Automatic Music Genre Classification with Deep Learning

Kenny Jones and Derrick Bonafilia

Project Overview

Attempted to identify genre from music samples and high-level music features

Used two data sets, GTZAN and MSD, in two separate neural networks, a DNN and a CNN respectively

Implemented networks in Tensorflow

Data Collection: A Trip down the Rabbit Hole

- Music is ubiquitous, but often copyrighted.
- Whole songs effectively entirely unavailable (free and legally at least)
- 30 second samples are legally iffy and are mostly unavailable except for in places where the music is also being sold

Potential Solutions

- GTZAN:
  - GTZAN is effectively the current benchmark dataset for genre classification.
  - 1000 thirty second song samples, 100 songs each for 10 genres
  - Presumably legal because of academic intention (also small scale)
  - Drawbacks: dataset is tiny!!!! And Deep Learning works better with lots of data
- Million Song Dataset:
  - Contains metadata and extracted features for the one million songs
  - Does NOT contain genre classifications
  - To avoid copyright right issues, the dataset chooses to not host any of the audio

First Approach

GTZAN is boring and old. Big Data is the trendy thing. 1 Million > 1 Thousand

Let’s go for it with the Million Song Dataset(MSD)

TU Wien in Vienna put together a rather rigorous mapping of songs in the MSD to their genre as classified by a voting mechanism over several website

TU Wien also put up a much easier to work with version of the MSD complete with their own audio features that they computed as well. 2 GB’s instead of 300GB, this seems feasible.

More Data Collection

We worked with the data we retrieved from this and found that a deep neural net was very marginally better than a one layer neural net. This makes sense, features have already been extracted.

This is the most ahead of time we've ever been on a project.

Let's see what we can make happen with a more raw audio approach.

Several other sources have managed to get the audio for the Million Song Dataset, I bet we can do it too.

3 API Keys and More than 10 Hours of Data Collection later..

Pretty much every song in the million song dataset has a 30 second sample on 7 digital that can be accessed via the 7digital API, there's even a 7digital id for each song in the MSD to ease the process

9/10 of these id's are wrong

API allows 4000 accesses per day.

It takes all of these accesses for one key to just to recalculate 2000 id's

It takes another to download the sample for all of these songs

Somehow after running a script to clean the data I only had 789 songs and I had burnt through 2 API Keys

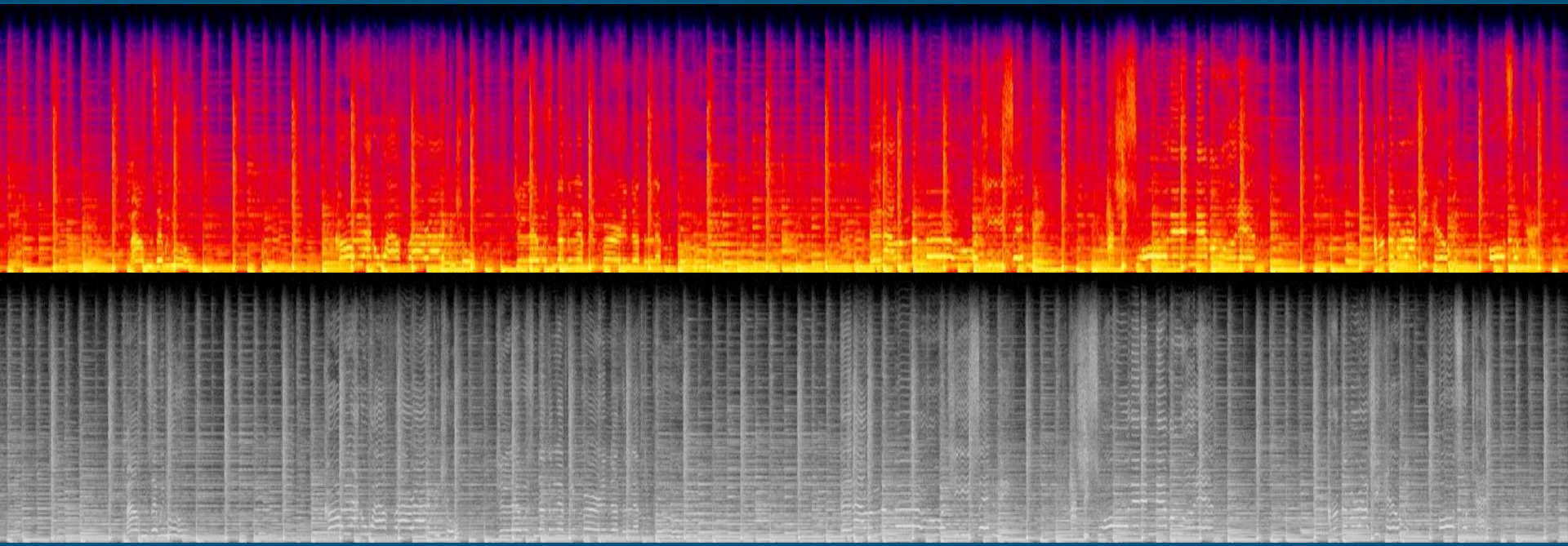
A Return to GTZAN

Reading a couple articles and listening to a few songs convinced me that humans could identify genre by just listening to a short sample of a song.

We could then create more data by slicing the song samples into pieces. 1000 songs samples turns into 5,000 or even 10,000

Plus we can now compare our results to lots of literature

Data Representation: Spectrograms



Spectrograms show the frequency of audio on the y-axis over the time on the x-axis.

TensorFlow-Variable initialization

```
x = tf.placeholder(tf.float32, shape=[None,iNum])
y_ = tf.placeholder(tf.float32, shape=[None,oNum])

h1= weight_variable([iNum, hNum1])
h2= weight_variable([hNum1, hNum2])
outW= weight_variable([hNum2, oNum])

b1 = bias_variable([hNum1])
b2 = bias_variable([hNum2])
outB = bias_variable([oNum])
```

TensorFlow - DNN Network Structure

```
# Hidden layer with RELU activation
layer_1 = tf.add(tf.matmul(x,h1), b1)
layer_1 = tf.nn.relu(layer_1)

# Hidden layer with RELU activation
layer_2 = tf.add(tf.matmul(layer_1, h2), b2)
layer_2 = tf.nn.relu(layer_2)

#dropout layer
keep_prob = tf.placeholder(tf.float32)
h_drop = tf.nn.dropout(layer_1, keep_prob)

# Output layer with linear activation
y_conv = tf.matmul(h_drop, outW) + outB
```

TensorFlow - CNN Network Structure

```
# First convolutional and pool layer

W_conv1 = weight_variable([5, 5, 1, 16])
b_conv1 = bias_variable([16])
h_conv1 = tf.nn.elu(conv2d(x_image, W_conv1) + b_conv1)
h_pool1 = max_pool_2x2(h_conv1)
# Second convolutional and pool layer
W_conv2 = weight_variable([5, 5, 16, 32])
b_conv2 = bias_variable([32])
h_conv2 = tf.nn.elu(conv2d(h_pool1, W_conv2) + b_conv2)
h_pool2 = max_pool_2x2(h_conv2)
# Third convolutional and pool layer
W_conv3 = weight_variable([5, 5, 32, 64])
b_conv3 = bias_variable([64])
h_conv3 = tf.nn.elu(conv2d(h_pool2, W_conv3) + b_conv3)
h_pool3 = max_pool_2x2(h_conv3)
```

```
W_fc1 = weight_variable([8 * 8 * 64, 512])
b_fc1 = bias_variable([512])
#Fully connected layer
h_pool2_flat = tf.reshape(h_pool3, [-1,
8*8*64])
h_fc1 = tf.nn.elu(tf.matmul(h_pool2_flat,
W_fc1) + b_fc1)
#Dropout layer
keep_prob = tf.placeholder(tf.float32)
h_fc1_drop = tf.nn.dropout(h_fc1,
keep_prob)
#Output layer
W_fc2 = weight_variable([512, 10])
b_fc2 = bias_variable([10])

y_conv = tf.matmul(h_drop, W_fc2) + b_fc2
```

TensorFlow Training and Testing Functions

```
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y_conv, y_))
train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv,1), tf.argmax(y_,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

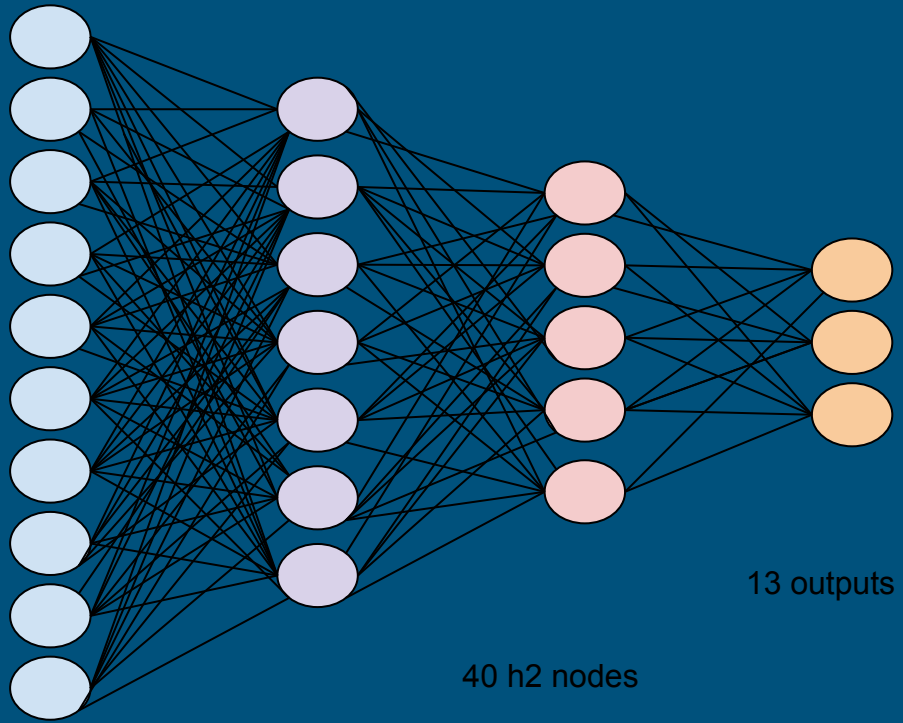
TensorFlow-Voting

We split the song into n slices, we run the model on all n of these keeping track of the probability computed for each genre for each slice and choose the genre with the highest probability

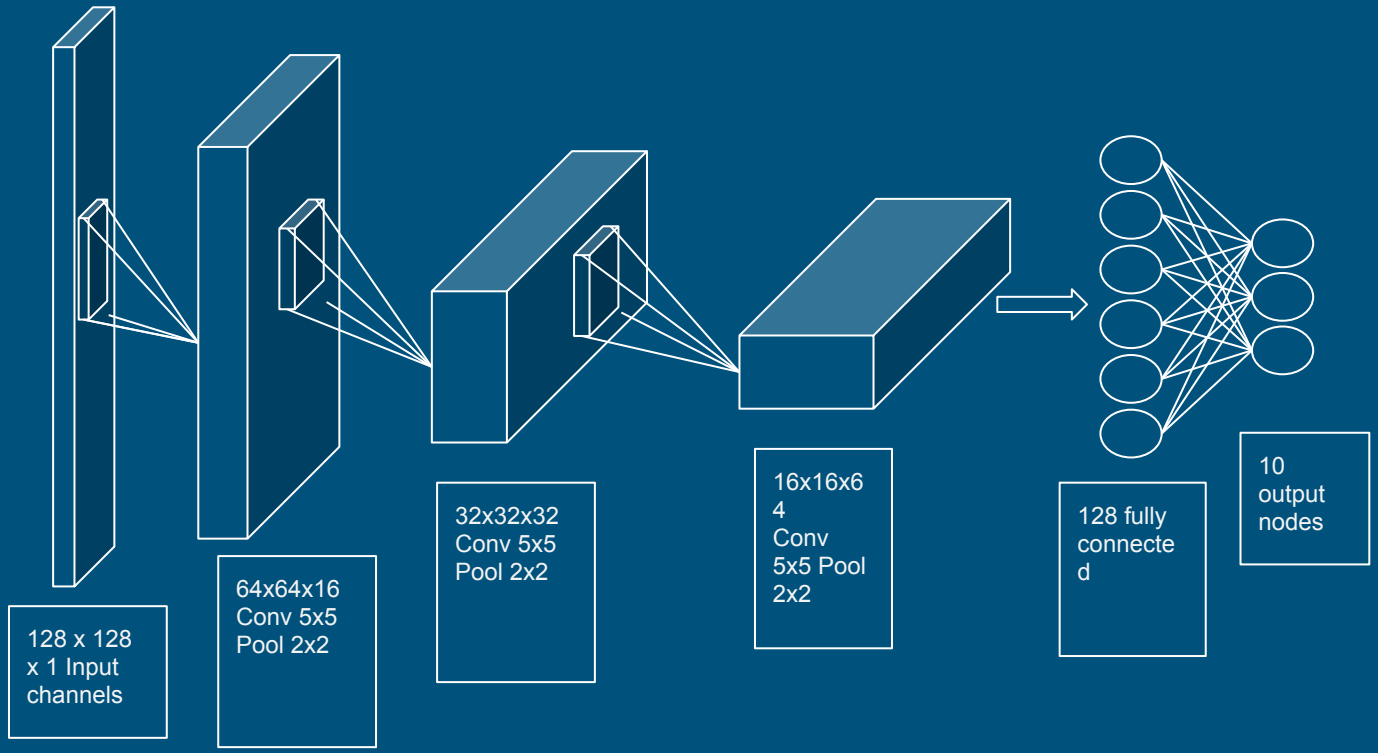
#Sum up output for each of the n slices and then return index with max value

```
def voting(song,genre):
    totalProb = numpy.zeros([10])
    for slice in song:
        prediction=probabilities.eval(feed_dict={x: numpy.array([slice]), keep_prob: 1.0},session = sess)
        totalProb = totalProb + prediction
    return numpy.argmax(totalProb)==numpy.argmax(genre)
```

Model - DNN Diagram



Model - CNN Diagram



Experimental Set up

MSD

- Used script from TU Wein to divide data set into a test and train partition

GATZAN

- From 1000 songs create randomized 900 * numSlice large training set and 100 sized testing set with equal representation from Genres

Results

67 percent hit on MSD with 2 layer DNN

83 percent hit on GTZAN spectrograms with 3 convolution and pooling layers on 5 sliced 128 x 128 resized images

85 percent hit on GTZAN spectrograms with 2 convolution and pooling layers on 10 sliced 128 x 128 not resized images

Results - Comparison to literature values

83 ± 1.1% accuracy on the Tzanetakis (GTZAN) datase Improved music feature learning

with deep neural networks - http://ieeexplore.ieee.org/document/6854949/?denied (2014)

In the original paper, (Tzanetakis, 2002), the best classification result reported was 61 % accuracy (4 % standard deviation on 100 iterations of a 10-fold cross validation) using Gaussian Mixture Models and the 30 dimensional MARSYAS genre features.

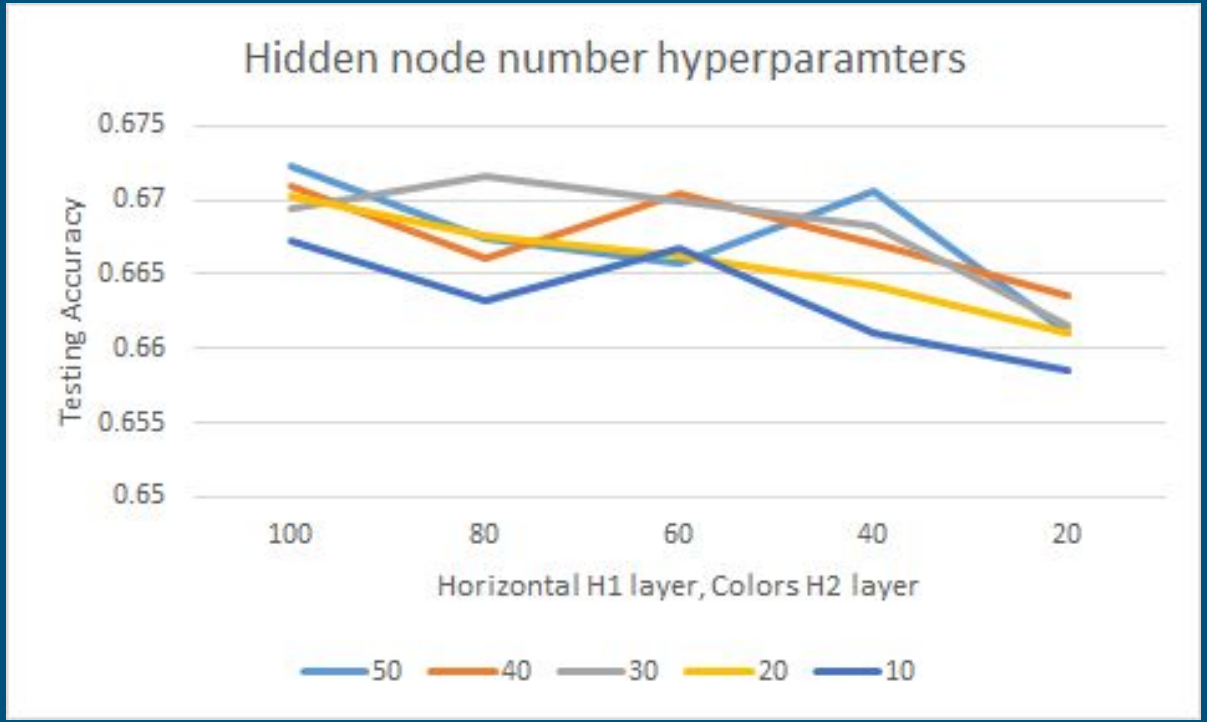
(Li et al., 2003) in “Toward intelligent music information retrieval” achieved 74.9 % classification accuracy in a 10-fold cross validation using Support Vector Machines (SVM) with pairwise classification and 78.5 % accuracy using SVM with one-versus-the-rest classification.

Paper purported to have 91 percent accuracy on GTZAN - Music genre classification via sparse representations of auditory temporal modulations

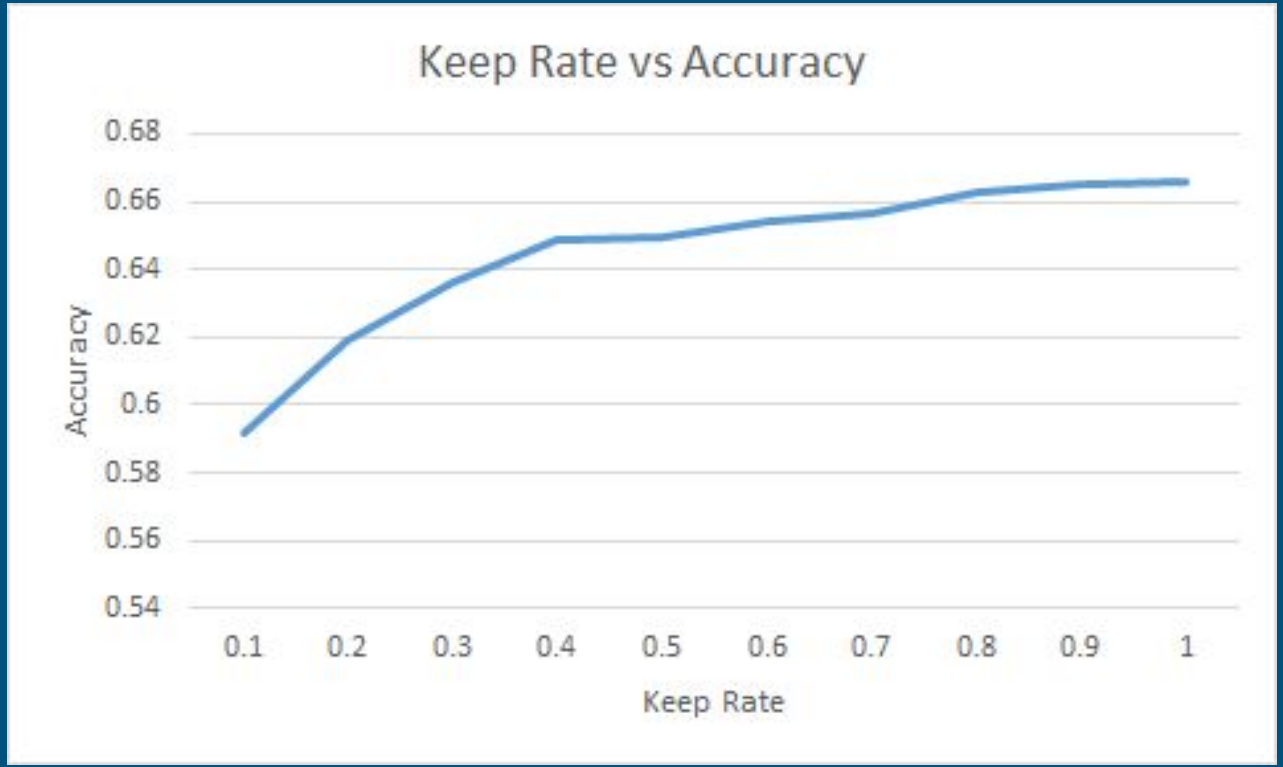
Follow up paper - On Automatic Music Genre Recognition by Sparse Representation Classification using Auditory Temporal Modulations

The only way we have found to obtain something close to the 91% mean accuracy reported in [31] is to limit the classification problem to the first five genres of GTZAN: blues, classical, country, disco, and hiphop.

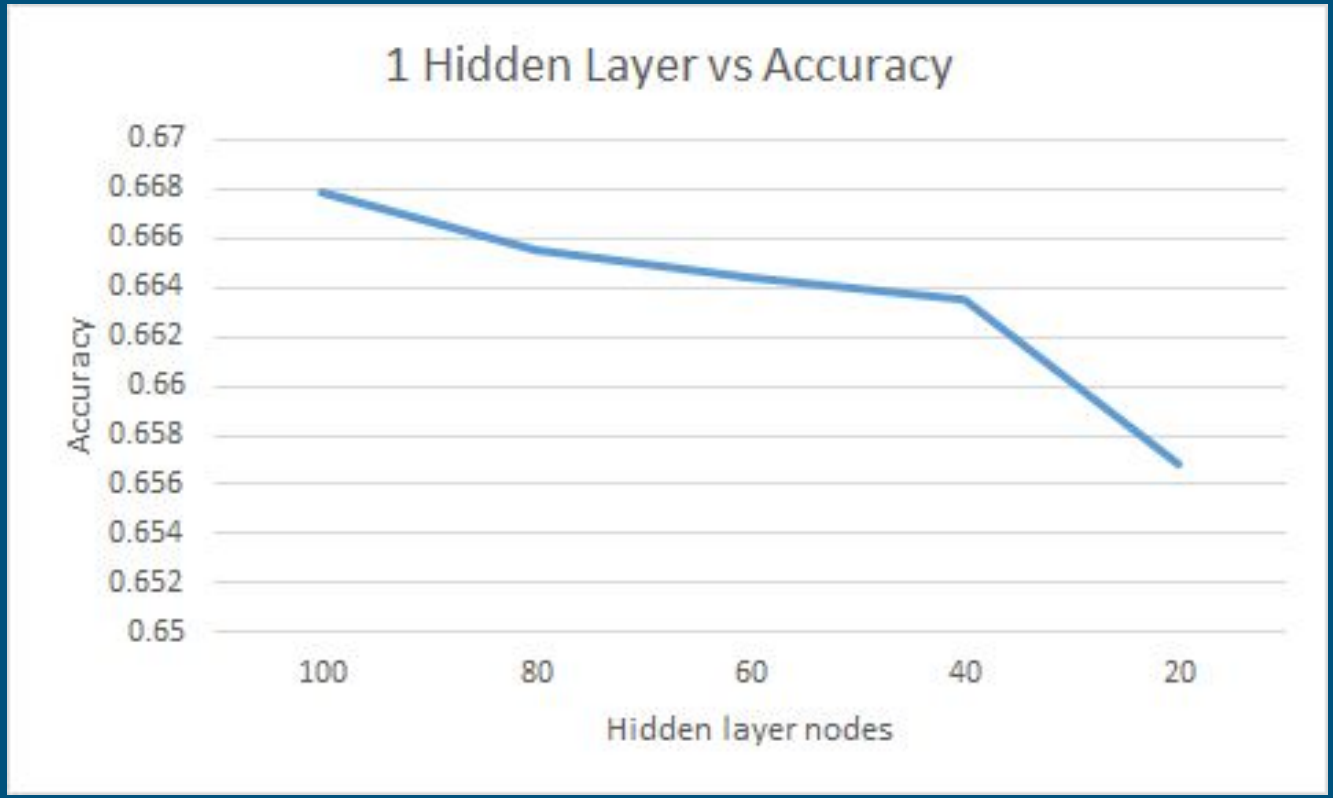
Results


Hidden node number hyperparamters
0.675
0.67
0.665
0.66
0.655
0.65
Testing Accuracy
100
80
60
40
20
Horizontal H1 layer, Colors H2 layer
50
40
30
20
10

Results


Keep Rate vs Accuracy
0.68
0.66
0.64
0.62
0.6
0.58
0.56
0.54
Accuracy
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Keep Rate

Results


1 Hidden Layer vs Accuracy
Accuracy
0.67
0.668
0.666
0.664
0.662
0.66
0.658
0.656
0.654
0.652
0.65
100
80
60
40
20
Hidden layer nodes

Some Difficulties

Small training set

Memory issues prevented up from running a larger model

Poor initialization values for the model sometimes led to the model not really learning at all

Future Work

GTZAN is bad and small, use a different data set or gather one ourselves

Experiment with spectrograms from different length time series and different sized level of detail

Try out one dimensional pooling across image in addition or in substitution of 2 dimensional pooling

Either parallelize GPU power or use cloud computing (Amazon EC2) to be able to process bigger images and larger datasets

Perform more rigorous cross-validation while testing

See if we can improve on results using three and four layer CNN for 10 sliced data

References

https://chatbotslife.com/finding-the-genre-of-a-song-with-deep-learning-da8f59a61194#.gle9s0vvy Julian Despois, “Finding the genre of a song with Deep Learning”

http://benanne.github.io/2014/08/05/spotify-cnns.html Sander Dieleman(Currently at Google Deep Mind and co-author of the AlphaGo paper this year!), “Recommending music on Spotify with deep learning”

http://cmmr2012.eecs.qmul.ac.uk/sites/cmmr2012.eecs.qmul.ac.uk/files/pdf/papers/cmmr2012_submission_17.pdf Sturm “On Automatic Music Genre Recognition by Sparse Representation Classification using Auditory Temporal Modulations”

http://ieeexplore.ieee.org/document/6854949/ Sigtia “Improved music feature learning with deep neural networks”

TensorFlow Tutorials:
https://www.tensorflow.org/versions/r0.12/tutorials/mnist/beginners/index.html#mnist-for-ml-beginners
https://www.tensorflow.org/versions/r0.12/tutorials/mnist/pros/index.html#deep-mnist-for-experts

Million Song Dataset http://www.ifs.tuwien.ac.at/~schindler/pubs/ISMIR2012.pdf

References (cont)

ieeexplore.ieee.org/iel5/89/21966/01021072.pdf "Musical Genre Classification of Audio Signals" George *Tzanetakis*

http://ieeexplore.ieee.org/document/1632041/ Toward intelligent music information retrieval Li