# Implementation of Frame Level BiLSTM for Piano Transcription

**Siddarth Selvakumar**
UVA

## Abstract

This project presents the implementation of a frame-level piano transcription model using a Bidirectional Long Short-Term Memory (BiLSTM) neural network. The model processes overlapping spectrogram segments to predict active piano notes at each time frame. A BiLSTM architecture was used with convolution layers and normalization layers producing an 88-dimensional piano roll. The approach demonstrates effective frame-wise note estimation for complex polyphonic piano music.

## 1 Introduction

Automatic music transcription (AMT) is the process of converting audio recordings into a form of music notation by analyzing rhythmic and melodic structures. AMT has been a wide area of research in the Music Information Retrieval (MIR) space, with many methods and frameworks developed to address its challenges. Despite the availability of tools and approaches, accurately transcribing audio still remains difficult due to complexities in the sound quality, chordal structure, and polyphonic nature of real-world music. In fact, there are multiple methods to transcribing the same piece.

A spectrogram graph-like representation of the audio signal that displays the spectral content of the audio over time. This extracts the frequency over time giving us access to perceived pitch, timbre, harmonics and other musical aspects from an audio file. It takes the FFT splice which provides the frequencies for each time frame and the time domain graph which provides information of the amplitude change over time. Using a Short Time Fourier Transform algorithm a 2-D representation of frequency over time can be created.

Deep Neural Networks (DNNs) provide powerful tools to extract patterns in complex input states to provide meaningful output information. DNNs only process input independently, however, music tends to be highly connected with information in the time step both before and after. To address this issue, a Recurrent Neural Network (RNNs), a type of neural network designed to handle sequences using feedback loops. Long Short-Term Memory Networks (LSTMs) are a type of RNNs which is designed to maintain and selectively forget information over longer time spans, mitigating the vanishing gradient problem that traditional RNNs suffer from. LSTMs use gated mechanisms—input, forget, and output gates—to control the flow of information. This allows for relevent memory of temporal occurences throughout the piece of music. To further capture these temporal dependencies, a Bidirectional LSTM (BiLSTM). This consists of 2 LSTM layers, one processing information after a certain time frame, and one processing information before a certain time frame. This helps improve the accurancy and temporal information that the model can pull from when making a prediction.

The dataset used contained classical music from composers such as Schubert, Mozart, and Bach. Classical music is often considered extremely fundamental when learning to play instruments such as piano due to its structural complexity, rhythmic variety, chord progression, and other fundamental skills in music. In addition, these works heavily use polyphonic notes which sound harmonically

Preprint. Under review.

sound well together. Since song transcription is not an exact art, it is important for the model to learn proper chords to be able to use in its generated songs own songs.

## 2 Methods

### 2.1 Pre-Processing

The data must be preprocessed in order to be able to be passed in the BiLSTM layers. The MusicNet Database contains information of classical music pieces matching their wav files and their midi files. Using the Python Librosa library, The wav files was sampled at 22050 Hz using the STFT method to create a spectrogram. The spectrogram was then projected onto the 128-bin Mel Spectrogram scale which better aligns with human hearing. The logarithm of this model was taken to model the decibel sound and the amplitude was normalized between -1 and 1 to reduce the difference between the different training files. At the current point in time, only songs with only piano music was analyzed. The spectrogram was then split into labeled overlapping segments to better preserve temporal connections between the parts of the song. The MusicNet database contained a csv file containing the start sample and end sample for every note along with its midi value between 0-128. Since piano is only capable of working with 88 notes, a 2D frame was created with the row representing the time frame and the columns representing each of the 88 different notes. The frame was set such that if a note was being played at any given time, the value of the row-column pair would be 1, and if not, it would be 0. The time frame was also split into the segments matching the split of the spectrogram and paired together as input and label for training.

### 2.2 Model Design

#### 2.2.1 Layers

Given the input $X \in \mathbb{R}^{256x128}$ from the spectrogram, it is passed into a LSTM layer which created with 128 hidden states and made to be Bidirectional layer which produces another LSTM with 128 hidden states from the opposite direction. It is then passed into a Convolution layer with a kernel size of 3 to identify smaller patterns such as onset spike detection. We then pass it into a layer of Normalization to stabilize the model. Finally, we pass the model through a Time Distributed Dense Layer of size 88 and activation model sigmoid. The Time Distributed method splits each of the Dense layers into its own time step and the Dense layer provides each of the 88 possible note outputs. To reduce extreme weight adjustment for the model, an L2 kernel regularizer was used. the final output is a $Y \in \mathbb{R}^{256x88}$ representing the time frame and midi note that is being played at a certain time.

#### 2.2.2 Loss Function

The Loss Function comprised of 3 parts, the Binary Cross Entropy(BCE), the excess note amount calculation, and a sparse bias. There's a weighted BCE because the likelyhood a note not being played is significantly higher than a note being played. There is a count penalty that balances count and is squared to heavily deincentivize guessing many notes wrong and primarily reduce the number of notes guesses. Finally the sparsity helps to reduce confidence on negative predictions if it is not sure to reduce false positives. The expression of the loss function is shown below.

$$\sum_{t,p} [\alpha \, y_{tp} \log \hat{y}_{tp} + (1 - y_{tp}) \log(1 - \hat{y}_{tp})] + \ \lambda_c \frac{1}{T} \sum_{t} (\hat{n}_t - n_t)^2 + \ \lambda_s \frac{1}{TP} \sum_{t,p} \hat{y}_{tp} \qquad (1)$$

$t$ – Time frame index.
$p$ – Pitch index (out of 88 keys on a piano)
$y_{tp}$ – Ground truth: 1 if pitch $p$ is active at time $t$, else 0.
$\hat{y}_{tp}$ – Model's predicted probability that pitch $p$ is active at time $t$.
$\alpha$ – Positive class weighting factor (to balance for sparsity).
$\lambda_c$ – Weight for the count penalty term.
$\hat{n}_t$ – Predicted total number of active notes at time $t$: $\sum_p \hat{y}_{tp}$.
$n_t$ – True number of active notes at time $t$: $\sum_p y_{tp}$.
$T$ – Number of time frames in the segment.

$P$ – Number of pitches (typically 88 for piano).
$TP$ – Total number of time-pitch combinations: $T \times P$.
$\lambda_s$ – Weight for the sparsity prior term.

### 2.3 Post-Processing

Finally, the model was trained with around 100 songs all containing only piano music with a batch size of 32 and the precision, recall, and precision-recall Area under the Curved information was obtained. The output array was then processed into a Midi file using the Pretty-Midi library on python. The output was then visualized using the Flat.io software.

## 3 Results and Discussion

Table 1: Model Training Metrics

| Name | Description |
|---|---|
| Precision | 0.2371 |
| Recall | 0.8721 |
| Precision Recall Area Under Curve | 0.5063 |

Table 1 summarises the frame-level metric scores obtained on the. All metrics are computed at a fixed decision threshold of 0.5 and averaged over every 256-frame spectrogram excerpt.

The main note of the metrics indicated a high recall, but low precision. The network correctly activates 87% of correct note frames. This means that it was able to determine when a note was played 87% of the time. However, only 24% of the predicted activations correspond to a true note from the recall. This indicates still a large number of false positives. Therefore the model would guess more notes in general than what was actually expected from the song. However, one good thing about the model is that it learned how to play chords with a fairly decent accuracy that the extra notes didn't sound excessively jarring.

The Precision–Recall Area Under Curve of 0.5063 reflects balance of the recall vs precision is fairly high and the model tends to recall more notes than to worry about precision. Generally in transcription, it is better to play the correct note as a home note and can add extra notes to the chord as the song will still maintain its harmonic structure.

Figure 1 shows an example of the transcription midi file that the model had produced. This was an excerpt from the piece *Pirate's of the Carribean - He's a Pirate*. This shows that there were a lot of additional notes in the chords from the false positives. However, many of these additional notes fit chordal structure from the classical music so it didn't break the sound of the song.

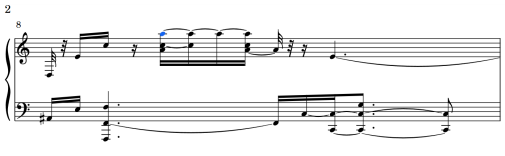

Figure 1: Example of transcription

## 4 Further Work

### 4.1 Improved Dataset

Some observations with the model was that it could make complex chords, but it was not able to play songs that are much simpler and only needed a single note. For example, when transcribing a song like Happy Birthday, it over complicated many of the notes and made them into chords.

This was likely due the dataset being comprised solely of complex hard songs which have a lot of chordal structure. To improve this, the model could have first been trained on this classical music, but fine tuned with a lower learning rate around pop music and simpler songs that is more likely to be requested. This may also help the issue of false-positives as the model will learn simpler structures.

### 4.2 Onset Detection

The paper Onsets and Frames: Dual-Objective Piano Transcription by Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, Douglas Eck explains an extremely effective approach combining frame models with onset models. Essentially, this paper, passes the onset information into the frame model to better model when a note should be played. This extra information improved recall rate by almost 3x.

### 4.3 Musical Structure

More ground rules that populate what makes a good transcription will make the model much more accurate. Currently, it is only training to detect whatever note is being played. However, if it was instead training on learning the home note and the chord type that is being played, it could better predict what note fits the music better. This would decrease the possible output space from 88! down significantly. This can be improved by adding more metrics to be measured rather than simple recall and precision. For example, A minor and C major have the same notes that make them up, but they are different harmonics when played. Having the model predict the key that a measure or segment is in can be significant in improving the ability for the model to provide a better transcription. I need to label what key each section of the song is in.

### 4.4 Future Timeline

The main improvement that I want to make in this project is to incorporate Onset detection and tempo prediction. This is help employ and lot more structure into the piece of music and help clean up some of the messy notes to make it more playable. This way, the model can predict exact note types (quarter notes, half notes, etc.) to more effectively create a human playable transcription.