Project Report
Advanced Machine Learning
Group 2

A Lightweight instrument-agnostic model for polyphonic note transcription and multipitch estimation

Amine-Rami BENNACER	Mai-Anh DANG	Ali EL GHOUL	Tatiana ORLOVIC
Master SAR	Master SAR	Master SAR	Master SAR
21416701	28608619	28717628	21400538

Sorbonne Université
Faculté des Sciences et de l'Ingénierie

Abstract

This project is embedded in the continuity of the article [2]. We are in the field of machine learning applied sound, and more specifically music transcription. This article tries to create a simple machine learning architecture, lightweight and adaptable to any kind of instrument. On the contrary to the majority of the literature around music transcription, which often shows heavy models, specifics to certain instruments, or only adapted to mono-phonic inputs.

The article shows that it is possible to get very good results with a much lighter model, even better or at least comparable to the most ”in vogue” models around. The trick ? Have several outputs, each capturing a different information, and using them all to transcript the music.

In this report, we will try to reproduce some of the results this paper showed. We will start by a introduction to the paper, what is at stake, what have they done, what we understood from the philosophy of the paper. Then we will develop on our choices of implementation, what we were able of struggled to do (with also a focus on the data). We will finally share our results and discuss about what we learned from the project, what we could have done better and what still needs to be done.

Presentation of the Paper - Introduction

What is at stake ?

We want to create a lightweight, instrument agnostic, polyphonic compatible model that transcripts music into posteriorgrams. Why so ? In the literature around when the article got published, the original researchers noticed that most of the music transcription models are either specific towards the inputs : specific according to an instrument to improve accuracy by using its specificity, specific also according to the nature of the sound : polyphonic or monophonic. Models can also be specific regarding the kind of output and prediction they make. They can return monophonic or polyphonic outputs to a certain degree, and they can choose to work on the frequencies detected (fo) or the notes.

The goal in the studied article is to get over those restrictions and see if we can't have it all ? If we have sufficient knowledge and we segment skillfully our data into meaningful quantities, maybe a simpler model can deal better with varying types of inputs and still transcript the music in a coherent way ?

Preprocessing the data

The first thing is determining in what format the data are better to be processed, and the lightest possible. For this task, they use a Hamonic Stacking (HCQT) method, that is basically the same as the CQT, but where the harmonics are stacked on top of the other. Then, it still need to be transcripted.

Achitecture of the model

In order to do that, the article implements a CNN architecture that has three outputs.

- Yo, which corresponds to the onsets of the notes (the instant they start)
- Yp, for pitch, which captures the pitches of the note. (Indeed you can hear the same note, but a pitch higher)
- Yn for the note itself.

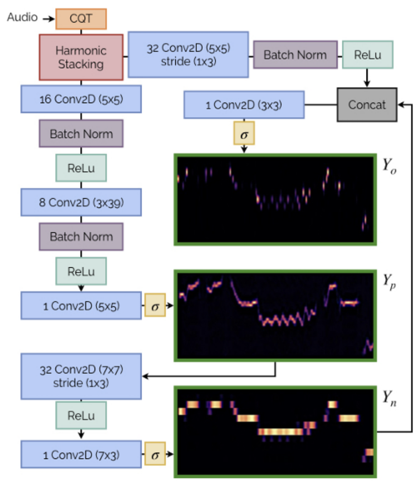


Fig. 1. The NMP architecture. The matrix posteriorgram outputs Y_o, Y_p, and Y_n are outlined in green. σ indicates a sigmoid activation.

Figure 1: Representation of the architecture in [2]

The goal with these different outputs is to capture the essence of the musical information, we need to know what note, how high it is and when it starts (also when it ends but we'll get to it).

The CNN architecture operates in three states. We can see the architecture in the figure 1. First, we extract Yp as the multipitch information. Its higher resolution (3 bins per semitone instead of 1 bin per semitone) allows it to grasp more precise information such as the vibrato. The small convolutional layers help "denoise" the signal to get rid of non-pitch information such as transients. Yp is also what 'looks the most' like the original signal. Then, adding another few convolutions acts like a "musical quantization" to get to Yn. And finally, they use Yn and some other layers to extract the onsets in the form of Yn.

Experiments The model returns some outputs to be trained with the pre-processed data. And then they expose a whole lot of results of experiments.

First of all, the article shows different versions of the model tested : the initial one, the one where they just do a CQT and not the HCQT (NMP-H) and the one where they leave Yp in unsupervised training (NMP-P). These "ablation experiments" allow the paper to show the importance of those steps in the model.

To measure the performance of their model, they compare it to some baseline, state of the art models, on the basis of mir_eval's metrics.

Without going into much more detail, we will now explore more in depth what we have and have not done in order to replicate those results, what were the difficulties and up to where we have been able to lead this project. The reader will find the github on the link [4].

Method

So what have we done ? What is the same, what is different, what did we have to do in order to get through this thing ? (Let's talk real). It might also be a good time to mention that none of us had ever to do some music processing or even sound processing, so it took us a lot of time to get acquainted with the methods, the files' types and the metrics specifics to this field. Which is a detail that will probably explain some (if not most) of our difficulties.

We decided to replicate the experiments for only the NMP model, considering that we did not want to have to train 3 different models, and the goal of those was to prove they were less performative, validating the researcher's hypothesis on the importance of the HCQT and Yp, which we consider proved.

We also decided to **focus mostly on the experiments using polyphonic datasets with several instruments** (several datasets with different instruments, but mostly polyphonic ensembles of the same instrument) - which correspond to the first table in their results. We made that choice because it was the main win of the paper. Therefore we set aside the experiments on monophonic datasets of specific instruments, we set aside the ablation experiments and the efficiency experiment. On the latter, it was anyway nearly impossible to replicate those, as they are conducted on a specific computer, which we do not possess.

Now, how did we proceed ?

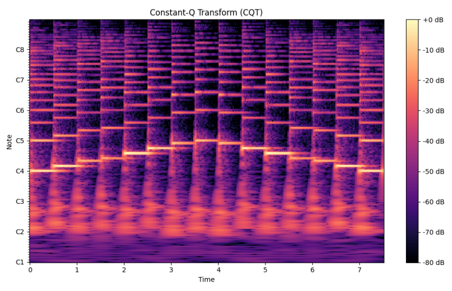
pre-processing (after feature extraction)

The first thing to do is get the data in the right format. To do that, we will divide the explanations in two parts. The really beginning of pre-processing the data will be more developed in the data section. Here, we will focus on the HCQT implementation and getting the right input shape for the model.

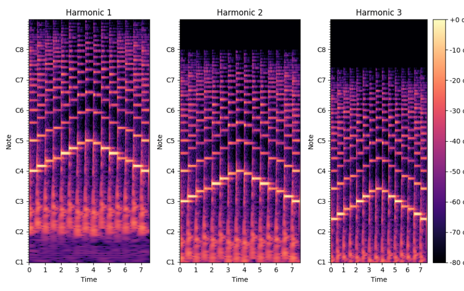
The HCQT is a 3 dimensional tensor indexed by the harmonics , frequencies and time . It measures the harmonic h at a frequency f and time t . In another way it's stacking CQT representations of sound on top of each other where HCQT[h] = CQT[fmin=h.fmin]

In CQT, the k-th bin measures $f_k = f_{\min} \cdot 2^{k/B}$, which limits harmonics $h \cdot f_k$ to $h = 2^n$, thereby missing odd harmonics. The HCQT representation aligns harmonics across the first dimension, where the k-th bin of CQT[h] corresponds to the frequency $f_k = h \cdot f_{\min} \cdot 2^{k/B}$, matching the h-th harmonic of the k-th bin in CQT[$h = 1$] (where $h = 1$ represents the fundamental frequency). This alignment allows 2D convolutional CNNs to efficiently capture locality in time, frequency, and harmonics..[1]

To work around this problem, compute a single CQT for the signal, then extract the frequency band ranging from octave i to octave n from the CQT. Append each extracted CQT to a list. Be careful with the dimensions—pad with zeros where there are gaps to match the original signal's dimensions. For example, if $h = 2$ that means $h = 1$ is deleted, translate the CQT results one octave down and pad one octave up.(inspired by the paper).[3]



CQT representation of the C major scale(resolution = 3bins/semitone)



HCQT of the C major scale(resolution = 3bins/semitone)

Model

The implementation of the model was actually one of the easiest parts. We implemented the model as shown in the figure 1. It might even be closer to the figure than the implementation of the paper, since they segment a lot into several cases and deal with the HCQT inside the model. Which we thought to do at first, using their version of the HCQT, but then we implemented ourselves.

No the difficulty was not really there, especially when all the research was about how to keep a model as light as possible. It is with everything around the model that we got some issues and had to battle against all odds do move forward.

Onsets and Frames

[5] The introduction of this work focuses on the improvements in polyphonic piano music transcription with a deep convolutional-recurrent neural network. The model predicts simultaneously note onsets and frames, conditioning framewise predictions on onset predictions. By requiring that notes can only be activated when their onset detection aligns, this dual-objective approach also improves transcription quality. It shows more than 100 % gain for note F1 scores on MAPS dataset. So do transcription, now with a velocity predicted, to allow for more expressive and natural outputs.

The way it proceeds is new in many ways. First, it incorporates the onset detection with the crane prediction in order to improve framewise transcription accuracy. It also adds another important information when one plays piano, which is the prediction of the velocity of the key note, i.e. how fast the key is being played. The model architecture combines different algorithms, including convolutional layers, bidirectional LSTM (long-short-term memory) as well as a dual loss function to train the onsets and the frame detectors. In order to evaluate the model, they use the MAPA dataset which they pre-process and utilize recall metrics. We could have used our Maestro database if we had had the time to train it and understand it, but it required a thorough knowledge in sound signal treatment. Finally, they used what is called an ablation study : in order to figure out which variation of their architecture worked best, they decided to run the algorithm without one or another method. It helped demonstrate namely the significance of onset detection, highlighting the performance's drastic decrease when it was omitted.

As a result, the authors managed to build a model which achieves a state-of-the-art performance on piano transcription. The paper encourages further work on larger and more diverse datasets, but well, we are not there yet at all. Although their work was impressive and quite frankly a milestone in polyphonic transcription, we did not have the requirements nor the capacity to achieve such results, but it would have been great to use this work as a validation.

Validation

After the model comes the post processing and the validation parts. For the validation, we wanted to use the *mir_eval* library, as is said in the article, to be able to compare our results with theirs. Unfortunately, this library is very unclear, the measures are difficult to find and most importantly, at first, it made the computer crash for some reason when we tried to test it. We then discovered that it was because of the fact that there is a *mir_eval* function in the *mir_eval* library. So the algorithm was lost to which function run when was launched.

To be then perfectly clear, we use the function *mir_eval.transcription.precision_recall_f1_overlap* in the library. (We write it down because it was very hard to find in the said library).

The Big difficulties that we encountered with Y_{train} and the model's output are that we didn't have the time to train the model on the datasets, but we managed to 'train' it on one song, partitioned into 95 samples. We overfitted the model on this one song and evaluated the outputs. The results are:

- F-measure: 0.4—0.5
- F-measure (no offset): 0.4—0.5
- Frame-level accuracy: 0.4—0.5

The validation was computed with an onset/note threshold of 0.2, which is quite low. Any higher threshold resulted in no estimated notes or pitches being detected. This suggests that the model's output needs to be more confident in order to correctly predict the onset and note events, or that the thresholds might not be properly aligned with the model's output scale. We know these values do not reflect the results of the paper, but they serve as proof that the model is being trained and learning from the data, even though it was trained on a single song. This shows progress, and further training on a larger dataset is needed to improve performance.

model set_loss: 0.0019 - val_loss: 0.0830 - val_multipitch_accuracy: 0.9969 - val_multipitch_loss: 0.0466 - val_note_accuracy: 0.9917 - val_note_loss: 0.0318 - val_onset_accuracy: 0.9992 - val_onset_loss: 0.0048t

Data

In this section we will examine the datasets used as well as the methods required to process our data accordingly to the defined model architecture.

Datasets

The training of the defined model requires a large amount of audio files which are produced using one or many different instruments (including vocals), for that purpose the article proposes a mix of seven (7) different datasets to be used for training. However, obtaining the data although being a simple task proved to not be so straightforward, which is why we start by retaking **table 1** from the proposed article and adding more details that we judged relevant 1.

Datasets	Availability	Polyphony	Instruments	Labels	Annotation Format
Molina	Publicly Available	Mono	Vocals	N	.csv
GuitarSet	Publicly Available	Mono / Poly	Ac. Guitar	N+P	.jams
MAESTRO	Publicly Available	Poly	Piano	N	.mid
Slakh	Publicly Available	Poly	Synthesizers	N	.mid
Phenicx	Publicly Available	Poly	Orchestral	N	.mid
iKala	Deleted	Mono	Vocals	N + P	-
MedleyDB	Restricted (access permission acquired)	Mono	Multiple	N+P	.csv

Table 1: Modified summary of the datasets used. The Labels column indicates which kind of annotations are available: (N) Notes, (P) Multi-pitch.

The datasets cover a range of polyphony types, and there is a diverse representation of instruments across the datasets. This diversity ranges from specialized datasets (e.g., MAESTRO for piano, Slakh for synthesizers) to more general datasets (e.g., MedleyDB and Phenicx) that include multiple instruments, and the addition of Vocals (represented in Molina and iKala.) adds more variety to the training data. However, despite their relevance, some datasets required additional effort to acquire, such as MedleyDB, where access permissions were necessary, while the unavailability of iKala makes our data uncompleted compared to that which was used to train the original model.

Additionally, the emphasis was made on the annotation format as we noticed that the metadata was saved differently, which coupled with each dataset organization and structure required manual preprocessing, and this is our next topic of discussion.

Feature extraction

The aim of the preprocessing step is to be able to extract from the data available the key features needed for training. While the article mentions the use of the mirdata library and remarks about each dataset, it fails in detailing the processing of generating the target variables.
The input data for our model consists of 2 seconds of audio sampled at 22050 Hz, which is fairly easy to reproduce, however our target variables are the corresponding Y_p, Y_o and Y_n to each audio, and there's no direct method to extract it as such, manual processing of labels is needed.
The challenge consists in the variety of annotation formats available, this makes the automation of the process harder. In our work, we gave special focus to the "midi" format as it is the most common, and the two largest datasets (Slakh and MAESTRO) are formatted this way.
Midi data saves every note event, onset and offset times within an audio recording for each instrument used, noting that notes are classified from 0 to 127 according to the MIDI Standard. To be in accordance with the input data, dividing the midi contents into 2 second segments with time realigning is performed. Finally after producing the corrected midi files, output data is generated by producing the different posteriograms Y_p, Y_o and Y_n as np.arrays, ready to be used for training. The Y_trains should share the same dimensions of the model's output, which it should be (number_of_time_frames, frequency_bins, 1).

Experimental results

As mentioned earlier, the model was over-fitted on one song to test the proper working of all components. In this section, we will explore the results of this model, even though they do not represent the true capacity of the model.

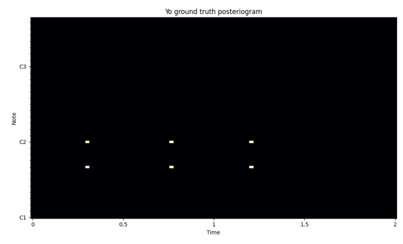
The model was trained on the `basson1.wav` file with the following parameters:

- `batch_size` = 20,
- `epochs` = 100,
- `validation_split` = 0.2.

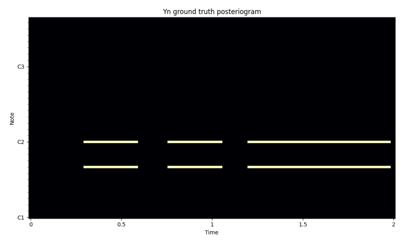
The results after the training session of the model are:

$$\text{set\_loss} = 0.0019, \quad \text{val\_loss} = 0.0830, \quad \text{val\_multipitch\_accuracy} = 0.9969, \quad \text{val\_multipitch\_loss} = 0.0466,$$

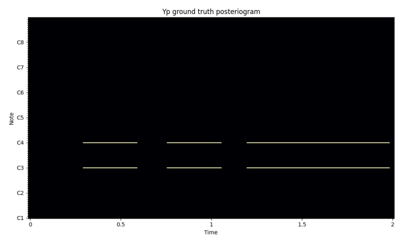
$$\text{val\_note\_accuracy} = 0.9917, \quad \text{val\_note\_loss} = 0.0318, \quad \text{val\_onset\_accuracy} = 0.9992, \quad \text{val\_onset\_loss} = 0.0048$$



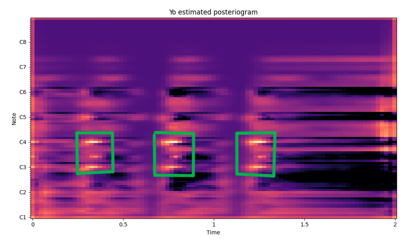
Ground Truth: Onset (Y_o)



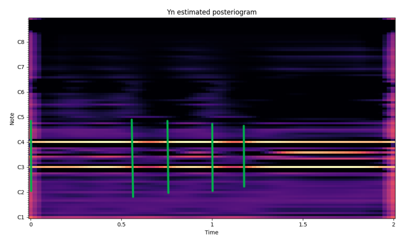
Ground Truth: Note (Y_n)



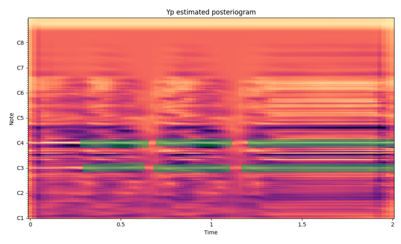
Ground Truth: Pitch (Y_p)



Estimated Onset (Y_o)



Estimated Note (Y_n)



Estimated Pitch (Y_p)

We find these graphs quite interesting, despite several inconsistencies present in them. First, the estimated graphs are very noisy, indicating that the model lacks confidence. However, we can still clearly observe identical patterns between the estimated and the ground truth, where these patterns are the brightest in the estimated graphs. Second, these identifiable patterns are unfortunately shifted upwards by approximately two octaves in the estimated graph compared to the ground truth, which suggests either an issue with the labeling of the training data or a problem with the visualization. Despite these challenges, these results represent a significant improvement compared to the initial ones we obtained.

Discussion

Alright, after having spent pages on our problems, difficulties, issues and other names, let's talk about what we could do to go further.

We are actually, despite everything we encountered, have something that runs and returns semi-valid results!

We were able to set a program that :

- Has almost all the datasets
- Loads the files, cuts them into 2 second-pieces
- Extracts the Ttrain
- Computes the files into a CQT
- Stacks the CQT int HCQT (that we re-implemented)
- Gets the input into and out of the model
- Train
- Outputs a fairly good prediction of onset, notes, and pitch spectrograms of a sound segment even though it still needs a lot of training.
- Takes the outputs and compares them in the mir_eval metrics

So indeed, we don't have real results to show, yet. We hope to have something to show for the oral examination (without promising anything). Still, on the basis of what we have, what could we improve ? What have we found ?

We still have lots of data to process, for now, we only have one batch, which was a milestone to setup all the parts together, now we need to extend to all the data and do a proper training. Once we have that, we could do a better validation.

Then, we could extend to other experiments. The first experiment would be easily transposable to the NMP-P model, for that we just need to let Yp be unsupervised and transpose the loss for this specific output. The ablation of the HCQT however would need to work on the shapes of the input of the model and adapt it to a 3D output instead of a 2D.

And something we really wanted to try, but haven't found the time to do yet, was to implement an Onset's and Frames-like method to visualize the concatenation of our model predictions. It was indeed not absolutely necessary to the implementation and this part of the article is more useful for the following users of $basic_pitch$, as it is a very visual and understandable visualization of the transcription, and allows the user to really grasp what has done the model, where it did right or wrong. It can also be useful in the training phase to see where are the problems : are they more from the notes, the pitches, is the multiplitch well taken into account? So we understand that it is a nice tool to have to adapt the model and see what is happening.

Conclusion

To conclude on this very interesting, traumatizing, stimulating project, what have we learned ?

First of all, we have to admit that we did not do the best project management we could have, and lacks of communication has made us loose a lot of time. We have been overwhelmed by reports to send (even on the 31st of December) by other courses and it proved to be difficult to get this project done in time. But we still managed to get almost everything done.

We have learned that there is a huge field and a wide variety of ways of treating sound and more specifically music data. We now know how to implement a HCQT method and have seen how a CNN, within its different layers, can be very adapted to extract different information from a file. If chosen wisely, those different outputs are a good approximation of what is important in music : when you read a partition to play an instrument, you need to know when to start, what is the note and how high/low it is (in what key usually).

For some of us, we have learned to use GitHub and still have a lot to learn.

It is time for us to close this chapter. Again, we are really sorry about the lack of results to show, especially when we know we are so close. Thank you for your attention and we will see you soon!

Maï-Anh, Amine, Ali and Tatiana.

Bibliography

References

[1] Rachel Bittner et al. "Deep Salience Representations for f0 Estimation in Polyphonic Music". In: Oct. 2017.

[2] Rachel M. Bittner et al. "A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation". In: *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2022, pp. 781–785. DOI: 10.1109/ICASSP43922.2022.9746549.

[3] "Github of the paper". In: 2022. URL: https://github.com/spotify/basic-pitch.

[4] Group2. "Github of the project". In: 2025. URL: https://github.com/PickleRickRoll/MLA-Project.

[5] Curtis Hawthorne et al. "Onsets and Frames: Dual-Objective Piano Transcription". In: *19th International Society for Music Information Retrieval Conference*. Paris, France, 2018. URL: https://goo.gl/magenta/onsets-frames-code.