Music Sentiment Classification

Sihan, Yanxin, Qianyu

keras.optimizers.Adam

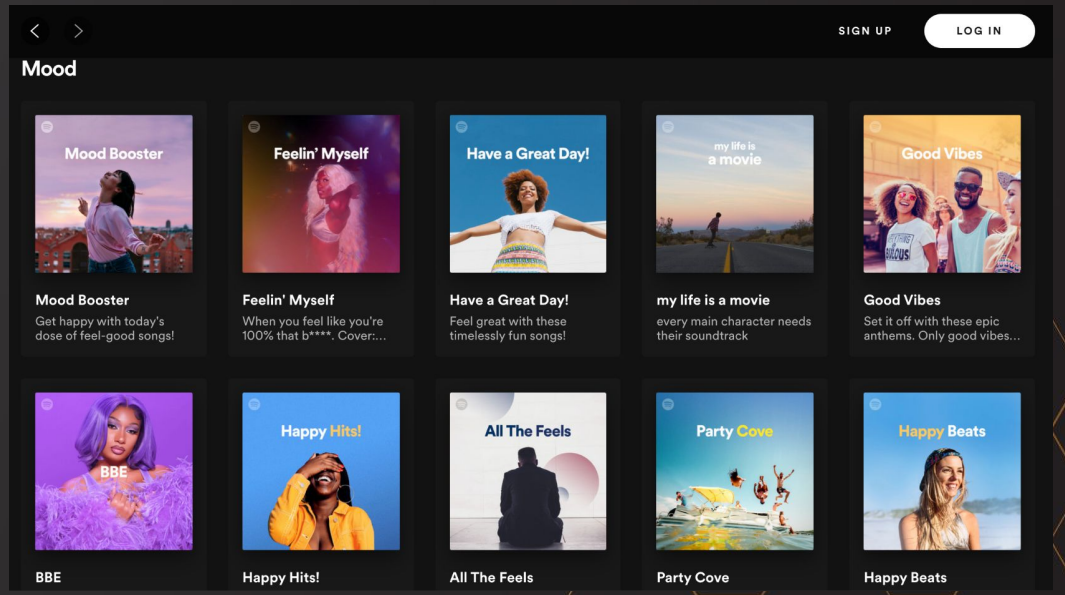
Presentation Category

1. Project Question Introduction
2. Data Crawling & Preprocessing
3. Feature vectors & General Model(SVM, Tree, etc.) Comparison
4. Audio feature and Convolutional Neural Network Analysis

Dataset -- overview

- **source: Spotify**
- **7195 songs (5510 with .wav)**
  - Calm: 2951 (2742 with .wav)
  - sad: 439 (232 with .wav)
  - energetic: 3554 (2364 with .wav)
  - happy: 251 (172 with .wav)
- **16 features**
  - provided by Spotify API
- **Label**
  - self-labelled
  - binary: positive & negative
  - multi-class

Spotify for Developers



Dataset -- preprocessing

- drop duplicated songs
- extract songs with preview_url (with audio)

```
calm.csv
calm_no_duplicates.csv
calm_no_duplicates_no_nan.csv
calm_vectors/
```

Dataset -- features

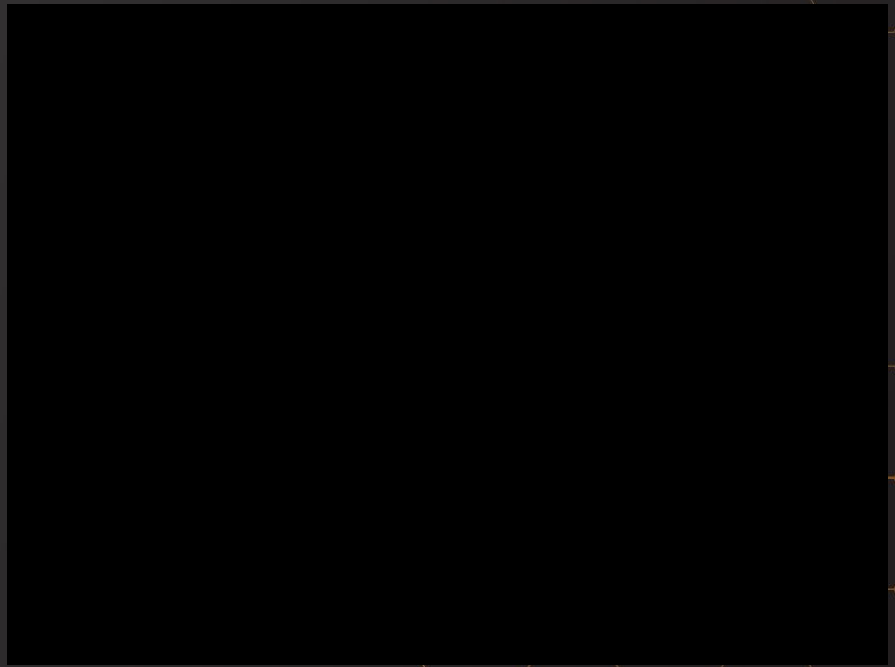
- object metadata
  - 16 features from Spotify API

id	0LweQRsfJ3pRAJJFy6DrR1
danceability	0.776
energy	0.692
key	2
loudness	-3.834
mode	1
speechiness	0.0555
acousticness	0.0075
instrumentalness	0
liveness	0.0588
valence	0.543
tempo	95.972
duration_ms	198174
time_signature	4
artist_name	Jessi
track_name	What Type of X
preview_url	https://p.scdn.co/mp3-preview/917623a4638d1005...
binary_label	1
multiclass_label	4

Name: 1, dtype: object

Dataset -- features

- **30 second preview**
  - **retrieve .wav**
  - **.wav -> vector**
    - which spectrogram
    - duration, start time
    - sr = 44100 ??
    - how to save numpy matrices?



Traditional models :)

SVM, Random Forest, etc.

Final Result

	precision	recall	f1-score	support
Sad	0.43	0.48	0.45	88
Healing	0.95	0.97	0.96	1926
joyful	0.90	0.86	0.88	816

first vs. last

	precision	recall	f1-score	support
Sad	0.43	0.48	0.45	88
Healing	0.95	0.97	0.96	1926
joyful	0.90	0.86	0.88	816

Final result: classifier = stacking (SVM, Decision Tree, Random Forest)

	precision	recall	f1-score	support
sad	1.00	0.06	0.11	53
calm	0.93	1.00	0.96	1926
happy	1.00	0.02	0.04	50
energetic	0.89	0.85	0.87	766

First model: SVM

Model Improvement

Improve the model

Better data

- **oversampling**
- **add more data**

Better model

- grid search for hyperparameter
- try different models
- stack models

Bther

- change classification criterion

Better Data

Oversampling helps: library SMOTE

More data helps！ ！ ！

before

	precision	recall	f1-score	support
sad	0.17	0.62	0.27	53
calm	0.94	0.95	0.94	1926
happy	0.15	0.58	0.23	50
energetic	0.94	0.58	0.72	766

after

	precision	recall	f1-score	support
sad	1.00	0.06	0.11	53
calm	0.93	1.00	0.96	1926
happy	1.00	0.02	0.04	50
energetic	0.89	0.85	0.87	766

Better Model

Grid Search for good hyperparameter

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, f1_score
param_grid = {'C': [1, 10, 100, 1000],
              'gamma': [1, 0.1, 0.01,0.001],
              'kernel': ['rbf']}
svc_clf = SVC()
scorer = make_scorer(f1_score, average = 'weighted')
grid = GridSearchCV(SVC(), param_grid, scoring=scorer, refit = True)
```

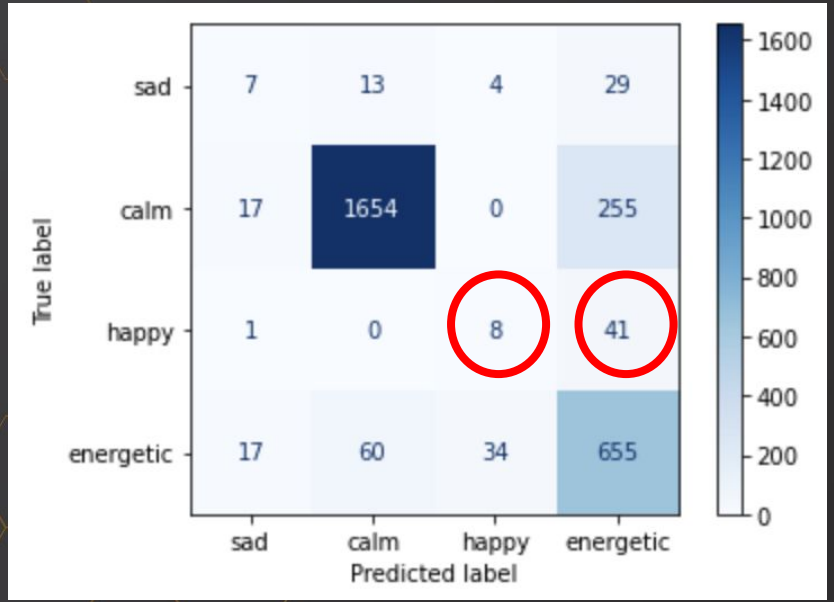
Stacking

```
models = [('dtree', dtree),
          ('rforest', rforest),
          ('svm', svm)]
stacking = StackingClassifier(
```

Try different models:

```
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

Other



Change classification criterion

Future Work

- More data : add data for the minority class
- Data of higher quality
- More features: extract features from the audio

Neural Network

A long journey of model optimization

Feature vector input:

- One-layer NN

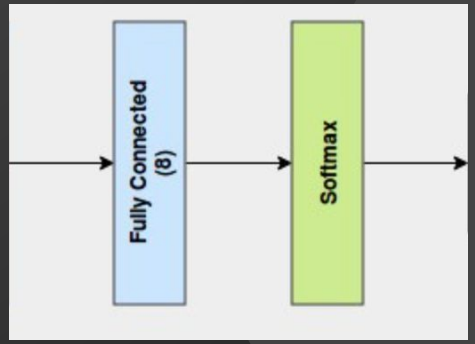
Raw audio input:

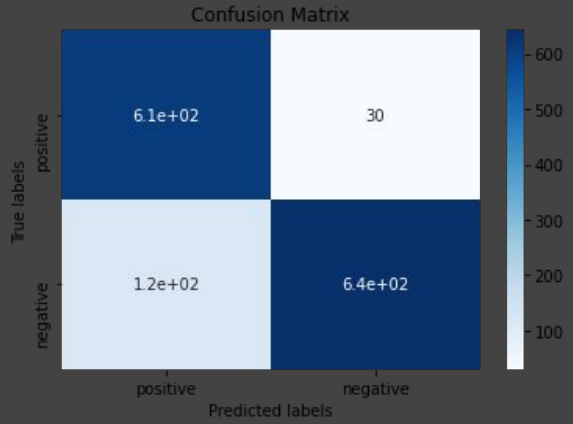
- Conv + FC
- Conv + LSTM + DNN
- Pure LSTM

Simple NN

One layer with 8 nodes

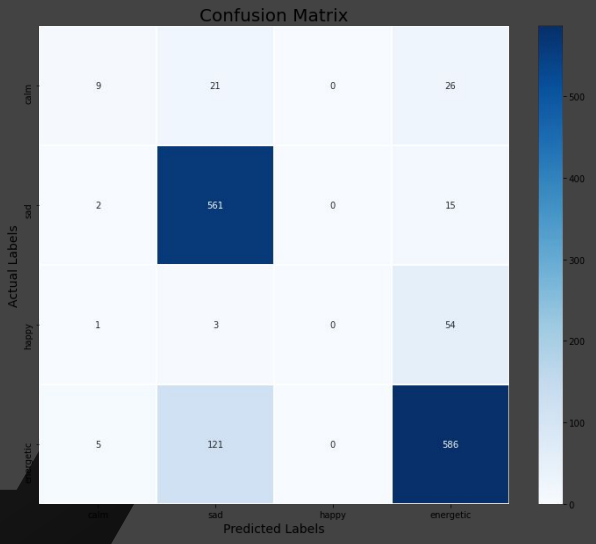
Accuracy Score: 0.894





However, what can we do if we are given a new song?

```
Model: "sequential_46"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_92 (Dense)             (None, 8)                 112
_________________________________________________________________
dense_93 (Dense)             (None, 4)                 36
=================================================================
Total params: 148
Trainable params: 148
Non-trainable params: 0
_________________________________________________________________
```



Conv + FC

3,552,513

Parameters

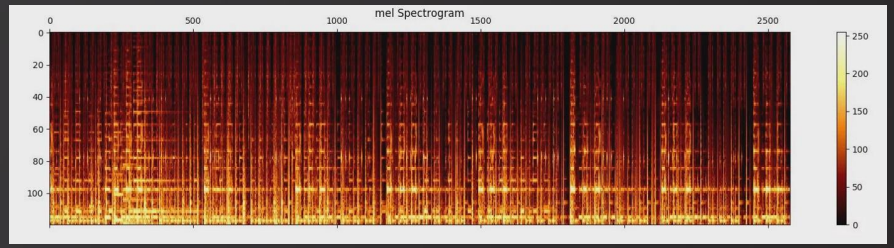
30+ mins

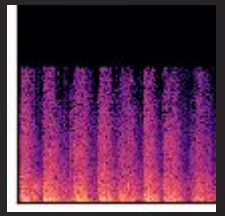
Training time

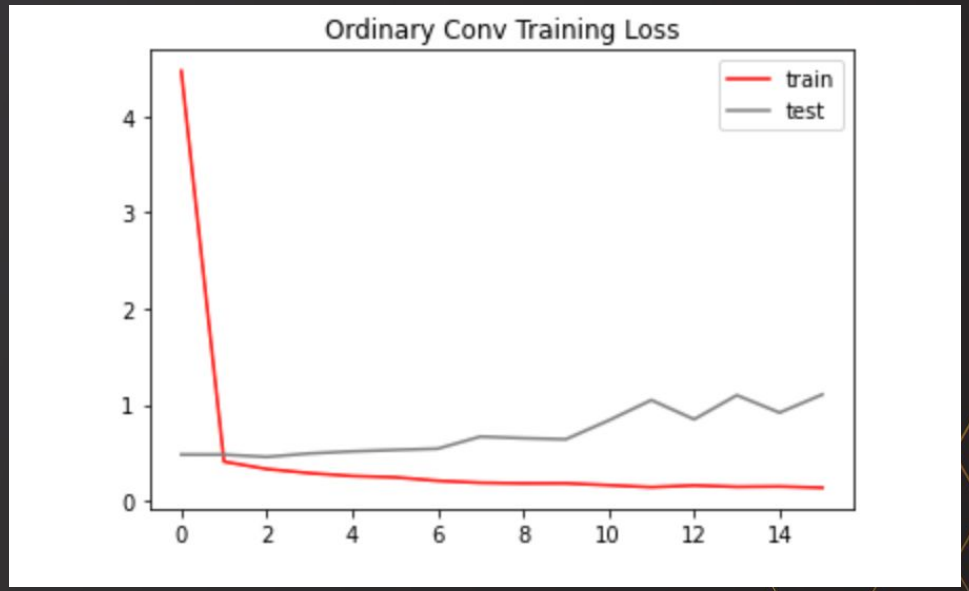
84.39%

Accuracy

Can we do better??







Batch-Normalization & Drop Out Rate

Early stopping

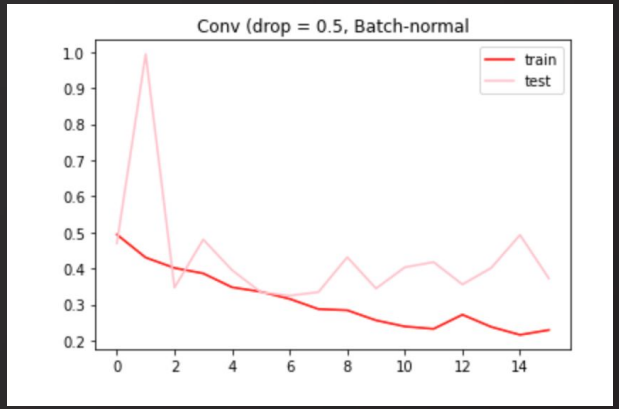
Callbacks=[es]

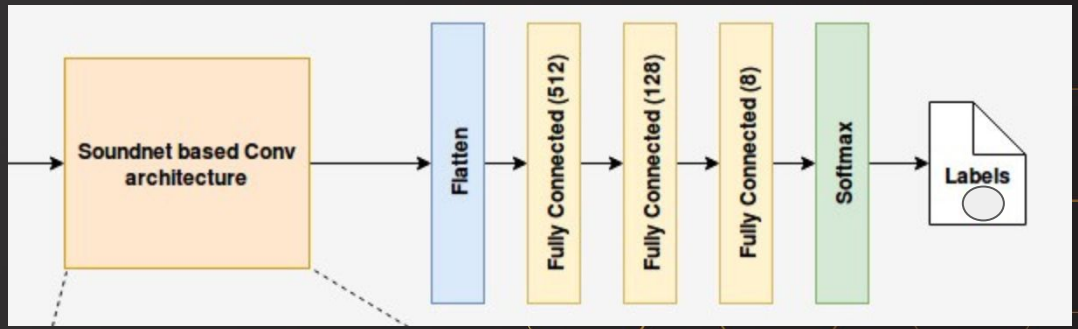
32

Batch size

89.11%

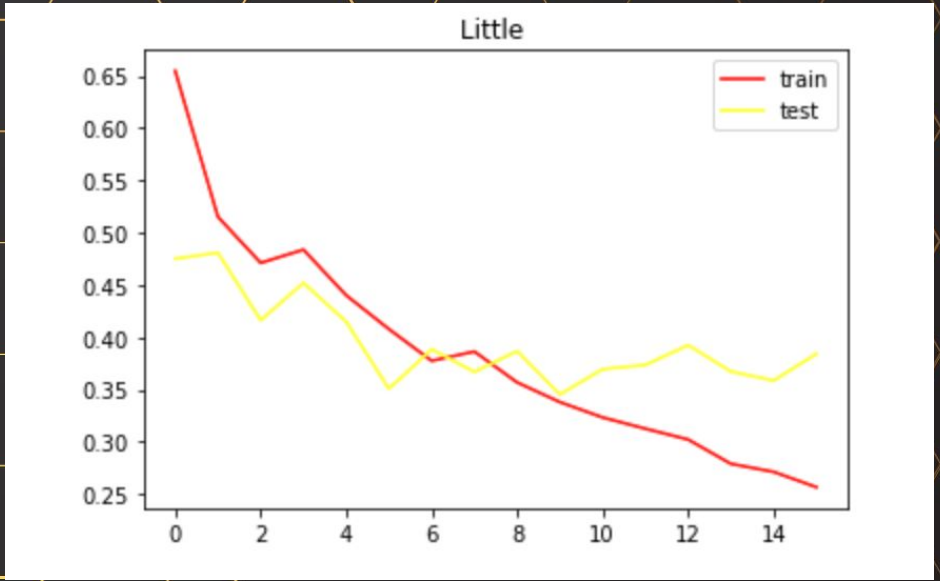
Accuracy





Can we do better?

We changed the filter size from square into rectangular…



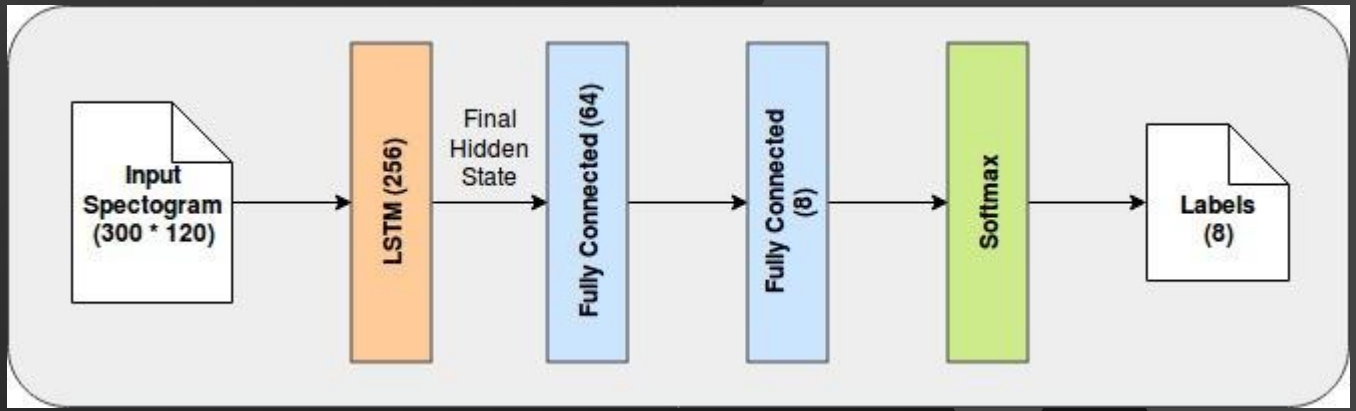
With only 446193 parameters!
1/10 of original numbers!

Accuracy: 89.11%

Pure LSTM

Low training rate

and bad behavior!



Conv + LSTM + DNN

Activation

LeakyReLU(alpha=0.01

Filter

kernel_size=(1, 3)

Time + LSTM

Bidirectional LSTM
(128, dropout=0.25,
return_sequences=True)

89.47%

Highest Training Accuracy !

10+ mins/Epoch ...
Super slow ... Orz ...