# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>3</td>
</tr>
<tr>
<td><strong>Training with Colab</strong></td>
<td>3</td>
</tr>
<tr>
<td>• Getting Started</td>
<td></td>
</tr>
<tr>
<td>• Configuring your Colab instance</td>
<td></td>
</tr>
<tr>
<td>• What should happen?</td>
<td></td>
</tr>
<tr>
<td>• Aside: Behind the Scenes</td>
<td></td>
</tr>
<tr>
<td>• Connect to Google Drive</td>
<td></td>
</tr>
<tr>
<td>• Training the model</td>
<td></td>
</tr>
<tr>
<td><strong>Install Docker</strong></td>
<td>7</td>
</tr>
<tr>
<td>• Signup and log into Docker</td>
<td></td>
</tr>
<tr>
<td>• Download and Install Desktop Docker</td>
<td></td>
</tr>
<tr>
<td>• TensorFlow needs a lot of computing resources</td>
<td></td>
</tr>
<tr>
<td><strong>Create/Fork Docker Image</strong></td>
<td>9</td>
</tr>
<tr>
<td><strong>Advanced: Build TensorFlow</strong></td>
<td>10</td>
</tr>
<tr>
<td><strong>Training and freezing models</strong></td>
<td>13</td>
</tr>
<tr>
<td>• Freeze</td>
<td></td>
</tr>
<tr>
<td>• Convert</td>
<td></td>
</tr>
<tr>
<td>• Extract &amp; Save</td>
<td></td>
</tr>
<tr>
<td><strong>Commit Docker</strong></td>
<td>16</td>
</tr>
</tbody>
</table>
Overview

Machine learning has come to the 'edge' - small microcontrollers that can run a very miniature version of TensorFlow Lite to do ML computations. The first demos available are for 'micro speech' which is detecting a couple words. The default words are 'yes/no' but the dataset contains many other words! This guide goes through how to train micro speech models on your own.

Training with Colab

To train with Colab, you'll need to have a good Internet connection and be able to leave your browser running with the Colab webpage open for several hours. Make sure your laptop battery is charged before you start this guide!

Getting Started

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:
Configuring your Colab instance

To get started, move your mouse cursor over the box to the left of the first code snippet, underneath the Configure training header. It will change to a "Play" icon. Click on this icon.

Configure training

The following `os.environ` lines can be customized to set the words that will be trained for, and the default values will result in the same model that is used in the micro.speech example. Run the cell to:

```python
import os

# A comma-delimited list of the words to train against.
# The options are: yes,no,up,down
# All other words will be used to train.
# SPLIT_WORDS = "yes", "no"
```

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 the following output, 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.

- Training these words: ['yes', 'no']
- Training steps in each stage: [15000, 3000]
- Learning rate in each stage: ['0.001', '0.0001']
- Total number of training steps: 18000

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 runtime, 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 few cells in the notebook do just that.

Connect to Google Drive

As mentioned in the last Aside section, the runtime (along with any files created) is lost when you close your browser tab. We'll need to find somewhere more permanent to store our trained model. Fortunately, Google provides a handy way to connect your Google Drive to the notebook. It appears just like a regular folder on the Colab runtime.

The next cell will connect your Google Drive to the Colab. You'll need to authorize the connection in a new browser tab. First, run the next cell, titled Connect to Google Drive. You should see something like this appear:
Click the URL, which will open a new browser tab. Go through the steps to select the Google account you want to use, and allow access to the Google Drive File Stream app. This was written by Google to work with Colab.

When you've successfully authorized it, you'll see this screen (the code in this image is blurred). Copy the code (or click the helpful copy button to the right), switch back to the Colab tab, and paste the code into the text box below the Enter your authorization code text.

```
import os.path
from google.colab import drive

DRIVE_STORAGE_PATH = '/content/drive/My Drive/speech'

def ensure_drive():
    if not os.path.exists('/content/drive/My Drive'):
        drive.mount('/content/drive', force_remount=True)

ensure_drive()
```

Go to this URL in a browser: https://accounts.google.com
Enter your authorization code:  

Training the model

Now you're ready to train your speech recognition model! Run the next few cells, titled Install Dependencies and Download Tensorflow.
If you want to visualize training while it's in progress, run the Optional: Visualize graph and training rate cell. This isn't required, though.

Finally, run the Create trained model cell. This will run for several hours, and you can't close your browser tab--so, be sure you can leave your computer running for a while.

**Model Output**

You can find your model output on your [Google Drive](https://drive.google.com), in a folder called *speech-recognition*. You should see something like the following screen.

---

**Install Docker**

We recommend training with Colab since you won't have to deal with Docker management. That said, it is possible to docker-ize training.

We need to be able to run a specific version/commit of TensorFlow and the dependancy requirements for TF are very extreme. We strongly suggest against trying to compile and run on your native computer OS - that way we don't get weird interactions with your OS, compiler toolchain, Python kit, etc. Also, TF really wants to run on a particular version of Linux and chances are you aren't running it.
Instead, we will be using Docker to containerize and separate the TF build so we have a compact, clean, dependable build. Docker is lighter than VMWare/vagrant, and has a very nice ‘hub’ backend for saving/restoring your images, all for free!

Signup and log into Docker

Sign up at https://hub.docker.com/signup

You don't need to pay for an account, but be aware the software images we'll be using are public so don't put any private data in em!

Download and Install Desktop Docker

Download Docker software for Windows or Mac, whichever matches your computer

TensorFlow needs a lot of computing resources

Give it as many CPUs and as much RAM as you can spare

You need to give it at least 8 GB of RAM or gcc will fail with a very annoying and somewhat confusing error like this (but on some other file)

```
ERROR: /root/tensorflow/tensorflow/core/kernels/BUILD:3371:1: C++ compilation of rule '//tensorflow/core/kernels:reduction_ops' failed (Exit 4)
gcc: internal compiler error: Killed (program cc1plus)
Please submit a full bug report,
```
Open a command terminal and try to login, use the same username/password as from the site

```
C:\\Users\\ladyada\\docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: ladyada
Password: [hidden]
login Succeeded
```

OK you're ready to go!

## Create/Fork Docker Image

Start with the official TensorFlow Docker image, like github you can pull/commit/push and implicitly fork when you do this between sources.

```
docker pull tensorflow/tensorflow
```

will get you the latest docker image from Google

```
C:\\Users\\ladyada\\Desktop\\ml\\docker pull tensorflow/tensorflow
Using default tag: latest
Latest: Pulling from tensorflow/tensorflow
5b7339215d1c: Pull complete
4b084e09f672: Pull complete
e31c30e77a94: Pull complete
eb46706e5b7: Pull complete
c6e2a2b6bc88: Pull complete
f2854be8b851: Pull complete
c204a719ea4c: Pull complete
2d62b866f6: Pull complete
c7610033dbfa: Pull complete
9ba4eb8370e: Pull complete
909b027dfab9d7be: Pull complete
Status: Downloaded newer image for tensorflow/tensorflow:latest
```

Log into the Docker image with

```
docker run -it tensorflow/tensorflow bash
```
Within the Docker root shell, install some dependencies with

```
apt-get install -y curl zip git
```

### Advanced: Build TensorFlow

If you need to compile TensorFlow from scratch, you can do it, but it's very slow to get everything compiled. Once it's compiled, it's really fast to train models!

We have to start this way, until there's more automated methods...so here's a guide on how we did it

```
While this method takes a long time it's the only way we were able to build models, hopefully there will be an easy to use pip installer soon!
```

We need to use version 0.23.1 of bazel (the build tool), so we'll install that specific version like this:

```
cd ~
curl -O -L https://github.com/bazelbuild/bazel/releases/download/0.23.1/bazel-0.23.1-installer-linux-x86_64.sh
chmod +x bazel-0.23.1-installer-linux-x86_64.sh
./bazel-0.23.1-installer-linux-x86_64.sh
```
You can verify it with `bazel version`

For some reason, the image is still using Python 2.7, so grab the future package so we can run python3 code

```
pip install future
```

We also need to get the right version of the 'estimator' package (we use it later)

```
pip uninstall tensorflow_estimator
pip install -I tensorflow_estimator==1.13.0
```
We need to build a specific commit of TensorFlow, so clone the repo then switch to that commit

```
$ git clone https://github.com/tensorflow/tensorflow.git
$ cd tensorflow
$ git checkout 4a464440b2e8f382f442b6e952d64a56701ab045
```

Go with the default configuration by running

```
yes *** | ./configure
```

Finally start the TensorFlow compile and speech training with

```
bazel run -c opt --copt=-mavx2 --copt=-mfma tensorflow/examples/speech_commands:train -- --model_architecture=tiny_conv --window_stride=20 --preprocess=micro --wanted_words="yes,no" --silence_percentage=25 --unknown_percentage=25 --quantize=1
```
This will create a micro model of the large speech data set with only "yes" and "no" words in the model (to keep it small/simple)

This will take many hours especially the first time! Go take a break and do something else (or, you can try using your computer but it will be slow because Docker is sucking up all the computational resources to compile 16,000 files)

After TensorFlow has completed compiling it will take another 2+ hours to run the training. In the end you will get something like this:

Training and freezing models

Start training a new micro speech model with

```
python tensorflow/examples/speech_commands/train.py -- --model_architecture=tiny_conv -- --window_stride=20 -- --preprocess=micro -- --wanted_words="yes, no" -- --silence_percentage=25 -- --unknown_percentage=25 -- --quantize=1
```
or, if using bazel

```
bazel run -c opt --copt=-mavx2 --copt=-mfma tensorflow/examples/speech_commands:train -- --model_architecture=tiny_conv -- window_stride=20 --preprocess=micro --wanted_words="yes,no" -- silence_percentage=25 --unknown_percentage=25 --quantize=1
```

This will run for a few hours

At the end you’ll get your final test accuracy and checkpoint file

```
Step 10000: validation accuracy = 89.48% (N=44445)
```

Checkpoint files are stored in /tmp

In this case we want /tmp/speech_commands_train/conv.ckpt-18000.* (the last place the trainer saved to)
Freeze

Take the trained weights and turn them into a frozen model on disk.

```python
tensorflow/examples/speech_commands/freeze.py --
model_architecture=tiny_conv --window_stride=20 --preprocess=micro --
wanted_words="yes,no" --quantize=1 --output_file=/tmp/tiny_conv.pb --
start_checkpoint=/tmp/speech_commands_train/conv.ckpt-100
```

or if using bazel something like:

```bash
bazel run tensorflow/examples/speech_commands:freeze --
model_architecture=tiny_conv --window_stride=20 --preprocess=micro --
wanted_words="yes,no" --quantize=1 --output_file=/tmp/tiny_conv.pb --
start_checkpoint=/tmp/speech_commands_train/tiny_conv.ckpt-18000
```

Convert

Convert the TensorFlow model into a TF Lite file

```bash
bazel run tensorflow/lite/toco:toco -- --input_file=/tmp/
tiny_conv.pb --output_file=/tmp/tiny_conv.tflite --
input_shapes=1,49,40,1 --input_arrays=Reshape_1 --
output_arrays='labels_softmax' --inference_type=QUANTIZED_UINT8 --
mean_values=0 --std_values=9.8077
```

The file can now be found in /tmp/tiny_conf.tflite
Extract & Save

Finally, you can use docker cp to copy the file from your container to your desktop. From the host computer (not the docker container) run docker cp CONTAINERID:/tmp/tiny_conf.tflite .

You should now have access to the file!

Here are some example files

updown.zip
yesno.zip
catdog.zip

Commit Docker

Now's a good time to 'save' our work. Run docker ps to list all your docker containers

you can 'save' this docker container to your account with

docker commit CONTAINER_ID USERNAME/mytensorflow

where the CONTAINER_ID is the 12 character that is to the left of the image name and USERNAME is your docker login name. So in my case, docker commit c2a0a7f0a7bb ladyada/mytensorflow
It will take a few minutes while Docker runs, eventually you'll get this on the terminal:

```
PS C:\Users\llinor> docker commit c2a87f0a7bb ladyada/mytensorflow
sha254:8f0a7bb ladyada/mytensorflow
PS C:\Users\llinor>
```

Then push it to docker hub with docker push username/containername

Then visit your dockerhub profile to see that you have in fact pushed the docker image