International Journal of User-System Interaction 14(3) 2021, 87-100

Automatic Music Transcription by Deep Learning

Dan Teodor Avramescu, Paul Stefan Popescu, Mihai Mocanu and Marian Cristian Mihaescu
University of Craiova
Department of Computers and Information Technology
avramescu.dan.m8s@student.ucv.ro,
{stefan.popescu,mihai.mocanu,cristian.mihaescu}@edu.ucv.ro

Abstract. Automatic music transcription represents a specific translation task that falls into information processing, where input information is sound. Similar, but more general, are building text transcripts from speech. We propose a data analysis pipeline that follows the roadmap of previous works but uses a distinct dataset and several custom hyperparameter settings. The results are not as good as previous ones as the implementation runs on commodity systems, and therefore refinements and more powerful techniques may further improve the results. Still, state-of-the-art DL libraries raise the potential for improvements in a susceptible application domain.

Keywords: automatic music transcription; deep learning.

DOI: 10.37789/ijusi.2021.14.3.1

1. Introduction

Deep learning (further referred to as *DL*) algorithms have been integrated into many application domains and proved their effectiveness. This paper tackles the problem of building the transcription of polyphonic music given as input a music file in MIDI format (*Rothstein 1992)*. Thus, automatic transcription represents an ancient task that could nowadays benefit from the high potential currently developed DL algorithms. Professional music software systems integrate various functionalities, and usage of DL for music transcription should be emulative simple.

The limitations of current systems are mostly related to the quality of the MIDI transcript. It may contain many errors (i.e., wrong or missing notes) exhibited by the best algorithms and do not correspond to human performance. Therefore, many applications such as musical education, validation of students' musical abilities of students or helping amateur musicians generate musical transcripts could not be obtained by other means.

Similarly, with the natural language processing task of building a transcript from a recorded speech, music transcription reduces to producing a paper or digital representation of a musical continuous sound wave.

The difficulty of building precise Automatic Music Transcription (AMT) systems comparable to human professionals derives from two reasons. Firstly, each note is represented by a complete spectrum of harmonics with variable intensity, which is highly sensitive to musical instruments. Secondly, the overlapping of harmonics introduced by polyphonic (i.e., many notes that exist simultaneously) reduces the problem of source separation. In our case, we do not retrieve any information about the instrument, and all transcript notes are combined on one side.

2. Related Work

The music may be represented by a continuous sound wave that represents the music itself. The MIDI representation may be regarded as an exemplary symbolic musical notation. Therefore, AMT is a fundamental problem in Computer Audition (CA) and Music Information Retrieval (MIR).

The AMT concept (*Piszczalski & Galler 1977)* has developed starting in 1977 due to the new field of digital audio engineering, which heavily relied on the availability of computer systems and their ability to be programmed to analyze a music recording detects the musical notes along with rhythm accents of percussion instruments. The AMT task explores a musical play and prints a partiture from that play.

An approach using deep neural networks (DNN) in (*Morin 2017)* had the objective to compare the performance between LSTM (Long short-term memory artificial neural network) and simple DNNs in the task of Automatic Music Transcription (AMT).

A model proposed in (*Wu 2019)* performed polyphonic transcription via a system of support vector machine (SVM) classifiers using 87 OVA binary note classifiers to perform classification with the advantage of simplicity. Then, a post-processing of the Hidden Markov Model (HMM) was adopted to smooth the results temporarily.

A recent model proposed in (*Poliner 2006)* used the semantic segmentation model for transcription, which is also widely used in image

processing. This model is improved initially from DeepLabV3+[1] and further combined with U-net architecture and focal loss.

Our code is adapted from the GitHub project (*BShakhovsky 2021)* to run on a different dataset. The network structure was reconfigured and a hyperparameter tuning was performed to achieve a decent performance on the new data.

3. Proposed System

In general terms, music is a type of sound. At least in western music – which is exclusively used in our study – there are 12 classes of pitch (i.e., the height of the sound), each of them being offset by a semitone or half step. Each note is in one of these pitch classes, in a particular octave (i.e., range of frequencies) which occurs in a specific quantity of time. A chord is defined as two or more notes played at the same time. The task of this study is to build the transcript for piano music, which has 88 keys from A0 to C8.

The proposed system takes input audio files (i.e., mp3, wav, etc.) and returns corresponding MIDI files as output. In our case, the system

The MP3 is a compressed audio file, while the MIDI file has no raw (i.e., wave) audio data. It provides information about the components of the note, such as pitch (height), volume (how loud or quiet it sounds), modulation (how long it lasts), as well as other characteristics. This MIDI file can be easily converted to a transcribed score, easily shared with other musicians for collaboration, and edited to change notes in that performance to enhance a composition individually.

The sound we hear playing the MIDI file is created digitally (with a device/program). It uses its data to play a song, similar to a musician playing by scores, while the MP3 is an audio recording. The MIDI pitch range is 0-127, but we will only use the interval between MIDI notes 21-108, representing A0 - C8 (standard piano pitches).

Data Preprocessing

We use a single train/validation/test division to satisfy the following criteria:

[1] DeepLabV3: https://github.com/VainF/DeepLabV3Plus-Pytorch

1) No composition should appear in more than one split; 2) Train/validation/test represents approximately 80/10/10% of the data set (over time). These proportions should be accurate globally and also within each composer. Maintaining these proportions is not always possible, as some composers have few compositions in the dataset; and 3) The validation and test splits should contain a variety of compositions. Modern compositions performed by many performers should be placed in the training split.

However, built models differ slightly from *Magenta (Music and Art Generation with Machine Intelligence)*[2] ones. Magenta is a research project exploring the role of machine learning in creating art and music. Primarily this involves developing new deep learning and reinforcement learning algorithms for generating songs, images, drawings, and other materials. But it's also an exploration in building smart tools and interfaces that allow artists and musicians to extend (not replace!) their processes using these models. Their model is too big for available GPU, so we decided to split it into four separate sub-models:

- Onsets - represent the time when a musical note begins.
- Offsets - represent (approximately) the length of time from the beginning of the track.
- Actives - represent the duration of each note over time.
- Volumes - represent how loud the note sounds.

Step 1. Model Configuration

For model configuration, which is presented in Figure 1, we used the approach from (*Hawthorne et al. 2017)* by using the *Librosa*[3] library (*McFee et al. 2015)* for computing the exact input data representation of mel-scaled spectrograms with log amplitude of the input raw audio with 229 logarithmically spaced frequency bins, a hop length of 512, an FFT window of 2048, and a sample rate of 16kHz.

Step 2. Piano Transcription and MIDI Preprocessing

The piano transcription follows a similar approach as (*Hawthorne et al.*

[2] Magenta: https://github.com/magenta/magenta
[3] Librosa, https://librosa.org/doc/latest/index.html

2018), with the difference that we switched to HTK frequency spacing (*Young, S. et al. 2006)* for the Mel-frequency spectrogram input.

We first translate "sustain pedal" control changes into longer note durations for MIDI preprocessing. If a note is active when sustain goes on, that note will be extended until either sustain goes off or the same note is played again. Regarding the model configuration, all onsets will span exactly two frames. Labeling only the frame containing the exact beginning of the onset does not work because of possible miss-alignments of the audio and labels. We experimented with requiring a minimum amount of time a note had to be present in a frame before it was labeled but found that the optimum value included any presence.

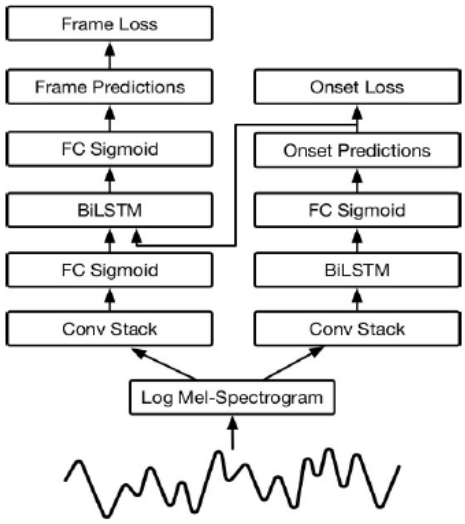


Figure 1. Network Architecture

Step 3. Hyperparameter Tunning

We have set up the number of time-frames for 5 seconds at the 16 kHz sample rate. Similar to [3], we split the training audio into smaller files. However, when we do this splitting, we do not want to cut the audio during notes because the onset detector would miss an onset while the frame detector would still need to predict the note's presence.

Step 4. Training and Validation

The model's logic is based on Google's [3] approach. The parameters are the

same with the following exceptions: 1) Batch normalization is used whenever possible (everywhere except the LSTM layer and the last fully-connected layer); 2) Dropout is not necessary at all, as there is no sign of overfitting; 3) It was impossible to keep the recommended batch size of 8 on our custom implementation, so I increased it to 64; 4) The model is divided into four .hdf5 files: Training Onsets Model, Training Offsets Model, Training Actives Model, Training Volumes Model.

We have used Frame-Based Accuracy Metric as a validation metric, which is stricter than F1-score. The standard accuracy is not representative, it would be almost 50% right at the beginning, and obviously, it would increase to 99% very quickly. The F1 score is better, but stricter accuracy is (*Dixon & Cambouropoulos 2000)*.

$$Acc = \frac{\sum_n Ntp[n]}{\sum_n Nfp[n] + \sum_n Nfn[n] + \sum_n Ntp[n]} =$$

$$= \frac{tp}{fp + fn + tp} = \frac{tp}{(pp - tp) + (rp - tp) + tp} = \frac{tp}{pp + rp - tp}$$

Specific Contributions in Model's Design

The onset detector is composed of the acoustic model, followed by a bidirectional LSTM with 128 units in both the forward and backward directions, followed by a fully connected sigmoid layer with 88 outputs representing the probability of onset for each of the 88 piano keys. Still, the convolutional acoustic stack is from (Kelz et al. 2016), which means a state-of-the-art model.

In our case, there is a bidirectional LSTM layer between the last two fully-connected layers, batch normalization is intensively used, dropout is not necessary, and the remaining parameters are from [5] with available implementation in GitHub4. Specifically, we increased the size of the

4 https://github.com/tensorflow/magenta/blob/master/magenta/models/onsets_frames_transcription/model.py

bidirectional LSTM layers from 128 to 256 units, changed the number of filters in the convolutional layers from 32/32/64 to 48/48/96, and increased the units in the fully-connected layer from 512 to 768. In terms of the batch, we used the size 64 for onsets, offsets, frame subnets, and 128 for CNN (convolutional neural network or ConvNet) volumes, instead of size eight as implemented by [3]. The batch size represents the number of samples processed before the model is updated. The number of epochs is the number of complete passes through the training data set. The batch size must be greater than or equal to 1 and less than or equal to the number of samples in the training data set.

4. Experimental Results

Dataset Description

The dataset consists of pairs (wave, midi) from the MAESTRO V1.0.0 archive, downloaded from Google Magenta [5]. The dataset archive contains 1184 music performances divided into .wav and .midi files that have a total size of 125 GB. Given the fact that the size of the files in the dataset and those that will be created after data processing is considerable, for quick training, we chose the MAESTRO dataset from the folder named "2015" that contains waveforms and MIDI representations for 43 pieces of piano recorded in the year 2015.

Numerical Accuracy Results

We have used LSTM as a recurring layer that only supports the Tensorflow backend, and we have used Adam optimizer as a stochastic optimization method.

The initial training and validation graphics for Onsets are presented in Figure 2, and we can observe that the loss and accuracy reach a plateau at around ten epochs. The model accuracy slightly goes above 0.7 while the loss goes below 0.01, so there is no need for further training.

[5] Magenta, https://magenta.tensorflow.org/datasets/maestro#download

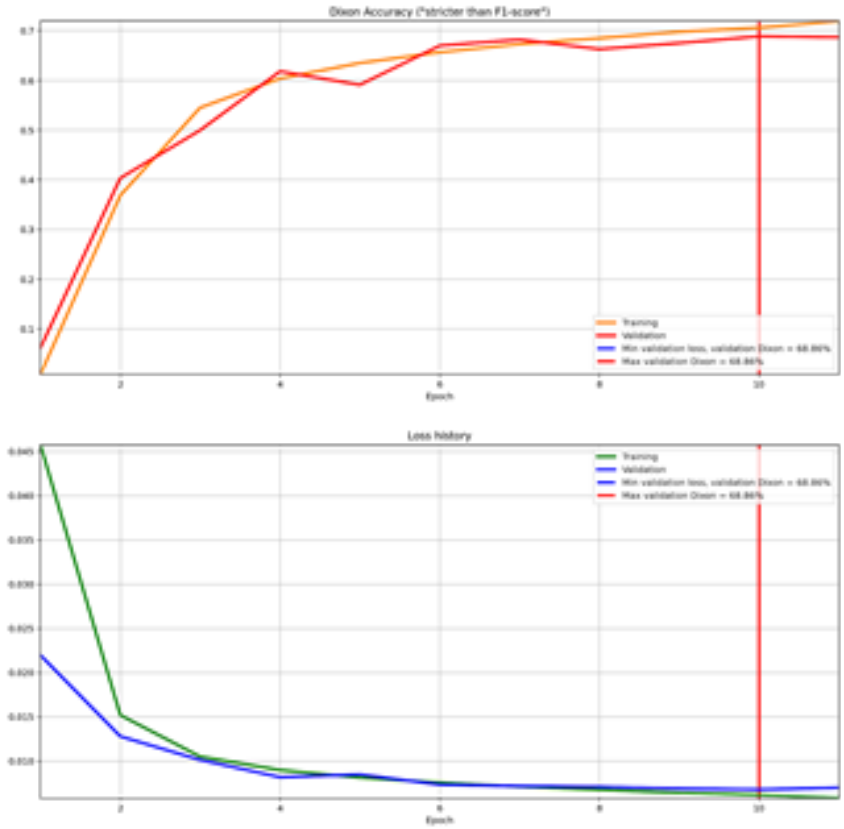


Figure 2. Training and validation for Onsets

We have added an offset detection head regarding the offsets for the RNN subnetwork. The offset head feeds into the frame detector but is not directly used during decoding. The offset labels are defined to be the 32ms following the end of each note.

Figure 3 presents the training and validation for offsets; in this case, the head of the offset fuels the frame detector, but it is not directly used while decoding. The offset tags are, in this case, the last 32 ms from the end of every note.

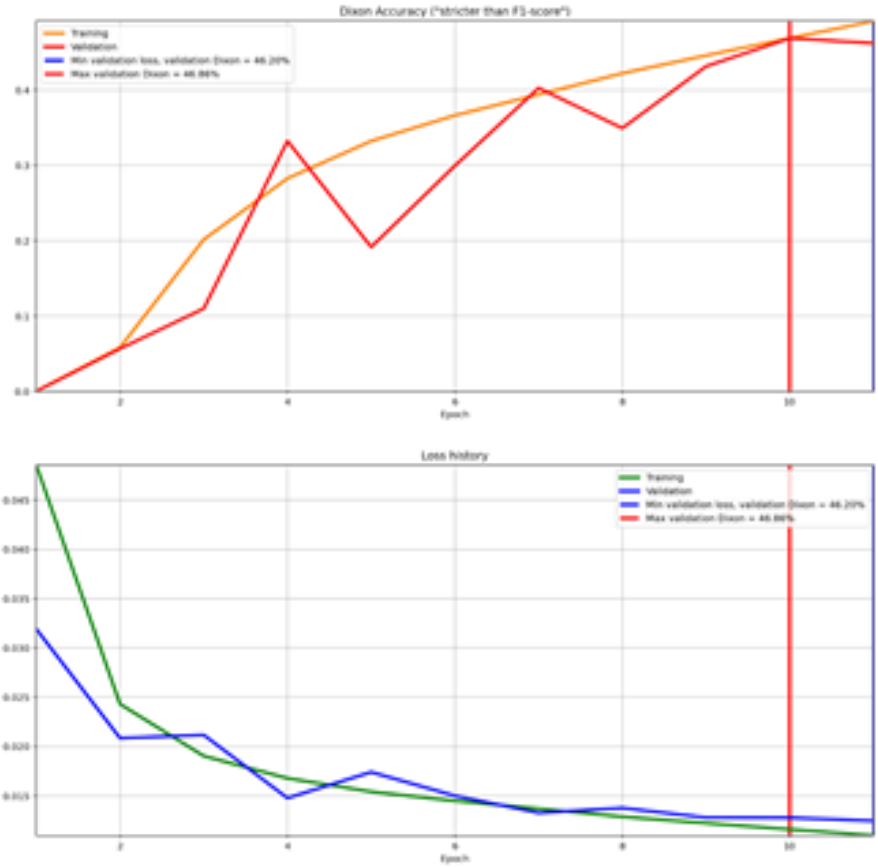


Figure 3. Training and validation for Offsets

Figure 4 presents the training for the activation detector, which is composed of a conventional CNN acoustic model with a sigmoid layer on top of it connected to 88 different outputs. We also used ten epochs for training until we reached a plateau.

Figure 5 presents the training and validation for volumes, and in this case, we considered going further than ten epochs as before have we reached a plateau around 17 epochs but still trained a total of 20 epochs.

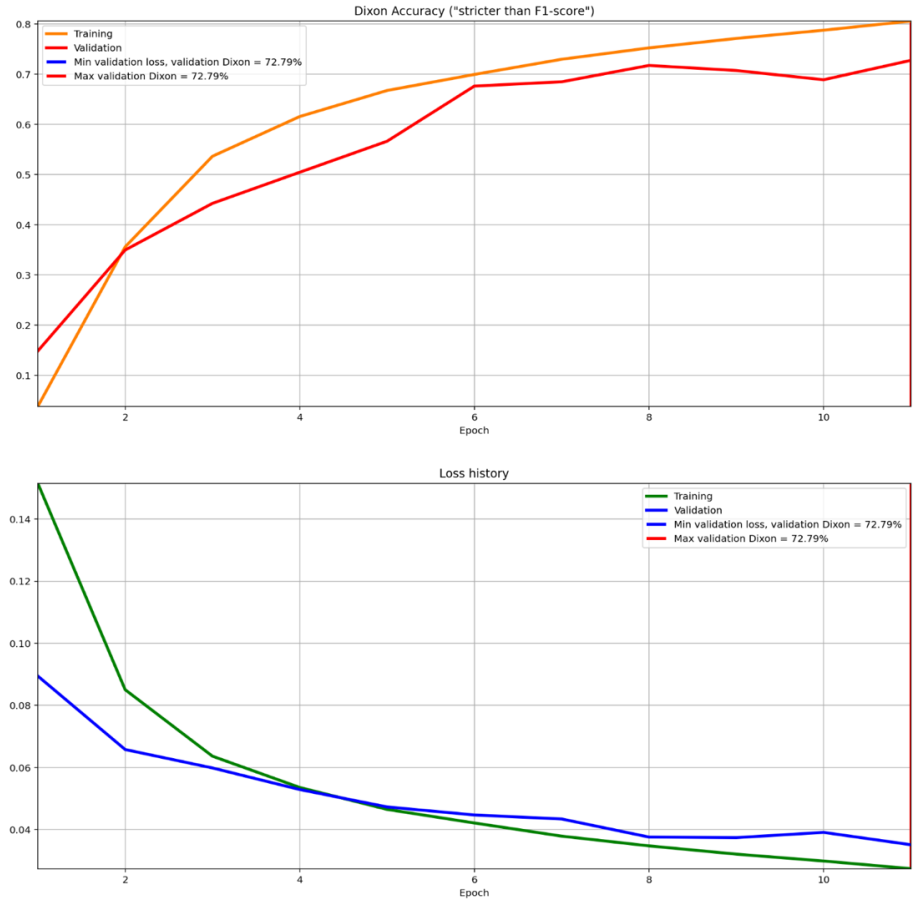


Figure 4. Training and validation for Actives

The frame activation detector comprises a separate acoustic model, followed by a fully connected sigmoid layer with 88 outputs. Its output is concatenated with the result of the onset detector and followed by a bidirectional LSTM with 128 units in both the forward and backward directions. Finally, the output of that LSTM is followed by a fully connected sigmoid layer with 88 outputs. During inference, we use a threshold of 0.5 to determine whether the onset detector or frame detector is active.

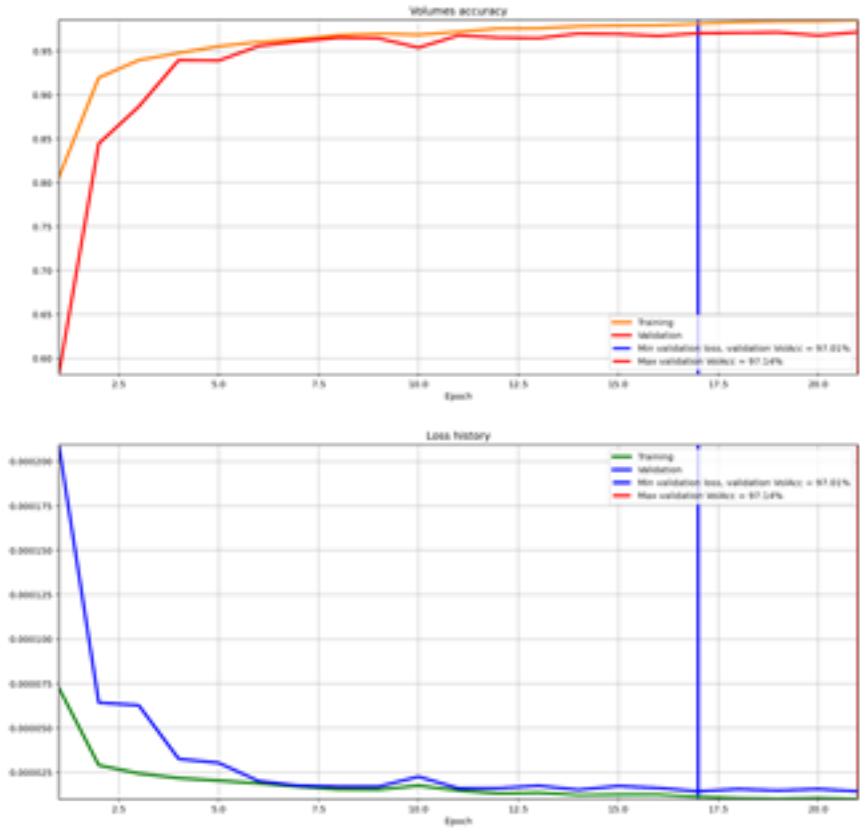


Figure 5. Training and validation for Volumes

The loss function is the sum of two cross-entropy losses: one from the onset side and one from the note side.

$$Ltotal=Lonset+Lframe$$

We decided to stop gradient propagation into the onset subnetwork from the frame network. We further extend the model by adding another stack to predict velocities for each onset. This stack is like the others and consists of the same layers of convolutions. This stack does not connect to the other two. The velocity labels are generated by dividing all the velocities by the maximum speed present in the piece. The lowest rate does not go to zero but rather to Vmin/Vmax. The stack is trained with the following loss averaged across a batch:

$$L_{vel} = \sum_{p=p_{min}}^{p_{max}} \sum_{t=0}^{T} \mathbf{I}_{onset}(p,t)(v_{label}^{p,t} - v_{predicted}^{p,t})^2$$

At inference time, the output is clipped to [0,1] and then transformed to a midi velocity by the following mapping:

$$vmidi=80 \cdot vpredicted+10$$

Firstly, we resize all ground-truth velocities in a transcript in the range [0, 1]. After notes are matched according to their pitch and onset/offset timing, we assemble pairs of the reference (ground-truth) and estimated velocities for compared notes, referred to as Vr and Ve, respectively. We then perform a linear regression to estimate a global scale and offset parameter such that the squared difference between pairs of reference and estimated velocities is minimized:

$$m, b = \arg\min_{m,b} \sum_{i=1}^{M} \|v_r(i) - (mv_e(i) + b)\|^2$$

Where M is the number of matches (i.e. number of entries in Vr and Ve). These scalar parameters are used to rescale the entries of Ve to obtain

$$\hat{v}_e = \{mv_e(i) + b, i \in 1, \ldots, M\}$$

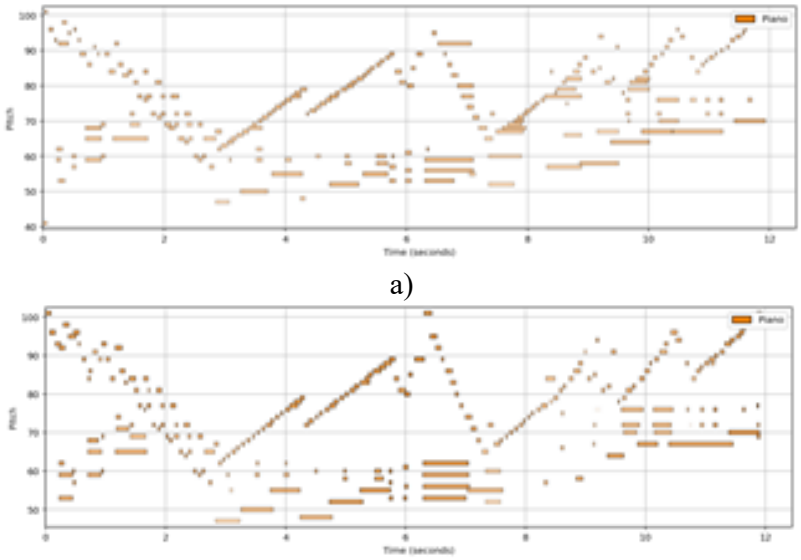
Finally, a match i is now only considered correct if, in addition to having its pitch and timing match, it satisfies Vi-Vri $< \tau$ for some threshold. We used τ=0.1 in all of our evaluations. The precision, recall, and F1 scores are then recomputed based on this new filtered list of matches.

Computing the total precision, recall, and F1 score has been done by using the mir eval library.

We compute two versions of the metrics: one that requires that onsets to be ± 50 ms from ground truth but ignoring the offsets, and one that requires that balances to be obtained as the duration of notes that are 20% off ground truth or 50 ms, whichever is greater.

Both scores of frames and notes are determined by play, and the average value of scores is presented as a final metric for a collection of plays.

The transcripts of poor quality may also lead to high frame scores due to false and repeated notes that should be in a continuous message. For the human ear, the decrease in transcription quality is best emphasized by the note-with-offsets scores.



a)

b)

Figure 6. Transcription example for "Abegg Variations, Op. 1 (Robert Schumann)" from the MAESTRO dataset.

In Figure 6, we present the transcription example for recording "Abegg Variations, Op. 1 (Robert Schumann)" from the MAESTRO dataset. The a) part present the post-processed output of the transcription-prediction system and b) presents the pitch ground truth of the recording.

Conclusions

This work focuses on the A2M (Audio to Midi) task, a hardly explored formulation consisting of building the MIDI representation from an audio file. The output sequence format is intended to be further processed by a computer.

We applied two deep learning methods: LSTM and CNN (a separate network) for music transcription. In general, the prediction accuracy is very promising given that the used data set is small and the neural network is not very deep. The trained model gives exact transcriptions for the piano sounds because the model was trained on a dataset that contains only piano compositions. But there are noticeable false-positive notes in high octaves where they surely should not be.

It also manages to transcribe audio files containing the sounds of many musical instruments and the piano, but the accuracy is lower. To further increase the accuracy of our music transcription, the neural networks could be trained on a larger dataset, including more complex music pieces such as orchestras.

References

BShakhovsky Polyphonic Piano Transcription: Recurrent Neural Network for generating piano MIDI-files from audio (MP3, WAV, etc.). (2021, May) Available: (https://github.com/BShakhovsky/PolyphonicPianoTranscription)

Dixon, S., & Cambouropoulos, E. (2000, August). Beat tracking with musical knowledge. In ECAI (pp. 626-630).

D. G. Morin (2017), Deep neural networks for piano music transcription. Available: https://github.com/diegomorin8/Deep-Neural-Networks-for- Piano-Music-Transcription.)

Poliner, G. E., & Ellis, D. P. (2006). A discriminative model for a polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007, 1-9.

Kelz, R., Dorfer, M., Korzeniowski, F., Böck, S., Arzt, A., & Widmer, G. (2016). On the potential of simple framewise approaches to piano transcription. arXiv preprint arXiv:1612.05153.

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015, July). librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference* (Vol. 8, pp. 18-25).

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., ... & Eck, D. (2017). Onsets and frames: Dual-objective piano transcription. arXiv preprint arXiv:1710.11153.

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C. Z. A., Dieleman, S., ... & Eck, D. (2018). Enabling factorized piano music modeling and generation with the MAESTRO dataset. arXiv preprint arXiv:1810.12247.

Piszczalski, M., & Galler, B. A. (1977). Automatic music transcription. *Computer Music Journal*, 24-31.

Rothstein, J. (1992). MIDI: A comprehensive introduction (Vol. 7). AR Editions, Inc.

Young, S. (2006) et al., *the HTK Book V3. 4* Cambridge University Press. Cambridge UK.

Wu, Y. T., Chen, B., & Su, L. (2019, May). Polyphonic music transcription with semantic segmentation. In ICASSP 2019-2019 *IEEE International Conference on Acoustics, Speech, and Signal Processing* (ICASSP) (pp. 166-170). IEEE.