Music Emotion Recognition (MER)




Sihan, Yanxin, Qianyu

Agenda

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

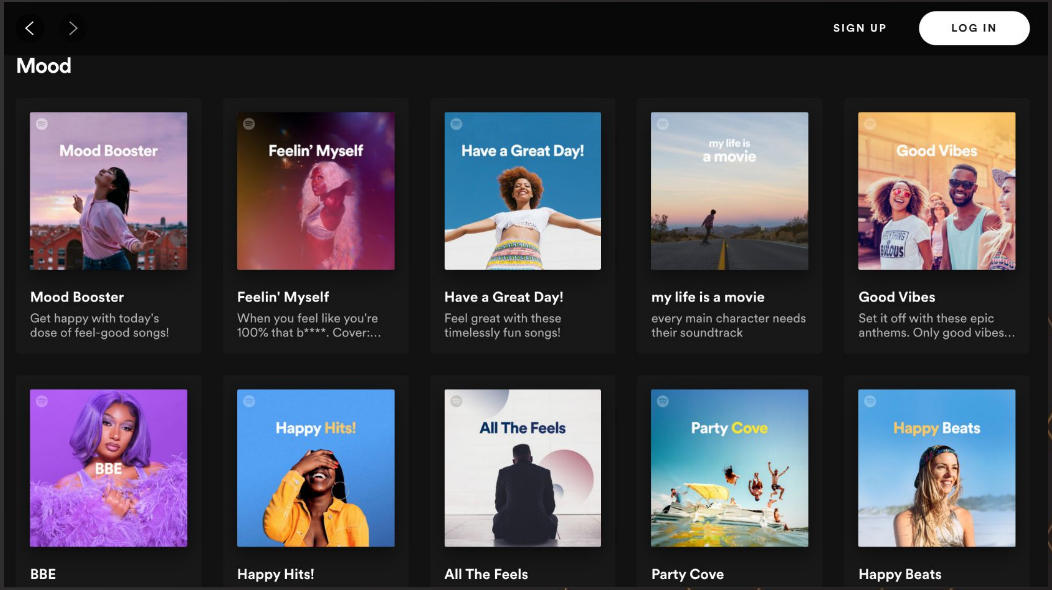
MER

- multi-label tagging task
- acoustic features <-> emotion tags
- application: music recommendation etc.
- models: SVM, random forests, CNN
  (Liu, Xin, et al.)

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/
```

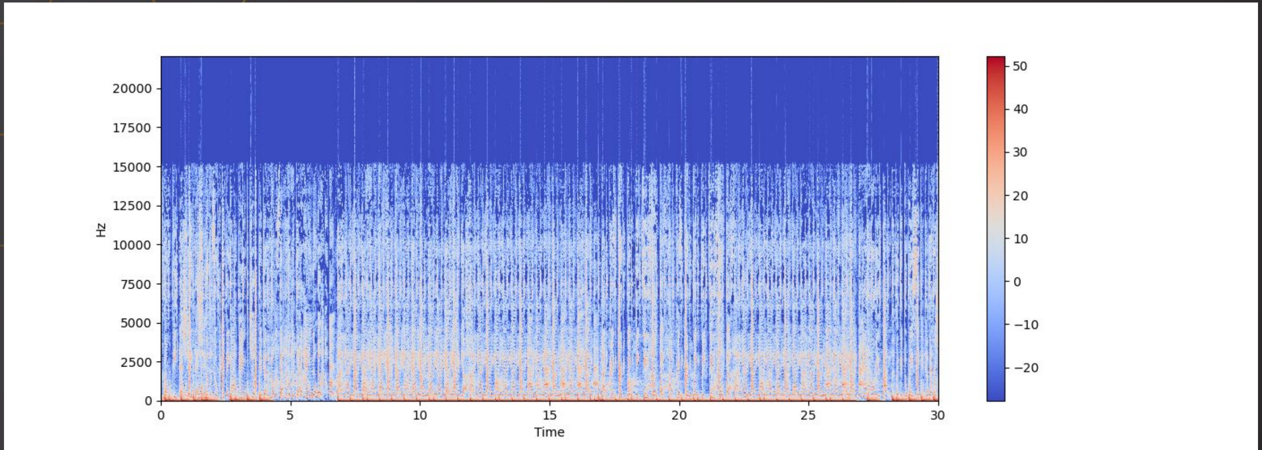
Dataset -- features

- object metadata
    - 16 features from Spotify API

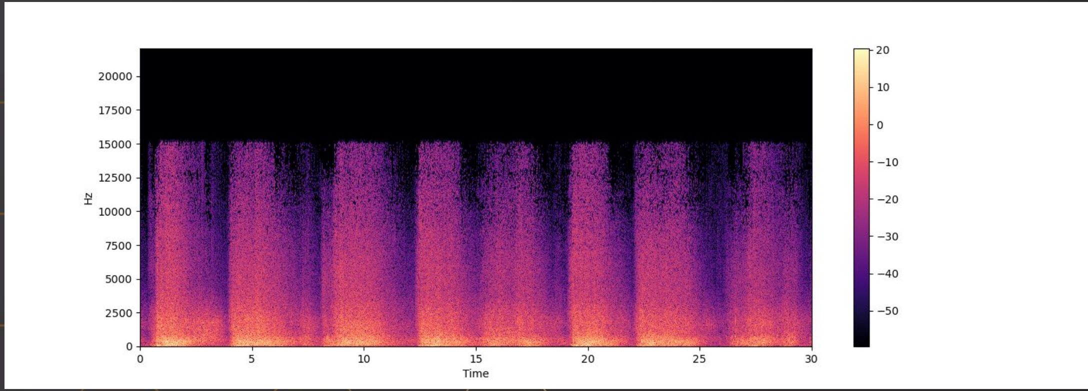
id	0LweQRsfJ3pRAJJFy6DrR1
danceability	0.776
energy	0.692
key	2
loudness	-3.834
mode	1
speechiness	0.0555
acousticness	0.0075
instrumentalness	0
liveness	0.0588
valence	0.543
tempo	95.972
duration_ms	198174
time_signature	4
artist_name	Jessi
track_name	What Type of X
preview_url	https://p.scdn.co/mp3-preview/917623a4638d1005...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

Name: 1, dtype: object

Dataset-features



calm



energetic

Spectrogram

- 2D discrete Fourier transform of audio
- visualize harmonic structure in audio

Dataset -- features

- 30 second preview
  - retrieve .wav
  - .wav -> matrix 128 * 216
  - intensities of spaced frequency ranges over discrete slices in time

```
x, sr = librosa.load(file, offset = 10,duration = 5)
ps = librosa.feature.melspectrogram(x, sr = sr)
```

```
data = np.array(data)
print(data.shape)
np.save(name + '.npy', data)
```

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

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

classifier = SVM

Model Improvement

--------------A long journey of model optimization--------------

Improve the model

Better data

- oversampling
- add more data

Better model

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

Other

- change classification criterion

Main Difficulty: class imbalance

calm	9629	68.91%
energetic	3829	27.40%
sad	266	1.90%
happy	250	1.79%

Better Data

Oversampling: library SMOTE

before

	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

classifier = SVM default parameters

after

	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

classifier = SVM default parameters + oversampling

More data helps！！！

	precision	recall	f1-score	support
sad	0.33	0.72	0.45	88
calm	0.94	0.96	0.95	1926
happy	0.16	0.58	0.25	50
energetic	0.94	0.62	0.75	766

SVM default parameters + oversampling + add data

Better Model

Hyperparameter tuning: Gridsearch

```
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
```

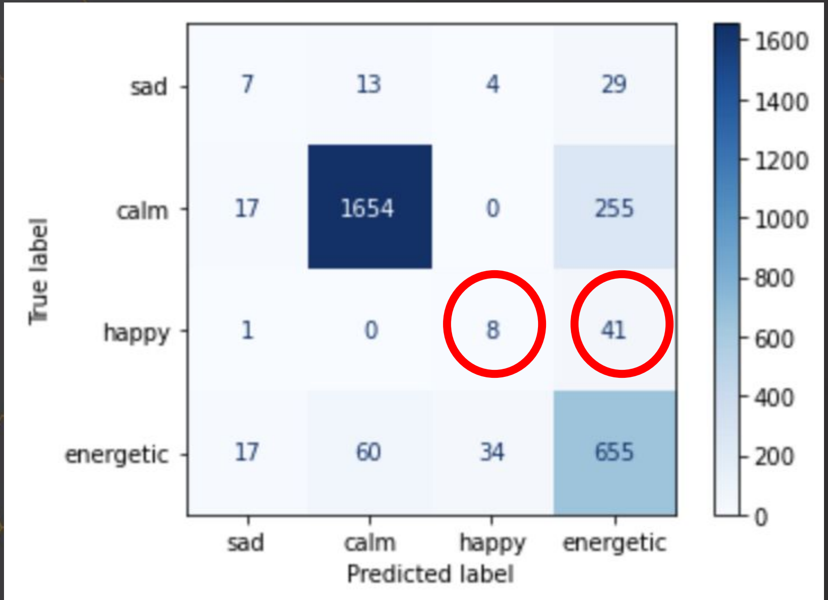
Stacking

```
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

Feature vector input:

- One-layer NN

Raw audio input:

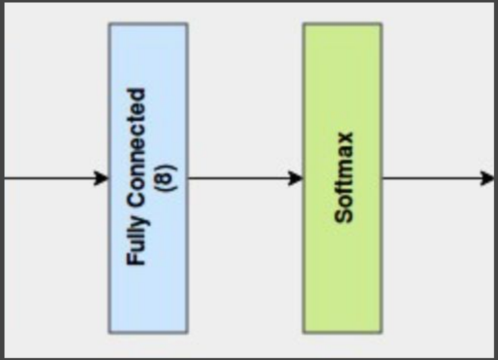
- Conv + FC
- Conv + LSTM + DNN

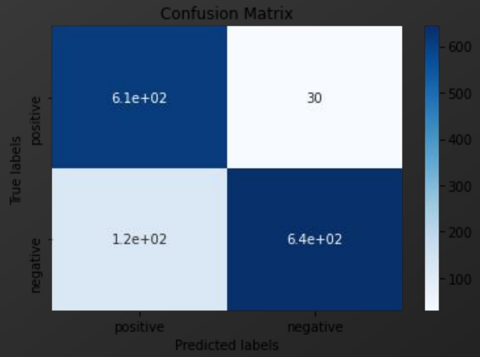
Simple NN

```
In [351]: #Normalize the features
          X = MinMaxScaler().fit_transform(X)
```

One layer with 8 nodes

Accuracy Score: 89.4%






```
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
_________________________________________________________________
```

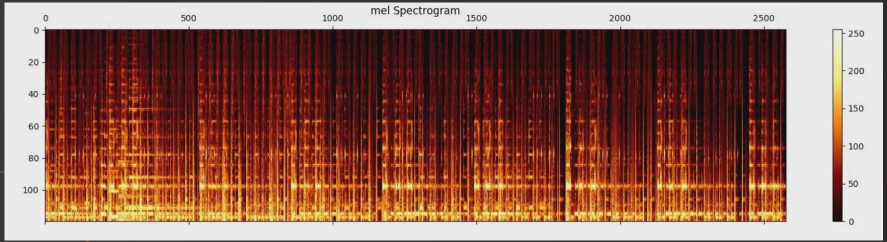
Baseline model:

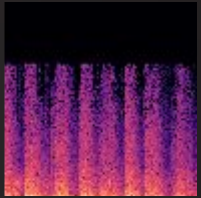
Three-layer CNN

Accuracy: 53.9%

Conv + FC

Conv + LSTM + DNN





(128, 216, 1)

Conv + FC

3,552,513

Parameters

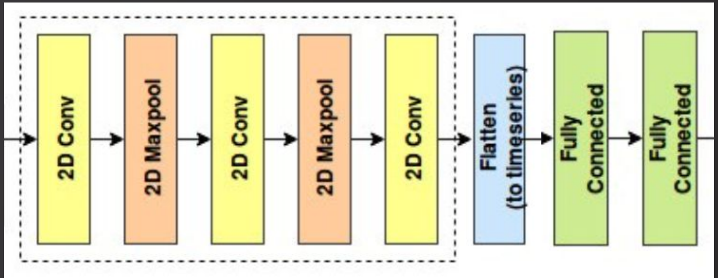
30+ mins

Training time

81%

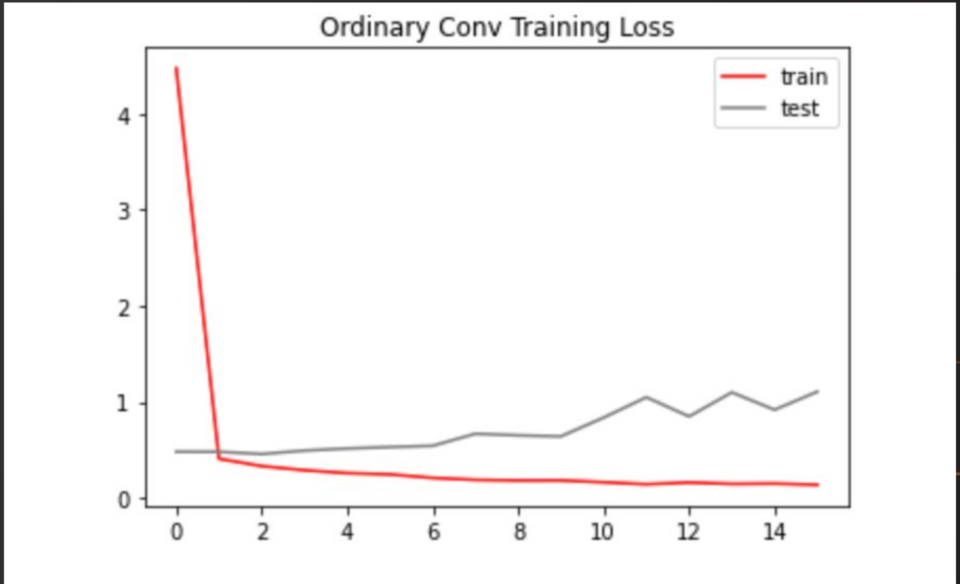
Best Val Accuracy

Can we do better??



Adam Optimizer

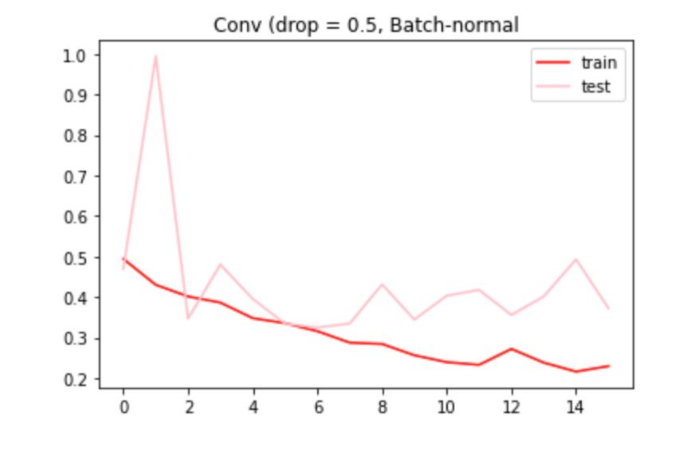
Cross-entropy Loss



Model Optimization

Callbacks=[es]
Early Stopping

Batch size
32



Batch-normalization
After Activation
Before drop out

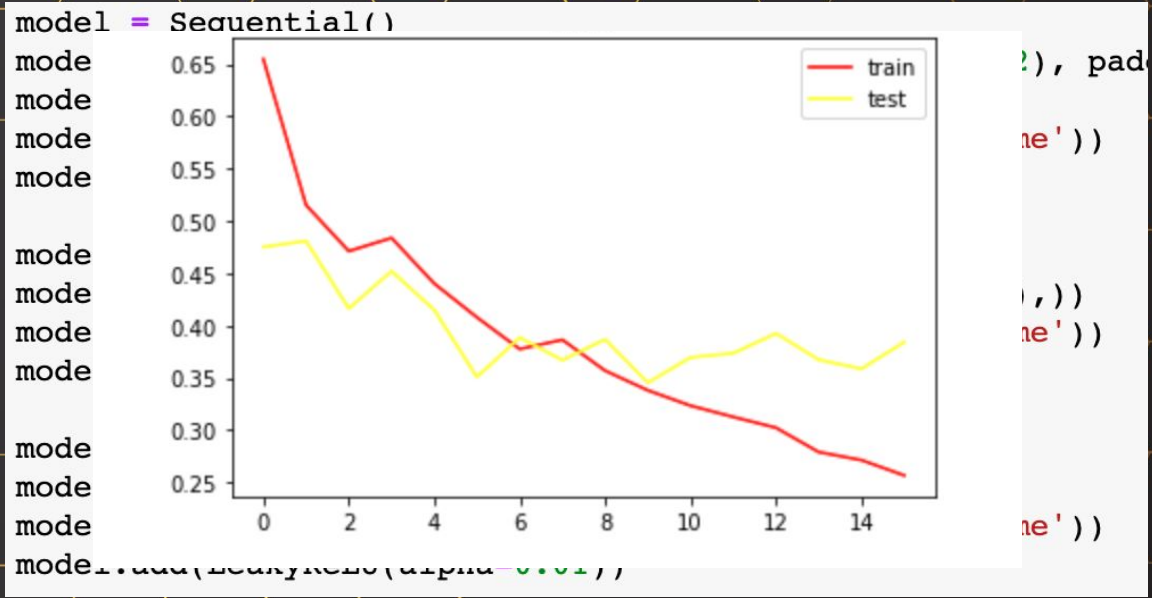
Drop Out rate
0.3 --> 0.5

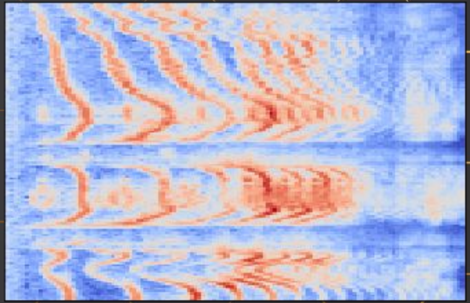
Accuracy
89.47%

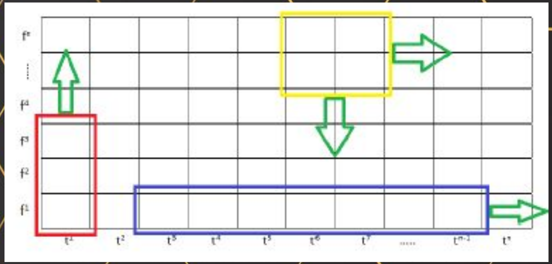
```
es = EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=3)
```

Can we do better?

We changed the filter size
from square
into rectangular...







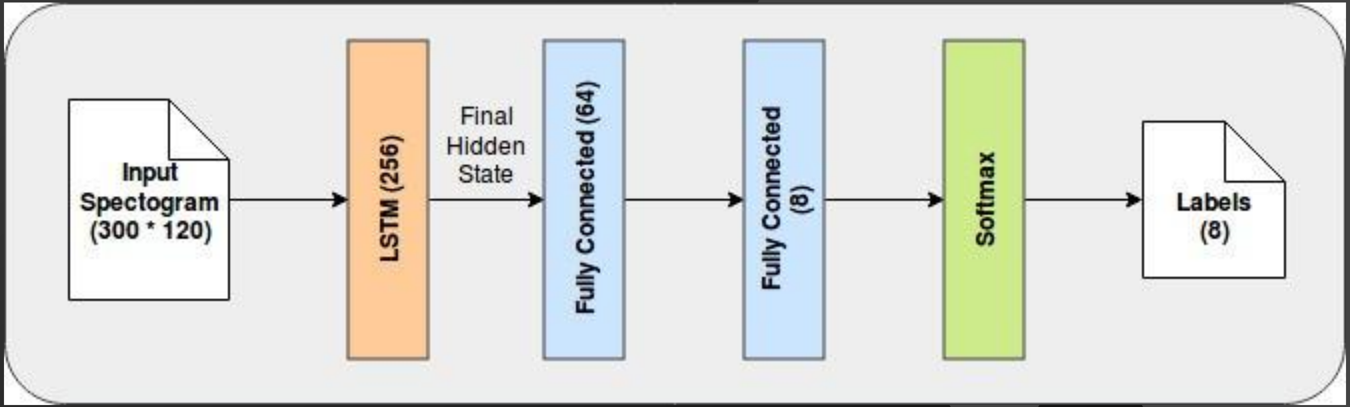
With only 446193 parameters!
1/10 of original numbers!

Accuracy: 89.11%

Pure LSTM

Low training rate

and bad behavior!



Conv + LSTM + DNN

Filter

kernel_size=(1, 3)

89.11%

High Training Accuracy !

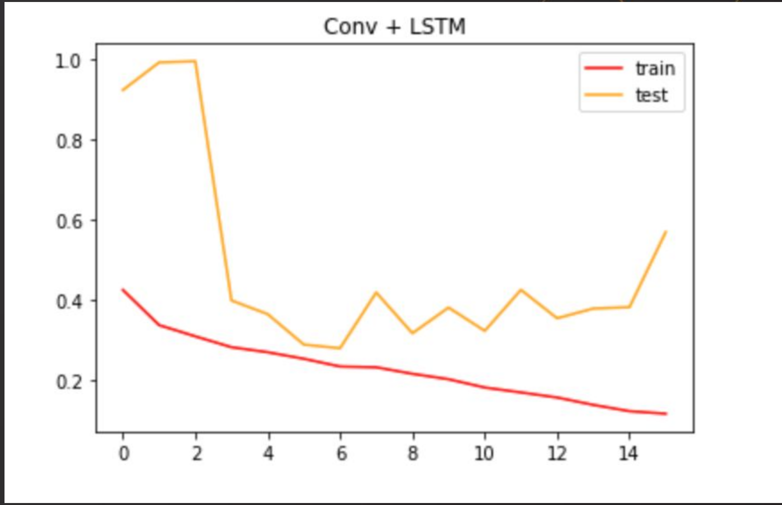
LeakyReLU(alpha=0.01)

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

Time + LSTM

Bidirectional LSTM

(128, dropout=0.25, return_sequences=True)



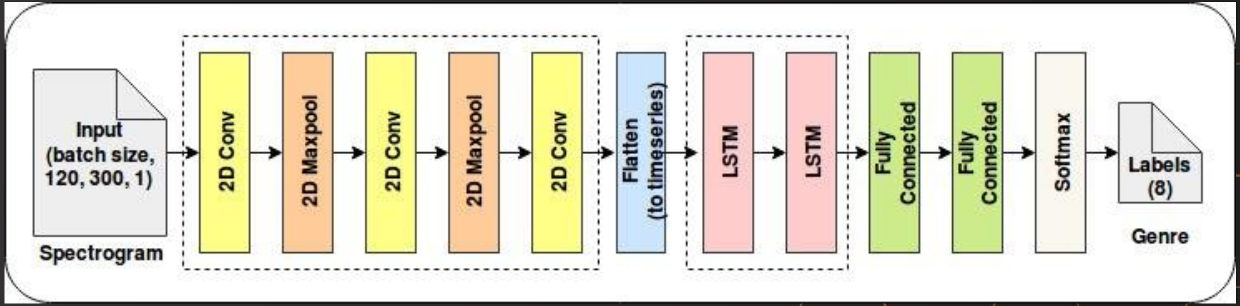


Image from Rajat Hebbar

Model Limitation & Future Improvement

1. **Data limitation**
   a. Subjective label / 5s Clips / quantity
2. **Audio Data Augmentation**
   a. Time Stretching / Pitch Shifting
   b. Dynamic Range Compression
   c. Background Noise
3. **Multi-class Emotion Recognition**
4. **New data type (SoundNet)**

Thank you!