Tones2Notes - Music Transcription using Concurrent Neural Networks

Atishay Jain 210050026
Avadhoot Jadhav 210050027
Megh Gohil 210050055
Veeresh Patil 210050163
Harshit Morj 210050062

***Abstract*—Automatic Music Transcription (AMT) refers to the task of converting a given audio into symbolic representations (musical notes or MIDI). In this project, the goal is to transcribe a given music into a MIDI file to get the musical notes played in it. Three distinct models have been implemented and evaluated. The architectural design of these and data processing techniques are informed by the ideas presented in this [1] paper. The chosen approach involves a special technique of regression on onset and offset times, and the use of Bi-directional Gated Recurrent Units (biGRU).**

***Index Terms*—Piano Transcription, MIDI, CRNN, Mel Log Spectrograms, MAPS**

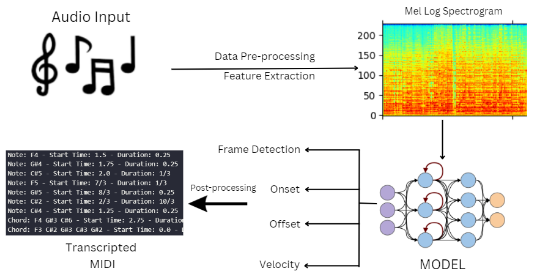
I. Introduction

Automatic music transcription is the task of transcribing audio recordings into symbolic representations, such as piano rolls, guitar fretboard charts, or Musical Instrument Digital Interface (MIDI) files. It is an essential topic of music information retrieval (MIR) and is a bridge between audio-based and symbolic-based music understanding. An AMT system can benefit several MIR tasks, such as score following, audio-to-score alignment, and score-informed source separation. In addition, it can be used in music education systems to assist music learners. It is also used in music productions for transcribing audio recording to MIDI for later editing, symbolic-based music information retrieval and studying unarchived music, such as jazz improvisations.

II. Utilization of Neural Networks for Piano Transcription

A. History and Idea

Neural networks have gained recent publicity in piano transcription. As a start, a deep belief network was proposed to learn feature representations for music transcription. Then on, fully connected neural networks, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been proposed to learn regressions from audio input to labelled ground truths. One problem of these frame-wise transcription systems is that the transcribed frames need to be elaborately post-processed to piano note events. In addition, those frame-wise piano transcription systems did not use onset targets which carry rich information on piano notes. Recently, onsets and frames systems have been proposed to predict both onsets and frame-wise pitches of notes simultaneously and have achieved state-of-the-art results in piano transcription.



```
Note: F4 - Start Time: 1.5 - Duration: 0.25
Note: G#4 - Start Time: 1.75 - Duration: 0.25
Note: C#5 - Start Time: 2.0 - Duration: 1/3
Note: F5 - Start Time: 7/3 - Duration: 1/3
Note: G#5 - Start Time: 8/3 - Duration: 0.25
Note: C#2 - Start Time: 2/3 - Duration: 10/3
Note: C#4 - Start Time: 1.25 - Duration: 0.25
Chord: F4 G#3 C#6 - Start Time: 2.75 - Duratio
Chord: F3 C#2 G#3 C#3 G#2 - Start Time: 0.0 -
```

Fig. 1. Overview of a Music Transcription System

Previous and most piano transcriptions, split audio recordings into frames. Then, piano rolls are created as targets for training, where a piano roll is a matrix containing a frame and a piano note dimension. Elements in a piano roll are assigned values of 1 or 0 indicating the presence or absence of onsets, offsets, or frames in the piano roll. These earlier models, however, predicted the presence of onsets and offsets exclusively over a single frame. Problems with such a model include: First, analysis shows that the attack of a piano note can last for several frames instead of only one frame. A note can be modeled in a more natural way with attack, decay, sustain, and release (ADSR) states. Second, the targets are sensitive to misalignment between audio recordings and labels, resulting in grave penalizing. Third, there is ambiguity when assigning labels for onsets and offset events. In addition, previous piano transcription systems predict the presence or absence of onsets and offsets frame-wise. So the transcription resolution is limited to the frame hop size.

To address the above problems, this project implements a regression-based high-resolution piano transcription system that can achieve arbitrary transcription resolution. In training, continuous targets representing the time difference between the center of a frame and its nearest onsets and offsets, are implemented. In post-processing, we demonstrate an analytical algorithm to convert the predicted continuous targets to precise onsets and offsets.

III. Dataset and Input Processing

There are mainly 3 useful datasets we found for Music Transcription -

- [2] MAPS - MIDI Aligned Piano Sounds is a database of piano recordings with CD quality (16-bit, 44-kHz sampled stereo audio) with related aligned MIDI files as ground truth. The overall size is $\approx$ 40 GB, about 65 hours of audio recordings
- [3] MusicNet - Collection of 330 freely-licensed classical music recordings, together with over 1 million annotated labels and the instrument that plays each note, which makes it useful for polyphonic transcription
- [4] MAESTRO - It is the largest among all, containing about 200 hours of paired audio and MIDI recordings from ten years of International Piano-e-Competition, with size $\approx$ 120 GB

The dataset used for this project is MAPS because the aim was monophonic transcription and it had a diverse set of piano audios with different settings. The original paper was implemented with the MAESTRO dataset, but since we had limited storage and resources, we went with MAPS. This dataset contains 7 folders each having a different setting of piano recording. The dataset was splitted as -

- Train, Validation - AkPnBcht, AkPnCGdD, SptkBGCl, AkPnBsdf, AkPnStgb, SptkBGAm, StbgTGd2
- Test - ENSTDkCl, ENSTDkAm

The first 5 folder's data was split into train and validation at random with ratio 4:1. The MIDI's and corresponding audios were saved in h5py files with attributes - File name, split, duration, Audio waveform, MIDI event dictionary. We used librosa and mido libraries for reading the audio and MIDI files.

A. Input Data and Spectrograms

Our inputs to this project are sounds or audio signals. We need a way to represent this abstract data in a form that's feedable to Neural Networks. A simple audio signal is just a variation of air pressure over time. To capture this, samples of air pressure are taken across time. What we have captured is a waveform for the signal, and this can be interpreted, modified, and analyzed with computer software. But, this is a jumbled mess from which hardly any useful information can be extracted.

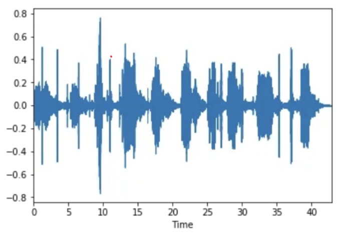


Fig. 2. Simple 2D representation of a song, with varying air pressure

An audio signal is comprised of several single-frequency sound waves. When taking samples of the signal over time, we only capture the resulting amplitudes. The Fourier transform is a mathematical formula that allows us to decompose a signal into its frequencies and the frequency's amplitude. In other words, it converts the signal from the time domain into the frequency domain. The result is called a spectrum., where the x-axis represents time, y-axis the frequency, and the color-axis the intensity, or amplitude.

With further tweaks, the y-axis is converted to a log scale, and the color dimension is converted to decibels (you can think of this as the log scale of the amplitude). This is because humans can only perceive a very small and concentrated range of frequencies and amplitudes. The resulting graph is known as a Spectrogram.

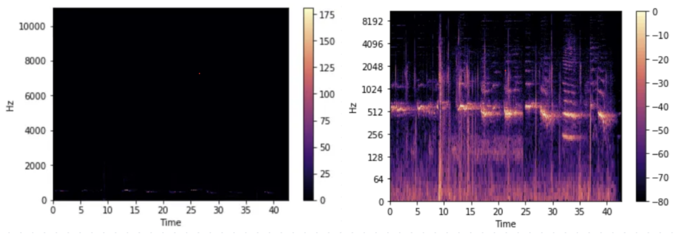


Fig. 3. Spectrogram (left) and log-Spectrogram (right)

However, studies have shown that humans do not perceive frequencies on a linear scale. We are better at detecting differences in lower frequencies than higher frequencies. For example, we can easily tell the difference between 500 and 1000 Hz, but we will hardly be able to tell a difference between 10,000 and 10,500 Hz, even though the distance between the two pairs is the same. Thus, a unit of pitch such that equal distances in pitch sounded equally distant to the listener. This is called the Mel scale. A mel spectrogram is a spectrogram where the frequencies are converted to the mel scale. This is what composes the true input to our neural network models.

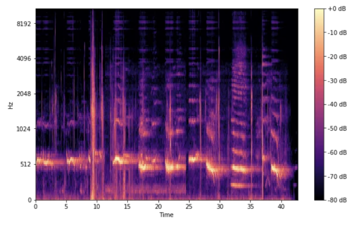


Fig. 4. Mel Log Spectrogram (suited for human perception)

B. *MIDI and Target Data*

Our final transcription targets are Musical Instrument Digital Interface (MIDI) files, which upon simple extraction are composed of nothing but a series of musical events. The events concerning us include musical notes' onsets and offsets. This information is encoded in the MIDI files in the form of a quadruple (note on/off, note, velocity, time) for every onset and offset of every note. Velocity here is just a representation of note intensity. These again are not neural network-friendly representations of the target data. According to our proposed frames-onsets system, we also need to extract the frame information from the said events.

C. *Regressed Onsets and Offsets times*

Following is a proposal for a high-resolution piano transcription system by predicting the continuous onsets and offset times of piano notes, instead of classifying the presence probabilities of onsets and offsets in each frame. We predict the time distance between the center of a frame to the precise onset or offset time of a note.

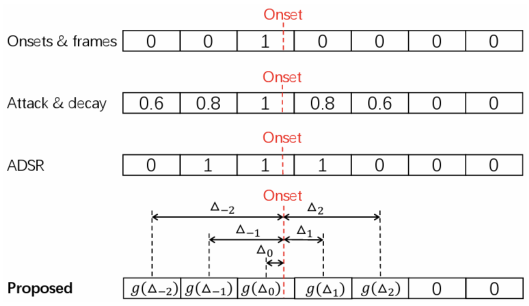


Fig. 5. Training Targets of Piano Transcription Systems

$$\begin{cases} g(\Delta_i) = 1 - \frac{|\Delta_i|}{J\Delta}, |i| \le J \\ g(\Delta_i) = 0; |i| > J, \end{cases}$$

where J is a hyper-parameter controlling the sharpness of the targets. A larger J indicates a "smoother" target and a smaller J indicates a "sharper" target.

IV. Loss Function

Here, T is the number of frames (which is 1001 for a 10-second input), and K is the number of piano notes (which is 88 in a piano). The target and predicted frames, regressed onsets, offsets, and velocities all have a size of $T \times K$.

Given I_{fr} as the target frame roll and P_{fr} as the predicted frame roll, the frame loss is given by -

$$l_{fr} = \sum_{t=1}^{T} \sum_{k=1}^{K} l_{bce}(I_{fr}(t,k), P_{fr}(t,k))$$

Given G_{on} as the target regress onset roll, G_{off} as the target regress offset roll, R_{on} as the predicted regress onset roll, and R_{off} as the predicted regress offset roll, the onset and offset losses are given by -

$$l_{on} = \sum_{t=1}^{T} \sum_{k=1}^{K} l_{bce}(G_{on}(t,k), R_{on}(t,k))$$

$$l_{off} = \sum_{t=1}^{T} \sum_{k=1}^{K} l_{bce}(G_{off}(t,k), R_{off}(t,k))$$

Given I_{vel} as the target velocity roll and P_{vel} as the predicted velocity roll, the velocity loss is given by -

$$l_{vel} = \sum_{t=1}^{T} \sum_{k=1}^{K} I_{on}(t,k) \cdot l_{bce}(I_{vel}(t,k), P_{vel}(t,k))$$

We only predict velocities for piano notes and frames containing onsets. The reason is that the onsets of piano notes carry rich information of their velocities, while the decay of the piano notes carry less information on velocities than onsets.

Here, l_{bce} denotes the binary cross entropy loss between the target and predicted values. The final total loss function of our models is given by

$$l_{note} = l_{fr} + l_{on} + l_{off} + l_{vel}$$

V. Data Pre-processing

As part of data pre-processing, according to our reference - Kong's paper, all audio recordings are converted to monophonic and are re-sampled to 16kHz. This cut-off frequency covers the frequency of the highest note C8 on a piano of 4186Hz. We split audio recordings into 10-second clips. Then, Short Time Fourier Transform(STFT) with a Hanning window size of 2048 is used to extract a spectrogram. Mel banks with 229 banks and cut-off frequencies of 30Hz and 8000Hz are used to extract the log-mel spectrogram. We use a hop size of 10ms between frames. For a 10-second audio clip, its corresponding log-mel spectrogram has a shape of (1001, 229), where the extra one frame comes from the "center=True" argument during feature extraction. Following the paper, the hyper-parameter J is set to 5, that is, each onset or offset will affect the regression values of 2xJ+1=11 frames. Each of these 10-second clips with 1001 frames acts as input to our model.

As for the pre-processing of target data-points, that is, the MIDI files, we know that we have the note events extracted from them, but we need to obtain data point targets for each input corresponding to the frame roll, regress input roll, regress output roll, and velocity roll. To achieve this, we perform a thorough feature extraction from the available note events. The onset roll, offset roll, and velocity roll are obtained directly from the events. On encountering an onset event at frame t for a note k with velocity v, we set the (t,k) entry of the onset roll to 1. On encountering an offset roll at frame t́ for note k corresponding to an onset roll, (t́,k) entry for the offset roll is set to 1, and all entries between (t,k) and (t́,k) for a frame roll are set to 1, and for a velocity roll are set to v. The regressed onset and offset rolls are also obtained as explained earlier from the onset and offset rolls by considering the exact event time as t_0 (as is given by the midi events).

NOTE: The velocities reported by the extracted MIDI events lie in the range [0, 128). To be compatible with the binary cross-entropy loss, to maintain the consistency across all 4 estimations, and also the fact that models perform better with smaller values as opposed to larger numbers (owing to lower derivatives), the velocity values are normalized to the range [0,1) before training, by simply dividing the values by 128. The predicted velocities are re-scaled back by multiplying with 128 before being reported as a final velocity value for an event.

VI. Models

In the following section, we discuss the various models implemented and analyzed under this project:

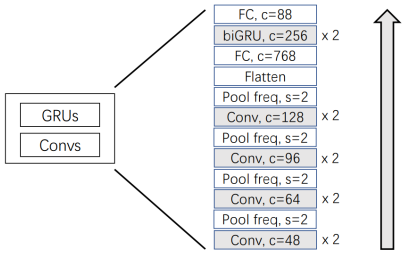
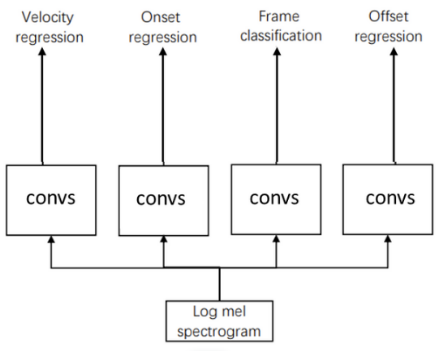


Fig. 6. Acoustic model

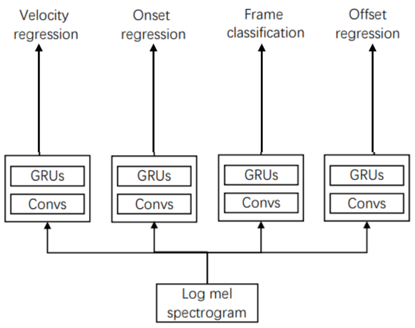
A. *Concurrent Convolutional Neural Network (CCNN)*



This model uses an acoustic model as its basic unit. The acoustic unit consists of 4 convolution blocks. Each convolution block contains 2 convolution layers with kernel size 3×3, their corresponding batch normalization layer to prevent internal covariant shift and ReLu activation function. At the end of the pass in each convolutional block, we use average pooling to reduce the number of features. After the convolutional layer, features are passed through fully connected layers to give an output of size 88 (number of classes, i.e. the piano notes) for each frame. Additionally, we add dropout layers to avoid overfitting the model.

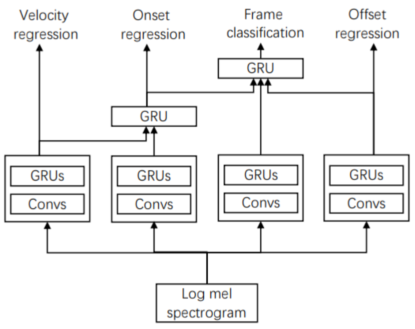
In the forward propagation of CCNN, the log mel spectrogram (1001×229) of the pre-processed waveform is passed through 4 acoustic units parallelly and independently, one each for velocity, onset, offset and frame predictions to obtain 4 separate (1001×88) outputs.

B. Concurrent Recurrent Neural Network (CRNN)



The input of our task i.e. Audio waveform is sequential data. So we want our model to capture the sequential dependencies and patterns in the input data. Also, the information of notes in the current frame depends on previous as well as next frames. In this model, therefore, we use bidirectional GRUs to capture these dependencies. Thus, we modify our acoustic unit by adding a bidirectional GRU unit after the convolutional layers. In forward propagation of CRNN thus, similar to that in CCNN, the input log mel spectrogram is passed through the 4 acoustic units, equipped with bi-GRU, independently and parallelly to predict the midi feature targets.

C. CRNN with Conditioning (Final Model)



According to the reference paper, the features of midi file i.e. onset, offset, velocity and frame classifications are not completely independent. This is because the detection of onsets and velocities can affect each other. For example, the velocity information of a piano note can be helpful to detect its corresponding onset. Similarly, the detection of the frame roll can be enhanced with the help of the onsets and offsets. For our model to be able to learn about these inter-feature dependencies, we concatenate the output of velocity and onset regression obtained by acoustic unit along their piano notes dimension, and apply a bi-GRU layer over this concatenated data, to predict the onsets. Similarly, for prediction of the frame roll, we concatenate the output of onset regression, offset regression and frame classification from the acoustic units, and then use a bi-GRU layer over this concatenated data for the final frame roll prediction.

In a single forward pass of this model, the input spectrogram of size $(T \times F)$ is batch normalized and passed to four acoustic units, equipped with bidirectional GRUs. Here, T is number of frames(1001) and F is number of mel frequency bins(229). This input is modelled by the acoustic model four times for each of the 4 features. In acoustic model, input is passed through 4 convolutional blocks each consisting of 2 convolutional layer with kernel size 3×3, 2 batch normalization layers and ReLU activation function. The four convolutional blocks have output feature maps of size 48, 64, 92, 128 respectively. The output after passing through these convolutional blocks is flattened using fully connected layer and then passed through bidirectional GRUs to learn sequential dependencies.

The final output of acoustic model is of size $(T \times K)$ where K is number of piano notes. After this, for prediction of onsets we concatenate the acoustic output of velocity and onsets along piano notes dimension to get a data of size $(T \times 2K)$ and pass this through a bidirectional GRU and fully connected layer to get final output of onset regression. Similarly, for notes classification, we concatenate the acoustic output of onset, offset and frame classification along piano notes dimension to get data of size $(T \times 3K)$ and pass this through a bidirectional GRU and fully connected layer to get the final output of frame classification.

D. Metadata

We use a batch size 8, and an Adam optimizer with a learning rate of 0.0005 for training. The learning rate is reduced by a factor of 0.9 every 10k iterations in training. A model checkpoint is saved every 5k iterations to be able to resume training from any point, or to test on any intermediate models. Prediction Statistics, after being evaluated on a subset of train data, an evaluation dataset, and an unseen test dataset, are logged ever 1k iterations. In the original paper, the model is trained for 200k iterations. But, owing to our constrained time and resources, we train it for 50k iterations. In inference, we set onset, offset and frame-wise thresholds to 0.3. A tolerance of 50ms is used for onset evaluation. A tolerance of 50ms and an off set ratio of 0.2 are used for offset evaluation. A velocity tolerance of 0.1 is used for velocity evaluation. The outputs are post-processed to MIDI events as described in the next section.

VII. Post-processing (Inference)

A. From model predictions to piano note events

After passing input through log mel spectrogram, model outputs frame wise predictions, onset regression, offset regression and velocity regression corresponding to the input spectrogram. The task left for AMT is to transcribe these predictions to piano notes in the form of high resolution note events. Note events can be represented as a tuple of <piano note, onset time, offset time, velocity>. From figure 2, a local maxima is obtained at on a particular frame. But, that may not be the actual time at which the note onset maximum occurs, since onsets are predicted on quantized frames of 10ms each. Why not to take onset time as the frame center? For our proposed high resolution transcription result!

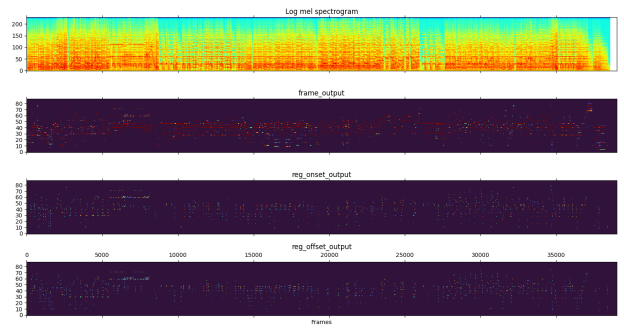


Fig. 7. Log mel spectrogram and Piano Roll of different features for an audio transcribed

To obtain the high resolution onset time, the following procedure can be employed: First, detect onset local maximum, and check if it is greater than the onset threshold defined. Consider B as the local maximum. Now, consider the adjacent frames of B i.e. A, C. We propose G as the precise onset time to be calculated. AG and GC are symmetric along the line GI. Without loss of generality, let us suppose output value at A is smaller than C then the distance BH is given by,

$$BH = \frac{x_B - x_A}{y_B - y_A}\frac{y_C - y_A}{2} \tag{1}$$

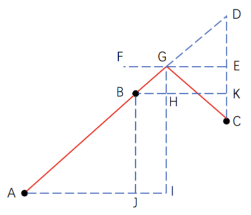


Fig. 8. precise calculation of onset, offset time. A, B and C are predicted outputs of three frames and G is the precise onset time while B is the local maximum for onset time

Derivation : In above figure, extend AB to D where CD is a vertical line. Take the median point of CD as E. Draw a horizontal line EF cross AD at G. Then, we calculate BH. We have $\triangle BGH \sim \triangle ABJ$, so $BH = \frac{AJ \cdot GH}{BJ}$. We know that $AJ = x_B - x_A$, $BJ = y_B - y_A$ and $GH = DK - DE$, and we know $AJ = BK$, so $\triangle ABJ \sim \triangle BDK$, so $DK = y_B - y_A$. Then, we have $DE = \frac{CD}{2} = \frac{DK+CK}{2} = y_B - \frac{y_A+y_C}{2}$, so $GH = \frac{y_C-y_A}{2}$, so $BH = \frac{x_B-x_A}{2}\frac{y_C-y_A}{y_B-y_A}$
In the case when value at A is larger than at C. The formula comes out to be

$$BH = \frac{x_B - x_C}{y_B - y_C}\frac{y_C - y_B}{2} \tag{2}$$

By the above proposed method, we will be able to calculate precise onset and offset time for a piano note. Algorithm 1 is the piano note transcription algorithm. For precise onset time calculation, first we check for local maximum and threshold. Then we refine the onset time by using above formulas. For offset time detection, everything is similar except for detection of offset. Offset is detected if either it is local maximum and value is above offset threshold θ_{off}, or if the frame prediction is below frame threshold θ_{fr}.

B. From piano note events to midi file

So far we converted the given audio file to piano note events. Now to play the transcribed notes, we have to convert the note events to midi file. This can be achieved through the following steps: For each note event, (represented as quadraple of note, onset time, offset time, velocity) create two messages- one for onset and one for offset. For onset, the value of message will be <time=onset time, note=piano note, velocity=velocity of note event>. For offset, the message value will be <time=offset time, note=piano note, velocity=0>. Note that for offset we are using velocity as 0 since after that time the note is not being played. Now we have a list of messages consisting of the time at which they started and

Algorithm 1 Piano notes transcription

Input: $R_{on}(t,k), R_{off}(t,k), P_{fr}(t), P_{vel}(t,k), \theta_{on}, \theta_{off}$ and θ_{fr}

Output: Piano notes

```
1:  for $k = 1, 2, ..., K$ do
2:    for $n = 1, 2, ..., N$ do
3:      //for onset detection and onset time
4:      if $R_{on}(t,k) > \theta_{on}$ and $R_{on}(t,k)$ is local maximum then
5:        Refine onset time according to (1) or (2)
6:        calculate note velocity by $P_{vel}(t,k) \times 128$
7:      end if
8:      //for offset detection and offset time
9:      if ($R_{off}(t,k) > \theta_{off}$ and $R_{off}(t,k)$ is local maximum) or $P_{fr}(t) < \theta_{fr}$ then
10:       Refine offset time according to (1) or (2)
11:     end if
12:   end for
13: end for
```

the velocity of that note and note itself. Next, sort the list with respect to time of the message. To transport this info into a midi file, we have used the mido library. We create two tracks using the mido library: the first track stores the metadata of the file (eg. tempo, time signature, end of track), and the second track has messages generated earlier from the note events. After this, we save the file using the library, and thus is ready to play, our beloved transcribed song.

VIII. Evaluation Metrics

For this project, the metrics used for prediction evaluation include mean absolute error (MAE), average precision (AP), and F1 score. These metrics are used for the following reasons:

A. Mean absolute error and Average precision

Music transcription involves capturing precise timings of note onsets and offsets. AP and MAE are well-suited to evaluate how well a system captures these temporal variations. Partial matching is very crucial in AMT as variations in note lengths and the exact timing of note boundaries are common. Average precision accommodate partial matching. Hence for frame predictions, we have used average precision. For onset, offset and velocity, there is no point of calculating AP since the values are not discrete and confined to frames but spread across, representing probabilities of presence. Therefore, accuracy can now be well visualized by the help of mean absolute error.

NOTE: Higher AP is better, whereas lower MAE is better.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{3}$$

B. F1 score

Higher the precision, means lower the model has predicted false positives. Higher the recall, means the model has predicted mostly true positives. F1 score is the harmonic mean of precision and recall.

$$F1 = \frac{2 \times P \times R}{P + R}, \; P \textit{ is precision and } R \textit{ is recall} \tag{4}$$

F1 is an important metric in automatic music transcription (AMT) because it provides a balanced measure of precision and recall, two critical aspects in evaluating the performance of transcription systems.

IX. Results

A. Comparison of Model performances on said data

The training of data has been done individually over 3 different models - CCNN, CRNN, CRNN with conditioning - as explained earlier. Owing to the fact that we haven't run the models for as long as the original paper intended, and also to the fact that the relation between the input and output data is too complex to be learned in a short time, we knew that our model was not going to overfit to the training data. We thus, also trained and tested our final model without using any dropout layers anywhere, as the function of a dropout layer is partially to reduce overfitting. All the 4 different models were analyzed over the validation dataset, and from figure below we observe that the final model (CRNN + Conditioning) performs the best among them

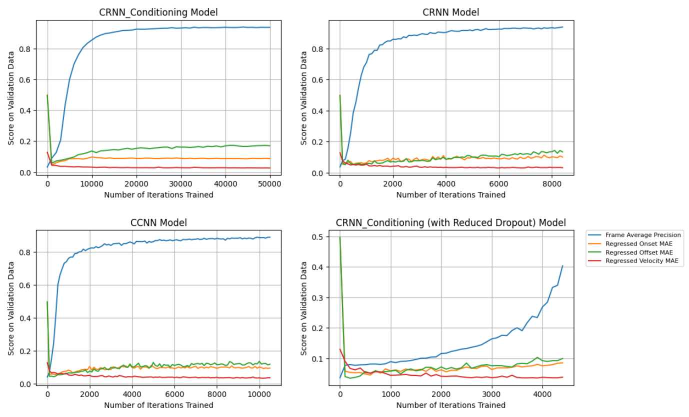


Fig. 9. Validation stats of different models

Below figure shows the F1 score of the different models, calculated on Test dataset. It can be observed that the final model predicts better than the other models in most of the test cases.

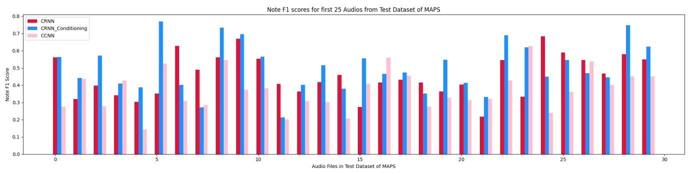


Fig. 10. Note F1 score of different models on Test data

B. Tuning of Hyperparameter J

We ran our final model with 4 different values of J - 2, 5, 7, and evaluated on the test data. We observe that there is not much difference in the Frame detection scores, but for Note detection, value of J around 5 turns out to be a better choice

TABLE I
HYPERTUNING J

Value of J	Notes			Frames		
	F1	precision	recall	F1	precision	recall
2	0.2320	0.4896	0.1564	0.7594	0.7658	0.7568
5	0.5161	0.5262	0.5072	0.8144	0.8189	0.8099
7	0.4836	0.5051	0.4666	0.8023	0.8132	0.7864

C. Comparing model with regressed data and un-regressed data

We also ran our final model architecture over standard target data, i.e. frame roll, onset roll, offset roll, and velocity roll without regression. The results very clear as to how training on regress onsets and offsets is a better proposition than the standard frames-onsets system.

TABLE II
EFFECT OF REGRESSION

Onset Offset Data Type	Notes			Frames		
	F1	precision	recall	F1	precision	recall
Regressed	0.5161	0.5262	0.5072	0.8144	0.8189	0.8099
Un-regressed	0.3094	0.2688	0.3717	0.8114	0.8181	0.8081

D. Transcribing any given Audio

The final goal was to make a model that can find and generate the musical notes of an audio/song. We ran our trained model for many audios - some from test dataset of MAPS, some generated by us (using a Synthesizer) and some from our favourite songs. The model performed pretty well for monophonic sounds, but if there are a number of multiple instruments being played then noise is introduced in the output. If vocals are added, then noise increases. You can view some sample audios and our results on them here

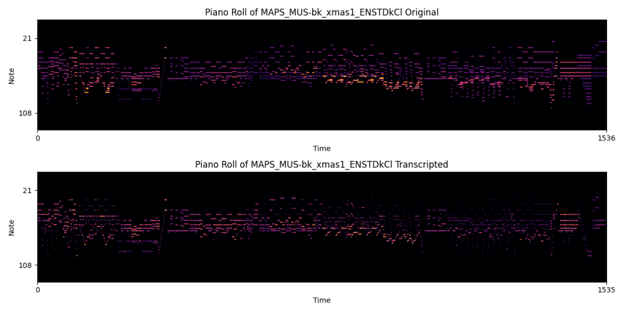


Fig. 11. Comparison of Piano Rolls of generated MIDI and Original MIDI for a song from Test dataset

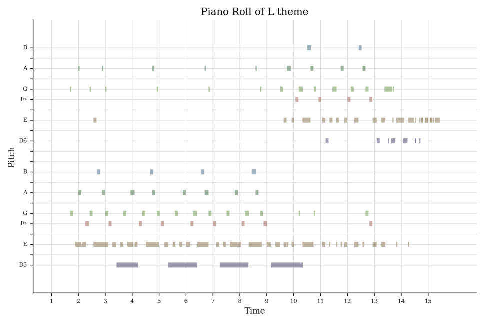


Fig. 12. Piano Roll of a sample audio generated by us (using Synthesizer)

REFERENCES

[1] Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. "High-resolution Piano Transcription with Pedals by Regressing Onsets and Offsets Times." arXiv preprint arXiv:2010.01815 (2020).

[2] Valentin Emiya, Nancy Bertin, Bertrand David, Roland Badeau. MAPS - A piano database for multipitch estimation and automatic transcription of music. [Research Report] 2010, pp.11. ⟨inria-00544155⟩

[3] John Thickstun, Zaid Harchaouiand Sham M. Kakade, "MusicNet". Zenodo, Nov. 30, 2016. doi: 10.5281/zenodo.5120004.

[4] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset." In International Conference on Learning Representations, 2019.

[5] These GitHub Repositories for data processing techniques and understanding music transcription - bytedance, BShakhovsky and Kong