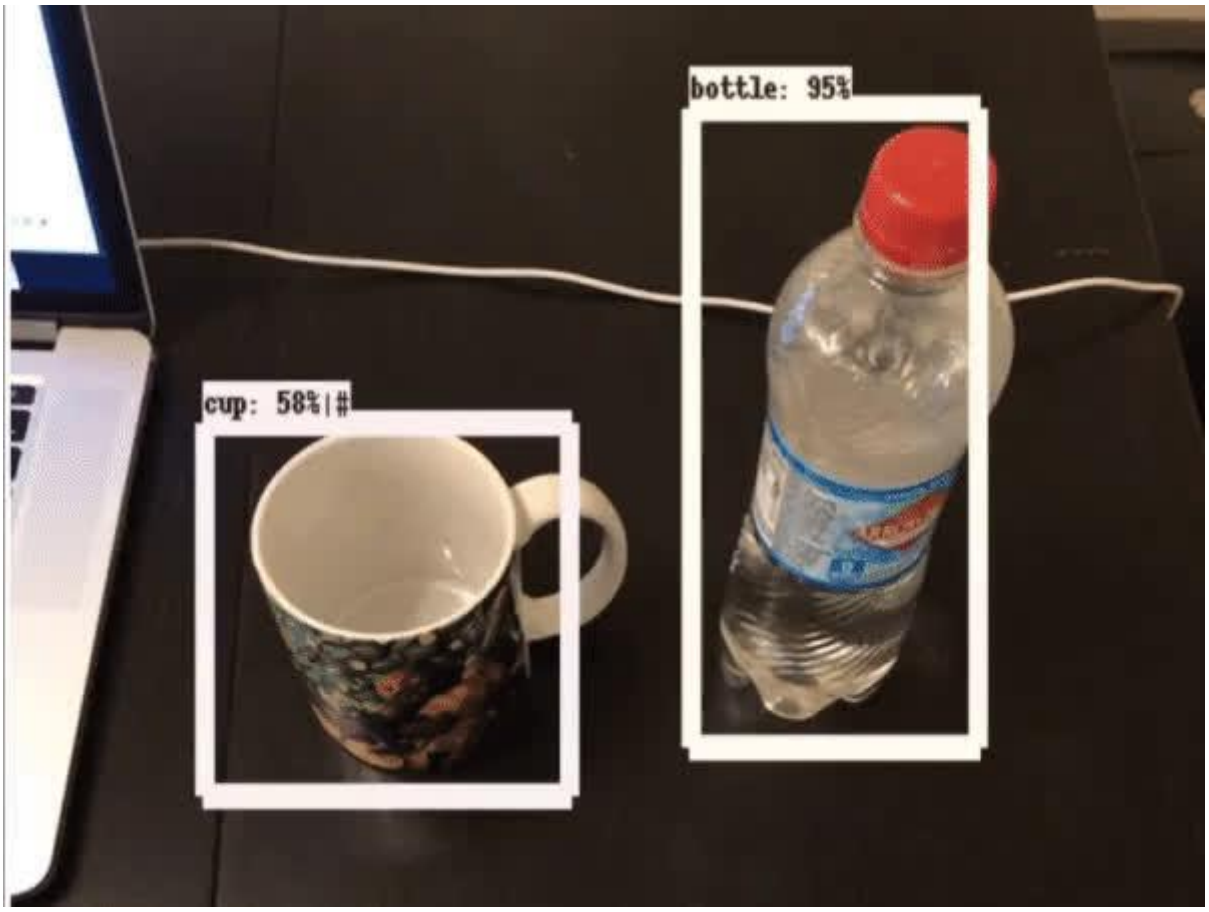
TensorFlow in your browser: Object Detection with Bounding Boxes

Created by Andrew Reusch



https://learn.adafruit.com/tensorflow-in-your-browser-object-detection-with-bounding-boxes

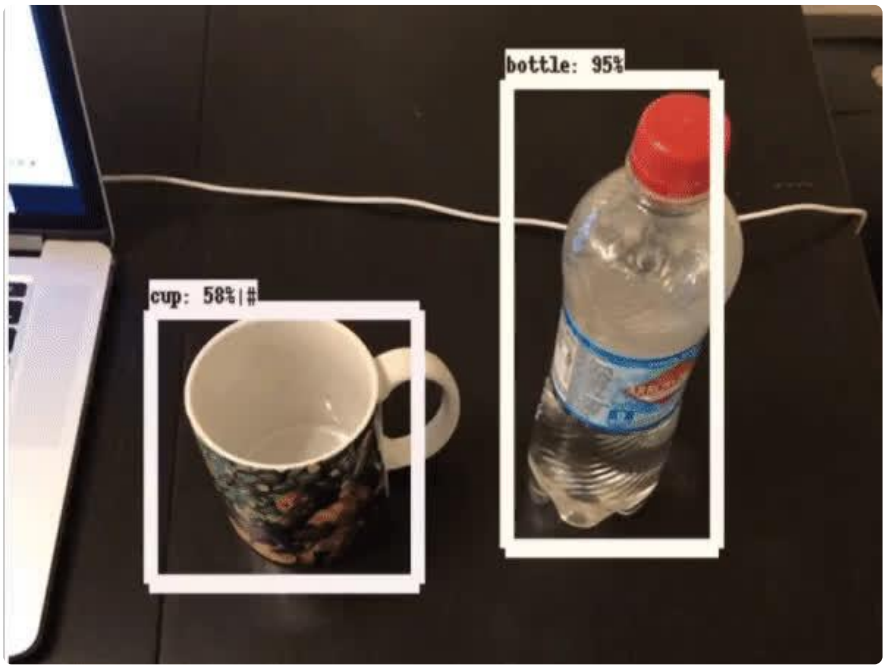
Last updated on 2023-08-29 04:18:08 PM EDT

Table of Contents

Welcome

You've heard about Machine Learning and AI - and you want to see what all the fuss is about. But you don't want to spend all your time installing bazel and Jupyter? Or maybe you're running an old computer or your only computer is a phone! What now, give up? Never! Thanks to Google Colab, you can run TensorFlow in a browser window, and all the computation is handled on Google's cloud service for free. It's a great way to dabble, without all the setup!

We've hacked together a Colab notebook that can see things using your computer, laptop, or phone camera! It takes live pictures from your camera and feeds them through the Mobilenet v2 + SSDLite model () to find and box the objects it sees. This way you can see what Mobilenet v2 + SSDLite can do, instantly!

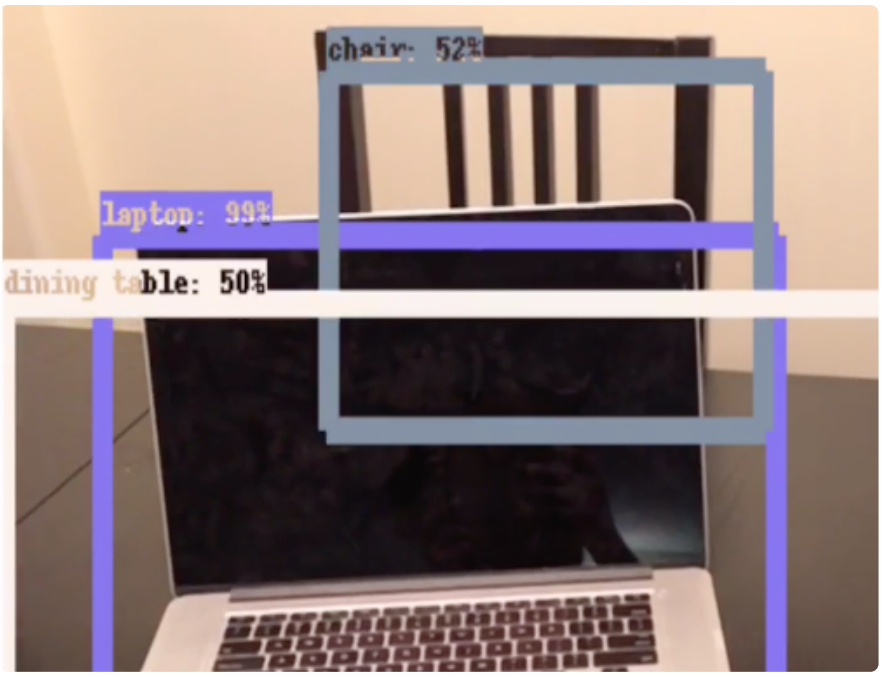


For this tutorial you will need a free Google account, a computer, phone or tablet with a camera or webcam, and a recent browser.

What is Object Detection?

Object Detection models find known objects in pictures. More advanced models also know how to bound the things they see--that is, they can tell you exactly where in the image it thinks they are.

Take a look below:



The algorithm produces three outputs here:

- The identified object, given both by name (water bottle) and an id number
- Confidence Level, a measure of the algorithm's certainty
- Bounding box, a box drawn around the image region that contains the object

Early object detection algorithms used hand-written heuristics to identify objects. For example: a tennis ball is usually round and green. While these had some successes, they were difficult to create and were prone to some hilarous false-positives.

Mobilenet v2

In recent years, a technology called neural networks has made it possible to let computers develop the heuristics on their own, by showing them a large number of examples. Mobilenet v2 is one of the well-known Object Detection models beacuse it's optimized to run on devices like your cell phone or a raspberry pi. When attached to another model known as SSDLite, a bounding box can be produced.

The authors of Mobilenet v2 + SSDLite claim it runs in 200ms on a Pixel 1. You won't be able to see speeds like this in your browser because this demo sends images securely over the Internet to Google's datacenter for processing. But, think of this as a helpful reference if you want to use this in your own project.

Mobilenet v2 + SSDLite can recognize 90 different objects.

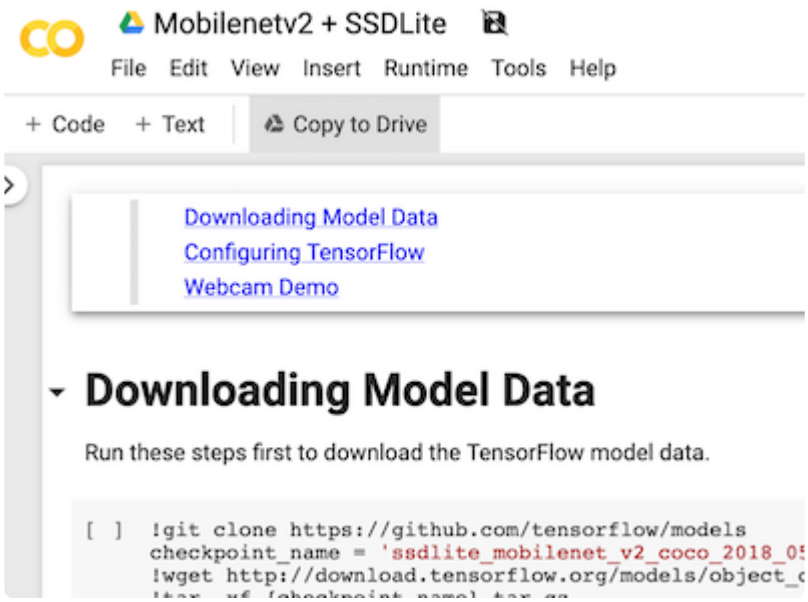
These objects are taken from the COCO dataset, a popular set of images used to develop object detection algorithms.

Opening the Notebook

To get started, click here to open our notebook in Colab.

When you click the link, it should take you to a page that looks like this:

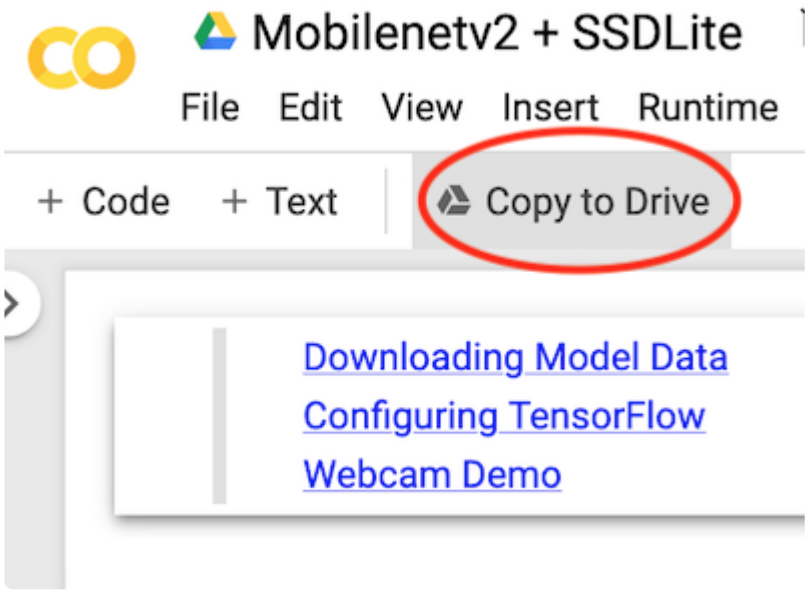
We recommend the Chrome browser on a desktop/laptop. On iOS (phone/tablet) use the default Safari browser.



Making changes

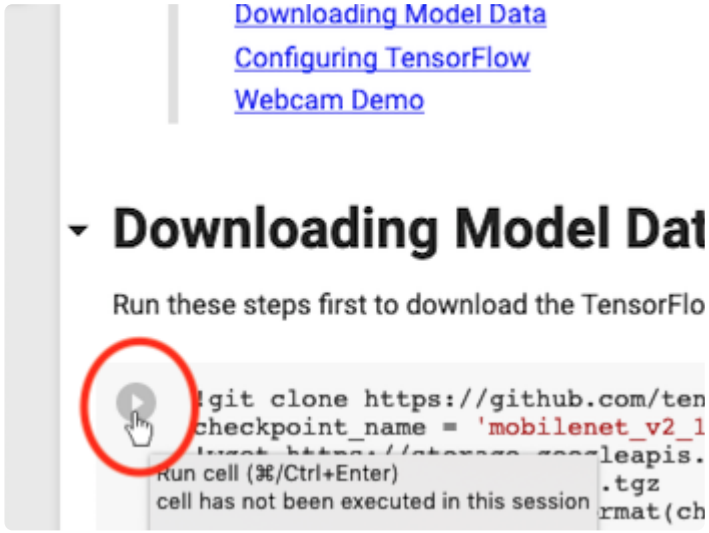
This link will open the notebook in playground mode, meaning it's read-only. But, you can make changes if you want to!

To make changes, you'll need to save a copy by clicking "Copy to Drive" like in the image below.



Running the Detector

To get started, move your mouse cursor over the `[ ]` box to the left of the first code snippet, underneath the Downloading Model Data header. It will change to a "Play" icon. Click on this icon.

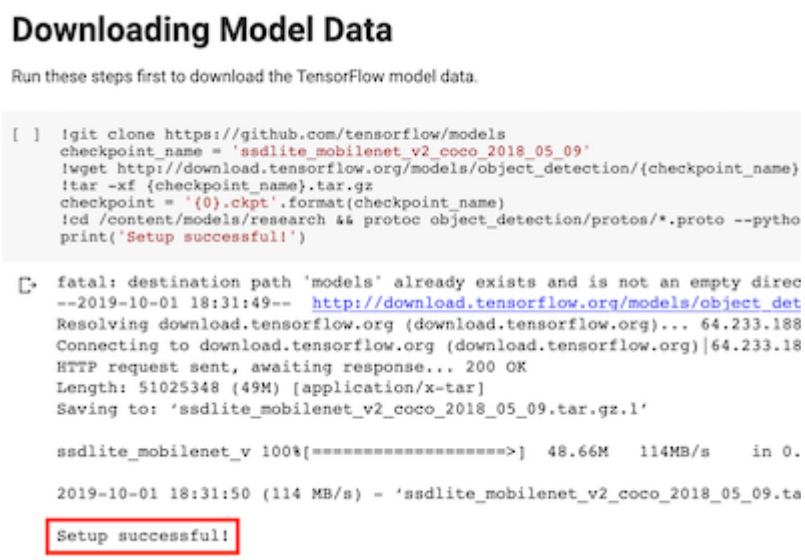


At this point, you may be prompted to sign in to your Google account. You'll need to sign in before you can continue with this guide.

What should happen?

After typically 20 seconds or so, you'll see the notebook come to life. The previous output will vanish and you'll see it replaced with the result of running on your new run time (see the section titled Aside below for more about what a runtime is).

When you see `Setup Successful!`, you know you've finished this step. You can open another copy of the notebook and compare it to our previous run, just to make sure it looks correct.



Aside: Behind the Scenes

Each time you open a Colab notebook, Google lets you temporarily use a computer in their datacenter to run your code. This computer is running a program called the runti me, which lets you play around with TensorFlow without having to worry about how fast your computer and without needing to buy an expensive graphics card.

When you close your Colab notebook, Google replaces your runtime with a brand new one, and releases your machine to someone else. This means that each time you come back, you'll need to set up the machine from scratch.

The first cell in the notebook does just this. It downloads the TensorFlow Model that will be used to recognize objects.

Configuring TensorFlow

Scroll down to the next code block, titled Configuring TensorFlow. Again, move your mouse over the `[ ]` box on the left, and click to run the code block.

When the code block finishes executing, you should see the following:

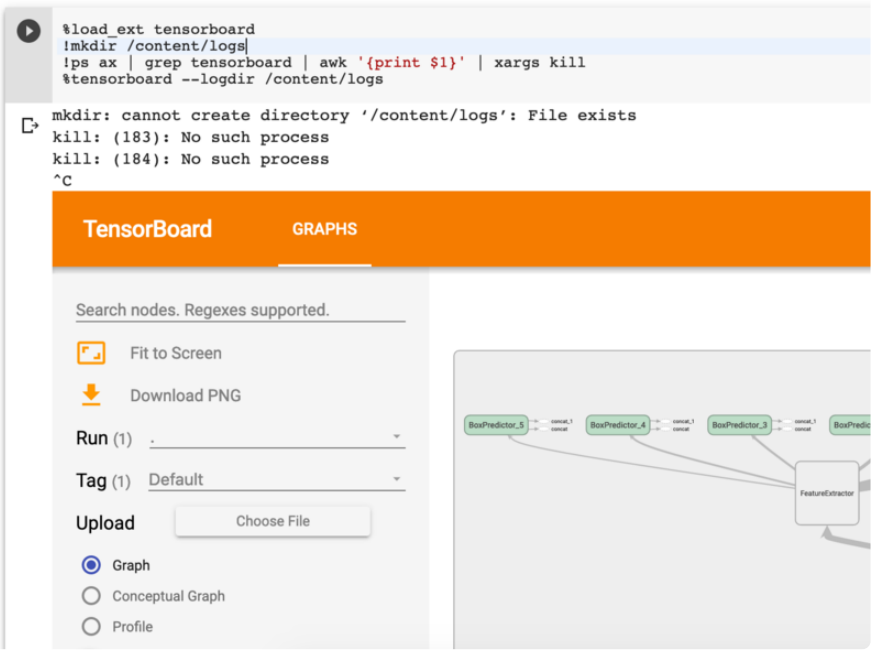
```
Model configured
```

Optional: Visualizing the graph

The code you just ran sets up a TensorFlow graph--a series of processing steps that translate the image from your camera into object labels and bounding boxes. You can visualize the graph using a tool called [TensorBoard]. This step isn't strictly necessary but it lets you peek inside the network to see the complexity behind the scenes.

To do this:

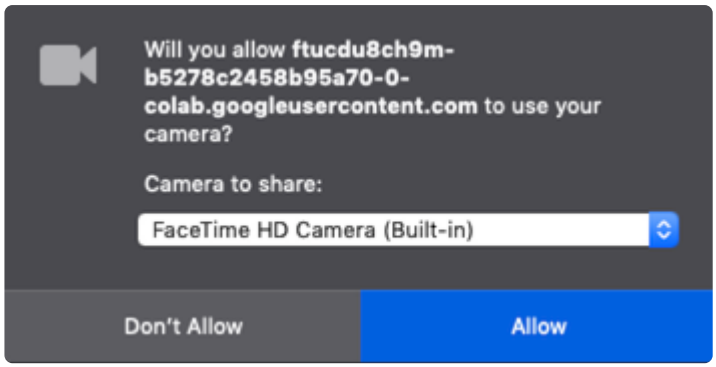
1. Under Optional: Visualize the Graph with TensorBoard, click Play.
2. Wait a few seconds after the block finishes executing. You should then see the TensorBoard UI appear below:



Starting the Demo

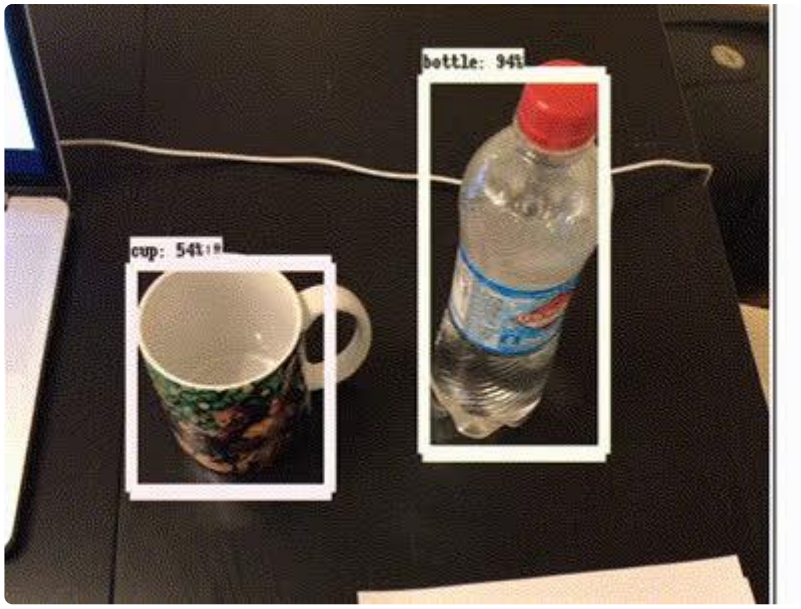
Now, scroll to the final code block, titled Demo and run it in the same way. This block of code won't stop until you tell it to.

After you click the play button, your browser should ask for permission to use your webcam. In Firefox, it looks like this:



Be sure to click Allow.

Finally, scroll down and you should see video from your camera appear on screen. Above the video, you'll see the output from the Object Detector.



You might notice that the model is a bit laggy. This is due to the way computation is split up in this example--the picture has to be sent from Javascript running on your machine to Python running in Colaboratory and back. Were this running on one machine in one programming language, it would be considerably faster!

Tips and Tricks

Play around with the detector to see how it performs best. Here are some hints from my experiments so far:

- Make sure the object takes up a significant part of the frame.
- Try with a uniform background for best results.

- Try changing the angle.