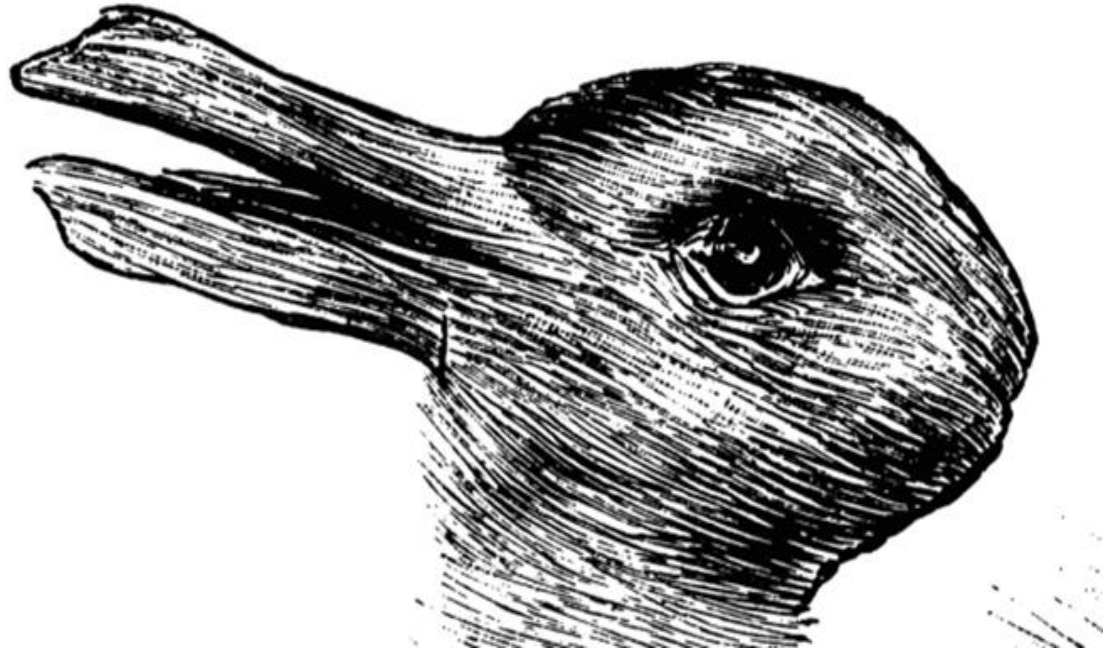




# Using Deep Learning to Estimate Multiple Song Tempos

**THIS IS  
A DUCK**  
(facing left)

**THIS IS  
ALSO  
A RABBIT**  
(facing right)





**ONE SONG  
CAN HAVE  
MORE THAN  
ONE TEMPO**

**EVEN IF THE  
TEMPO  
DOESN'T  
CHANGE  
DURING THE  
SONG**

Jackson 5

“I Want You Back”

100 Beats per Minute 🎵

Good for typical walking pace



Jackson 5

“I Want You Back”

200 Beats per Minute 🎵

Good for brisk running pace



## **AUTOMATIC TEMPO ESTIMATION METHODS**

- ◎ Sometimes unreliable.
- ◎ Generally produce only one answer.

*Try to devise one that will be more reliable  
and can give more than one answer when  
appropriate*

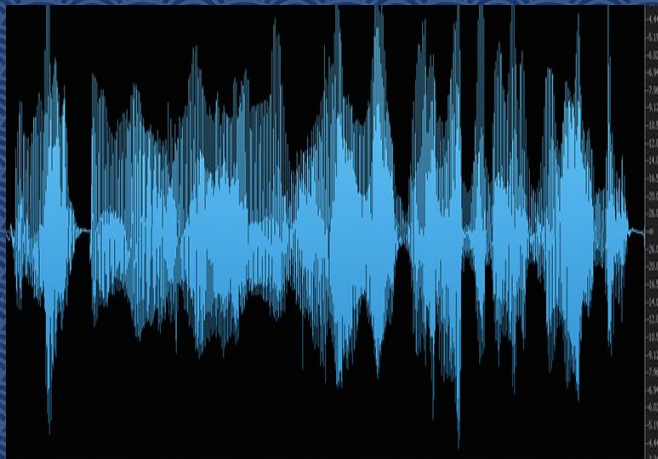




## The Data

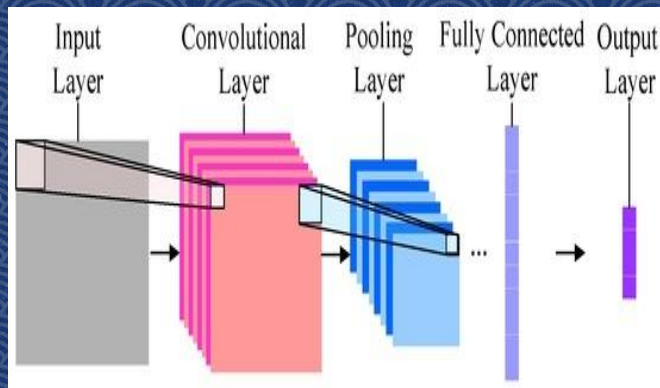
My MP3 collection:  
about 150 audio files, one song each,  
hand-labelled with 1, 2, or 3 tempos each





## The Methods

- Mel Spectrogram computed by LibROSA
- Convolutional Neural Network





## CHALLENGES

- ◎ Not enough data
- ◎ What loss function to use for multiple tempos?
- ◎ Some songs change tempo
- ◎ How to represent data to generate features

## **DEALING WITH TEMPO CHANGES**

- ◎ Cut off beginning and end of each song (since many have tempo changes there).
- ◎ Drop songs with tempo changes in the middle.



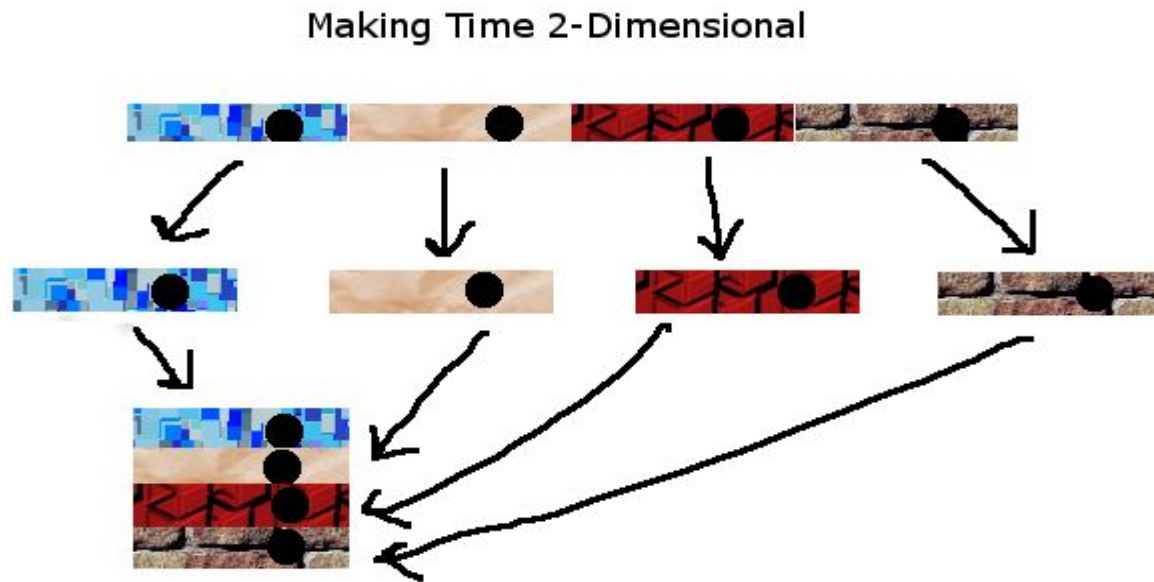
## MAKING TRAINING DATA

- ◎ Create training examples by taking multiple random samples from each song. few->many
- ◎ Use peaks of LibROSA tempogram() function as “candidate tempos”
- ◎ Each sample gets assigned a candidate tempo (randomly chosen from those peaks)
- ◎ Simple binary loss: Is candidate tempo close to one of the hand-labeled tempos?



## HOW TO REPRESENT THE DATA

Divide clip  
into one  
section for  
each beat at  
candidate  
tempo, and  
stack the  
sections





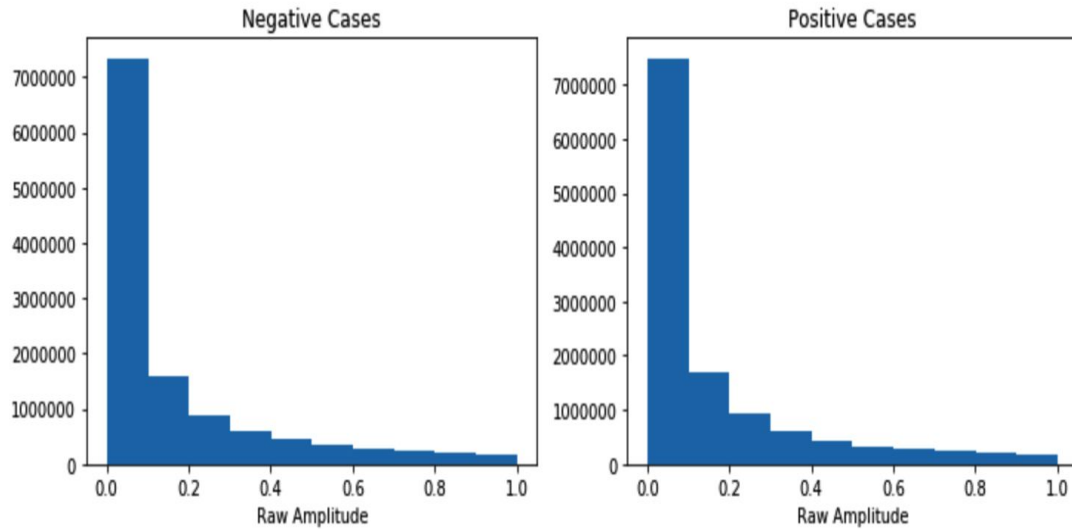
**THEN USE THE  
STACKED DATA AS  
INPUT TO A  
2-DIMENSIONAL  
CONVOLUTIONAL  
NEURAL NETWORK**



The background of the slide is a solid blue color with a repeating white wave pattern, resembling a traditional Japanese 'nami' pattern. The pattern consists of overlapping, stylized waves that create a sense of movement and depth. The waves are arranged in a grid-like fashion, filling the entire background.

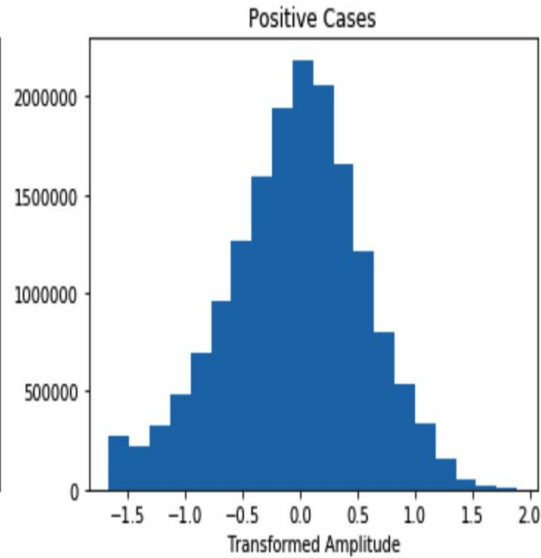
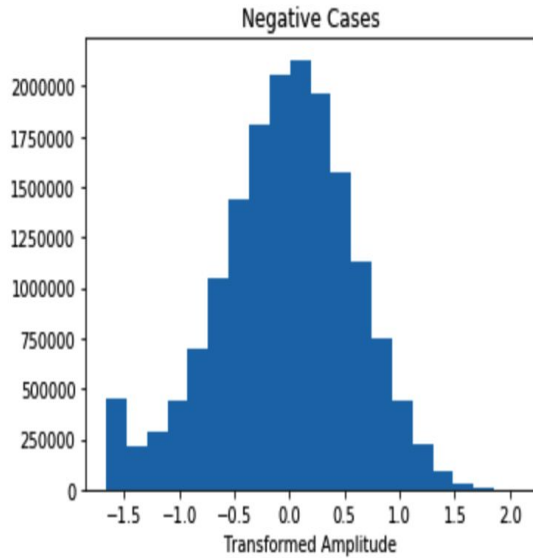
# How the data look

# Distribution of Raw Data



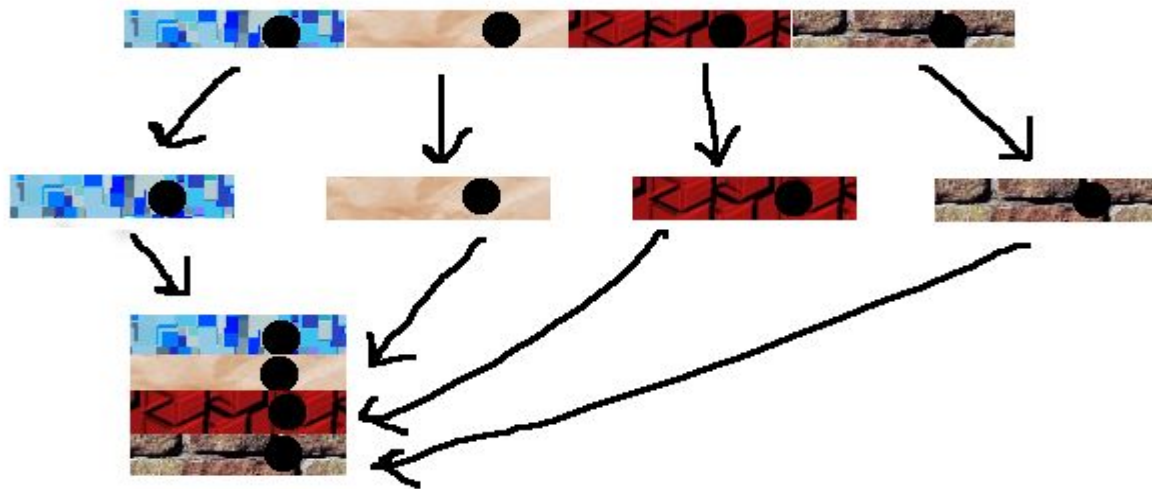


# Transformed Data

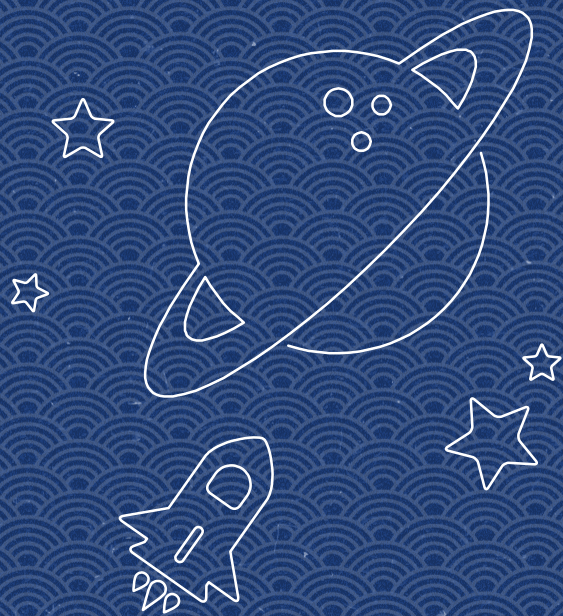


**FOR  
POSITIVE  
(CORRECT)  
CASES,  
EACH BLOCK  
OF DATA  
SHOULD  
HAVE A  
CONSISTENT  
VERTICAL  
PATTERN, AS  
BEATS GET  
REPEATED**

Making Time 2-Dimensional



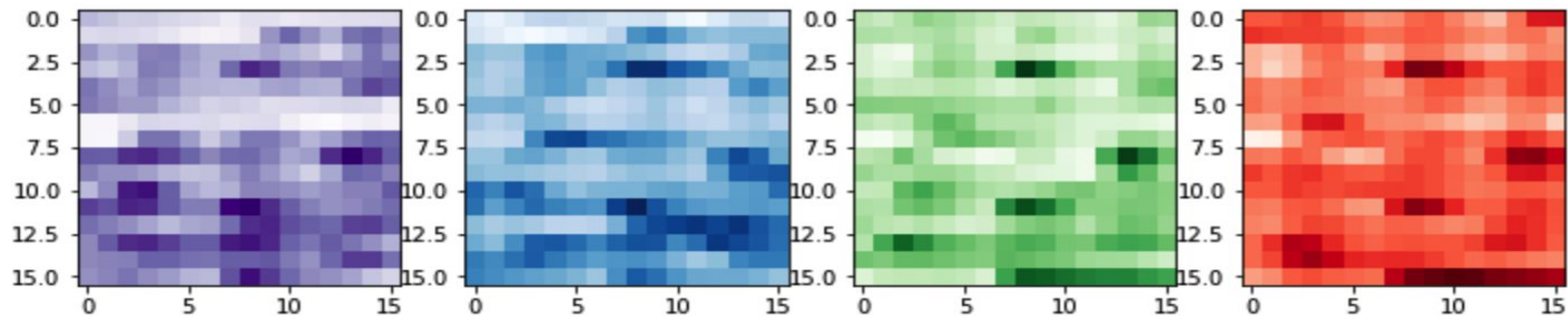




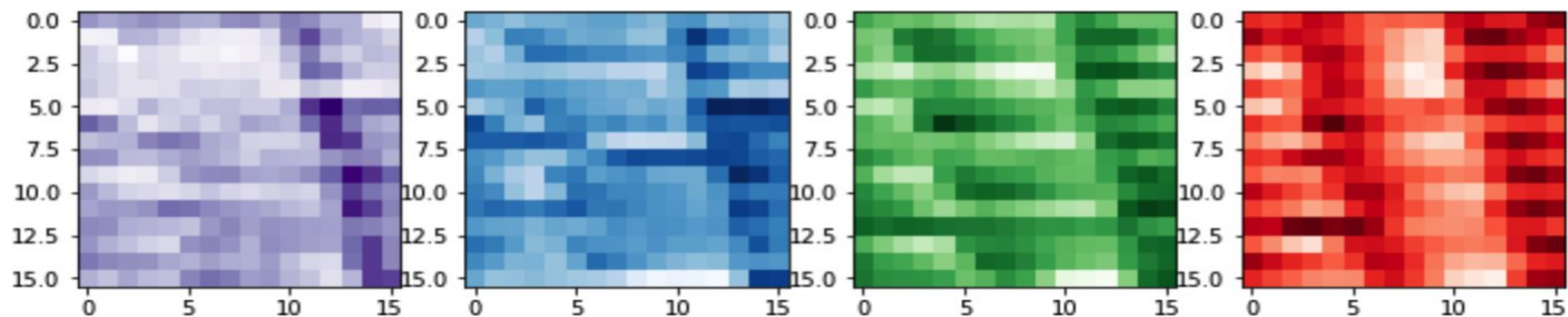
# What do we actually see?



Negative case, preprocessed:



Positive case, preprocessed:



Here the spectrogram is divided into 4 sections, with the lowest pitches in purple and the highest in red. These cases are fairly typical. In the negative case, as expected, there are no clear patterns. In the positive case, there are almost-vertical patterns, but they are downward-sloping.



**STATISTICS TO  
REPRESENT  
VERTICAL  
PATTERNS**

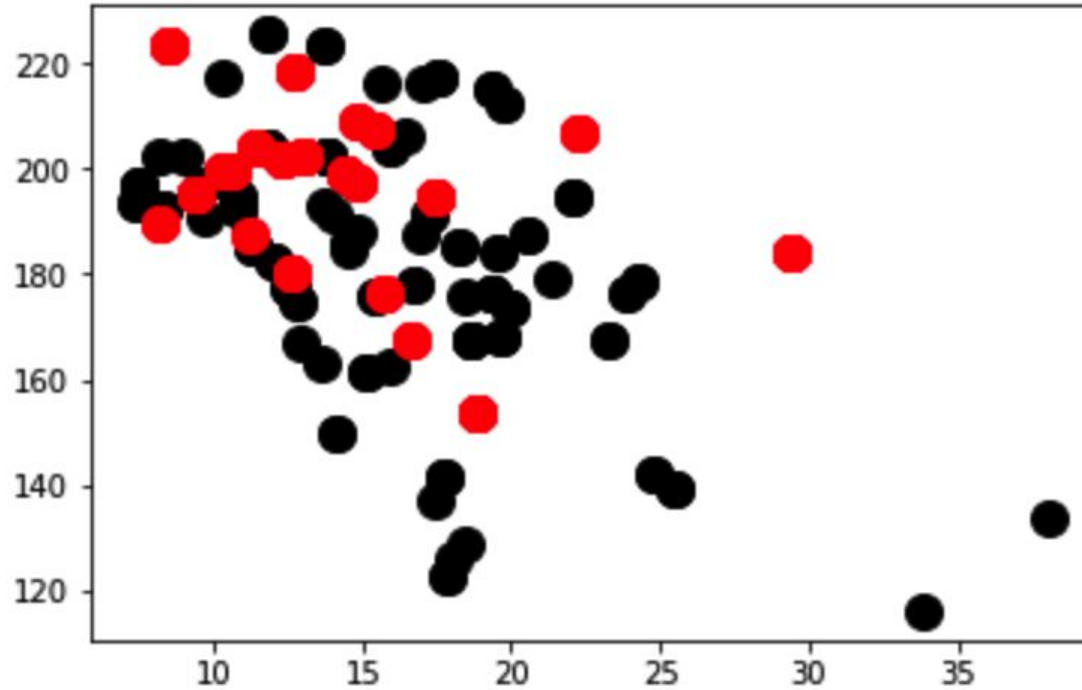
Average variance along  
vertical dimension

Should be low for  
positive cases

Average correlation  
between successive  
rows

Should be high for  
positive cases

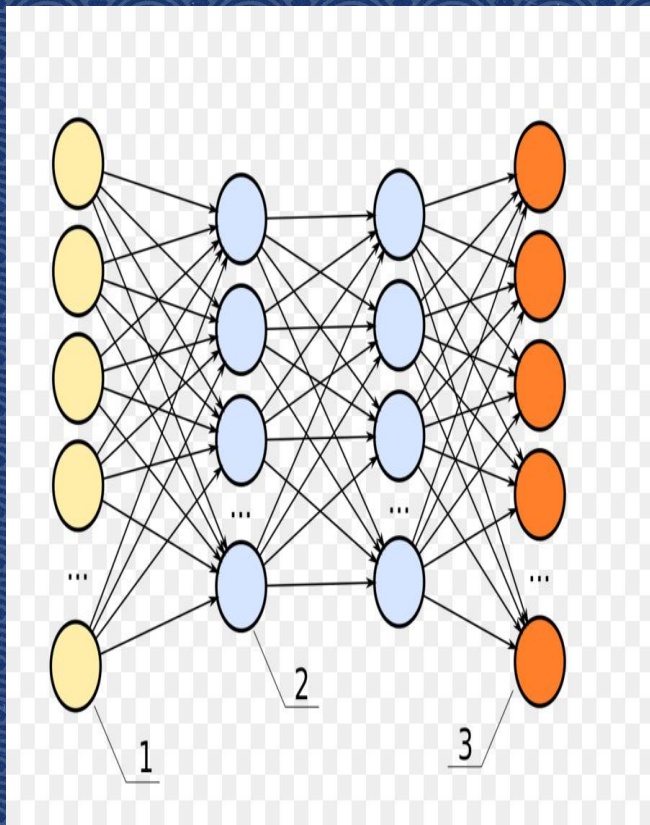
**average  
correlation**



**average variance**

The positive (red) cases, as expected, tend to have high average correlation and low average variance. The negative (black) cases are more broadly distributed, but they are more common elsewhere.





# The Neural Network



# STRUCTURE

Output: Sigmoid  
Dense(20)  
Dropout(.8)  
1x4 Convolution, 3 filters  
Batch Norm  
4x6 Convolution, 25 filters  
Dropout(.3)  
1x4 Convolution, 12 filters  
Batch Norm  
1x1 Convolution, 32 filters  
Dropout(.1)  
1x1 Convolution, 64 filters  
Input: 16x16, 128 channels



## **SALIENT FEATURES**

- ◎ Narrow convolutions (1x4 and 4x6) instead of square convolutions
- ◎ Alternating batch normalization and dropout between successive convolutions
- ◎ Very high dropout before top layer

## RESULTS

- ◎ Validation accuracy 0.91 (after 60 epochs using final version of model)
- ◎ This is quite good, considering that many cases are ambiguous



## EXAMPLE RESULTS

- **53rdAnd3rdMP 45**  
**Predicted: 0.00090704486   Actual: False**
- **AHardRainsAGonnaFall 56**  
**Predicted: 0.021726133   Actual: False**
- **AHardRainsAGonnaFall 62**  
**Predicted: 0.017301498   Actual: True**
- **InTheHillsOfShiloh 64**  
**Predicted: 0.13423629   Actual: False**
- **CaliforniaDreamin2MP 112**  
**Predicted: 0.93683857   Actual: True**



## NEXT STEPS

So, so many: for example

- ◎ Look at results per song. Subjectively, how does model do at getting correct paces? Are typical failures legitimately ambiguous cases, or is the model really missing something?
- ◎ Get data from other human labelers.
- ◎
- ◎ Get more data for underrepresented genres (and time signatures). And a greater variety of artists.

....and so on

See <https://github.com/andyharless/paces/blob/master/README.md>





**THANKS!**

[github.com/andyharless](https://github.com/andyharless)