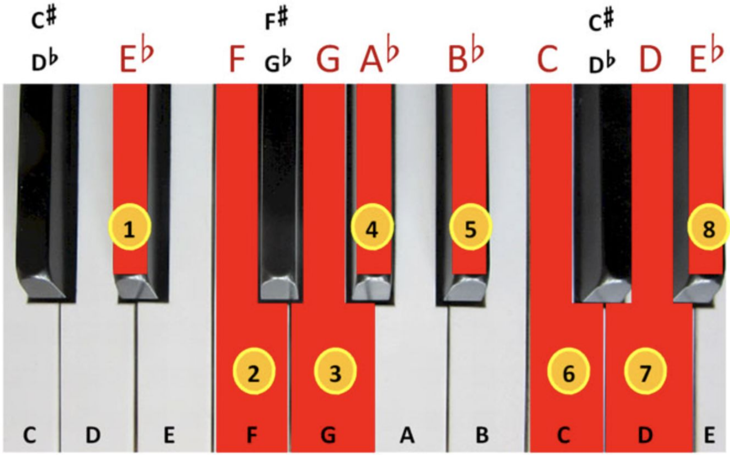
Key Detection on Pop Music

Boning Wang
Jiajun Wu
Yichuan Wang
Jiajun Chen

What is the key of music

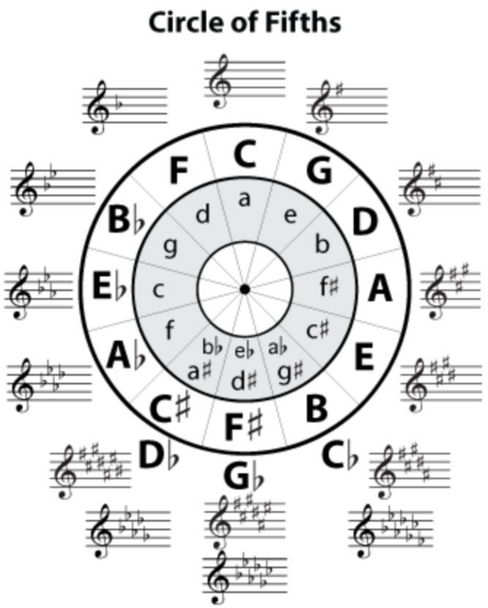
- The key is a form of music organization structure[1].
- In an octave, the notes of music can be divided into 12 semitones.
- Notes in a major scale follows the structure W-W-H-W-W-W-H
- Natural minor W-H-W-W-H-W-W
- Key refers to the note at the first



[2]

Relative Keys

- Major and minor keys are the most common in pop and classical music
- Each major key has a corresponding minor key that shares the same notes[3]
- To simplify problem, we regard all the keys of songs as major scale



[3]

1. Related projects

There have been extensive research and projects on key detection

- ➔ **Krumhansl-Schmuckler profile[1]**
  Based on judgement of music professional
- ➔ **K-nearest neighbor[4]**
  Compare dataset with templates
- ➔ **SVM[4]**
  Raise to high dimension to enable linear separation

Most of the current key detection output only one key

Problem:

Is it possible that multiple keys appear in a single song?

Of course! Change in key is called **modulation** which is a frequently used technique

2. Dataset

We manually created our dataset and labels

➔ **What**

411 songs in English, Mandarin Chinese, Cantonese, and absolute music

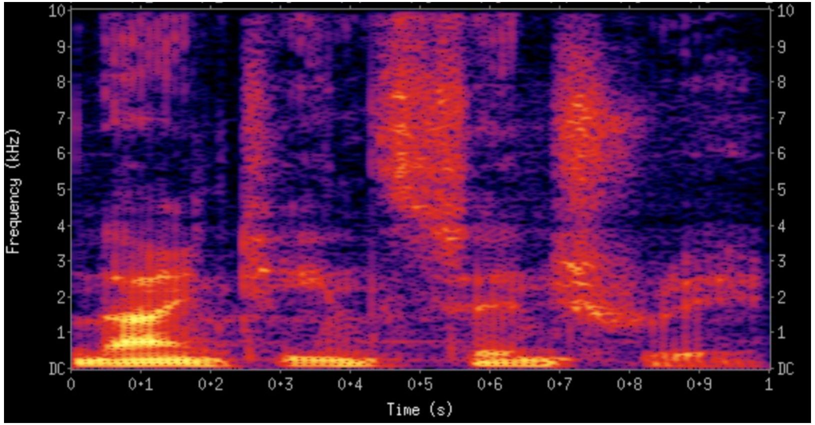
96 of them involves modulation

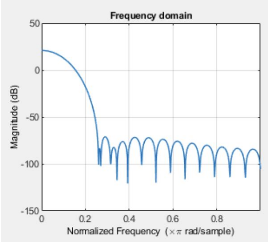
➔ **How**

In order to obtain specific features of keys and modulation, we divided each song to multiple clips according to beats so that clips from fast songs and slow songs contains similar amount of information.

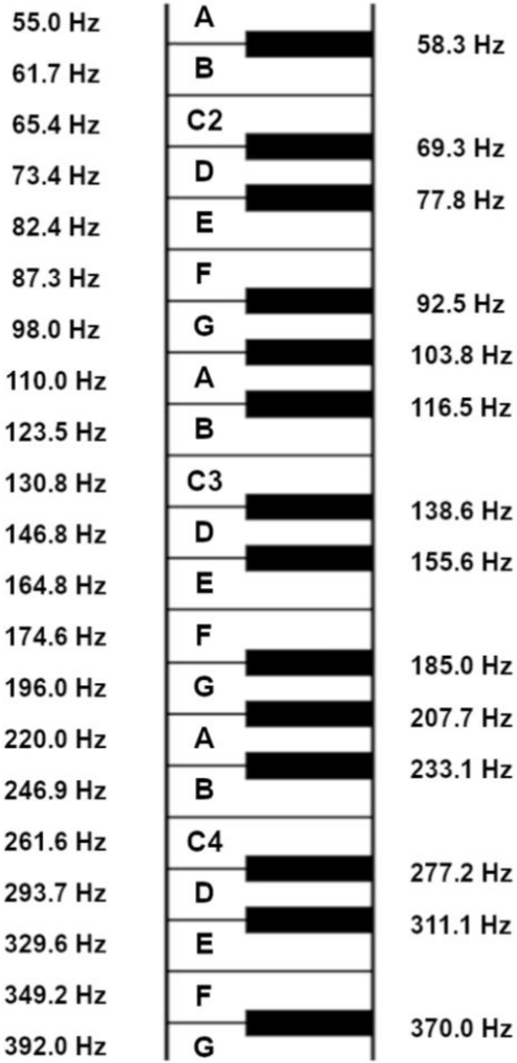
Short time Fourier transform (STFT)

- Analyze the waveform in both temporal and spectral domain
- Obtain existing notes in an instant frame
- 4096 samples window length, 3 times zero padding, Blackman Harris window, 512 samples hop





[5]



[6]

Problem:

Auditory perception of frequency is in logarithmic scale

12288 bins * 1k+ samples * 411 songs!

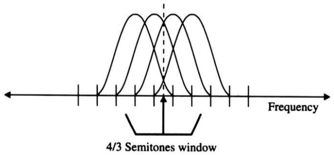
Harmonic Pitch Class Profile (HPCP)

- Since key information is octave-independent, we can reduce the dimension by HPCP[7]
- Convert frequency bins to notes
- Put all Cs regardless of octave to common bins, and so do C#, D, D#, ...
- Choose frequencies in the range [150 Hz, 4000 Hz]

Harmonic Pitch Class Profile (HPCP)

Spectrum → Find peaks above -70dB → Convert frequency to the distance to a reference frequency in semitones → Find the remainders to make the distance octave-independent → Calculate weights given a peak for all HPCP bins → Multiply weights to the peak amplitudes and find the sum → Normalize to range between 0 and 1 → HPCP

$$w(n, f_i) = \begin{cases} \cos^2\left(\frac{\pi}{2}\frac{d}{0.5 \cdot l}\right) & \text{if } |d| \le 0.5l \\ 0 & \text{if } |d| > 0.5l \end{cases}$$



[7]

3. Neural Network Model

We use **Transformer** as the model

➔ **What**

Transformer is basically a structure of encoder and decoder. For more details please refer to paper “Attention is all you need[8]”.

Pre-Processed Data

8077 Excel files (2-d data) with:

Row -> sequence(frames)

Column -> classes

Left is an example of data feature files

Batch size -> 128

data .shape->[128,1700-2200, 36]

label.shape->[128]

	0	1	2	3	4	5	6
0	0.033689	0.001303	0.039276	0.039738	0.000208	0.249280	0.468065
1	0.022338	0.018375	0.015515	0.053248	0.100832	0.575867	0.079538
2	0.007917	0	0.003066	0.036448	0.085027	0.325971	0.027779
3	0.013328	0.002999	0.002734	0.024093	0.035579	0.389987	0.023864
4	0.070231	0	0.000751	0.025165	0	0.556256	0.198688
5	0.181349	0.000068	0.004167	0.012642	0.001198	0.008551	0.762397
6	0.242881	0.312428	0.002741	0.035075	0	0.001544	1
7	0.045791	0.276284	0	0.007714	0	0.000928	0.356831
8	0.066947	0.288974	0.000316	0.019443	0	0.001608	0.185873
9	0.539211	0.000082	0.024481	0.022690	0	0.003655	0.493757
10	0.624937	0.002694	0.005866	0.024007	0	0.048249	0.558417
11	0.053505	0.003640	0.002762	0.032999	0	0.185626	0.407206
12	0.007281	0.008026	0.001313	0.014814	0.021013	0.200124	0.249095
13	0.0		000564	0.004975	0.029604	0.365294	0.016349
				0	0.000466	0.414776	0.015000
				0	0.000247	0.377155	1
				000591	0.005375	0.027548	0.586309
				002632	0.010955	0.003746	0.152378
				005552	0.003188	0.025781	0.066838
				010707	0.017909	0.048932	0.116863
				007666	0.005802	0.001535	0.205276
				012019	0.004728	0.041856	0.720257
				013123	0	0.042953	0.719687
				006753	0.038163	0.095069	0.043344
				015275	0.191490	0.003333	0.120419
				012923	0	0.154732	1
				0	0.003665	0.158104	1
27	0.075584	0.005724	0	0.000319	0.004627	0.007773	0.895073

How We Process

- build a dataset class
- Read data and label into X and y with torch.read_excel
- Instantiate
- Load into DataLoader

Data Loader

Each Clip has different sequence length.

We use *pad_sequence* from *torch.nn.utils.rnn* (zero padding used)

After padding, each batch have difference sequence length, which is the longest sequence in the batch

Split data into 80% train, 10% val, 10%test

```
Batch data shape: torch.Size([128, 1800, 36])
Batch labels shape: torch.Size([128])
output shape: torch.Size([128, 12])
Batch data shape: torch.Size([128, 1978, 36])
Batch labels shape: torch.Size([128])
output shape: torch.Size([128, 12])
Batch data shape: torch.Size([128, 2190, 36])
Batch labels shape: torch.Size([128])
output shape: torch.Size([128, 12])
Batch data shape: torch.Size([128, 1879, 36])
Batch labels shape: torch.Size([128])
output shape: torch.Size([128, 12])
Batch data shape: torch.Size([128, 1884, 36])
Batch labels shape: torch.Size([128])
out                         8, 12])
Bat                        e([128, 2200, 36])
Bat                        ze([128])
out                        8, 12])
Bat                        e([128, 2006, 36])
Bat                        ze([128])
out                        8, 12])
Bat                        e([128, 1884, 36])
Bat                        ze([128])
out                        8, 12])
Bat                        e([128, 1879, 36])
Bat                        ze([128])
out                        8, 12])
Bat                        e([128, 2027, 36])
Batch labels shape: torch.Size([128])
```

Tip

Stories become more credible when they use concrete details such as the specific complex moves Alberto learned through Translate and his 30 goals in 21 games performance stats.

Model Architecture

Input Embedding(36->512 d_model)

3 Encoder Layer

- 8 heads, 2048 dim_feedforward

Aggregation([128,1900,512]->[128,512]) 512 is dimension of model, d_model

Fully Connected(512->12) 12 classes

Story for illustration purposes only

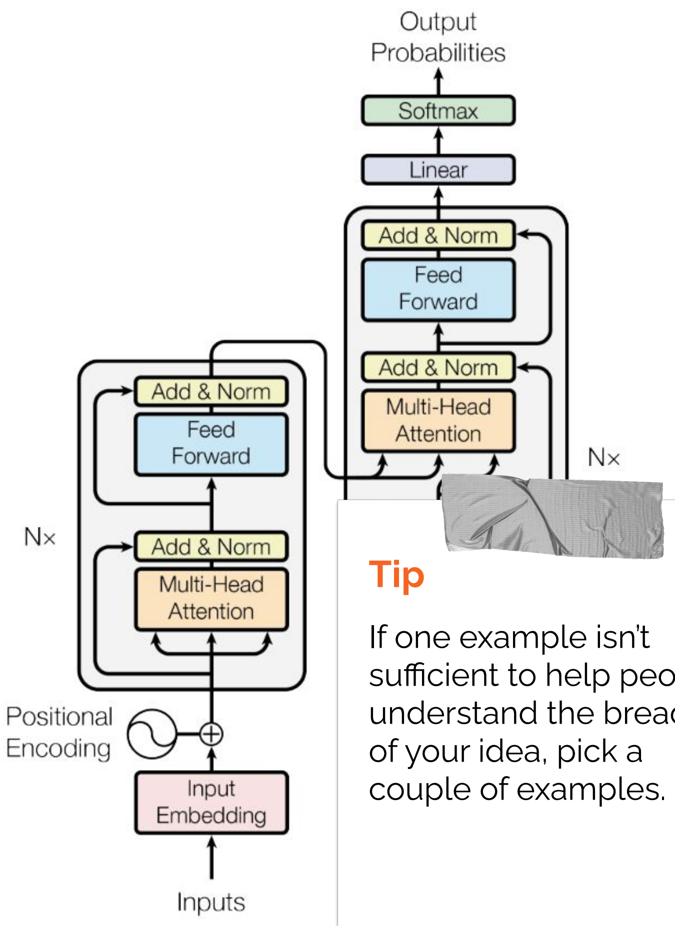
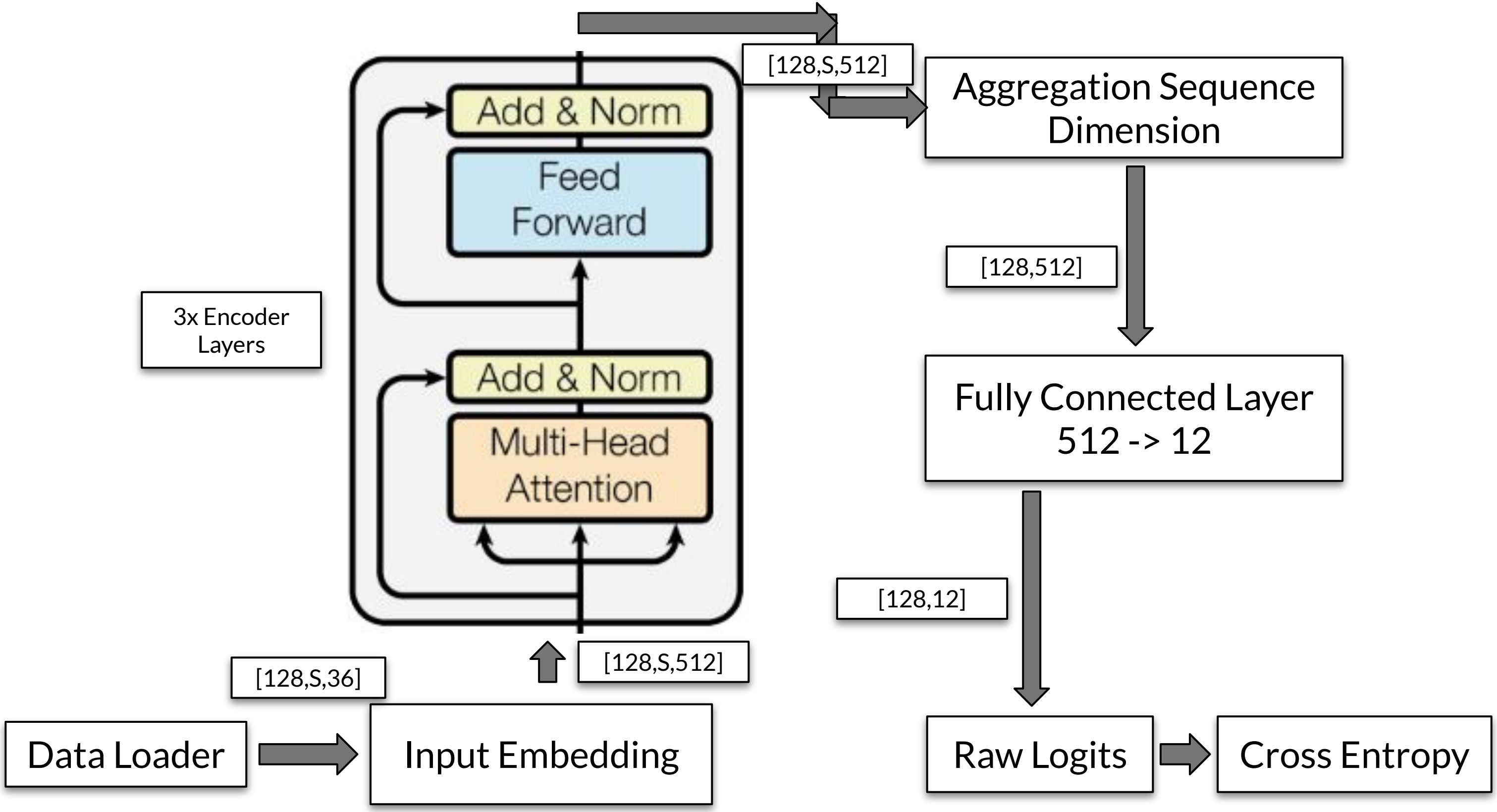


Figure 1: The Transformer - model architecture.

Tip

If one example isn't sufficient to help people understand the breadth of your idea, pick a couple of examples.

Model Architecture (Data Flow)



Optimizer and Criterion

Optimizer -> Adam, lr=0.001

Criterion -> CrossEntropyLoss

- Input Shape ([128,12],[128])
    - First input: logits for 12 classes with 128 data per batch
    - Second input: 128 labels from 0-11 for 12 classes

Tip

Ideally, speak of people in very different situations, but where each could benefit from your solution.

Training

Device: A100 from Google Colab

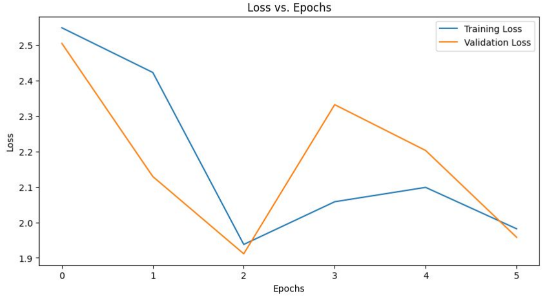
We train three times (16 hours)

- 1st 1 epoch (test run 12.62%)
- 2nd 6 epoch(24.38%)
- 3rd 3 epoch (25.87%)

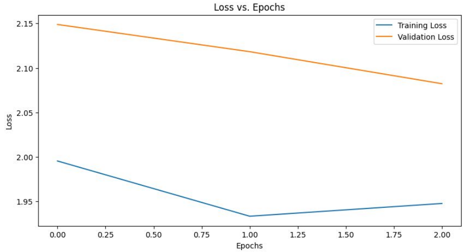
First training does not print out training loss and validation loss. Second training print out train and val loss. Third time the same. Every time after train we save the model and reload it back.

Training Result

- Epoch 1/10, Training Loss: 3.4215, Val loss: 2.4502
- Epoch 7/10, Training Loss: 1.9819, Val loss: 1.9580



- Epoch 10/10, Training Loss: 1.9477, Val loss: 2.083



- 25 more epoches on training

4. Prediction

We use the model trained to predict all the existing keys of a new song and use a greedy algorithm to find the modulation point.

- **Preprocess**
  Apply the similar procedures as that of the training set: cut a song to clips by beats, apply STFT, and convert to HPCP
- **Predict**
  Use the model to predict local keys of clips and detect the modulation point

Modulation point locating

- After we find the local key of each clip, we observe whose key is different from that of the previous one.
- We move the the clip in which a new key appears from the time of half of the previous clip to half of the current one and find the probability vectors
- We assume that the time when ratio between the two elements in the probability corresponding vector reach a certain threshold is the modulation point

Potential Problems

Musical Problems:

Percussion interference

Dominant keys confusion

Tonicizations and mixtures

Preprocessing Problems:

Peak threshold

Normalization

Weight width

Model Problem:

No Positional Encoding

Didn't use Decoder

Sequence Aggregation

Reference

[1]Y. Rou, H. Yang, H. Xu, and Y. Zhou, "Music Tonality Detection Based on Krumhansl-Schmuckler Profile," 2019.
[2]Merriam Music, "The Complete Guide to Music Key Signatures," June 10, 2019. [Online]. Available: https://www.merriammusic.com/school-of-music/piano-lessons/music-key-signatures/.
[3]Music Theory for Beginners, "Relative Minor Scales," in Piano Music Theory, June 1, 2016. [Online]. Available: https://piano-music-theory.com/2016/06/01/relative-minor-scales/.
[4]S. Campbell, "Automatic Key Detection of Music Expert from Audio," McGill University, Aug. 2010.
[5]MathWorks, "blackmanharris," [Online]. Available: https://www.mathworks.com/help/signal/ref/blackmanharris.html.
[6]robrt60, "Ultimate Guide to Musical Frequencies," iDrumTune, May 8, 2021.
[7]E. Gómez, "Tonal Description of Polyphonic Audio for Music Content Processing," INFORMS Journal on Computing, University Pompeu Fabra, Aug. 2006.
[8]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," presented at the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 2017.

Key Detection on Pop Music

Boning Wang
Jiajun Wu
Yichuan Wang
Jiajun Chen