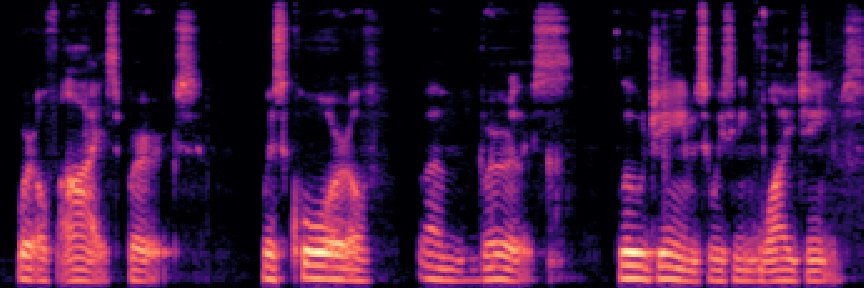
In auditory research, a spectrogram is a graphical representation of audio that has frequency on the vertical axis, time on the horizontal axis, and a third dimension of color represents the intensity of the sound at each time x frequency location.

Here is an example of a voice spectrum:



In this spectrogram, we can see many frequencies that are multiples of the fundamental frequency of the note being played. These are called harmonics in music. The vertical lines throughout the spectrogram are the brief pause between each word we say. So it appears the spectrogram contains lots of information about the nature of different sounds. Here is a script that will convert each wav file into a spectrogram. Each spectrogram is stored in a folder corresponding to its category:

for ele in i:

count=-1

with open('Data/'+ele+'/'+ele+'\_df.csv') as csvfile:

spamreader = csv.reader(csvfile, delimiter=',')

for row in spamreader:

count+=1

if count == 0:

continue

if not os.path.exists('spectrograms/' + row[2]):

os.makedirs('spectrograms/' + row[2])

os.makedirs('spectrograms/test/' + row[2])

y, sr = librosa.load("Data/"+ele+"/"+row[1])

# make and display a mel-scaled power spectrogram

S = librosa.feature.melspectrogram(y, sr=sr, n\_mels=128)

# Convert to log scale (dB). Use the peak power as reference.

log\_S = librosa.power\_to\_db(S)

fig = plt.figure(figsize=(12,4))

ax = plt.Axes(fig, [0., 0., 1., 1.])

ax.set\_axis\_off()

fig.add\_axes(ax)

# Display the spectrogram on a mel scale

# sample rate and hop length parameters are used to render the time axis

librosa.display.specshow(log\_S, sr=sr, x\_axis='time', y\_axis='mel')

# Make the figure layout compact

plt.savefig('spectrograms/' + row[2] + '/' + row[1] + '.jpg')

plt.close()

The other nice thing about using the spectrogram is that we have now changed the problem into one of image classification, which can be used on one the most sophisticated Machine Learning models—inception v3. Transfer learning is where we take a neural network that has been trained on a similar dataset, and retrain the last few layers of the network for new categories. The idea is that the beginning layers of the network are solving problems like edge detection and basic shape detection, and that this will generalize to other categories. Specifically, Google has released a pretrained model called Inception. We can run the script to retrain on our spectrograms in 8000 steps on the terminal:

python retrain.py \

--bottleneck\_dir=bottlenecks \

--how\_many\_training\_steps=8000 \

--model\_dir=inception \

--summaries\_dir=training\_summaries/basic \

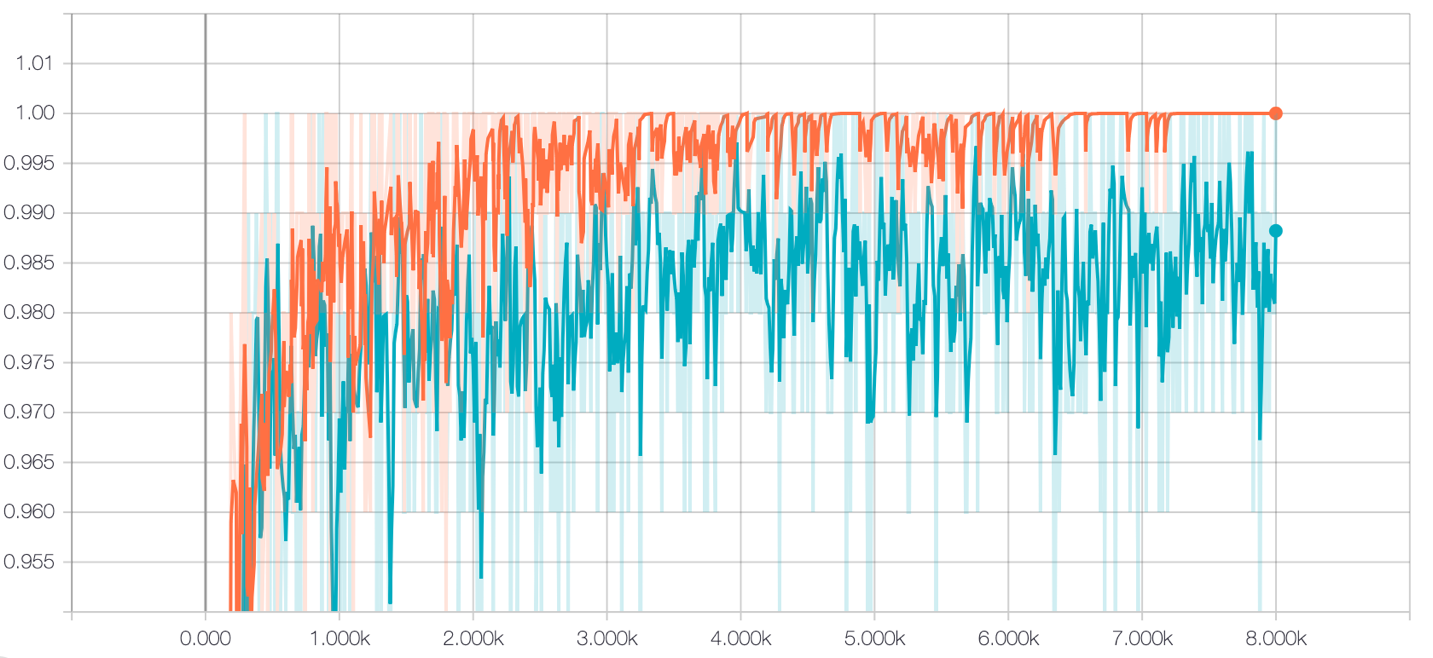
--output\_graph=retrained\_graph.pb \

--output\_labels=retrained\_labels.txt \

--image\_dir=spectrograms

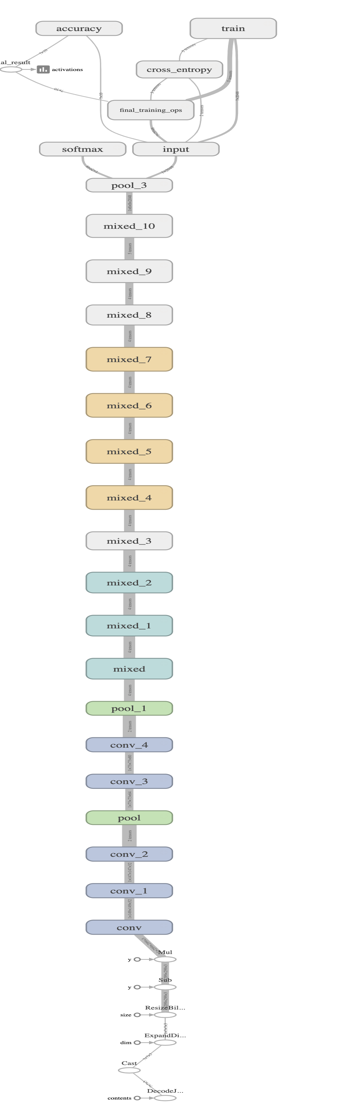
After training we can run another command to review the training progress, accuracy and model structure in our browser:

tensorboard --logdir training\_summaries



After around 3k iterations the accuracy tops off at 99.5% on the validation set. Not bad for a fairly naive approach to sound classification.

Here is the Model Structure:



Then use to test the sanity test files, we need to convert the .wav to spectrum first:

y, sr = librosa.load('testN/rg\_audio/rg\_val\_sent002.wav')

# make and display a mel-scaled power spectrogram

S = librosa.feature.melspectrogram(y, sr=sr, n\_mels=128)

# Convert to log scale (dB). Use the peak power as reference.

log\_S = librosa.power\_to\_db(S)

fig = plt.figure(figsize=(12,4))

ax = plt.Axes(fig, [0., 0., 1., 1.])

ax.set\_axis\_off()

fig.add\_axes(ax)

# Display the spectrogram on a mel scale

# sample rate and hop length parameters are used to render the time axis

librosa.display.specshow(log\_S, sr=sr, x\_axis='time', y\_axis='mel')

# Make the figure layout compact

#plt.show()

plt.savefig('spectrograms/test/rene\_g/rg2.jpg')

plt.close()

We can run this script on the terminal to test:

python label\_image.py \

--graph=retrained\_graph.pb\

--labels=retrained\_labels.txt\

--image=spectrograms/test/mikio\_d/md2.jpg

After test all the 12-sanity test, the Inception v3 model have a final accuracy of 75, where both of the Mikio’s simple failed and one of the Serhad’s simple failed. Even the result was a bit disappointing, we found out that our model returned really high certainty on the simple that the test result is positive.