CS230

Piano Music Transcription Using Convolutional Neural Networks

Zehui Wang
Department of Computer Science
Stanford University
wzehui@stanford.edu

Abstract

In this project we aim to solve piano music transcription problem. We preprocess the WAV file into frames of features via constant Q transform (CQT) and feed the features into a convolutional neural network (CNN) model which outputs the pitches within each frame. The CNN method incorporates context information in music and achieves a F-measure of 0.6527, outperforming the baseline model that uses a normal deep neural network.

1 Introduction

Automatic Music Transcription (AMT) is a system that automatically parses the notation, pitch and velocity of an acoustic music input into human-readable representations. Music transcription is considered to be time-consuming and require expert knowledge in music, so researchers have investigated various AMT approaches to tackle this problem. However, concurrently sounding notes usually cause a complex interaction and overlap of harmonics in the acoustic signal, thus it is hard for AMT systems to be comparably accurate with human performance [7].

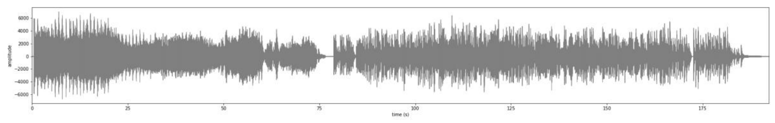
In this project we use a CNN approach to solve this problem. The input to our system is piano music in .WAV format. We then apply constant Q transform (CQT) [2] to the input so that the audio file is divided into multiple frames. Each frame has a ground truth label, thus we can use our neural networks to predict the pitch within each frame. The output is then processed to a MIDI (Musical Instrument Digital Interface) file, which can be directly converted to music scores.

2 Related work

The approaches to music transcription can be divided into frame-based methods and note-based methods. The frame-based approaches estimate pitches in each time frame and generate frame-level results. The note-based transcription approaches directly estimate the notes without dividing them into fragments [9]. Traditional AMT methods involve acoustic model and spectrogram analysis. Generally the input magnitude spectrogram is represented by a weighted combination of basis spectra which corresponds to specific pitches [1]. However, this method can lead to low accuracy due to the mismatch of the base and the musical pitch. With the trend of deep learning, many deep neural networks have been used to identify pitches and achieve substantial progress [5]. [10] investigates the predictive power of simple LSTM networks for polyphonic MIDI sequences, using an empirical approach. Such systems can be used as a music language model which, combined with an acoustic model, can improve AMT performance. [8] proposes to use convolutional neural network (CNN) to estimate the probabilities of pitches at each detected onset which achieves the state-of-the-art performance.

3 Dataset and Features

3.1 Dataset


(a) Audio files in .WAV format.

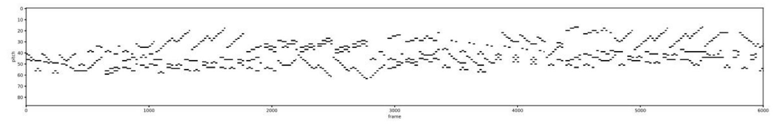

(b) Labels in MIDI format.

Figure 1: Dataset visualization

The dataset we use for this project is MAPS (MIDI Aligned Piano Sounds) dataset [4], which is composed of recordings with CD quality (16-bit, 44-kHz sampled stereo audio) and the related aligned MIDI files as ground truth. The contents of MAPS is divided into four sets: isolated notes, random chords, usual chords from Western music and pieces of piano music. For the purpose of this project, only full piano music pieces will be used. These audio files are subdivided into 9 categories depending on the instruments and recording conditions. Two of the categories are recorded in real Disklavier piano, and the rest are recorded using different MIDI synthesizers. We use real piano music as dev set and test set, and synthesizers as training set. However, due to some network issues, some of the downloaded files were broken. In the end we used training data that contains 19.37 hours of piano music, and test and dev data that contain 5.01 hours and 4.04 hours.

3.2 Data preprocessing

Following data preprocessing methods proposed in [7], we transform the input audio to a time-frequency representation. We use the constant Q transform (CQT) since it is fundamentally better suited as a time-frequency representation for music signals [2]. We compute CQTs over 7 octaves with 36 bins per octave and a hop size of 512 samples, resulting in a 252 dimensional input vector, with frame rate of 31.25 frames per second. We then process ground truth labels so that each frame has its correspondent pitch label. Specifically, each frame is labeled using a multi-hot binary vector of length 88, representing 88 notes of a piano keyboard. Finally, a post-processing is adopted to convert numpy arrays to MIDI files. To normalize the data, we use minmax normalization, that is, subtracting the min values across features and dividing by the range and then subtracting the mean.

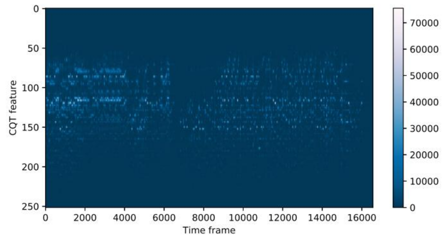


Figure 2: CQT features computed from .wav files. X-axis represents each time frame. Y-axis represents the corresponding feature vector.

4 Methods

The baseline model adopts the DNN model implemented in [6]. It has 3 dense layers with 256 units in each layer. Each hidden layer is followed by ReLU as the activation function. The output layer uses sigmoid activation. The input size and output size depend on features and ground truth. As explained in Section 3.2, the input size is 252 and output size 88 in alignment with the features of our data. The model uses Adam optimizer with Mean Squared Error (MSE) as the loss function.

One problem of the baseline approach is that it neglects the context of music and predicts purely from frequency features within one frame. To address this problem, we use a CNN model that takes an input of a context window of frames, of which the center is the target frame. Zeros paddings are used in the beginning and the end of the input. Figure 3 illustrates the architecture of our model. The context window is fed into a series of convolutional layers and then flattened and followed by two fully connected layers. This model also uses Adam optimizer with Mean Squared Error as the loss function. While doing experiments of CNN model, we also tried binary cross entropy loss function for this multi-label problem, which takes the average of all binary cross entropy over all classes. Specifically, the loss of a single example defined as

$$Loss_{CE} = \frac{1}{n}\sum_{i=1}^{n}(-y_i \log \hat{y}_i - (1-y_i)\log(1-\hat{y}_i))$$

where y is the ground truth vector , $\hat{y}$ is the predicted vector, and n is the number of classes, which is 88 in our case.

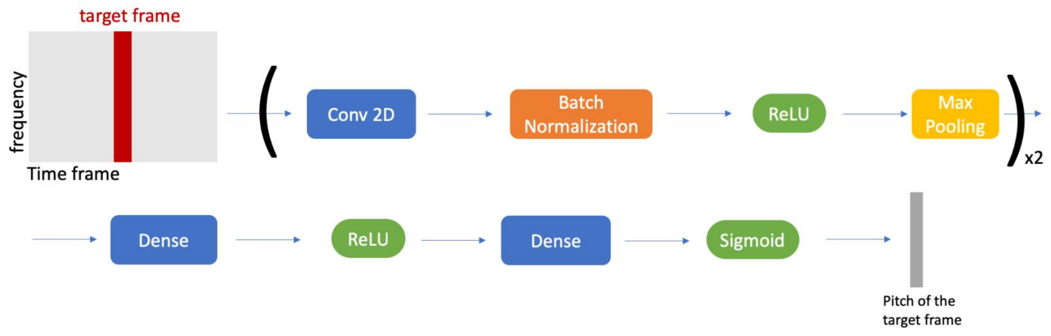


Figure 3: Overview of the CNN model

5 Experiments/Results/Discussion

5.1 Evaluation metrics

We adopt the frame-based evaluation method used in [7] to assess the performance of the neural networks. Frame-based evaluations are made by comparing the transcribed binary output and the MIDI ground truth frame-by-frame. Specifically, the frame-based metrics are defined by as follows:

$$\texttt{Precision}(P) = \frac{\texttt{TP}(t)}{\texttt{TP}(t) + \texttt{FP}(t)}$$

$$\texttt{Recall}(R) = \frac{\texttt{TP}(t)}{\texttt{TP}(t) + \texttt{FN}(t)}$$

$$\texttt{Accuracy}(A) = \frac{\texttt{TP}(t) + \texttt{TN}(t)}{\texttt{TP}(t) + \texttt{FP}(t) + \texttt{TN}(t) + \texttt{FN}(t)}$$

$$\texttt{F-measure}(F) = \frac{2PR}{P+R}$$

where TP(t) is the number of true positives for the event at time frame t, FP is false positives, FN is false negatives and TN is true negatives. The average of T frames is computed over the entire dataset to get final scores. Due to the characteristics of music data, labels are unevenly distributed (more 0s than 1s). Therefore, we will use F-measure as our primary metric.

5.2 Implementation details

The experiments were implemented with Keras [3] and run on the AWS virtual machine. The source code is available on github.[1] For all experiments, an early-stopping is forced if dev loss stops decreasing after 20 epochs.

Baseline model We use Diego González Morín's work [6] as our baseline model. The model is composed of 3 fully connected layers of 256 hidden units. Each hidden layer is followed by ReLU activation. The output layer uses sigmoid and outputs a vector of size 88. It uses mini batch size of 100, dropout of 0.2 and number of epoch 50. We train the model with Adam optimizer of learning rate 0.001. The dev loss stops decreasing after 22 epochs. The F-measure of test set is 63.93, accuracy 46.98.

CNN model Here we describe the architecture of the CNN model that achieves the best result. It uses a context window of size 7. With input shape described in Figure 3, we use two convolutional layers of kernel size (25, 5) and (5,3). Each is followed by a batch normalization layer, a ReLU activation layer and a max pooling layer of size (3,1), which means that we do max pooling only along the frequency axis, since we only want to generalize frequency features. To regularize, we use dropout of 0.5 in all layers, as well as L2 regularization of 0.0001 in dense layers. We train the model for 65 epochs using Adam optimizer with learning rate of 0.0001 and mini-batch size of 256. The best model achieves the F-measure of 0.6527, accuracy of 0.4850.

5.3 Experiment results

Table 1: Experiment Results

No.	Model	Loss Function	Parameter Searching [1]	F-measure	Accuracy
1	Baseline	MSE		0.6393	0.4698
2	Baseline	MSE	dropout 0.3	0.6398	0.4703
3	Baseline	MSE	hidden units 125	0.6243	0.4538
4	**CNN**	**MSE**		**0.6527**	**0.4850**
5	CNN	MSE	dropout = 0.2	0.3812	0.2355
6	CNN	MSE	learning rate $= 0.001$	0	- [2]
7	CNN	MSE	$L2 = 0.00005$	0.6368	0.4671
8	CNN	MSE	$L2 = 0$	0.4172	0.2636
9	CNN	MSE	window size = 9	0.5964	0.4249
10	CNN	Cross entropy loss		0.6328	0.4628
11	CNN	Cross entropy loss	$L2 = 0.00005$	0.5653	0.3940
12	CNN	Cross entropy loss	$L2 = 0$	0	- [3]

[1] The parameters listed in this column are compared to those described in Section 5.2. Empty entry means it uses the same parameters in Section 5.2.
[2, 3] In these experiments, both true postives and false positives are zero (the model predicts all 0s).

We present our experiment results in Table 1. As mentioned in Section 5.2, the baseline model is borrowed from [6], which is a complete deep neural network model with fine-tuned parameters. In order to include the context information in music, we implement our CNN model to slide a context window through each frame in the input. We try two loss functions: mean squared error and binary cross entropy loss. We also try different values of dropout and L2 regularization to reduce overfitting.

To examine the quality of our prediction, we plot our model predictions as well as ground truth labels in Figure 4. We will further discuss them in the next section. We also convert the predictions to MIDI files and visualize the pitches.[2]

[1]The source code of our project can be found in `https://github.com/zehuiw/piano_music_transcription`

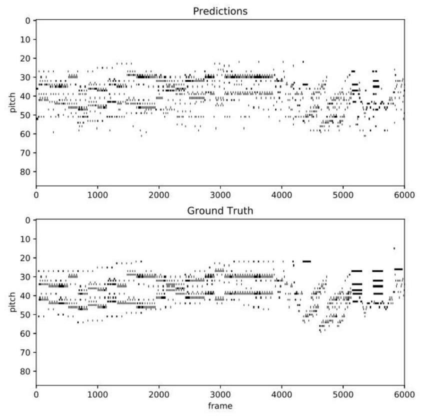
[2]The transcript music can be listened at `https://youtu.be/u-_W-XH7EJg`. It contains both ground truth and predicted music.

5.4 Analysis

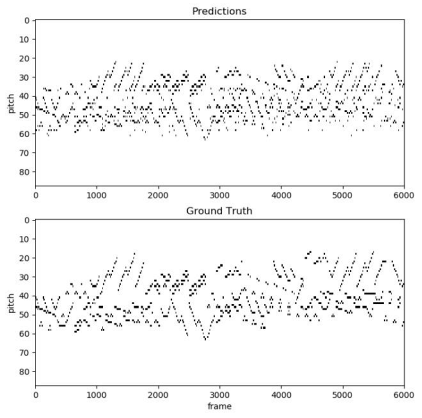
Compared with the oracle performance of F-measure of 0.7245 in [7], we analyze that the major reason could be the less amount of data, as mentioned in Section 3.1. Apart from that, we also observe a few other things as illustrated below.

Regularization Due to the scope of this project, we did not do a wide range of parameter searching, instead we focused on how dropout and L2 influence the performance. As shown in table 1, experiment 7 and 8 prove that even a small amount of L2 regularization can greatly reduce overfitting and improve the performance. We also found that comparing experiment 8 and 12, cross entropy loss is more sensible than mean squared error loss in terms of L2 regularization.

Error analysis We found that our model could not predict precisely when music notes last a long time. As shown in (a) of Figure 4, compared to the ground truth in the lower image, the upper image predicts poorly during frames 5000 to 6000, where the duration of the notes is long. After examine the training data, we suspect the problem could be that the majority in the dataset are short notes, while long notes appear less often, usually only at the end or the transition of a music piece.



(a)



(b)

Figure 4: Examples of model outputs.

Data mismatch Despite the regularization we added, the model still had an issue of overfitting. The baseline F-measure on training set was 0.8656, 35.4% higher than test set. We suspect that there might be a data mismatch problem. As discussed in Section 3.1, we use real piano music as dev set and test set, and synthesizers as training set. To solve this problem, we planned to put 20% of test data to the training set and retrained the model, however, we noticed that the new test data would give a different measure result than before on the same model (0.6465 when tested on 80% of the test data, 0.6527 on full test data). Due to time constraints and the concern that small test data would give unconvincing results, we do not continue the experiments, which leaves room for future work.

6 Conclusion/Future Work

In this project we implemented a CNN model to solve piano music transcription problem, achieving an F-measure of 0.6527. It works better than baseline neural networks because it takes account of the context information. We explored different different regularization settings and gave error analysis to our prediction results. Future work includes fine tuning CNN architectures and parameters, running experiments on RNN structure, as well as diving into data mismatch problem.

7 Contributions

Except the data prepossessing and performance evaluation scripts, which were borrowed from [6], Zehui did the rest of the project on her own, including writing CNN model, running experiments, writing MIDI converter scripts, etc.

References

[1] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[2] Judith C Brown. Calculation of a constant q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[3] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[4] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010.

[5] Rainer Kelz and Gerhard Widmer. An experimental analysis of the entanglement problem in neural-network-based music transcription systems. *arXiv preprint arXiv:1702.00025*, 2017.

[6] Diego Gonzalez Morin. Deep neural networks for piano music transcription. `https://github.com/diegomorin8/Deep-Neural-Networks-for-Piano-Music-Transcription`.

[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*, 24(5):927–939, 2016.

[8] Qi Wang, Ruohua Zhou, and Yonghong Yan. A two-stage approach to note-level transcription of a specific piano. *Applied Sciences*, 7(9):901, 2017.

[9] Qi Wang, Ruohua Zhou, and Yonghong Yan. Polyphonic piano transcription with a note-based music language model. *Applied Sciences*, 8(3):470, 2018.

[10] Adrien Ycart, Emmanouil Benetos, et al. A study on lstm networks for polyphonic music sequence modelling. ISMIR, 2017.