Automatic Polyphonic Music Transcription

ECE 208/408 *Spring 2025*

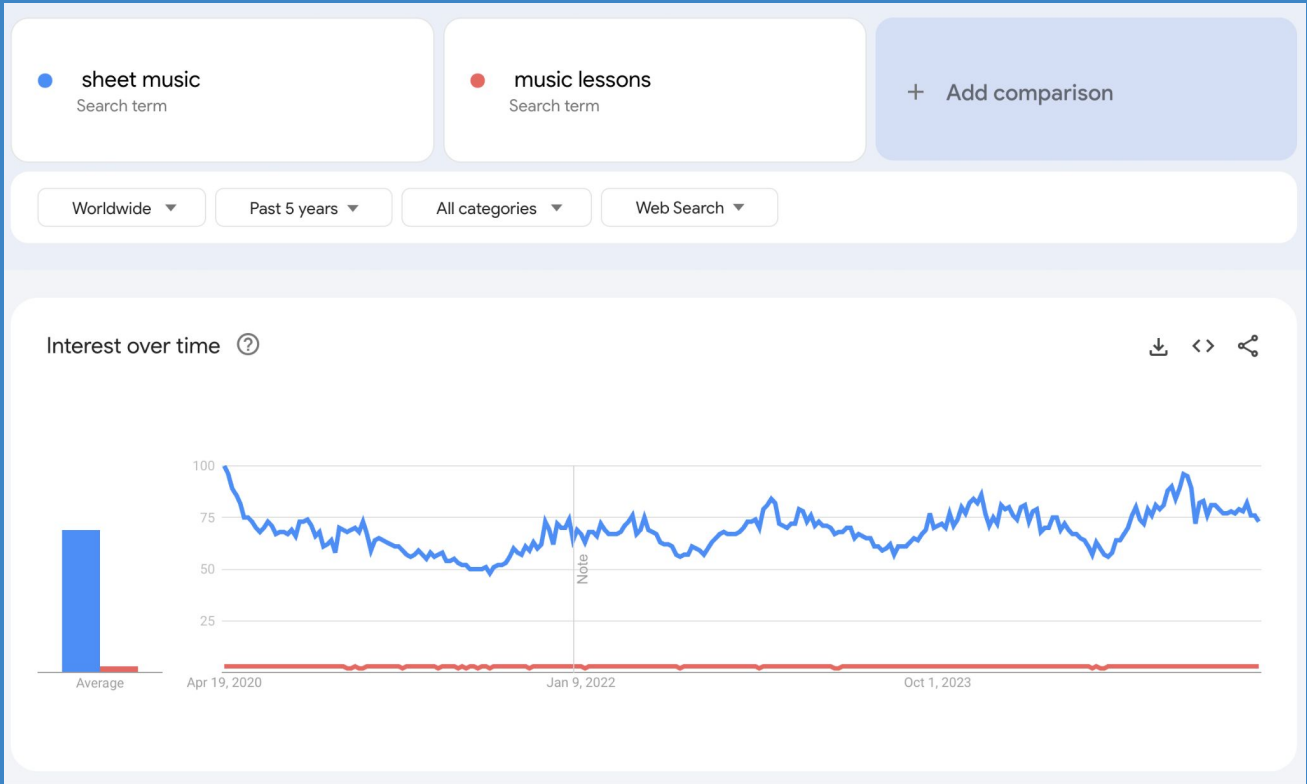
R. Faran, Y. Wu, S. Xu

01

Motivation

AUDIO TRANSCRIPTION

Continued need for accessible music



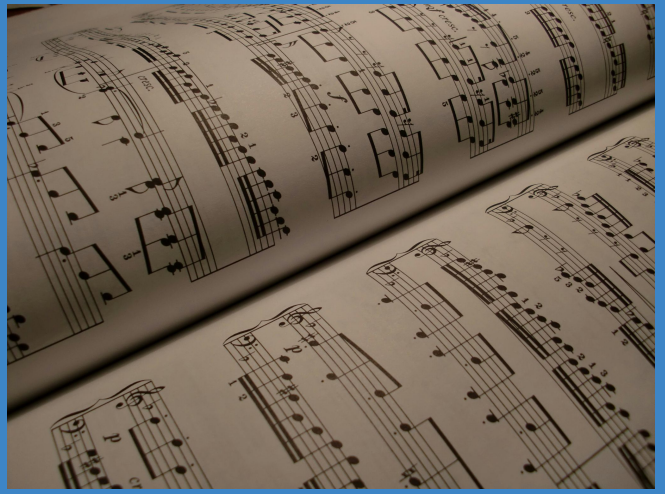
MUSIC TRANSCRIPTION

Lack of dedicated transcribers
- -Very few professional musicians can make a living off of transcription alone

Years of training required
- -Often overlap with composers or arrangers

Music Publishers are constantly looking for better and cheaper methods

Current models inaccurate, especially for multiple notes



"Although some may manage to make a living working exclusively as transcribers, for most, this job is just one of many ways for highly trained musicians to use their skill sets in an entrepreneurial fashion"

https://www.berklee.edu/careers/roles/transcriber

Why Classical Music?

High cost of entry/lessons limits accessibility
- -$60/hr or even higher!

Very old genre
- -Centuries worth of pieces and decades of recordings

Common curriculum of study for musicians and schools
- -Education and scholarship

Accessibility of dataset: most pop music is copyrighted

02

Setup

Music Transcription Steps

01 DATASET CURATION - Faran

- .wav files included
- Manually Transcribed and verified by trained musicians
- Variety of instrumentation
- Split training and test data

02 PRE-PROCESSING - Faran

- Load dataset
- STFT and Chroma data
- Load training labels (.csv)
- Infer Timing for labels

03 TRAIN A MODEL - Wu, Xu

- CNN
- HMM
- Predictive Modeling

04 TUNE HYPERPARAMETERS- Wu

- Number of layers
- Batch and hop sizes
- Layer Connectivity

05 OUTPUT EVALUATION - Faran, Xu

- .wav Input
- MIDI Output
- Accuracy

DATASET CURATION

Needed a robust and well-defined dataset; focus on polyphonic transcription

MusicNet Dataset	• 330 Classical pieces with varied instrumentation ○ Piano, String Quartet, Woodwinds, etc. • Verified by trained musician, 4% labeling error • Pre-Split Training and Testing Data
Training Input	• .wav files • .csv files for chroma information • MIDI available for evaluation stage
Desired INPUT	• .wav
Desired OUTPUT	• MIDI

03

Our Model

PRE-PROCESSING

Loading the Dataset and Labels

-reorganizing the csv, extracting labels and link it to the corresponding wav file
Only uses start_time, end_time and notes

-load .wav as mono 1D Numpy array
-Downsample the audio to 22050 Hz

start_time	end_time	instrument	note	start_beat	end_beat	note_value
9182	90078	43	53	4.0	1.5	Dotted Quarter
9182	33758	42	65	4.0	0.5	Eighth
9182	62430	1	69	4.0	1.0	Quarter
9182	202206	44	41	4.0	3.5	Whole
9182	62430	1	81	4.0	1.0	Quarter
33758	62430	42	60	4.5	0.5	Eighth
62430	90078	42	69	5.0	0.5	Eighth
62430	119774	1	84	5.0	1.0	Quarter
62430	119774	1	72	5.0	1.0	Quarter

start_time	end_time	note
9182	90078	53
9182	33758	65
9182	62430	69
9182	202206	41
9182	62430	81
33758	62430	60
62430	90078	69
62430	119774	84
62430	119774	72
90078	119774	65
90078	119774	57
119774	145886	74

PRE-PROCESSING

Audio

-broken into fixed blocks

-builds a label matrix based on the block-level slices and the total possible MIDI note values for training (128)

-reformatted for use in the model

Labels

-MIDI note onset/offsets aligned with audio block timing

-converts .csv into a usable DataFrame format

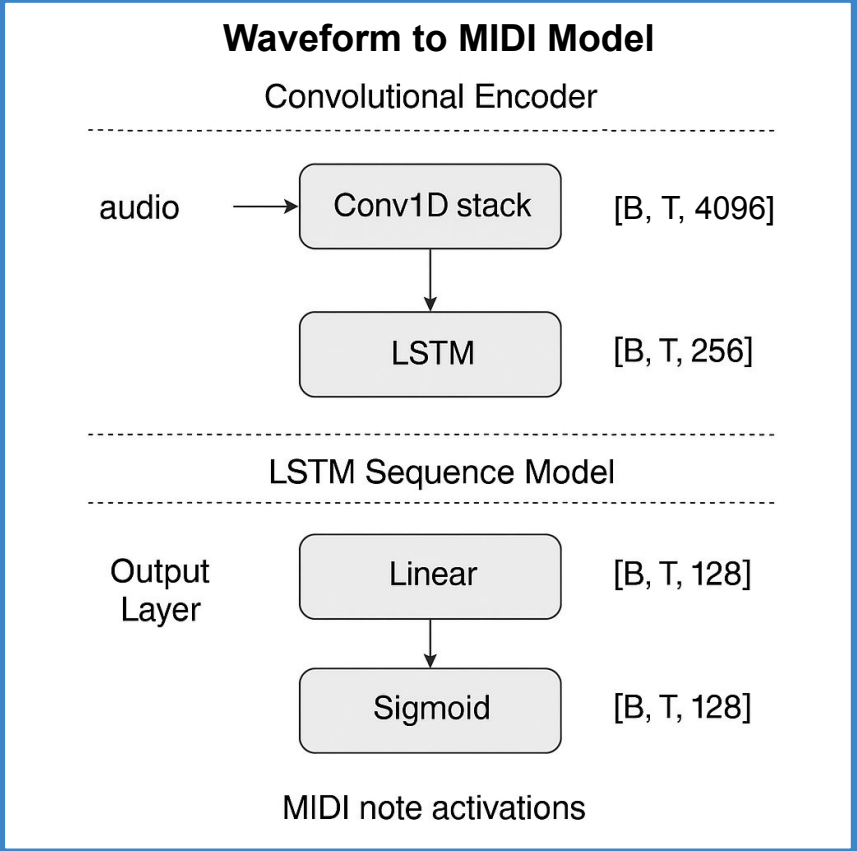
CNN

Convolutional Neural Network

-Effective at recognizing local patterns and learning hierarchical features

-Generalizes well for wider numbers of pieces

-research shows overfitting in Hidden Markov Model implementations



DEFINED MODEL

Input (B * T, 1, 4096)	B: Batch size; T: Temporal length (audioblocks) 4096: block size (samples per block)
⬇️ Convolution Encoder	1D Convolutional layer: Extract time-domain features to each block
⬇️ LSTM Sequence Model	Long Short-Term Memory Sequence: Captures temporal dependencies across blocks
⬇️ Fully Connected Layers	Maps temporal features to pitch activation space
⬇️ Softmax	Interpreted as a probability distribution over possible notes for each block
⬇️ Output	Generate output and convert to MIDI format

EXPERIMENTS

-STFT before CNN or not?
    -negligible benefit in small-scale; eliminated due to computational costs

-Number of convolution layers (underfitting)

-Identifying individual instruments
    -eliminated due to time limitations
    -filtering eliminated some of the overtones that help with identification of instruments

EVALUATION

Use model trained with training set to predict testing set

Evaluate on accuracy, precision, plot visualizations such as ROC curve

- -Pitch Error Rate (Ep): # Correct Pitches / Ground Truth
- -Extra Note Rate (Ee): # False Positive / Total # Ground Truth
- -Onset Error Rate (EoN): # Incorrect onsets / Total # Ground Truth Onsets
- -Missing Note Rate (Em): # Missed notes / Total # Reference Notes

Tune hyper parameters

Small-scale models ranged from 65-85% accuracy

ISSUES

Most models focus on simple melodies and monophonic transcription
- -polyphonic music is much harder to identify individual notes
  - -very computationally expensive! *Long* training times

GPU access issues
- -unable to log into CIRC for over a week
- -kicked out of Google Colab for using too many resources
- -unstable connections to web servers wiped out training processes multiple hours into training; had to restart several times

Generalization
- -limited to classical music due to dataset

Continued issues in instrument identification

THANK YOU

R. Faran, Y. Wu, S. Xu

Automatic Polyphonic Music Transcription

Questions?

TYPICAL PRE-PROCESSING

Extract pitch and timing data to be used in the model

STFT	• Time-Frequency Analysis • Feature Extraction • Energy distribution over Time
Chroma	• Chromagram • Specifically pitch content • Better at recognizing changes in harmonic structure
Pitch	• Derived from both Chroma features and STFT
Timing	• Derived from STFT

PRE-PROCESSING

Audio

-waveform broken into fixed-length blocks (10 blocks of 4096 samples each)

-builds a label matrix based on the block-level slices and the total possible MIDI note values for training (128)
    -[10 128]

-reformatted for use in the model

Labels

-MIDI note onset/offsets aligned with audio block timing

-converts .csv into a usable DataFrame format

-time converted into frame/block indices

-fills binary matrix [blocks, 128] with 1's where notes are active

PRE-PROCESSING

Example Loaded Annotations of Training Label

```
Annotations DataFrame:
    start_time  end_time  instrument  note  start_beat  end_beat  \
0         9182     90078          43    53         4.0       1.5
1         9182     33758          42    65         4.0       0.5
2         9182     62430           1    69         4.0       1.0
3         9182    202206          44    41         4.0       3.5
4         9182     62430           1    81         4.0       1.0

       note_value  duration
0  Dotted Quarter     80896
1          Eighth     24576
2         Quarter     53248
3           Whole    193024
4         Quarter     53248
```

CONVOLUTION ENCODER

Purpose:

- Each 4096-sample block is processed by a small CNN to extract local time-domain features.
- Operates on 1D audio waveform (like a short-time filter bank).

Steps:

- The Conv1D layers (with kernel size 9, stride 2) reduce the time resolution.
- MaxPool1D further reduces dimensionality.
- Final shape after convolution per block: [B, T, 256]

The CNN compresses each block (4096 samples) into a feature vector of length 256.

LSTM SEQUENCE MODEL

Purpose:

- Models temporal relationships between the blocks in a sequence.
- Each block has now been reduced to a 256-dim feature vector → LSTM sees a sequence of vectors: [B, T, 256].

Parameters:

- 2-layer LSTM
- Hidden size: 128
- Batch first: true (input shape is [B, T, F])

Output:

- Produces output for each time step (block) → shape [B, T, 128]

OUTPUT LAYER

Applies softmax across 128 MIDI pitches.

Interpreted as a **probability distribution over possible** notes for each block.

Each timestep gives a probability distribution for the 128 possible MIDI note values (127 notes or note off)

MASKED BINARY CROSS-ENTROPY LOSS

Computes BCE loss

-uses making padded positions to ensure only for active time steps

Averaged over valid tokens