Expert Systems with Applications 162 (2020) 113769

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Data representations for audio-to-score monophonic music transcription

Miguel A. Román [*], Antonio Pertusa, Jorge Calvo-Zaragoza

U.I. for Computing Research, University of Alicante, Carretera San Vicente del Raspeig s/n, 03690 Alicante, Spain

ARTICLE INFO

Article history:
Received 7 April 2020
Revised 24 June 2020
Accepted 15 July 2020
Available online 26 July 2020

Keywords:
Automatic music transcription
Audio processing
Neural networks
Audio to score
Monophonic music

ABSTRACT

This work presents an end-to-end method based on deep neural networks for audio-to-score music transcription of monophonic excerpts. Unlike existing music transcription methods, which normally perform pitch estimation, the proposed approach is formulated as an end-to-end task that outputs a notation-level music score. Using an audio file as input, modeled as a sequence of frames, a deep neural network is trained to provide a sequence of music symbols encoding a score, including key and time signatures, barlines, notes (with their pitch spelling and duration) and rests. Our framework is based on a Convolutional Recurrent Neural Network (CRNN) with Connectionist Temporal Classification (CTC) loss function trained in an end-to-end fashion, without requiring to align the input frames with the output symbols. A total of 246,870 incipits from the Répertoire International des Sources Musicales online catalog were synthesized using different timbres and tempos to build the training data. Alternative input representations (raw audio, Short-Time Fourier Transform (STFT), log-spaced STFT and Constant-Q transform) were evaluated for this task, as well as different output representations (Plaine & Easie Code, Kern, and a purpose-designed output). Results show that it is feasible to directly infer score representations from audio files and most errors come from music notation ambiguities and metering (time signatures and barlines).

1. Introduction

Automatic music transcription (AMT) is a music information retrieval (MIR) task whose final goal is to extract a human-readable and interpretable representation (i.e., a music score), from an audio signal. A music score is a guide to perform a piece of music, and it can be represented in different ways. The most extended representation is the modern staff notation used in common Western music, which can be encoded as symbolic data for computational purposes, namely a digital music score. Precise high-level musical information can be extracted from digital music scores, making them suitable to process, retrieve and classify music content (Corrêa & Rodrigues, 2016). There are different digital music score formats such as Plaine & Easie Code (Brook, 1965), Kern (Sapp, 2005) or MusicXML (Good, 2001), among others.

The most obvious application of AMT is to help a musician write down the music notation of a performance from its audio recording, which is a time-consuming task when it is done by hand. Additionally, AMT can also be useful for other MIR tasks, such as plagiarism detection, artist identification, genre classification, real-time score following, music style transfer or accompaniment generation. In general, AMT enables symbolic music algorithms to analyze recorded music.

In a modern music score (Schobrun, 2005), sounds are represented by *notes* placed on a *staff*, which is made of 5 horizontal lines representing the flow of time. The vertical position of the note in the staff determines its pitch, and the shape of the note symbol determines its duration. Silence is represented by *rest* symbols, whose duration is also determined by its shape. A staff begins with a *clef* symbol, which defines the pitch reference for the written notes. After the clef, a *time signature* determines the metric information or rhythm, and a *key signature* determines which note pitches must be altered according to the tonality of the musical piece. Based on the time signature, notes and rests are divided in *measures* of equal duration using *barlines*. Notes whose duration go beyond the current measure are joined with another note of the same pitch in the next measure using a curved line named *tie*. However this is only the basics, as a modern music score usually contains other information to indicate changes in *dynamics* (i.e., volume of the played notes) and *tempo* (i.e., duration of the played notes).

In order to extract a modern score from an audio signal, it is necessary to estimate the notes (with their pitch and duration) and the rests (duration) together with other symbols intended to

[*] Corresponding author.
E-mail addresses: mroman@dlsi.ua.es (M.A. Román), pertusa@dlsi.ua.es (A. Pertusa), jcalvo@dlsi.ua.es (J. Calvo-Zaragoza).

https://doi.org/10.1016/j.eswa.2020.113769
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

aid the interpretation of the score, such as key signatures, metering information (time signature and barlines) and clefs. However, this is just the basic information since a score is also a mean of communication between the composer and the performer, in which the former expresses how the musical piece should be performed by the latter. Thus, a score also contains a myriad of symbols intended to affect the acoustic characteristics of the written notes (e.g., slurs, crescendo, rubato, pizzicato). The meaning of these symbols is vague and in some cases has changed over different periods (e.g., grace notes, tempo markings). As a score does not unequivocally represent a musical piece and vice versa, the output of any AMT system can only be an approximation of the score that a musician used to perform the recorded audio.

Experienced musicians are capable of writing down a score by listening to an audio recording, notwithstanding slight differences compared with the original score due to inherent notation ambiguities. Music students are initially trained by transcribing monophonic scores (only one note playing simultaneously) in melodic dictations. The polyphonic scenario (many notes playing simultaneously) is much more complex and even some expert musicians are not able to transcribe full scores correctly.

In this work, we propose to focus on a hardly explored formulation, namely the Audio-to-Score (A2S) task, consisting of extracting a full score from the audio file in an end-to-end fashion. To the best of our knowledge, addressing audio-to-score transcription in one step is only found in our preliminary work (Román, Pertusa, & Calvo-Zaragoza, 2018; Román, Pertusa, & Calvo-Zaragoza, 2019), and in Carvalho and Smaragdis (2017).

In Román et al. (2018), the AMT problem is formulated as an automatic speech recognition (ASR) problem. Using monophonic (i.e., only one note playing simultaneously) audio as input, and a sequence of symbols (analogously to the spoken words) as outputs enables the application of several methods that were originally developed for ASR. In particular, Román et al. (2018, 2019) adopted the Convolutional Recurrent Neural Network architecture (Shi, Bai, & Yao, 2017a), which is the same architecture employed by Deep Speech 2 (Amodei et al., 2015). The present approach extends the contents of Román et al. (2018) by training the model with a much larger dataset (246,870 vs 71,400), using different instruments for the synthesis of the audio excerpts (instead of only piano), evaluating different input and output representations, and analyzing the nature of errors.

The main goal of this work is to find the most adequate input and output representations for this problem, as well as to design a proper neural network architecture to tackle the A2S task from these data.

2. Background

Even though the final goal of AMT is to provide a valid score out of an audio recording, most authors working on this task focus on one intermediate goal (Benetos, Dixon, Duan, & Ewert, 2019), which is pitch estimation. AMT systems can be roughly classified into four categories (Benetos et al., 2019). (1) Frame-level AMT or pitch estimation outputs the fundamental frequencies present at all time frames of the input audio. This is usually performed in each frame independently, although contextual information is sometimes considered in a post-processing stage. (2) Note-level (or note tracking) AMT not only estimates pitches in each individual frame, but also connects pitch estimates over time, producing a piano-roll representation as the output similarly to frame-level methods, but characterizing each note with pitch, onset and offset times. (3) Stream-level AMT goes one step further by clustering notes belonging to the same audio source characterized by its timbre. (4) Finally, notation-level AMT aims at the final goal of AMT, generating a human-readable score, which is the task addressed in the present work.

Regarding frame-level AMT, established algorithms such as SPICE (Gfeller et al., 2019) or YIN (de Cheveigné & Kawahara, 2002) and its variants (such as probabilistic YIN (Mauch & Dixon, 2014)) can obtain very reliable results for monophonic pitch estimation, which can be considered an almost solved task. In consequence, most AMT researchers only target the polyphonic pitch estimation problem, which is an extremely challenging task that still leaves room for improvement. Polyphonic pitch estimation methods include signal processing based techniques (Su & Yang, 2015), Negative Matrix Factorization (NMF) (Benetos & Dixon, 2013) and neural networks (Kelz et al., 2016) among others.

For note-level AMT, deep neural networks achieve state of the art results (Hawthorne et al., 2018; Kim & Bello, 2019; Ycart, McLeod, Benetos, & Yoshii, 2019a; Ycart, Stoller, & Benetos, 2019b), particularly using Convolutional Neural Networks (CNN) to output a piano-roll representation.

At present, the main focus of AMT research is on note-level AMT, still an open problem. For this reason, there are very few works (Arora & Behera, 2015) addressing stream-level AMT, which is closely related to instrument source separation.

Likewise, few researchers address notation-level AMT, which is the subject of this work. Regarding polyphonic sources, Nakamura, Benetos, Yoshii, and Dixon (2018a) uses a pipeline of multi-pitch estimation, note tracking, rhythm quantization, and score typesetting for polyphonic music. The results are still unsatisfactory according to the authors, who also suggest the integration of a music language model to improve the metrics. Cogliati, Temperley, and Duan (2016) approaches the problem by focusing on rhythm quantization of MIDI files, but do not tackle AMT end to end. The first work that formulates notation-level AMT as a sequence-to-sequence translation is Carvalho and Smaragdis (2017), but they intend to show that a neural network is capable of learning the task and the nature of the experiments is still very simple (i.e., note duration is constant, all samples limited to one measure of 4/4 metering).

Nonetheless, to the best of our knowledge there is no previous work addressing monophonic notation-level AMT other than Román et al. (2018). Although frame-level pitch estimation can be accurately performed on monophonic audio, the task of obtaining a full monophonic score has not received attention from the community. As a matter of fact, the result of a monophonic pitch estimation or pitch tracking method, which shows the evolution of pitches over time, is not useful in most situations since it lacks musical meaning, whereas obtaining a complete score from a singer or a monophonic instrument would be very useful for musicians and music students. For example, jazz trumpet players could use a monophonic notation-level AMT system to extract the score out of their recorded performances, which very often include solo improvisations never played or written beforehand.

Although monophonic pitch estimation is a relatively simple task, the extraction of the full score out of the pitches is not trivial. Problems such as rhythmic quantization (i.e., note and rest durations, barline locations, time signature estimation) and pitch spelling (i.e., G♯ vs. A♭) are still open, notably if we consider the real duration of note symbols (e.g., ♩ 𝅗𝅥 ♪) is never the same from one composition to the other, and even from one performance to the other. Moreover, there can be different ways to represent the same audio content using the western score notation. These intrinsic challenges of monophonic notation-level AMT are common to polyphonic notation-level AMT. Hence, some of the lessons learned in this work can be useful for future notation-level AMT works addressing polyphonic music.

Notation-level transcription can be divided into various subtasks, including pitch estimation, note tracking, pitch spelling, instrument recognition, rhythmic quantization and extraction of music dynamics. Each of these sub-tasks is very challenging on its own, even for trained musicians, leading many researchers to address AMT only as a frame or stream level estimation problem (Cemgil, Kappen, & Barber, 2003; Benetos et al., 2019).

As shown in Fig. 1, recent AMT methods that produce a notation syntax as output usually split the task into two main stages: they first convert the audio signal into the piano-roll representation, and in a second stage, they convert the estimated piano-roll into the final musical notation. Commercial software like Audioscore[1] follows this two-stage approach. In this case the first stage is fully automatic, but the second stage requires human intervention to set the time signature and barlines manually. Nonetheless, this approach has some drawbacks, since a wrong detection in a given stage (such as the multi-pitch detection) can be propagated through the next processing stages (Román et al., 2019). This is the main motivation to use end-to-end methods, where pitches can be better predicted together with the time information and dynamics, and vice versa, thanks to the underlying music language model that can be learned from the training data.

All notation-level AMT methods, including both monophonic and polyphonic, suffer from two key limitations that hinder its progress. The first one is the scarcity of quality datasets. Even though there are many audio recordings with their corresponding scores, they are not aligned and, since they can not be split, samples are usually very long to fit the training of deep neural networks. Unlike Automatic Speech Recognition (ASR), there is no trivial way to break a continuous musical piece into smaller excerpts based on the score. The second key limitation is the lack of proper evaluation metrics that can measure how close two scores are from each other, apart from a character by character comparison. There are some attempts to develop evaluation metrics for notation-level AMT (e.g., (McLeod & Steedman, 2018)), but they are still far from becoming a *de facto* standard that allows a proper comparison between different methods.

3. Data acquisition

As previously mentioned, one of the key limitations of notation-level AMT is the lack of quality datasets for training neural networks, namely a vast collection of musical excerpts of real audio data together with the corresponding aligned score in a textual format (e.g., MusicXML).

Our end-to-end approach has the advantage of not requiring audio frames aligned with the corresponding score notation symbol, which allows us to build our own dataset from existing digital scores by synthesizing the associated audio. Synthetic data is not optimal for training neural networks but it has been found an effective and inexpensive alternative in the absence of real data (Tremblay et al., 2018).

Fig. 2 outlines the pipeline we developed to create our ground truth. Our data source is the Répertoire International des Sources Musicales (RISM) (RISM, 2017) online catalog, which contains more than 1 million music incipits, encoded in Plaine& Easie Code (Brook, 1965) more commonly known as PAE format. In the *Data Extraction* block, we select a subset of more than 300.000 incipits and extract the PAE contents along with other metadata such as clef, key signature, time signature, and target instrument. The next two blocks in the pipeline are the *Score Parsing* block and the *Audio Synthesis* block. The former removes multi-rests and grace notes, which will not be considered for this work, and discards scores with invalid format and with incorrect measure lengths. The latter converts the resulting PAE files into MIDI by sequentially running the tools `pae2kern` and `hum2mid` available at the humdrum extra toolkit (Sapp, 2013).

The General MIDI program was based on the instrument metadata extracted from RISM and a random tempo between 80 and 120 is chosen to ensure variability in the note duration. Audio files were synthesized from MIDI files via FluidSynth with the default sound font (FluidR3 GM Bank) at 22,050 Hz sampling rate and compressed to mono FLAC audio format to reduce disk space while preserving the audio content in a lossless manner.

After this initial processing stage, we ended up with 313,493 samples of FLAC (input) and PAE (output) files, which will form our training data. Aiming at curating the dataset, the *Filtering block* in Fig. 2 filters out those samples that do not have their clef in [G2, F4, C1, C3, C4], their key signature in [0–5 sharps, 0–5 flats], their time signature in [4/4, 2/2, 3/4, 2/4, 3/8, 6/8, 9/8, 12/8, 6/4, 4/2, 3/2], their note and rest durations in [1, 1/2, 1/4, 1/8, 1/16, 1/32] (including dotted versions), or their note pitches in the [C2-B6] range. The samples removed by this filter are music notation outliers, extremely rare music scores that are not sufficiently represented in our data.

Additionally, we discarded samples containing double dots, double flats, double sharps, measure repetitions, and PAE rhythmic patterns, as well as samples that change the initial clef/key/time signatures. And lastly, we also discarded audio samples longer than 18 s and those with a high number of output symbols per second, to guarantee a minimum 1:1 ratio between input frames and output symbols required for CTC loss function.

After this final filtering stage, we ended up with 246,870 samples that were randomly divided into training (70%), validation (15%) and test (15%) sets. The same sets were used for all the experiments to ensure results can be compared across all input/output representations and models.

4. Data representation

Under this end-to-end scenario, the input of the proposed model is a sequence of audio frames, and the output is a sequence of text characters representing the score. Given the nature of our approach, we intend to analyze the impact of the data representation with regards to both input and output formats. In the following sections, we describe the different alternatives that we have considered.

4.1. Inputs

All input representations were obtained from an audio waveform sampled at 22,050 Hz, and they apply the same window stride to ensure that input sequences have equal length regardless of its representation.

- *Raw audio*. Raw waveforms were directly used as input by adding SincNet (Ravanelli & Bengio, 2018) as the first layer of the model. SincNet layer extracts audio features from waveforms by discovering meaningful filters for our problem. SincNet is based on parameterized 'sinc' functions, which implement band-pass filters whose coefficients (low and high cutoff frequencies) are learned during training. A set of 144 filters were used with a window length of 2048 samples and a window stride of 512 samples. A Hamming window was selected to control spectral leakage after applying the filters. The initial filter coefficients were initialized with Mel-scaled bins across the valid frequency range (from 0 to Nyquist).
- *STFT*. The magnitude spectrogram from the Short Time Fourier Transform was also evaluated, using a Hamming window of 2048 samples and hop size of 512 samples.

[1] https://www.neuratron.com/audioscore.htm.

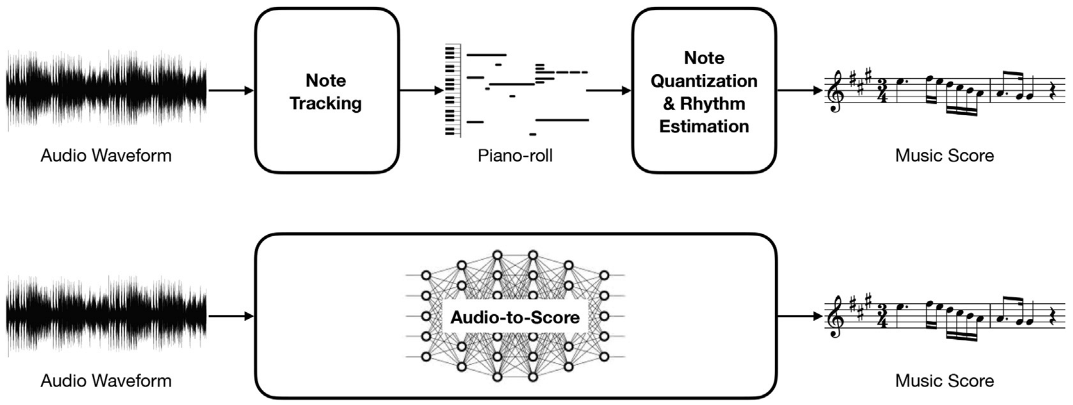


Fig. 1. Overview of notation-level automatic music transcription methods. Top: two-stage methods based on piano-roll estimation. Bottom: end-to-end methods based on deep neural networks and the object of this study.

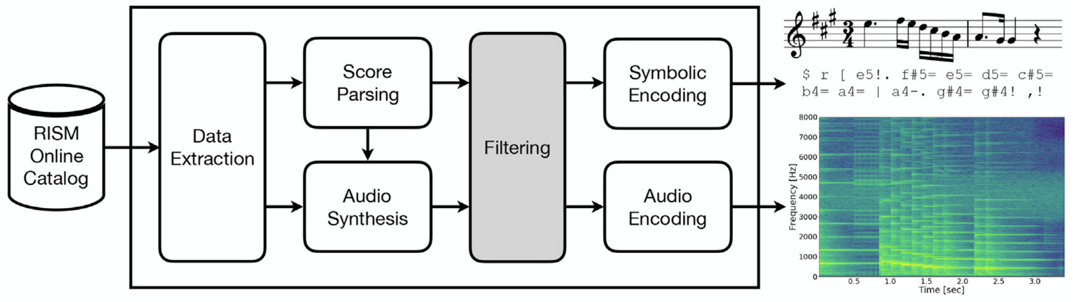


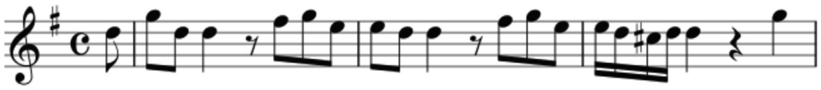
Fig. 2. Training data acquisition pipeline.

- *LogSTFT*. A log-spaced filter bank was applied to the previous STFT to get 24 bins per octave, starting from C2 and extending across 6 octaves.
- *CQT*. Constant-Q transform (Schörkhuber & Klapuri, 2010) spectrogram, using 24 bins per octave, starting from C2 and extending across 6 octaves. In this type of spectrogram, the window size is variable in order to guarantee the same resolution in the frequency axis at the cost of penalizing the resolution in the time axis. However, the hop size is constant and set to 512 samples in order to ensure the same number of frames as the previous representations.

4.2. Outputs

Different output representations were obtained from the original input incipits encoded in standard PAE format. Fig. 3 shows an example of a score excerpt and how it was encoded to the 3 output representations we considered.

- *CTC-friendly*. This is a semantic notation we created to ease Connectionist Temporal Classification (CTC) decoding, explained in more detail in the following section. It is formed by a sequence of words, separated by spaces, of one of these types: clef, key signature, time signature, barline, rest, note, tie. A single-character word was used to encode clef, key signature, time signature, barline and tie. Multiple characters were used to encode notes, including the pitch spelling (e.g., a♯4, c2, b♭3) followed by one character to encode the canonical duration, plus the standard notation dot character to extend the duration when needed. Explicit sharp and flat notes were encoded regardless of key signature or measure accidentals. Rests were encoded using one special symbol followed by the same duration encoding used for notes. The alphabet size is 52 characters, and the average word size is 2.53 characters.
- *PAE-based*. This notation is based on the PAE (Brook, 1965) format with the following changes to match the requirement of our output sequences: (1) clef, key signature and time signature were encoded following the PAE format and placed at the beginning of the incipit without their special characters and always in the same order; (2) notes, rests, ties, and barlines were split by spaces; (3) note/rest modifiers (i.e., octave, duration, accidental) were placed right before the first note or rest that they affect to; (4) note modifiers always follow this precedence: duration, octave, accidental; (5) there are implicit sharp and flat notes based on the key signature and measure-bound accidentals. The alphabet size is 24 characters and the average word size is 1.87 characters.
- *Kern-based*. This notation is based on the Humdrum (Huron, 1995) Kern format, which represents music in lines of tab-separated columns. Each line represents an instant of time where one or more note onsets exist, whereas each column, or spine in Humdrum's nomenclature, represents a different pentagram or voice in a score. As we are dealing with monophonic excerpts, our kern files have one spine only. We take advantage of this fact to create a one-line representation of the score by replacing the carriage return characters with spaces. Moreover, we make the following changes to match our output sequence format: (1) clef, key signature and time signature were encoded as described in the previous PAE format section; (2) comment records and interpretation records were removed; (3) middle notes of ties do not have any special character. Only the initial and final tie characters (i.e., '[' and ']')



CTC-friendly encoding

```
$ t ( d5- | g5- d5- d5! ,- f#5- g5- e5- | e5- d5- d5! ,- f#5- g5- e5- |
e5= d5= c#5= d5= d5! ,! g5! |
```

PAE-based encoding

```
G-2 xF 4/4 8''D / G D 4D 8- F G E / E D 4D 8- F G E / 6E D xC D 4D - G /
```

Kern-based encoding

```
G-2 xF 4/4 8dd = 8gg 8dd 4dd 8r 8ff# 8gg 8ee = 8ee 8dd 4dd 8r 8ff# 8gg 8ee =
16ee 16dd 16cc# 16dd 4dd 4r 4gg =
```

Fig. 3. Example of score encoding to the output representations we considered.

were considered and they were placed next to the affected notes; (4) there are explicit sharp and flat notes regardless of key signature or measure-bound accidentals. The alphabet size is 31 characters and the average word size is 2.64 characters.

5. Method

We built our architecture based on Román et al. (2018), which is an improved version of the Convolutional Recurrent Neural Network (CRNN) trained with Connectionist Temporal Classification (CTC) loss function.

Our notation-level AMT approach aims at estimating which music score, modeled as a structure containing symbols from a fixed alphabet of music notation, would define the input audio most likely. This is what we denote by the Audio-to-Score (A2S) task. Note that A2S resembles what a human would expect to get if it intends to visualize an audio file as a music score (e.g., MusicXML), unlike traditional AMT where the output is a construction intended to be further processed by a computer (e.g., piano-roll, MIDI, etc.).

Let $\mathscr{X}$ be the domain of audio files and Σ the alphabet of music score symbols. The aim of our A2S is to compute a function that maps any audio file into a sequence of symbols, i.e., a function $f : \mathscr{X} \rightarrow \Sigma^*$. The A2S task is therefore formulated as retrieving the most likely sequence of music symbols $\hat{\mathbf{s}}$ given an $\mathbf{x} \in \mathscr{X}$:

$$\hat{\mathbf{s}} = \arg \max_{\mathbf{s}\in\Sigma^*} P(\mathbf{s}|\mathbf{x}) \tag{1}$$

Following successful approaches in other sequence labeling tasks, we address the A2S with an end-to-end approach based on statistical models. For learning the posterior probability of Eq. (1), we resort to Convolutional Recurrent Neural Networks (CRNN).

A CRNN consists of one block of *convolutional* layers connected to a block of *recurrent* layers (Shi, Bai, & Yao, 2017b). Each convolutional layer usually includes a max-pooling layer (or filters with a stride greater than 1) that reduces the dimensionality of its output. The convolutional block is in charge of learning relevant features from the input and the recurrent layers process these features in terms of sequences of musical symbols.

Activations of the last convolutional layer can be seen as a sequence of feature vectors representing the input audio file, $\mathbf{x}$. Let M be the width of the input sequence $\mathbf{x}$. The length of the resulting features after the convolutional layer will be $J = \gamma M$, where $\gamma \leqslant 1$ is defined by the specific max-pooling parameters.

These activations are then fed to the first layer of the recurrent block, and the unit activations of the last recurrent layer are considered estimates of the posterior probabilities per frame:

$$P(\sigma|\mathbf{x},j), \quad 1 \leqslant j \leqslant J, \quad \sigma \in \Sigma \cup \{\epsilon\} \tag{2}$$

where ϵ is a special *"non-character"* symbol that is necessary to detect two or more consecutive instances of the same symbol (Graves, 2008).

5.1. Training

Convolutional neural networks can be trained through gradient descent using the well-known *Back Propagation* algorithm. RNN networks can be trained similarly using *Back Propagation Through Time* (Williams & Zipser, 1995). Therefore both the convolutional and recurrent blocks of a CRNN can be jointly trained by providing audio files annotated at the frame level.

In this work, we follow an end-to-end approach, which means that for each audio file we only provide its corresponding target transcription of score symbols, without any kind of explicit information about its segmentation into frames. A CRNN can be uniformly trained without this information by using the Connectionist Temporal Classification (CTC) loss function (Graves, Fernández, Gomez, & Schmidhuber, 2006). The CTC training procedure is a form of Expectation–Maximization, similar to the backward-forward algorithm used for HMM training (Rabiner & Juang, 1993), that distributes the loss among all the frames to locally maximize Eq. (1) with respect to the ground-truth sequence.

To improve convergence and prevent overfitting, Batch Normalization layers (Ioffe & Szegedy, 2015) are also added between any other layer excluding the input and output layers.

5.2. Decoding

For solving Eq. (1), the most likely symbol is computed for each input feature vector of the recurrent block j:

$$\hat{\sigma}_j = \arg \max_{\sigma\in\Sigma\cup\{\epsilon\}} P(\sigma|\mathbf{x},j), 1 \leqslant j \leqslant J \tag{3}$$

Then, an approximately optimal sequence of musical symbols is obtained as $\hat{\mathbf{s}} \approx \mathscr{F}(\hat{\sigma})$, where $\hat{\sigma} = \hat{\sigma}_1 \ldots \hat{\sigma}_J$ and $\mathscr{F} : \Sigma\prime^{\star} \rightarrow \Sigma^{\star}$ is a function which first merges all the consecutive characters such that $\hat{\sigma}_j = \hat{\sigma}_{j-1}$ and then deletes all the non-character symbols $(\sigma_j = \epsilon)$ (Graves et al., 2006). This is known as greedy decoding.

In our work, we replaced greedy decoding with CTC-based beam search decoding, which applies beam search algorithm to find the n-best paths with one particularity: when exploring new paths, if two or more of them lead to the same sequence (after merging consecutive characters and deleting the non-character symbol), they are grouped into a single path and their probabilities are added up. After all frames are decoded, the path with the highest probability is chosen as the output sequence.

5.3. Implementation details

In order to adapt the topology for the different input/output configurations and improve the accuracy, we made the following changes to the model described in Román et al. (2018).

First, we adapted the convolutional stage depending on the input representation to ensure the same number of features per frame in the recurrent stage, and we also reduced the size of the filters to increase the resolution in the frequency and time axis. Table 1 summarizes the convolutional stage hyperparameters we selected following this rationale.

In addition, zero-padding was applied not only at the input layer (to match the length of the largest sequence of the batch), but in the same manner at the output of all the layers of the convolutional stage.

For the raw waveform input representation only, we added an extra filter bank layer before the convolutional layers with learnable filter bands, based on Ravanelli and Bengio (2018), and changed 2D convolutions by 1D convolutions 5. The reason behind this change is to give freedom to the model to learn filters that might not be adjacent in the frequency axis.

Finally, we also reduced the number of recurrent layers from 3 to 2, increased the number of hidden units from 1,024 to 1,536 and replaced Gated Recurrent Units (GRU) with LSTM cells. We made all these changes after trying different values and finding a trade-off between reasonable performance and computational cost.

Table 1
Convolutional layers hyper-parameters based on input representation.

Input	Number of Filters	Kernel Size	Stride
Raw	2048, 2048	3, 3	1, 1
STFT	8, 8	$9 \times 3, 3 \times 3$	$2 \times 1, 1 \times 1$
LogSTFT	16, 16	$3 \times 3, 3 \times 3$	$1 \times 1, 1 \times 1$
CQT	16, 16	$3 \times 3, 3 \times 3$	$1 \times 1, 1 \times 1$

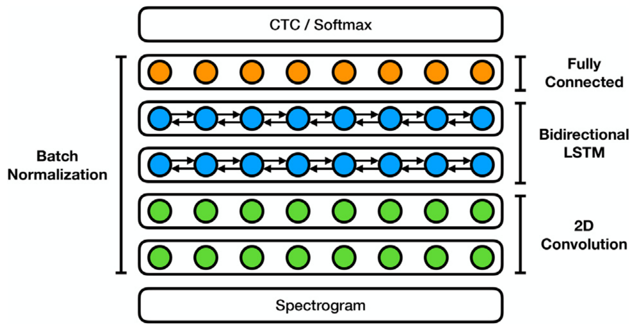


Fig. 4. CRNN with spectrogram input.

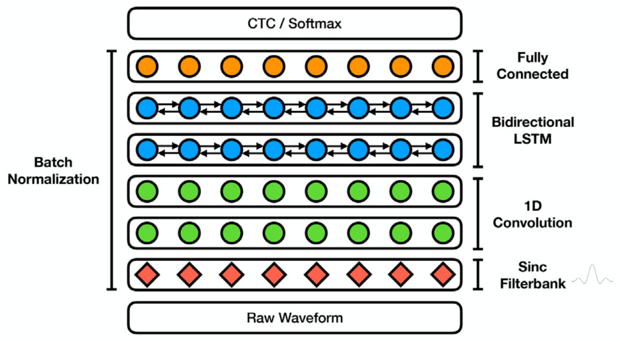


Fig. 5. CRNN with raw waveform input.

Table 2
Selected hyper-parameters for all the models.

Batch size	20
Epochs	25
Optimizer	SGD with Nesterov
Learning rate	3e-4
Momentum	0.9
Learning rate annealing	1.1

Fig. 4 depicts the high level CRNN architecture used for spectrogram-based input representations and 5 depicts its counterpart for raw waveform input representation.

In order to produce comparable results for all the training models, we used the same set of samples for the training, validation and test. With the same motivation, we set the same training hyperparameters summarized in Table 2.

6. Evaluation

6.1. Metrics

Unfortunately, there are not existing standard A2S evaluation metrics that can be adopted for notation-based methods, since this is still an open problem due to the complexity of the score notation. For example, a good metric should count score errors only once, hence if the time signature is wrongly predicted, barlines should not be counted as errors based on their ground truth location. Likewise, errors caused by music notation ambiguities should be dismissed, for example, a quarter dotted note should be equivalent to a quarter note tied to an eighth note.

Having this in mind, we borrowed the most common evaluation metrics from the ASR task, namely Word Error Rate (WER) and Character Error Rate (CER), to validate our results. As our output is also made by words separated by spaces, we can use these metrics effortlessly. WER is an indication of the number of words incorrectly predicted and is measured as the edit distance of the sequence of output words with respect to the ground truth. CER is an indication of the number of characters incorrectly predicted and is measured as the edit distance of the sequence of output characters (after removing the spaces) with respect to the ground truth. At training time, we keep the model with the lowest WER in the validation set, calculated after each epoch. We chose to minimize WER as it gives an indication of how many musical symbols (e.g., notes) should be manually edited by a musician to fix the output score, while CER does not provide a direct relationship between errors and musical symbols.

Additionally, we also used the evaluation metrics of Nakamura, Benetos, Yoshii, and Dixon (2018b) in order to provide a reference for AMT readers who are knowledgeable of existing note-level AMT methods, even though our work belongs to the notation-level AMT category. These evaluation metrics also provide an alternative way to compare our experiments beyond using WER and CER. The total number of notes in the ground truth is denoted by N_{GT}, that of estimated notes by N_{est}. The number of notes with pitch errors is denoted by N_p, that of extra notes by N_e, and that of missing notes by N_m. The number of matched notes is defined as $N_{match} = N_{GT} - N_m = N_{est} - N_e$. Then we define the pitch error rate as $E_p = N_p/N_{GT}$, extra note rate as $E_e = N_e/N_{est}$, and missing note rate as $E_m = N_m/N_{GT}$. Onset/offsets errors are also reported in Nakamura et al. (2018b). As we are dealing with note durations instead of onsets/offsets, we include an alternative error metric E_d which is calculated similarly to the pitch error E_p but using note duration errors, denoted by N_d. Thus, we define the duration error rate as $E_d = N_d / N_{GT}$.

6.2. Results

Table 3 shows the WER and CER metrics for all the combinations of input and output representations, measured on the test set using the CTC beam search decoding with 10 search paths. Concerning the input signal, it can be observed that spectrogram-based representations perform similarly, whereas a raw input representation provide significantly worse results. With respect to the output representation, CTC-friendly reports the best results on average, whereas both PAE and Kern-based output representations perform similarly, with a slightly worse performance. According to these results, the best combination is that of LogSTFT input with CTC-friendly output, that yields 9.96 and 3.15 of WER and CER, respectively.

The results of applying (Nakamura et al., 2018b) evaluation metrics as explained in the previous section are reported in Table 4. As can be noted, CTC-friendly output representations score the lowest in all error categories. However, the choice of the input representation hardly affects the magnitude of errors for any given output representation.

Overall, extra note errors (E_e) are very scarce in all scenarios, which is expected for monophonic transcription, and duration errors (E_d) are notably higher when considering rests as well as notes. PAE-based output representations exhibit the worst pitch error (E_p) and duration error (E_d), which makes sense considering the implicit duration, octave, and accidentals for most notes in this particular representation. On the other hand, Kern-based output representations score the worst missing notes errors (E_m), due to the higher number of symbols required to represent a note.

For a better comparison of the evaluated input/output representations, we group notation-level transcription errors into 8 categories that are shown in Table 5.

As can be seen, highest errors are related to *Barline* and *TimeSig*, which is expected as many incipits do not contain enough contextual information to properly predict them. We should notice here that audio is not synthesized with dynamic expression, thus without accents indicating strong and weak beats inside a measure, which also hampers the estimation of metric information. For spectrogram-based input representations, CQT has considerably higher *Barline* and *TimeSig* error rates, which indicates that the variable window size used to create this kind of spectrogram is negatively affecting the time-based predictions as expected (i.e., CQT improves frequency resolution at the expense of lower time resolution).

Focusing on the output representation, Kern-based has the highest *Format* and *Tie* errors, which suggests that its higher verbosity and the different way to represent ties (i.e., tie character is added to each note instead of being an independent word between notes) are more difficult to learn. On the other hand, CTC-friendly representation displays the lowest *Note* errors of all, while PAE-based again exhibits the highest *Note* errors, which is explained by the way it encodes notes with implicit modifiers (i.e., whenever a note modifier such as the change of duration is incorrectly predicted, it will very likely affect the prediction of multiple subsequent notes). Nonetheless, PAE-based output representation has the lowest *Format* errors, due to having the smallest alphabet size. Based on *Format* errors, the addition of a language model to the CTC decoding stage seems a very promising approach to increase accuracy.

Regarding *Clef*, *KeySig* and *TimeSig* errors, CTC-friendly generally shows better results, suggesting that the minimal character encoding for these symbols benefits its prediction accuracy. Overall, *KeySig* error rates are significantly lower than *Clef* error rates, hinting a higher correlation between note spellings and key signature than between note spellings and clef.

Fig. 6 depicts the distribution in absolute terms of transcription errors based on its category for each input/output representation. We notice a very similar distribution regardless of the input representation, being *Barline* and *TimeSig* the most prominent source of errors in all cases. *Note* errors are also high due to the fact that they are the most frequent symbol of every score. This figure also illustrates the highest proportion of *Note* errors for PAE-based representation and *Format*/*Tie* errors for Kern-based representation, as we previously noticed.

Finally, we have also measured WER and CER for different types of instruments using our best model (i.e., logSTFT and

Table 3
Word Error Rate (WER)/ Character Error Rate (CER) using different inputs and outputs. Best result is highlighted in bold text.

	Raw	STFT	LogSTFT	CQT	Average
CTC	12.18/ 3.94	10.29/ 3.24	**9.96**/ **3.15**	10.59/ 3.32	10.76 / 3.41
PAE	11.65/ 4.82	11.11/ 4.53	11.08/ 4.51	11.10/ 4.53	11.24/ 4.60
Kern	12.10/ 4.57	11.53/ 4.22	11.31 / 4.15	11.29/ 4.18	11.56/ 4.28
Average	11.98/ 4.44	10.98/ 4.00	10.78/ 3.94	10.99/ 4.01	

Table 4
Note-level transcription errors for different inputs and outputs. Best result is highlighted in bold text.

	E_p	E_m (+rests)	E_e (+rests)	E_d (+rests)
Raw + CTC	0.50	0.61 (0.67)	0.12 (0.13)	1.19 (1.49)
Raw + PAE	1.74	0.70 (0.78)	0.42 (0.40)	2.27 (2.56)
Raw + Kern	1.32	1.23 (1.34)	0.23 (0.22)	1.20 (1.46)
STFT + CTC	0.42	0.32 (0.35)	0.09 (0.09)	0.76 (1.06)
STFT + PAE	1.50	0.48 (0.52)	0.15 (0.14)	1.41 (1.72)
STFT + Kern	1.07	0.84 (0.90)	0.16 (0.15)	0.81 (1.10)
LogSTFT + CTC	0.44	**0.25 (0.28)**	0.09 (0.08)	**0.71 (1.04)**
LogSTFT + PAE	1.49	0.48 (0.53)	0.17 (0.16)	1.36 (1.70)
LogSTFT + Kern	1.12	0.86 (0.93)	0.16 (0.15)	0.82 (1.11)
CQT + CTC	**0.40**	0.38 (0.44)	**0.08 (0.08)**	0.78 (1.08)
CQT + PAE	1.53	0.47 (0.50)	0.15 (0.15)	1.52 (1.83)
CQT + Kern	1.16	0.73 (0.77)	0.15 (0.15)	0.84 (1.16)

Table 5
Transcription errors by category for different inputs and outputs. *Clef*, *KeySig* and *TimeSig* represent the prediction errors of the single word representing clef, key signature and time signature, respectively. *Note* shows the errors in the prediction of note pitch and duration, plus errors of missing and extra notes, whereas *Rest* represents the errors of rest duration, plus errors of missing and extra rests. *Barline* shows the errors in barline location, based on the predicted time signature. Finally, *Tie* accounts for invalid ties in the predicted score, and *Format* represents the formatting errors present in the output representation. Best result is highlighted in bold text.

	Clef	KeySig	TimeSig	Note	Rest	Barline	Tie	Format
Raw + CTC	19.93	13.38	38.07	2.41	5.74	32.27	10.03	0.79
Raw + PAE	21.20	14.53	38.47	5.12	7.07	31.60	5.84	0.38
Raw + Kern	22.53	14.39	38.92	3.96	6.58	33.38	43.93	1.25
STFT + CTC	**19.74**	13.74	37.99	1.58	**4.74**	28.93	11.97	0.51
STFT + PAE	21.91	13.31	38.50	3.53	5.59	30.50	**4.65**	0.38
STFT + Kern	21.02	14.03	39.74	2.88	5.46	34.65	39.59	0.93
LogSTFT + CTC	19.98	**13.26**	37.81	**1.48**	4.93	**28.04**	8.45	0.47
LogSTFT + PAE	21.48	13.59	38.15	3.50	6.08	29.91	4.74	0.35
LogSTFT + Kern	21.41	13.95	39.55	2.95	5.58	31.54	36.04	0.95
CQT + CTC	20.27	13.33	**36.74**	1.65	4.96	32.41	16.95	0.72
CQT + PAE	20.58	14.01	38.09	3.65	5.72	31.19	4.84	**0.33**
CQT + Kern	21.75	14.01	39.74	2.88	5.50	33.96	33.21	0.88

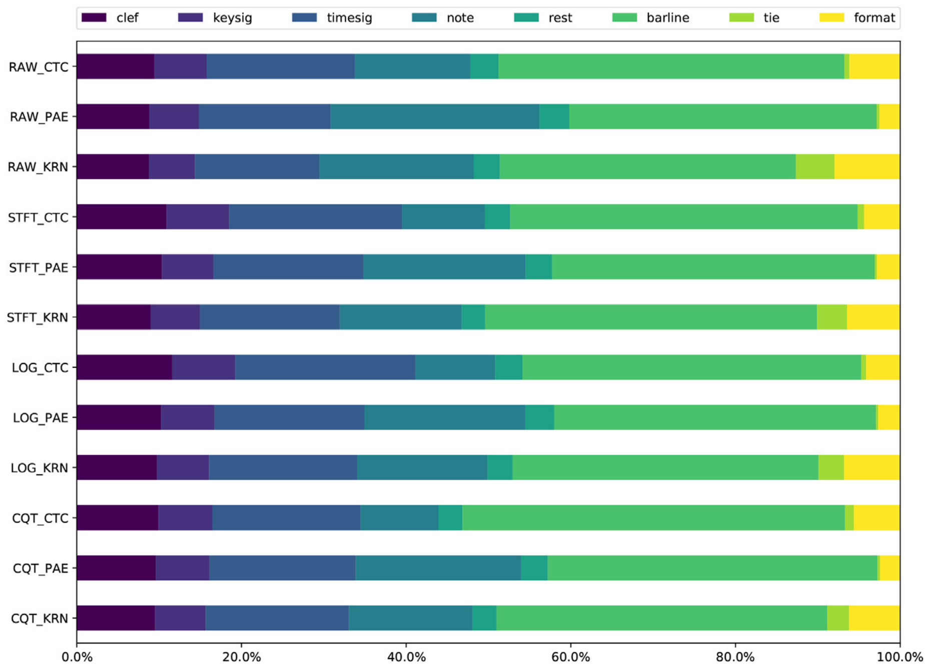


Fig. 6. Distribution of transcription errors by category for each input and output representation.

Table 6
WER and CER of test samples split by instrument type using the best model.

	WER	CER	Samples
Piano	10.20	3.26	20961
Harpichord	11.45	3.50	1508
Organ	12.05	4.04	2020
Strings	9.60	3.06	10293
Winds	11.08	3.65	1938

CTC-friendly), as shown in Table 6. These results reveal small differences in error rates, which suggests that the model can learn to transcribe multiple timbres with little effect on performance. No further comparative analysis can be derived from these results, since instruments are not equally represented in our test set.

7. Conclusions and future work

Although monophonic pitch estimation is considered an almost solved problem, producing a human-readable score from monophonic audio still presents some challenges we are addressing in this work. Our Audio-to-Score task goes one step beyond more traditional AMT systems based on pitch estimation, by also learning other musical information such as note duration, rests, clef, key signature, time signature, barlines, and ties.

In this work, we analyzed different input/output representations to perform notation-level AMT in an end-to-end manner using Convolutional Recurrent Neural Networks. For this, 246,870 incipits from RISM were synthesized using different timbres and random tempos, and the raw audio, STFT, LogSTFT and CQT were extracted to be used as input. To encode the music score in a textual format, PAE, Kern and a designed CTC-friendly language were considered. Different configurations of the CRNN were experimentally tuned for each input and output representation.

We tested this framework for all input and output combinations, performing a thorough error analysis to bring a comparative evaluation. The best results were obtained using logarithmic STFT as input and CTC-friendly as output format, which can be converted with simple symbol manipulations to Kern or PAE, making it a valid language to represent a music score. The comparative evaluation shows that using a proper representation of data can lead to shorter training cycles and lower error rates.

We classified prediction errors based on their musical meaning, i.e., clef, time signature, key signature, note pitch, note duration, rest, barline location, tie, and format errors. We observed that time-related symbols have the highest error rates, including barline location, time signature and note/rest duration, in that specific order.

In general, these results are promising to build a practical application for musicians that performs monophonic A2S music transcription, considering that approximately 10% of the predicted symbols require manual editing to yield the correct score. However, it is still not clear how these results would translate to polyphonic music. AMT for polyphonic music is still an open problem, but as Natural Language Processing thrived just by scaling up end-to-end language models, we believe AMT could experience a similar leap of progress once we procure sufficient amount of data and build music language models that can be trained with very long audio sequences.

Future work includes adding a language model to the decoder, which is expected to boost the performance by ensuring the output is syntactically and gramatically correct, thus making it suitable to render a valid graphic score with a music engraving software like Verovio (Swiss RISM Office, 2017). In addition, we plan to validate this framework with polyphonic audio, as well as to evaluate alternative neural network architectures such as *seq2seq* auto-regressive models with attention mechanisms, which perform better in the analogous speech recognition task by predicting output characters conditioned to those already predicted, while CTC-based models assume statistical independence of output characters. Moreover, auto-regressive models allow output sequences longer than its corresponding input sequences (Sutskever, Vinyals, & Le, 2014), a key advantage for polyphonic music that requires much more characters than monophonic music to output a score, with the same number of audio frames.

CRediT authorship contribution statement

Miguel A. Román: Software, Validation, Data curation, Investigation, Writing - original draft, Visualization. **Antonio Pertusa:** Conceptualization, Supervision, Writing - review & editing. **Jorge Calvo-Zaragoza:** Methodology, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been supported by the Spanish "Ministerio de Ciencia e Innovación" through Project HISPAMUS (No. TIN2017-86576-R supported by EU FEDER funds).

References

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A., Jun, B., LeGresley, P., Lin, L., & Zhu, Z. (2015). Deep speech 2: End-to-end speech recognition in english and mandarin. ArXiv e-prints.

Arora, V., & Behera, L. (2015). Multiple F0 estimation and source clustering of polyphonic music audio using PLCA and HMRFs. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP), 23*, 278–287.

Benetos, E., & Dixon, S. (2013). Multiple-instrument polyphonic music transcription using a temporally constrained shift-invariant model. *The Journal of the Acoustical Society of America, 133*, 1727–1741. https://doi.org/10.1121/1.4790351.

Benetos, E., Dixon, S., Duan, Z., & Ewert, S. (2019). Automatic music transcription: an overview. *IEEE Signal Procesing Magazine, 36*, 20–30.

Brook, B. S. (1965). The simplified plaine and easie code system for notating music: a proposal for international adoption. *Fontes Artis Musicae, 12*, 156–160. http://www.jstor.org/stable/23504707.

Carvalho, R. G. C., & Smaragdis, P. (2017). Towards end-to-end polyphonic music transcription: transforming music audio directly to a score. In *IEEE Workshop for Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE.

Cemgil, A. T., Kappen, H. J., & Barber, D. (2003). Generative model based polyphonic music transcription. In *IEEE workshop on applications of signal processing to audio and acoustics* (pp. 181–184).

Cogliati, A., Temperley, D., & Duan, Z. (2016). Transcribing Human Piano Performances into Music Notation. In *Proc. of the 17th International society for music information retrieval conference, ISMIR 2016, New York, USA*.

Corrêa, D. C., & Rodrigues, F. A. (2016). A survey on symbolic data-based music genre classification. *Expert Systems with Applications, 60*, 190–210. https://doi.org/10.1016/j.eswa.2016.04.008.

de Cheveigné, A., & Kawahara, H. (2002). YIN, A fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America, 111*, 1917–1930. https://doi.org/10.1121/1.1458024.

Gfeller, B., Frank, C., Roblek, D., Sharifi, M., Tagliasacchi, M., & Velimirović, M. (2019). Spice: Self-supervised pitch estimation. arXiv:1910.11664.

Good, M. (2001). MusicXML for notation and analysis. In W. B. Hewlett & E. Selfridge-Field (Eds.), *The virtual score: representation, retrieval, restoration, volume 12 of computing in musicology* (pp. 113–124). MIT Press.

Graves, A. (2008). Supervised Sequence Labelling with recurrent neural networks. Ph.D. thesis Technical University Munich.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. In *23rd International conference on machine learning international conference on machine learning* (pp. 369–376). ACM.

Hawthorne, C., Elsen, E., Song, J., Roberts, A., Simon, I., Raffel, C., Engel, J., Oore, S., & Eck, D. (2018). Onsets and frames: dual-objective piano transcription. In *Proc. of the 19th international society for music information retrieval conference* (pp. 50–57).

Huron, D. (1995). *The humdrum toolkit: reference manual*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd international conference on machine learning, ICML 2015, 6–11 July Lille, France* (pp. 448–456).

Kelz, R., Dorfer, M., Korzeniowski, F., Böck, S., Arzt, A., & Widmer, G. (2016). On the potential of simple framewise approaches to piano transcription. In *17th International conference on music information retrieval*.

Kim, J. W., & Bello, J. P. (2019). Adversarial learning for improved onsets and frames music transcription. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 670–677). http://archives.ismir.net/ismir2019/paper/000081.pdf.

McLeod, A., & Steedman, M. (2018). Evaluating automatic polyphonic music transcription. In *Proceedings of the 19th international society for music information retrieval conference, ISMIR 2018, September 23–27, 2018 Paris, France* (pp. 42–49).

Nakamura, E., Benetos, E., Yoshii, K., & Dixon, S. (2018a). Towards complete polyphonic music transcription: integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE.

Nakamura, E., Benetos, E., Yoshii, K., & Dixon, S. (2018b). Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice hall.

Ravanelli, M., & Bengio, Y. (2018). Speaker recognition from raw waveform with sincnet. In *2018 IEEE spoken language technology workshop (SLT)* (pp. 1021–1028).

RISM. (2017). Répertoire International des Sources Musicales. http://www.rism.info/..

Román, M. A., Pertusa, A., & Calvo-Zaragoza, J. (2018). An end-to-end framework for audio-to-score music transcription on monophonic excerpts. In *Proceedings of the 19th international society for music information retrieval conference, ISMIR 2018, Paris, France*.

Román, M. A., Pertusa, A., & Calvo-Zaragoza, J. (2019). A holistic approach to polyphonic music transcription with neural networks. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 731–737). http://archives.ismir.net/ismir2019/paper/000089.pdf.

Sapp, C. S. (2005). Online database of scores in the humdrum file format. In *Proceedings of the 6th international conference on music information retrieval, London, UK*. . http://ismir2005.ismir.net/proceedings/3123.pdf.

Sapp, C. S. (2013). humextra. https://github.com/craigsapp/humextra.

Schobrun, M. (2005). *The everything reading music book: a step-by-step introduction to understanding music notation and theory. Everything series*. Adams Media.

Schörkhuber, C., & Klapuri, A. (2010). Constant-Q transform toolbox for music processing. In *Proc. of the 7th sound and music computing conference, Barcelona, Spain*.

Shi, B., Bai, X., & Yao, C. (2017a). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*, 2298–2304.

Shi, B., Bai, X., & Yao, C. (2017b). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*, 2298–2304.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Proceedings of the 27th International conference on neural information processing systems – NIPS'14* (Vol. 2, pp. 3104–3112). Cambridge, MA, USA: MIT Press.

Su, L., & Yang, Y. (2015). Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 23*, 1600–1612. https://doi.org/10.1109/TASLP.2015.2442411.

Swiss RISM Office. (2017). verovio. https://github.com/rism-ch/verovio.

Mauch, M., & Dixon, S. (2014). PYIN: A fundamental frequency estimator using probabilistic threshold distributions. In Proceedings of the IEEE international conference on acoustics, speech, and signal processing (ICASSP 2014). In press.

Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. CoRR, abs/1804.06516. http://arxiv.org/abs/1804.06516. arXiv:1804.06516.

Williams, R. J., & Zipser, D. (1995). Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin & D. E. Rumelhart (Eds.), *Backpropagation: theory, architecture and applications* (pp. 433–486). Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

Ycart, A., McLeod, A., Benetos, E., & Yoshii, K. (2019a). Blending acoustic and language model predictions for automatic music transcription. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), Proceedings of the *20th International society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 454–461). http://archives.ismir.net/ismir2019/paper/000054.pdf.

Ycart, A., Stoller, D., & Benetos, E. (2019b). A comparative study of neural models for polyphonic music sequence transduction. In A. Flexer, G. Peeters, J. Urbano, & A. Volk (Eds.), *Proceedings of the 20th international society for music information retrieval conference, ISMIR 2019, Delft, The Netherlands, November 4–8, 2019* (pp. 470–477). http://archives.ismir.net/ismir2019/paper/000056.pdf.