AUTOMATIC MUSIC TRANSCRIPTION FOR POP MUSIC

Jonathan Yaffe
Tel Aviv University
jonathany@mail.tau.ac.il

Ben Maman
Tel Aviv University
benmaman@mail.tau.ac.il

Amit H. Bermano
Tel Aviv University
amberman@mail.tau.ac.il

ABSTRACT

Automatic Music Transcription (AMT) is the process of converting audio recordings of music into written musical notations with minimal human intervention, enabling the transformation of sound into tangible, editable, and analyzable forms. Current AMT approaches are restricted to piano and (some) guitar recordings, due to difficult data collection. In this project, we attempt the task of AMT for pop music. This task is hard due to data collection difficulty and also the wide variety of pop instruments and sounds. The main idea is to take a model that was pre-trained mainly on classic music and retrain it for pop music. We provide an "all around" model that can transcribe a wide variety of instruments and genres. Our project page with inference examples is available at `https://yoni-yaffe.github.io/pop_alignment`. We also have a colab page where you can try and transcribe your own audio files.

***K*eywords** Automatic Music Transcription (AMT)

1 Introduction

Automatic music transcription is a technological marvel that bridges the worlds of sound and notation. It involves the intricate process of converting audio recordings of music into a tangible, symbolic representation, such as sheet music or a MIDI file, with minimal to no human intervention. This transformative technology holds the promise of revolutionizing the way we interact with music, enabling a wide range of applications, from aiding musicians in transcribing their compositions to facilitating the analysis of vast music databases. Automatic music transcription combines elements of signal processing, machine learning, and deep neural networks to decode the intricate layers of melody, harmony, and rhythm inherent in musical compositions. It not only enhances music accessibility but also empowers musicians, researchers, and the music industry to explore new creative horizons and unlock the hidden treasures of musical expression.

The complexity of the task makes transcribing multi-instrument music often more challenging than transcribing single-instrument music. Unlike single-instrument pieces, multi-instrument music requires the transcriber to identify and separate the different instrument sounds, decipher their individual rhythms and melodies, and then accurately notate them on the score. Additionally, in sections where different instruments play overlapping or harmonized parts, distinguishing between them becomes challenging.. This can require a high level of skill and expertise, as well as extensive knowledge of music theory and instrumentation. Nonetheless, transcribing multi-instrument music can also be more rewarding, as it allows for a more nuanced and detailed understanding of the composition, and can capture the intricacies of the musical performance.

The success of deep neural networks in image classification have attracted more and more attention from the AMT community. However DNN require massive amount of training data. Data collection is very difficult because manual annotation require a lot of work and it is often not accurate. The two instruments with the best gathered data are Piano and Guitar. For Guitar the annotation is done semi automatically with human verification. For piano data collection is simpler because of unique equipment (the Disklavier). Sensors record the movements of the keys, hammers, and pedals during a performance. Because piano has much more data most AMT literature focus on that.

For pop music the situation is completely different. There is only manually annotated data by volunteers. The problem is that the data is low quality. Here are some problems with the data:

- People tend to change octave or tone of certain instruments
- People sometimes skip certain parts of a song or annotate only some of the instruments.
- people tend to give their own interpretation and change several notes
- Instrument labeling - instrument labeling. The instruments are not labeled consistently. For example Human voice is sometimes labeled as a violin sometimes as a trumpet.
- Timing - The timings of the labels are not accurate and sometimes way off. For that reason it is impossible to use this data to train as is.

In this work we will try to overcome these difficulties.

We manually filtered the Lakh MIDI Dataset Clean[1], modified some of the songs(mainly fixed wrong octave in some of the songs) and created a new pop dataset with 300 songs with reasonable quality. In this work we adopt the method proposed in [2] - a method for simultaneously training a transcriber and aligning the scores to their corresponding performances, in a fully-automated process. We first take a model that was trained mainly on classic music and then use [2] method to align the midi to the audio performance and then retrain the model for pop music. In the training process we include classical music as well. Our contributions are as follows:

- We propose a method to train a model on pop music without accurate labels while keeping high metrics for classical music. We show in the evaluation section that in this method we keep high metrics on the baselines. The model can also transcribe pop as shown in the github page.
- We show that for multi instrument transcription, training two models one that predict pitch only(instrument insensitive) and one that predict notes for each instrument can get better results.

2 Related Work

We continue the work of [2]. We use the $Note_{EM}$ process. As detailed in the article the process consist of three parts: 1. Take an of-the-shelf architecture trained on synthetic data and bootstrap its training on synthetic data. 2. The E-step, use the resulting network to predict the transcription of unlabeled recordings. The unaligned score is then warped based on the predictions as likelihood terms, and used as labeling 3. The M-step, the transcriber itself is trained on the new generated labels.

In this work we replace the model in stage 1 with a model that was trained on real recordings using the $Note_{EM}$ process. Then we start the process again on the pop dataset. The idea is that after training on classic music, the model will be good enough to detect pitch and then we can retrain for pop.

We use the architecture proposed by [3]. The approach is to predict pitch onsets events and then use the predictions to condition framewise prediction. During inference we do not allow for a new note to start unless the onset detector detect the start of the note. However only the onset stack will be trained. We freeze the velocity, offset and the frame stack.

3 Method

As detailed in [2] the key observation is that a weak transcriber can still produce accurate predictions if the global content of the outcome is known up to a warping function. In our case we have the unaligned labels for the pop music and we can take a pre-trained model and run the EM algorithm (Section1). We will also be using methods to deal with the low quality data as alignment and pseudo labels(Section 3.2). We will also use pitch shift augmentation(Section 3.5) to increase training data.

3.1 The EM Algorithm

Expectation–maximization (EM) algorithm is an iterative algorithm that aims to estimate parameters of a statistical model when some of the data is unobservable or latent. It alternates between two key steps: the "Expectation" step computes the expected values of the unobservable variables given the current estimates of the model parameters, and the "Maximization" step updates these parameters to maximize the likelihood of the observed data while incorporating the expected values obtained in the previous step. This iterative process continues until convergence is achieved, yielding maximum likelihood estimates for the model parameters. In our case, we start by computing probabilities for each note using the current parameters. Then the unaligned labels are wrapped using DTW (the E step). Then the labels are used to retrain the model (M step).

3.2 Pseudo Labels and Alignment

In order to train on the pop data we modify the labels in the following ways: 1. We use a previously trained model on classic music to predict notes probabilities. Then we use the probabilities to run DTW and align the unaligned labels. That way we can overcome the inaccuracies in the labels. The inaccuracies can be different timings and also wrong tempo.

After aligning the labels, we also add pseudo labels. We use the probabilities from the previous model and we use two thresholds T_{pos} and T_{neg}. Predicted notes with probability higher than T_{pos} are added to the labels. That way we can transfer the learning of the previous model to the new one. We use $T_{pos} = 0.5$ instead of 0.75 like proposed in [2] because we assume that the pre-trained model predict pitch well. In addition we do not back propagate through notes with probability lower than T_{neg}. Doing that we can get rid of wrong labeling or inaccurate labeling of the data. We use $T_{neg} = 0.01$ like in the original article. The main advantage of using pseudo labels is preserving knowledge of the model when training on pop music. In addition with Pitch Shift Augmentation 3.5 we also teach the model to detect the pseudo labels with different shifts. Pop music labels can be very inaccurate so we want to improve the quality of the labels. This method turns out to be crucial for pop training.

3.3 Instrument classes

Pop music has a wide variety of instruments so it is infeasible to detect each instrument with the current data. Thus we split the instruments into classes. The classes are Piano, Guitar, Bass and Human Voice. The human voice category contains strings and wind instruments(such as Violin, Cello, Flute and Synth). The full mapping can be found on the GitHub page. Human voice class is large because in the pop dataset human voice is inconsistently labeled so we want to catch all cases or at least most of them. In addition the pop dataset is not very big so we want as few classes as possible.

3.4 Instrument Sensitive detection

Best results were obtained when training two models. One that detects only pitch(insensitive for instrument) and one that is sensitive for instruments (detect each class in 3.3). During inference we use the pitch model to predict the notes. Then we assign an instrument for each note using the probabilities of the instrument model(we simply take the instrument with the maximum probability). We noticed that more instrument classes causes slower learning.

3.5 Pitch Shift Augmentation

As proposed in [4] we augment the data by stretching or shrinking our input audio with linear interpolation. We make pitch shifted copies by -5 to 5 semitones(overall 11 copies). In addition we apply continuous uniform distributed shift for each copy in the range of -0.1 to 0.1 semitones. Using the pitch shift help to deal with the problem of lack of data and also makes the model more robust to tuning in different recordings because usually in pop music the instruments are tuned perfectly.

3.6 Thresholds Fine Tuning

After training the model, we obtain probabilities for each note at any given time. Because the model does not perform the same for all notes we can adjust the threshold probability for each note. The default is 0.5. For each note we used the Maestro[5] dataset to calculate the note f1 score for only the specific note. Then we perform a binary search to find a local maximum. We do this for every note separately and come up with a new thresholds vector with a threshold for each note. Notes that did not appear get the default value 0.5.

3.7 Modulated Transcriber

We tried a new approach for detecting instruments. In the default configuration we have the audio as input and then we get predictions for notes and instruments each instrument separately. In this setting we also add the instrument as input and use it as "instrument embedding". Then the model predict only the pitch based on the embedding. That way we can have a one model that can predict all instruments with less parameters. This option can be more valuable when we need to predict a large amount of instrument. This method did not prove to be better at our experiments but we believe that the idea need to be explored with greater effort.

4 Experiments

4.1 Dataset

We used three Datasets in the training process:

4.1.1 Pop Dataset based on the Lakh Midi dataset

As mentioned before we formed a new dataset that is a subset of the Lakh MIDI Dataset Clean[1] and is licensed under CC-BY 4.0. The new dataset consist of ~ 300 songs and ~ 18 hours. Each song was checked manually to confirm that the midi file is on the same octave and that the midi sound reasonable. Another problem that needed to be checked is songs length. Many midi files sometimes omit part of the song and that can create big differences between the audio and midi. Another problem is that the audio and the midi are not synchronized. To solve that issue we align t we also add the pseudo labels in order to preserve the knowledge from the previous model. In addition to preserving knowledge we also teach the models to detect the pseudo labels with different pitch shifts. We also notice that instrument assignment was not consistent for human voice. In some notations human voice was notated as a string instrument sometimes as guitar and sometimes as a trumpet. To overcome this issue the class of human voice consist of many instruments.

4.1.2 Pop dataset collected from Unaligned supervision article

We used a dataset that was collected by [2] that contains ~ 17 hours of pop music.

4.1.3 Musicnet-EM

Musicnet-EM dataset is a modification of the Musicnet Dataset[6]. The dataset contains anotations of classical music. The dataset contains ~ 34 hours of recordings. As discussed in [2] the dataset was aligned as discussed in 3.2. The dataset can be found on `https://github.com/benadar293/benadar293.github.io`

4.2 Training

We used the architecture from [3] but we only trained the onset stack. weights for velocity, frame and offset were frozen for training. We use convolutional filters of size 64/64/128, and linear layers of size 1024. We re-sampled all recordings to 16kHz sample rate, and used the log-mel spectrogram with 229 bins as the input representation. We used hop length 512. We used the mean BCE loss, with an Adam optimizer, with gradient clipped to norm 3, and batch size 32.

We used a pre-trained model from [2] (a model that detects only pitch), aligned the labels and also calculated new pseudo labels as discussed in 3.2. The Musicnet EM4.1.3 dataset was left unchanged. We used Pitch Shift Augmentations3.5 for all the datasets. We trained 5 models:

1. Pop pitch first iteration - we trained a model that detects only pitch and we trained only the onset stack. The model was trained on all of the three datasets. We use a pre-trained model from [2] as the starting point. We use 100k iterations for the training.
2. Pop pitch second iteration - this model is trained using the same setup as the first model but it we use 'Pop pitch first iteration' as the starting point and train it for another 100k iterations. Here as well we train only the frame stack. This model detects only pitch.
3. Frame model - For this model we take 'Pop pitch first iteration' as the starting point and train it on the same dataset for another 100k iterations but this time we freeze the onset stack and train only the frame stack. This model detects only pitch.
4. Pop relabel pseudo labels - For this model we use 'Pop pitch second iteration' to relabel the pop datasets and then we run 100k iterations on the new relabeled dataset. We use the pre-trained model from [2] as the starting point. We train only the onset stack. This model detects only pitch. This model is basically two iterations of the EM Algorithm of [2] instead of 1.
5. Instrument sensitive model - An instrument sensitive model. We train the model only on the pop datasets after making alignments and pseudo labels based bassed on the pre-trained model. We train only the onset stack.
6. Frame model with threshold vector - this model is simply the frame model after training with a costumed threshold vector that was calculated as discussed in 3.6. we can see in the bar chart 1 that most of the thresholds are under 0.5. low thresholds increase the recall and decrease the precision.

Each model was trained using 4 Nvidia Tita XP. Each model training took $\sim$16 hours.

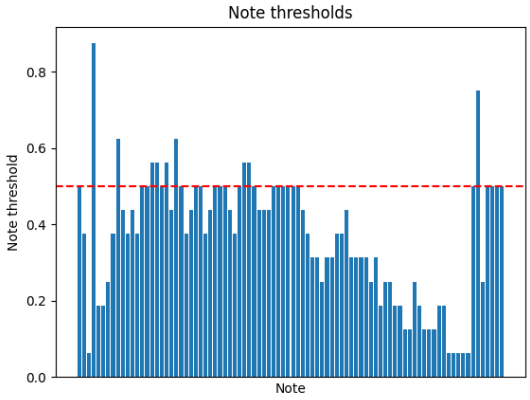

Figure 1: Note Thresholds bar chart

4.3 Evaluation

For evaluation We use the following datasets:

1. Musicnet EM - the test set of 4.1.3.
2. Guitar set [7] - Guitar set, a dataset contains $\sim$ 3 hours of guitar recordings. The dataset contains recordings of a variety of musical excerpts played on an acoustic guitar. We used the entire dataset for evaluation. The dataset was not used at all during training.
3. Maestro [5] - We use the test set of Maestro dataset. This dataset is considered to be the most accurate and large of the datsets used in evaluation. The data was not used during training but for the last model,'Frame model with threshold vector', we calculated the new thresholds based on the dataset.
4. URMP [8] - The dataset comprises a number of simple multi-instrument musical pieces assembled from coordinated but separately recorded performances of individual tracks
5. URMP BON - this is the same dataset as URMP but when evaluating we set the onset threshold to be 500s instead of 50ms we discuss the effect of the onset tolerance in 4.4

We add two tables of the note scores1 and the frame score 2 of the models

Model	**MusicNet-EM**			**GuitarSet**			**Maestro**			**URMP**			**URMP-BON**		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Pre Trained Model	93	88	90	**93**	72	81	92	80	85	**85**	74	**79**	**98**	86	**92**
Pop pitch first iteration	93	90	91	86	79	82	93	85	89	73	62	67	**98**	82	89
Pop pitch second iteration	**95**	90	**93**	90	77	83	**94**	87	**90**	71	59	64	**98**	83	89
Frame model	**95**	90	**93**	90	77	83	**94**	87	**90**	71	59	64	**98**	83	89
Pop relabel pseudo labels	94	89	92	86	**82**	84	**94**	84	89	64	55	59	94	87	90
Instrument sensitive model	80	86	83	82	80	81	78	76	77	77	**76**	76	90	**89**	89
threshold fine tune	94	**92**	**93**	90	78	**84**	93	**89**	**90**	69	60	64	97	87	91
Instrument note-with-inst	60	66	63	87	76	80	66	63	64	60	59	59	72	71	71

Table 1: Note Scores Table

4.4 URMP dataset tolerance

We noticed during the evaluation that for URMP dataset we get low scores when the onset threshold is set to default (50ms) and high scores when we increase the threshold. We made evaluations of the urmp dataset with different onset thresholds and gathered the results to 3. As we can see after we increase the threshold to 100ms the scores change

Model	MusicNet-EM			GuitarSet			Maestro			URMP			URMP-BON		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
Pre Trained Model	80	72	75	**79**	70	73	45	77	50	**88**	69	77	**87**	69	77
Pop pitch first iteration	77	62	66	**79**	74	76	47	72	48	82	46	56	83	46	56
Pop pitch second iteration	82	69	74	78	74	75	46	77	51	86	74	72	86	74	72
Frame model	**90**	**79**	**84**	76	80	77	**75**	74	**74**	83	**75**	**78**	83	**75**	78
Pop relabel pseudo labels	82	68	73	77	**77**	76	46	76	50	84	65	73	75	74	74
Instrument sensitive model	68	63	59	78	76	**77**	38	**78**	44	87	50	62	**87**	51	62
threshold fine tune	89	**79**	74	76	**81**	**78**	**75**	74	**74**	82	**77**	66	82	**77**	**80**

Table 2: Frame Scores Table

dramatically. This issue does not happen with other datasets. That can indicate that the timing of the annotations in the dataset are not very accurate.

Tolerance(ms)	50			100			250			500			1000		
	P	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
note score	71	59	64	93	78	84	96	81	87	96	81	87	96	81	87
frame score	86	74	72	86	64	72	86	74	72	83	45	56	86	74	72

Table 3: URMP evaluation with different onset tolerances

4.5 Inference

Inference examples can be found on the project page on `https://yoni-yaffe.github.io/pop_alignment`. We tested on a wide variety of pop songs.

For inference we used two models one for pitch detection (the frame model) and one for instrument detection.

5 Discussion

Our analysis reveals that a model trained solely on pitch outperforms one that predicts pitch in conjunction with instruments. Additionally, we observed that relabeling the pop data twice did not lead to improvements; in fact, it slightly diminished the metrics. Regarding the fine-tuning of thresholds, the overall impact was not substantial. While the f1 score remained constant, an increase in recall and a decrease in precision were noted. From the bar chart, a trend emerges where thresholds for middle notes tend to be closer to 0.5. This is likely due to the higher prevalence of these notes. The metrics for notes with instruments are notably lower than those for pitch scores alone. This discrepancy may stem from inconsistent labeling of instruments and the diversity of instruments in the pop dataset, complicating generalization. We also observed a more significant gap between note pitch and note-with-instrument metrics in the MusicnetEM dataset compared to the Guitar-set. Furthermore, the model demonstrated higher accuracy with guitar sounds, likely because the dataset for the instrument model was predominantly composed of pop music, excluding the Musicnet-EM dataset, which has a substantial representation of piano notes.

6 Conclusion

In this paper, we attempted the task of AMT for pop. As expected pop music is complex and finding good data to train on can be really hard. From the experiment we can learn that the new pop model can transcribe pop music well while still preserving high metrics for classical music. We see that the pseudo-labels help to preserve the knowledge of the pre-trained model. For the instrument sensitive model we noticed that guitar was the most classified instrument. Human voice class was much less frequently recognized because human voice has a wide variety of sounds and it also contains wind and string instrument so the data for this instrument was not of high enough quality. In addition as mentioned before instrument labeling is not consistent in the dataset, many instruments fall into the category of human voice. In the threshold fine tuning we see that the fine tuning didn't affect the f1 score significantly, it increased the precision and decreased the recall. However our method produced a reasonable data to train on. We believe that more work can be done to improve training such as adding guitar data and also human voice data.

References

[1] Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, 2016. `https://www.kaggle.com/datasets/imsparsh/lakh-midi-clean`.

[2] Ben Maman and Amit H. Bermano. Unaligned supervision for automatic music transcription in the wild, 2022.

[3] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription, 2018.

[4] John Thickstun, Zaid Harchaoui, Dean P. Foster, and Sham M. Kakade. Invariances and data augmentation for supervised music transcription. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2241–2245, 2018.

[5] 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.

[6] John Thickstun, Zaid Harchaoui, and Sham Kakade. Learning features of music from scratch. In *ICLR*, 2017. Available: `https://www.kaggle.com/datasets/imsparsh/musicnet-dataset`.

[7] Author(s) Name(s). Title of the paper introducing guitarset. In *Conference or Journal Name*, page Page Numbers. Publisher, Year of Publication. `https://guitarset.weebly.com/`.

[8] Bochen Li, Xinzhao Liu, Karthik Dinesh, Zhiyao Duan, and Gaurav Sharma. Creating a multi-track classical music performance dataset for multi-modal music analysis: Challenges, insights, and applications. *IEEE Transactions on Multimedia*, 2018. Available: `https://labsites.rochester.edu/air/projects/URMP.html`.