CS229 Final Report
Automatic Melody Transcription

Bingbin LIU (06191517) Laetitia SHAO (06186108) Xiaoyan WU (05860654)

Fall 2017

Category: Audio & Music

Github:`https://github.com/ClaraBing/CS229-MusicTranscription/`

Contributions

Laëtitia worked on the dataloader used to load the data for training the CNN in `PyTorch`, preprocessed the annotations from MedleyDB to change the frequency into bins, built the inference pipeline to generate the MIDI files and generated the features visualization plot.

Bingbin worked on the spectrogram datasets, and experimented with the CNNs and merging features.

Xiaoyan worked on training and testing LSTM, providing starter code of the baseline CNN, running the librosa baseline and generating confusion matrix visualization for CNN output.

1 Introduction

Automatic Music Transcription means automatically generating a musical representation of the notes played in a music piece. Instead of transcribing every notes played in a polyphonic song, our project tackles the subproblem of melody transcription, which focuses on retrieving the melodic line (i.e. the dominant notes) from the combinations of all instruments and voices in the song.

Automatic Transcription is generally a two- part process, namely pitch estimation and pitch tracking. This report focuses on pitch estimation, while the latter was tackled in the CS221 project. We propose to recover pitches by examining patterns in audios' spectrograms using Convolutional Neural Networks. Specifically, an input audio is first divided into equally spaced time slides; each time slice will then be transformed into a spectrogram, on which the CNN performs multi-class classification to produce an output note.

This report is organized as follows: we first provide a quick overview of related work in Automatic Music Transcription and pitch estimation. We then present the details of the different models we have experimented with. Finally, we provide result analysis in section 4 and discuss about future works.

2 Related work

The problem of pitch estimation and pitch tracking is rendered non-trivial when considering noisy environments and polyphonic audio sources, where multiple instruments play at the same time and the main melody stems from the overlapping harmonics.

Previously, well-known methods addressing pitch tracking mostly rely on heuristics related to signal processing theory. The Cepstral Method ([6]) consists of pre-processing the audio signal using a Fourier-like transformation and retrieving the peaks of the resulting signal. Spectrograms and Fourier analysis are often used in Music Information Retrieval.

More recently, CNN-based methods have gained increasing popularity. For example, in [5] and [8], Convolutional Neural Networks are used together with Hidden Markov Models to perform pitch estimation and pitch tracking on noisy inputs about speech recognition. In the context of music transcription, [8] applies CNN to augmented spectrograms, where the outputs are evaluated as discretized "pitch states". Our project is mainly based on the second work, and experimented with deeper network, more input features, and different network structures. We were able to improve the baseline accuracy by 300%.

3 Method

Methods we try include librosa (a python library) baseline, CNN and LSTM. We also make improvements by merging multiple features on the CNN model.

3.1 Modeling

In music theory, a pitch uniquely corresponds to the frequency of the vibration that produces the certain pitch. A common formula links the frequency given in

Hertz, to a note that can be transcribed on a music sheet; a table of the mapping between frequencies and pitches can be found here. We use this representation to quantize the commonly seen continuous frequencies into 109 bins, corresponding to 108 notes covering 9 octaves, and an additional bin representing silence. Typically, using the equal-tempered scale where we fix the frequency of A_4 to be 440Hz, the formula that links any frequency to its corresponding bin is

$$n = \frac{\log(\frac{f}{f_0})}{\log(\sqrt[12]{2})} + 58 \tag{1}$$

where f_0 is the base frequency of 440Hz, and n is the index of the bin.

This allows us to index our classes using n where n ranges from 0 to 108. Therefore, our problem is transferred into a 109-way classification task on the spectrogram of a fixed-length input audio.

3.2 System overview

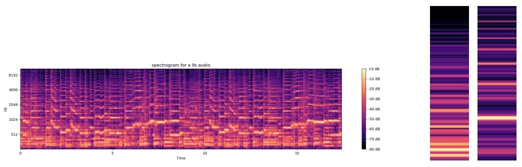

Figure 1: Example of spectrograms obtained from a 9-second audio file, and two example spectrogram slices

First, an input audio is divide into fixed-length segments. Then for each segment, a spectrogram (1) is created as the visual representation of the audio in the frequency domain. The spectrogram will then be used as the input of a convolutional neural network, which outputs a vector of probability scores for each of the 109 frequency classes. The most probable frequency classes along with their probabilities will then be provided to the second part of our system, which implements pitch tracking to determine the most probable melodies (for CS221).

3.3 Dataset

We use the MedleyDB dataset[1], which consists of 123 annotated songs expanding multiple genres including classical, pop, jazz and rock, and contains different types of instruments and vocals. Each song has a mixed multitrack audio and several single-track raw recordings. Both types of audios are annotated with timestamps in second, and corresponding frequencies in Hertz.

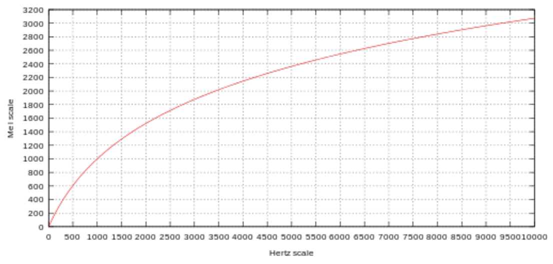

Figure 2: Plot of Mel scale v. frequency in hertz

We first prepare an image database for the audios. Each entry in the database is a spectrogram for a fixed-length audio, paired with a class label calculated from the frequency following Equation 1. To find a proper length for the audios, we experimented with both 11.6 milliseconds and 46 milliseconds. Since our final goal is to produce human-readable musical scores, we decided to use 46-millisecond ones because they provide a richer context, while 11.6-millisecond ones are unnecessarily fine-grained and could be noisy. Figure 1 shows an example of the spectrogram of a 9-second audio where there are human-interpretable patterns, and 2 example spectrograms for time slices corresponding to 835Hz and 327Hz.

3.3.1 Generating spectrograms

In our experiments we used two methods to generate the spectrograms: Mel and Constant-Q transform. Both spectrograms first use methods related to short-term Fourier transform to transform the input audio from time domain to frequency domain, then map the output frequencies to a log scale.

For Mel spectrogram, the frequencies are mapped to the *Mel scale* and quantized into 256 equally spaced bins.

For CQT, each octave is quantized into 12 bins, and the ratio between central frequencies of consecutive bins is held constant (i.e. constant Quotient), i.e. the central frequency for the k_{th} bin is $f_k = (\frac{1}{2})^{12k} \cdot f_{min}$, where f_{min} is the minimum frequency of an octave.

Qualitatively, Mel is often considered a decent representation for timbre, while CQT provides better information about pitch. A comparison between the two types of spectrograms for the same time slice is shown in 3.

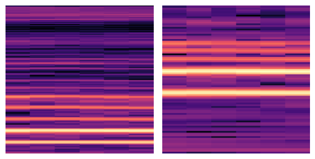

Figure 3: Comparison of Mel (left) and CQT (right)

In our experiments, we used 142961 images from 60 songs for training, 25275 images from 12 songs for validation, and 28552 images from 13 songs for testing.

3.4 Baseline 1. librosa

We started with a baseline method from `librosa`, a Python library for audio analysis. It provides a `piptrack` method which implements an algorithm that locates peaks of the input signal as pitch outputs for each time step. We select the pitch with the largest magnitude, and compare it with the ground truth annotations. This baseline method gives a **25%** testing accuracy.

3.5 Baseline 2. CNN with 5 convolutional layers

The baseline CNN contains 5 convolutional layers. The first layer has a kernel of size 7. For the remaining four layers, we set the kernel size to be homogeneous of size 3. All convolutional layers have padding that preserves the spatial dimension. A max pooling layer is inserted after every convolutional layer except the fourth, resulting in a feature map with spatial dimension of 16×16. The features are then linearized and forwarded into two fully connected layers. Finally, negative log likelihood is used to evaluate the results obtained from a 109-way classifier.

Training is conducted on mini-batches of 32 images. To obtain better generalization power, a batch normalization layer is inserted after each convolution layer, and a dropout layer (rate=0.6) is added after the first fully connected layer. We also used 0.9 momentum in the SGD optimizer. A descending learning rate is used, with descending interval differing for different network structures (explained in the next section). We used a Google Cloud VM with 4 CPUs and 2 GPUs.

3.6 Feature Merging

Since we have two different types of spectrograms from Mel and CQT, we tried different ways of merging them together, including stacking, early fusion, late fusion and model averaging.

Stacking: We started with the easiest way, which is to concatenate Mel and CQT as input to the network. The first CNN layer is modified to accommodate the new input size, and the rest of the network is kept unchanged.

Early fusion: Mel and CQT go through the first convolution layer separately, gets concatenated before the second layer and go through the remaining network together.

Late fusion: Mel and CQT go through all convolution layers separately except the last, where they get concatenated and convolved to the proper size before entering the fully connected layers. Late fusion is considered more effective when the inputs are complementary, which suits our cases where Mel and CQT focuses on timbre and pitch respectively.

Model averaging: similar to boosting, this method takes the two resulting probabilities from Mel and CQT, and output the class with the highest averaged probability.

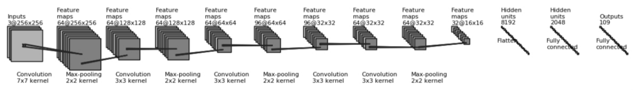

Figure 4: Architecture of the baseline CNN

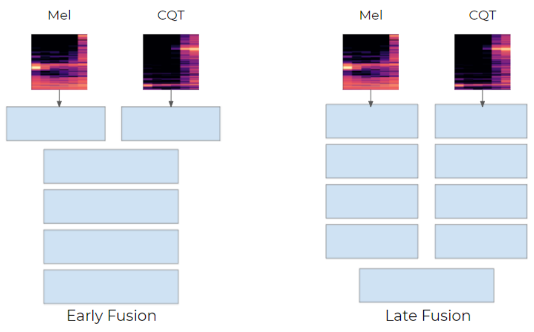

Figure 5: Early fusion (left) and late fusion (right)

3.7 LSTM

With the assumption that there may be temporal relations between notes, we experimented with a 2-layer LSTM with 1024 hidden dimensions and 109 embedding dimensions, which is the number of classes we have.

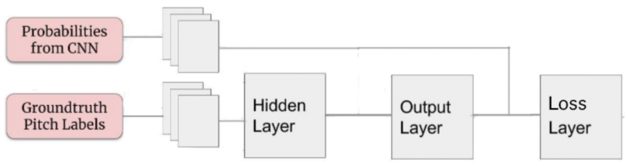

Figure 6: Architecture of the LSTM network

The optimizer of LSTM is using Adam algorithm. We choose this over stochastic gradient descent because Adam is converging faster on our data. The input is the output from CNN, which is a vector with length of 109 for each frame, containing a probability for each bin. And the labels to learn are the ground truth binary vectors at each timestamp, representing if a bin is present or not. We choose multilabel soft margin loss as our loss function since we have multiple labels. This function implements a multi-class multi-classification hinge loss, so given 2D tensors x,y, the loss is calculated using the following formula.

$$loss(x, y) = \frac{1}{n} \sum_{ij} (\max(0, 1 - (x_{y_j} - x_i)))$$

The best testing accuracy achieved by LSTM is 0.55, which is not better than the CNN. By consulting instructors in MUSIC 421 Deep Learning for Music and Audio and referring to a paper on music generation [3], we realized that pitch estimation may not be a problem suitable for LSTM. For example, our dataset contains multiple genres of music with different temporal relations between notes, so it may be hard for the LSTM to get a concrete solution on the whole dataset. This may explain why the LSTM is not working well on this problem.

4 Result and Error Analysis

In this section, we analyze the result from CNN model, and use multiple visualizations to understand and improve the performance.

4.1 Results

Table 1 offers a recap of the results (mostly top-1 accuracy) we obtained with our different models and a comparison with the original baseline. Our CNN approach has been proven to be effective, improving the librosa baseline performance by about 300%. It was shown that Mel consistently gives better results than CQT. Moreover, feature merging performed well on the training set; specifically, the two fusion methods both outperform naive stacking, and model averaging is also slightly better than the two models alone. However, the advantage of feature merging is less obvious on the validation set, which may indicate over-fitting. The testing set showed the largest performance difference in using Mel and CQT, which may partly explain why merging features made results even worse. In this case where the two types of spectrograms differ significantly, model averaging may simply copy the better model, which does not hurt the results but also offers no improvement. Overall, it seems that the performance drop from training to validation to testing may be over-fitting, which we will discuss in Future Work.

4.2 Confusion Matrix

We used a confusion matrix to visualize the performance of CNN, which shows a strong correlation between our model predictions and the ground truths. There are two observations from the confusion matrix. First, most of the errors are missing predictions, i.e. mistakingly predicting a note as silence. We think this is mainly due to severely imbalanced classes; as can be seen in figure 9, the silence class dominates both the training and testing set. We will discuss possible solutions in Future Work. Second, we found that the audios only contain the middle range of pitches, which means

Method	Training	Validation	Testing
Librosa	–	–	25.0%
Top-5	99.3%	98.6%	96.7%
CQT	83.8%	75.7%	66.3%
Mel	84.0%	76.5%	68.6%
Stacking	85.9%	75.6%	67.1%
Early fusion	**87.6%**	77.1%	67.9%
Late fusion	86,6%	77.2%	68.3%
Model averaging	84.5	77.4%	68.6%

Table 1: Recap of different accuracies obtained

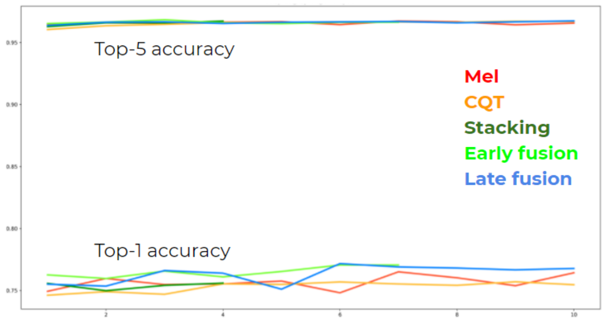

Figure 7: Accuracy across models on validation set

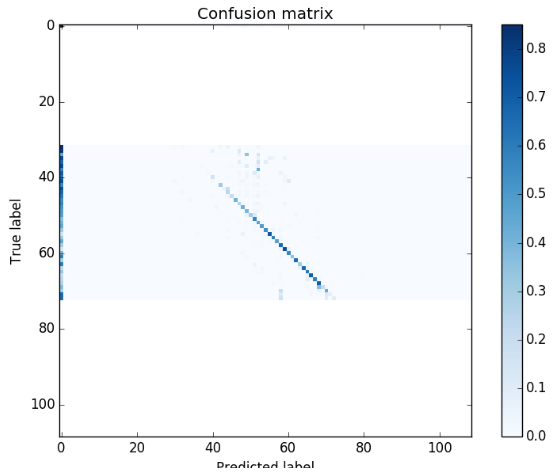

Figure 8: Confusion matrix for the CNN

we may reduce the number of output classes to reduce the model size.

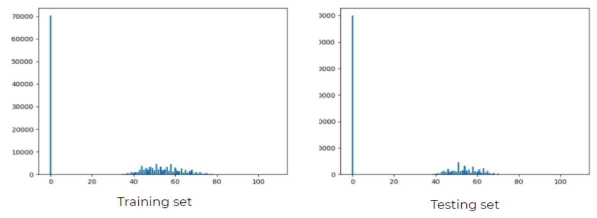

Figure 9: Number of samples for each class in training/testing set

4.3 Features visualization

We used t-SNE to help visually understand the learned features, which are descriptors of dimension 8192 (32*16*16) from the last convolution layer. Since t-SNE is computationally expensive, we first apply PCA to reduce the number of features to 50, then use t-SNE to embed the vector in a two dimensional space for visualization. Figure 10 shows the result of the visualization, which suggests that the CNN is able to learn features that allow samples from the same class to be clustered together.

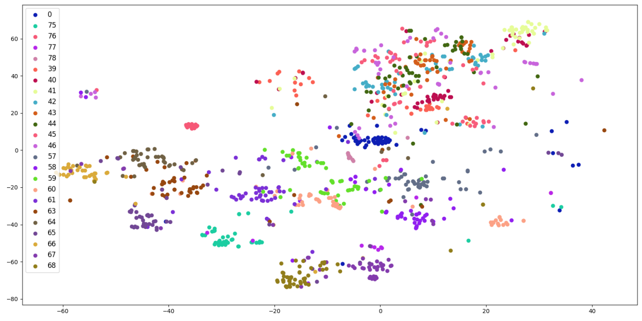

Figure 10: Feature vectors embedded in a 2D space. Each color represents a different ground truth label.

4.4 Result visualization

We generated MIDI files from the audio file which can then be visualized using a MIDI sequencer software like *Musescore*. The following figure (11)contains samples of generated music scores compared with the ground truth. The length of each note is inferred from the number of adjacent time frames it was detected in.

From the visualization of the generated MIDI files, we observed that most differences between our output and ground truth come from missing detections. Comparing outputs obtained from different genres and types

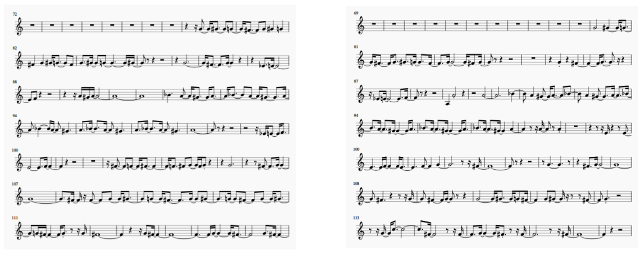

Figure 11: Result on a Schumann piece. Left: Original MIDI file created from the given annotations. Right: Inference result. (Note that there is a quarter beat of offset between the two)

of music, it also seems that the transcription is more accurate on classical pieces (11), and fails more likely on modern genres. We think this may be related to the fact that instruments usually have simpler harmonic structures while vocals are generally recognized as the most difficult to handle.

5 Future Work

Our performance has largely improved the baseline, but there are two problems that we are not handling well yet, imbalanced classes and multiple pitch output.

5.1 Handle imbalanced classes

Missing detections is the dominating type of error. Since the occurrences of notes is significantly smaller than silence (i.e. empty note), we may instead treat this as a detection problem, where we train a network to predict whether there is a note present, and proceed to a second network to determine which note it is only if the first network predicts a positive.

5.2 Multiple pitch output

Currently we are only predicting on single track audios, and our next step is to handle multiple tracks. This may induce more complexities about overlapping harmonics from different instruments, which may be helped with a set of class-specific binary masks.

6 Conclusion

Our project aimed at tackling the problem of Automatic Music Transcription. We used CNN to perform pitch estimation on generated spectrogram images and experimented with different architectures. Our current method up to 76% accuracy on a diversified range of music genres and instruments. Further analysis of our performance opened up two different ways we could improve and extend our model.

References

[1] Rachel M Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb: A multitrack dataset for annotation-intensive mir research.

[2] Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *Acoustics, speech and signal processing (ICASSP), 2012 ieee international conference on*, pages 121–124. IEEE, 2012.

[3] Madalane Boltz. Boltz, m.g. the generation of temporal and melodic expectancies during musical listening. percept. psychophys. 53, 585-600. 53:585–600, 07 1993.

[4] Alain De Cheveigné and Hideki Kawahara. Yin, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[5] Zhaozhang Jin and DeLiang Wang. Hmm-based multipitch tracking for noisy and reverberant speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(5):1091–1102, 2011.

[6] A Michael Noll. Cepstrum pitch determination. *The journal of the acoustical society of America*, 41(2):293–309, 1967.

[7] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(5):927–939, 2016.

[8] Hong Su, Hui Zhang, Xueliang Zhang, and Guanglai Gao. Convolutional neural network for robust pitch determination. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 579–583. IEEE, 2016.

[9] David Talkin. A robust algorithm for pitch tracking (rapt). 1995.

[10] JCJD Wise, J Caprio, and T Parks. Maximum likelihood pitch estimation. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(5):418–423, 1976.