Isolated guitar transcription using a deep belief network

Gregory Burlet[1] **and Abram Hindle**[1]

[1]**Department of Computing Science, University of Alberta, Edmonton, Canada**

ABSTRACT

Music transcription involves the transformation of an audio recording to common music notation, colloquially referred to as *sheet music*. Manually transcribing audio recordings is a difficult and time-consuming process, even for experienced musicians. In response, several algorithms have been proposed to automatically analyze and transcribe the notes sounding in an audio recording; however, these algorithms are often general-purpose, attempting to process any number of instruments producing any number of notes sounding simultaneously. This paper presents a polyphonic transcription algorithm that is constrained to processing the audio output of a single instrument, specifically an acoustic guitar. The transcription system consists of a novel note pitch estimation algorithm that uses a deep belief network and multi-label learning techniques to generate multiple pitch estimates for each analysis frame of the input audio signal. Using a compiled dataset of synthesized guitar recordings for evaluation, the algorithm described in this work results in an 11% increase in the f-measure of note transcriptions relative to a state-of-the-art algorithm in the literature. This paper demonstrates the effectiveness of deep, multi-label learning for the task of polyphonic transcription.

Keywords: Deep learning, music information retrieval, instrument transcription

INTRODUCTION

DON'T ENTIRELY AGREE

Music transcription is the process of converting an audio signal into a music score that informs a musician which notes to perform and how they are to be performed. This is accomplished through the analysis of the pitch and rhythmic properties of an acoustical waveform. In the composition or publishing process, manually transcribing each note of a musical passage to create a music score for other musicians is a labour-intensive procedure (Hainsworth and Macleod, 2003). Manual transcription is slow and error-prone: even notationally fluent and experienced musicians make mistakes, require multiple passes over the audio signal, and draw upon extensive prior knowledge to make complex decisions about the resulting transcription (Benetos et al., 2013).

In response to the time-consuming process of manually transcribing music, researchers in the multi-disciplinary field of music information retrieval (MIR) have summoned their knowledge of computing science, electrical engineering, music theory, mathematics, and statistics to develop algorithms that aim to automatically transcribe the notes sounding in an audio recording. Although the automatic transcription of monophonic (one note sounding at a time) music is considered a solved problem (Benetos et al., 2012), the automatic transcription of polyphonic (multiple notes sounding simultaneously) music "falls clearly behind skilled human musicians in accuracy and flexibility" (Klapuri, 2004). In an effort to reduce the complexity, the transcription problem can be constrained by limiting the number of notes that sound simultaneously, the genre of music being analyzed, or the number and type of instruments producing sound. A constrained domain allows the transcription system to "exploit the structure" (Martin, 1996) and consequently reduce the difficulty of transcription. This parallels systems in the more mature field of speech recognition where practical algorithms are often language, gender, or speaker dependent (Huang et al., 2001).

LEVERAGES A KNOWN PRIOR ON THE ODS. DATA DISTRIBUTION.

Automatic guitar transcription is the problem of automatic music transcription with the constraint that the audio signal being analyzed is produced by a single electric or acoustic guitar. Though this problem is constrained, a guitar is capable of producing chords of six notes simultaneously, which still offers a multitude of challenges for modern transcription algorithms. The most notable challenge is the estimation

of the pitches of notes comprising highly polyphonic chords, occurring when a guitarist strums several strings at once.

Yet another challenge presented to guitar transcription algorithms is that a large body of guitarists publish and share transcriptions in the form of tablature rather than common western music notation. Therefore, automatic guitar transcription algorithms should also be capable of producing tablature. Guitar tablature is a symbolic music notation system with a six-line staff representing the strings on a guitar. The top line of the system represents the highest pitched (smallest diameter) string and the bottom line represents the lowest pitched (highest diameter) string. A number on a line denotes the guitar fret that should be depressed on the respective string. An example of guitar tablature below its corresponding common western music notation is presented in Figure 1.

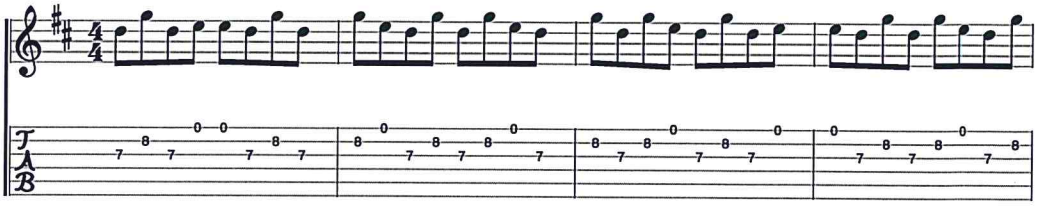

Figure 1. A system of modern guitar tablature for the song "Weird Fishes" by Radiohead, complete with common western music notation above.

A solution to the problem of isolated instrument transcription has substantial commercial interest with applications in musical games, instrument learning software, and music cataloguing. However, these applications seem far out of grasp given that the MIR research community has collectively reached a plateau in the accuracy of automatic music transcription systems (Benetos et al., 2012). In a paper addressing this issue, Benetos et al. (2012) stress the importance of extracting expressive audio features and moving towards context-specific transcription systems. Also addressing this issue, Humphrey et al. (2012, 2013) propose that effort should be focused on audio features generated by deep belief networks instead of hand-engineered audio features, due to the success of these methods in other fields such as computer vision (Lee et al., 2009) and speech recognition (Hinton et al., 2012). The aforementioned literature provides motivation for applying deep belief networks to the problem of isolated instrument transcription.

NOT EXACTLY

This paper presents a polyphonic transcription system containing a novel pitch estimation algorithm that addresses two arguable shortcomings in modern pattern recognition approaches to pitch estimation: first, the task of estimating multiple pitches sounding simultaneously is often approached using multiple one-versus-all binary classifiers (Poliner and Ellis, 2007; Nam et al., 2011) in lieu of estimating the presence of multiple pitches using a single classifier; second, there exists no standard method to impose constraints on the polyphony of pitch estimates at any given time. In response to these points, the pitch estimation algorithm described in this work uses a deep belief network in conjunction with multi-label learning techniques to produce multiple pitch estimates for each audio analysis frame.

MULTINOMIAL REGRESSION, ACT.

After estimating the pitch content of the audio signal, existing algorithms in the literature are used to track the temporal properties (onset time and duration) of each note event and convert this information to guitar tablature notation.

RELATED WORK

The first polyphonic transcription system for duets imposed constraints on the frequency range and *timbre* of the two input instruments as well as the intervals between simultaneously performed notes (Moorer, 1975).[1] This work provoked a significant amount of research on this topic, which still aims to further the accuracy of transcriptions while gradually eliminating domain constraints.

In the infancy of the problem, polyphonic transcription algorithms relied heavily on digital signal processing techniques to uncover the fundamental frequencies present in an input audio waveform. To this end, several different algorithms have been proposed: perceptually motivated models that attempt to

[1] *Timbre* refers to several attributes of an audio signal that allows humans to attribute a sound to its source and to differentiate between a trumpet and a piano, for instance. Timbre is often referred to as the "colour" of a sound.

model human audition (Klapuri, 2005); salience methods, which transform the audio signal to accentuate the underlying fundamental frequencies (Klapuri, 2006; Zhou et al., 2009); iterative estimation methods, which iteratively select a predominant fundamental from the frequency spectrum and then subtract an estimate of its harmonics from the residual spectrum until no fundamental frequency candidates remain (Klapuri, 2006); and joint estimation, which holistically selects fundamental frequency candidates that, together, best describe the observed frequency domain of the input audio signal (Yeh et al., 2010).

The MIR research community is gradually adopting a machine-learning-centric paradigm for many MIR tasks, including polyphonic transcription. Several innovative applications of machine learning algorithms to the task of polyphonic transcription have been proposed, including hidden Markov models (HMMs) (Raphael, 2002), non-negative matrix factorization (Smaragdis and Brown, 2003; Dessein et al., 2010), support vector machines (Poliner and Ellis, 2007), artificial shallow neural networks (Marolt, 2004) and recurrent neural networks (Boulanger-Lewandowski, 2014). Although each of these algorithms operate differently, the underlying principle involves the formation of a model that seeks to capture the harmonic, and perhaps temporal, structures of notes present in a set of training audio signals. The trained model then predicts the harmonic and/or temporal structures of notes present in a set of previously unseen audio signals.

Training a machine learning classifier for note pitch estimation involves extracting meaningful features from the audio signal that reflect the harmonic structures of notes and allow discrimination between different pitch classes. The obvious set of features exhibiting this property is the short-time Fourier transform (STFT), which computes the discrete Fourier transform (DFT) on a sliding analysis window over the audio signal. However, somewhat recent advances in the field of deep learning have revealed that artificial neural networks with many layers of neurons can be efficiently trained (Hinton et al., 2006) and form a hierarchical, latent representation of the input features (Lee et al., 2009).

Using a deep belief network (DBN) to learn alternate feature representations of DFT audio features, Nam et al. (2011) exported these audio features and injected them into 88 binary support vector machine classifiers: one for each possible piano pitch. Each classifier outputs a binary class label denoting whether the pitch is present in a given audio analysis frame. Using the same experimental set up as Poliner and Ellis (2007), Nam et al. (2011) noted that the learned features computed by the DBN yielded significant improvements in the precision and recall of pitch estimates relative to standard DFT audio features.

After note pitch estimation it is necessary to perform note tracking, which involves the detection of note onsets and offsets (Benetos and Weyde, 2013). Several techniques have been proposed in the literature including a multitude of onset estimation algorithms (Bello et al., 2005; Dixon, 2006), HMM note-duration modelling algorithms (Benetos et al., 2013; Ryynänen and Klapuri, 2005), and an HMM frame-smoothing algorithm (Poliner and Ellis, 2007). The output of these note tracking algorithms are a sequence of note event estimates, each having a pitch, onset time, and duration.

Humphrey 2014

Barbancho 2012

These note events may then be digitally encoded in a symbolic music notation, such as tablature notation, for cataloguing or publishing. Arranging tablature is challenging because the guitar is capable of producing the same pitch in multiple ways. Therefore, a "good" arrangement is one that is *biomechanically easy* for the musician to perform, such that transitions between notes do not require excessive hand movement and the performance of chords require minimal stretching of the hand (Heijink and Meulenbroek, 2002). Solutions to the problem of tablature arrangement include graph-search algorithms (Radicioni and Lombardo, 2005; Radisavljevic and Driessen, 2004; Burlet and Fujinaga, 2013), neural networks (Tuohy and Potter, 2006), and genetic algorithms (Tuohy and Potter, 2005; Burlet, 2013).

DEEP BELIEF NETWORKS

Before introducing the developed pitch estimation algorithm, it is worthwhile to review the structure and training procedure of a deep belief network. The intent of deep architectures for machine learning is to form a multi-layered and structured representation of sensory input with which a classifier or regressor can use to make informed predictions about its environment (Utgoff and Stracuzzi, 2002).

Recently, Hinton et al. (2006) proposed a specific formulation of a multi-layered artificial neural network called a deep belief network (DBN), which addresses the training and performance issues arising when many hidden network layers are used. A preliminary unsupervised training algorithm aims to set the network weights to good initial values in a layer-by-layer fashion, followed by a more holistic supervised fine-tuning algorithm that considers the interaction of weights in different layers with respect to the desired network output (Hinton, 2007).

Unsupervised Pretraining

In order to pretrain the network weights in an unsupervised fashion, it is necessary to think of the network as a generative model rather than a discriminative model. A generative model aims to form an internal model of a set of observable data vectors, described using latent variables; the latent variables then attempt to recreate the observable data vectors with some degree of accuracy. On the other hand, a discriminative model aims to set the value of its latent variables, typically used for the task of classification or regression, without regard for recreating the input data vectors. A discriminative model does not explicitly care how the observed data was generated, but rather focuses on producing correct values of its latent variables.

Hinton et al. (2006) proposed that a deep neural network be composed of several restricted Boltzmann machines (RBMs) stacked on top of each other, such that the network can be viewed as both a generative model and a discriminative model. An RBM is an undirected bipartite graph with m visible nodes and n hidden nodes, as depicted in Figure 2. Typically, the domain of the visible and hidden nodes are binary such that $\mathbf{v} \in \{0,1\}^m$ and $\mathbf{h} \in \{0,1\}^n$, respectively, such that

$$P(h_j = 1|\mathbf{v}) = \frac{1}{1+e^{-W_j\mathbf{v}}} \quad \text{and} \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1+e^{-W_i^T\mathbf{h}}}, \tag{1}$$

where $W \in \mathbb{R}^{n\times m}$ is the matrix of weights between the visible and hidden nodes. For simplicity, Equation 1 does not include bias nodes for $\mathbf{v}$ and $\mathbf{h}$.

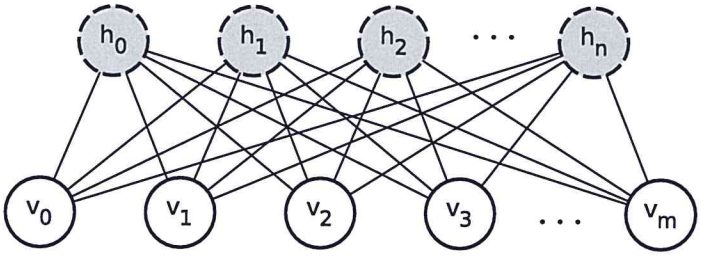

Figure 2. A restricted Boltzmann machine with m visible nodes and n hidden nodes. Weights on the undirected edges have been omitted for clarity.

Each RBM in the DBN is trained sequentially from the bottom up, such that the hidden nodes of the previous RBM are input to the subsequent RBM as an observable data vector. The unsupervised training of a single RBM involves iteratively modifying the model weights according to a learning signal that measures how well the generative model reflects reality. More specifically, the objective of the generative model is to maximize the log likelihood of the training data vectors by calculating the gradient of this objective function with respect to each edge weight.

Supervised Fine-tuning

The unsupervised pretraining of the stacked RBMs is a relatively efficient method that sets good initial values for the network weights. Moreover, in the case of a supervised learning task such as classification or regression, the ground-truth labels for each training data vector have not yet been considered. The supervised fine-tuning step of the DBN addresses these issues.

One method of supervised fine-tuning is to add a layer of output nodes to the network for the purposes of (logistic) regression and to perform standard backpropagation as if the DBN was a multi-layered neural network (Bengio, 2009). Rather than creating features from scratch, this fine-tuning method is responsible for modifying the latent features in order to adjust the class boundaries (Hinton, 2007).

After fine-tuning the network, a feature vector can be fed forward through the network and a result realized at the output layer. In the context of pitch estimation, the feature vector represents the frequency content of an audio analysis frame and the output layer of the network is responsible for classifying the pitches that are present.

ISOLATED INSTRUMENT TRANSCRIPTION

The workflow of the proposed polyphonic transcription algorithm is presented in Figure 3. The algorithm consists of an audio signal preprocessing step, followed by a novel DBN pitch estimation algorithm. The

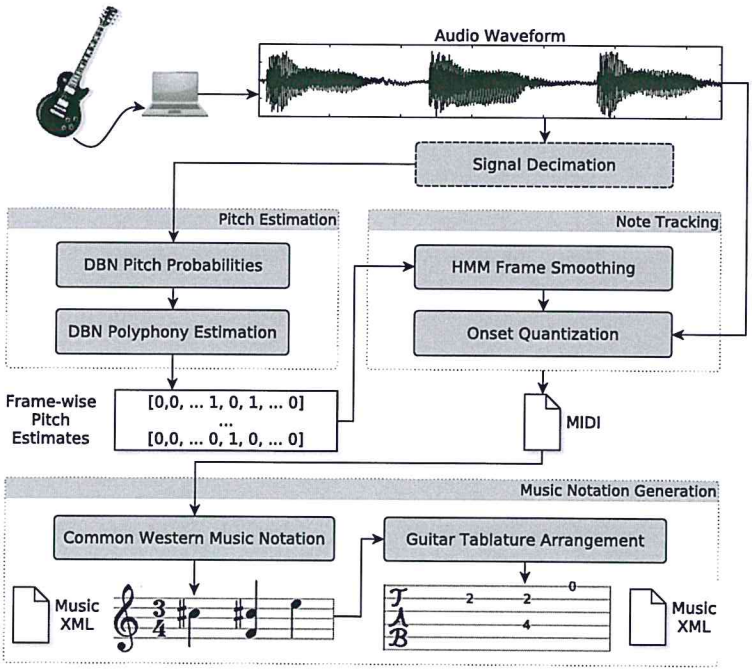

Figure 3. Workflow of the proposed polyphonic transcription algorithm, which converts the recording of a single instrument to a sequence of MIDI note events that are then translated to tablature notation.

note-tracking component of the polyphonic transcription algorithm uses a combination of the existing frame-smoothing algorithm developed by Poliner and Ellis (2007) and the existing spectral flux onset estimation algorithm described by Dixon (2006) to produce a MIDI file. MIDI is a binary file format composed of tracks holding a sequence of note events, which each have an integer pitch from 0–127, a velocity value indicating the intensity of a note, and a tick number indicating when the note event occurs. This sequence of note events is then translated to guitar tablature notation using the graph-search algorithm developed by Burlet and Fujinaga (2013).

Audio Signal Preprocessing

The input audio signal is first preprocessed before feature extraction. If the audio signal is stereo, the channels are averaged to produce a mono audio signal. Then the audio signal is decimated to lower the sampling rate f_s by an integer multiple, $k \in \mathbb{N}^+$. Decimation involves low-pass filtering with a cut-off frequency of $f_s/2k$ Hz to mitigate against aliasing, followed by selecting every k^{th} sample from the original signal.

Note Pitch Estimation

The structure of the DBN pitch estimation algorithm is presented in Figure 4. The algorithm extracts features from an analysis window that slides over the audio waveform. The audio features are subsequently fed forward through the deep network, resulting in an array of posterior probabilities used for pitch and polyphony estimation. INDEP?

First, features are extracted from the input audio signal. The *power spectrum* of each audio analysis frame is calculated using a Hamming window of size w samples and a hop size of h samples. The power spectrum is calculated by squaring the magnitude of each frequency component of the DFT. Since the power spectrum is mirrored about the Nyquist frequency when processing an audio signal, half of the spectrum is retained, resulting in $m = \lfloor w/2 \rfloor + 1$ features. The result is a matrix of normalized audio features $\Phi \in [0,1]^{n \times m}$, such that n is the number of analysis frames spanning the input signal.

The DBN consumes these normalized audio features; hence, the input layer consists of m nodes. There can be any number of stochastic binary hidden layers, each consisting of any number of nodes.

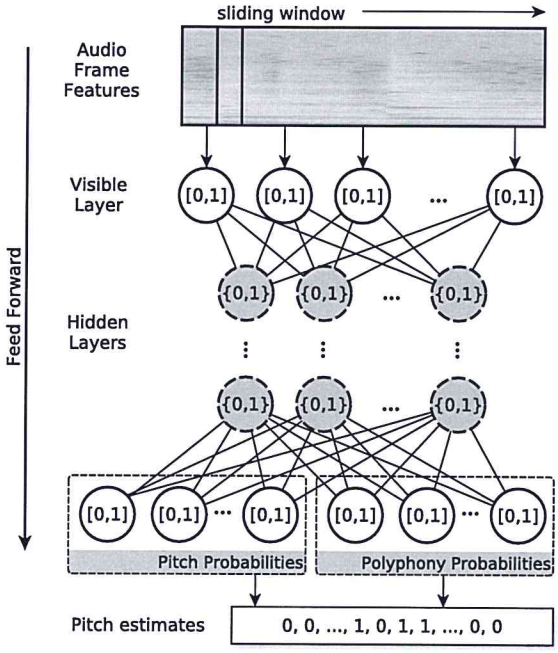

Figure 4. Structure of the deep belief network for note pitch estimation. Edge weights are omitted for clarity.

The output layer of the network consists of $k+p$ nodes, where the first k nodes are allocated for pitch estimation and the final p nodes are allocated for polyphony estimation. The network uses a sigmoid activation as the non-linear transfer function.

The feature vectors Φ are fed forward through the network with parameters Θ, resulting in a matrix of probabilities $\mathrm{P}(\hat{Y}|\Phi,\Theta) \in [0,1]^{k+p}$ that is then split into a matrix of pitch probabilities $\mathrm{P}(\hat{Y}^{(pitch)}|\Phi,\Theta)$ and polyphony probabilities $\mathrm{P}(\hat{Y}^{(poly)}|\Phi,\Theta)$. The polyphony of the i^{th} analysis frame is estimated by selecting the polyphony class with the highest probability using the equation

$$\rho_i = \underset{j}{\operatorname{argmax}} \left(\mathrm{P}(\hat{Y}_{ij}^{(poly)}|\Phi_i,\Theta) \right). \tag{2}$$

Pitch estimation is performed using a multi-label learning technique similar to the *MetaLabeler* system (Tang et al., 2009), which trains a multi-class classifier for label cardinality estimation using the output values of the original label classifier as features. Instead of using the matrix of pitch probabilities as features for a separate polyphony classifier, increased recall was noted by training the polyphony classifier alongside the pitch classifier using the original audio features. Formally, the pitches sounding in the i^{th} analysis frame are estimated by selecting the indices of the ρ_i highest pitch probabilities produced by the DBN. With these estimates, the corresponding vector of pitch probabilities is converted to a binary vector $\hat{Y}_i^{(pitch)} \in \{0,1\}^k$ by turning on bits that correspond to the ρ_i highest pitch probabilities.

For training and testing the algorithm, a set of pitch and polyphony labels are calculated for each audio analysis frame using an accompanying ground-truth MIDI file. A matrix of pitch annotations $Y^{(pitch)} \in \{0,1\}^{n\times k}$, where k is the number of considered pitches, is computed such that an enabled bit indicates the presence of a pitch. A matrix of polyphony annotations $Y^{(poly)} \in \{0,1\}^{n\times p}$, where p is the maximum frame-wise polyphony, is also computed such that a row is a one-hot binary vector in which the enabled bit indicates the polyphony of the frame. These matrices are horizontally concatenated to form the final matrix $Y \in \{0,1\}^{n\times(k+p)}$ of training and testing labels.

The deep belief network is trained using a modified version of the greedy layer-wise algorithm described by Hinton et al. (2006). Pretraining is performed by stacking a series of restricted Boltzmann

YOU PROBABLY WANT A SOFTMAX OVER THE POLYPHONY EST., THO COMBINING THE XENTROP W/ A NLL LOSS MIGHT BE NON-TRIVIAL...

NO RELU?

MIGHT NOT BE NECESSARY

machines and sequentially training each in an unsupervised manner using 1-step contrastive divergence (Bengio, 2009). Instead of using the "up-down" fine-tuning algorithm proposed by Hinton et al. (2006), the layer of output nodes are treated as a set of logistic regressors and standard backpropagation is conducted on the network. Rather than creating features from scratch, this fine-tuning method is responsible for modifying the latent features in order to adjust the class boundaries (Hinton, 2007).

The canonical error function to be minimized for a set of separate pitch and polyphony binary classifications is the cross-entropy error function, which forms the training signal used for backpropagation:

$$E(\Theta) = -\sum_{i=1}^{n}\sum_{j=1}^{k+p} Y_{ij}\ln \mathrm{P}(\hat{Y}_{ij}|\Phi_i,\Theta) + (1-Y_{ij})\ln(1-\mathrm{P}(\hat{Y}_{ij}|\Phi_i,\Theta)) \quad (3)$$

The aim of this objective function is to adjust the network weights Θ to pull output node probabilities closer to one for ground-truth label bits that are on and pull probabilities closer to zero for bits that are off.

The described pitch estimation algorithm was implemented using the *Theano* numerical computation library for Python (Bergstra et al., 2010). Computations for network training and testing are parallelized on the graphics processing unit (GPU). Feature extraction and audio signal preprocessing is performed using *Marsyas*, a software framework for audio signal processing and analysis (Tzanetakis and Cook, 2000).

So this is actually a little weird, as there's an implicit weighting between the pitch & polyphony surfaces... Could be controlled by a hyperparam?

Note Tracking

Although frame-level pitch estimates are essential for transcription, converting these estimates into note events with an onset and duration is not a trivial task. The purpose of note tracking is to process these pitch estimates and determine when a note onsets and offsets.

Frame-level Smoothing

The frame-smoothing algorithm developed by Poliner and Ellis (2007) is used to postprocess the DBN pitch estimates $\hat{Y}^{(pitch)}$ for an input audio signal. The algorithm allows a frame-level pitch estimate to be contextualized amongst its neighbours instead of solely trusting the independent estimates made by a classification algorithm.

Formally, the frame-smoothing algorithm operates by training an HMM for each pitch. Each HMM consists of two hidden states: `ON` and `OFF`. The transition probabilities are computed by observing the frequency with which a pitch transitions between and within the `ON` and `OFF` states across analysis frames. The emission distribution is a Bernoulli distribution that models the certainty of each frame-level estimate and is represented using the pitch probabilities $\mathrm{P}(\hat{Y}^{(pitch)}|\Phi,\Theta)$. The output of the Viterbi algorithm, which searches for the optimal underlying state sequence, is a revised binary vector of activation estimates for a single pitch. Concatenating the results of each HMM results in a revised matrix of pitch estimates $\hat{Y}^{(pitch)}$.

Onset Quantization

If the HMM frame-smoothing algorithm claims a pitch arises within an analysis frame, it could onset at any time within the window. Arbitrarily setting the note onset time to occur at the beginning of the window often results in "choppy" sounding transcriptions. In response, the onset detection algorithm that uses spectral flux measurements between analysis frames (Dixon, 2006) is run at a finer time resolution to pinpoint the exact note onset time. The onset detection algorithm is run on the original, undecimated audio signal with a window size of 2048 samples and a hop size of 512 samples. When writing the note event estimates as a MIDI file, the onset times calculated by this algorithm are used. The offset time is calculated by following the pitch estimate across consecutive analysis frames until it transitions from `ON` to `OFF`, at which point the time stamp of the end of this analysis frame is used. Note events spanning less than two audio analysis frames are removed from the transcription to mitigate against spurious notes.

Output of the polyphonic transcription algorithm at each stage—from feature extraction to DBN pitch estimation to frame smoothing and quantization (note tracking)—is displayed in Figure 5 for a four-second segment of a synthesized guitar recording. The pitch probabilities output by the DBN show that the classifier is quite certain about its estimates; there are few grey areas indicating indecision.

Music Notation Arrangement

The MIDI file output by the algorithm thus far contains the note event (pitch, onset, and duration) transcriptions of an audio recording. However, a MIDI file lacks certain information necessary to write

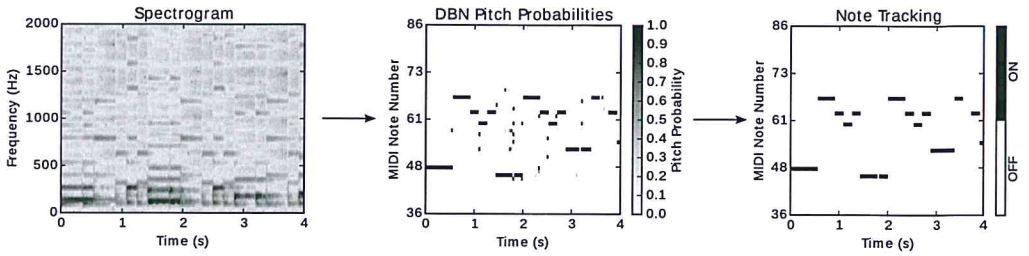

Figure 5. An overview of the transcription workflow on a four-second segment of a synthesized guitar recording.

sheet music in common western music notation such as time signature, key signature, clef type, and the value (duration) of each note described in divisions of a whole note.

There are several robust opensource programs that derive this missing information from a MIDI file using logic and heuristics in order to generate common western music notation that is digitally encoded in the *MusicXML* file format. MusicXML is a standardized extensible markup language (XML) definition allowing digital symbolic music notation to be universally encoded and parsed by music applications. In this work, the command line tools shipped with the opensource application *MuseScore* are used to convert MIDI to common western music notation encoded in the MusicXML file format.[2]

The graph-based guitar tablature arrangement algorithm developed by Burlet and Fujinaga (2013) is used to append a guitar string and fret combination to each note event encoded in a MusicXML transcription file. The guitar tablature arrangement algorithm operates by using Dijkstra's algorithm to search for the shortest path through a directed weighted graph, in which the vertices represent candidate string and fret combinations for a note or chord, as displayed in Figure 6.

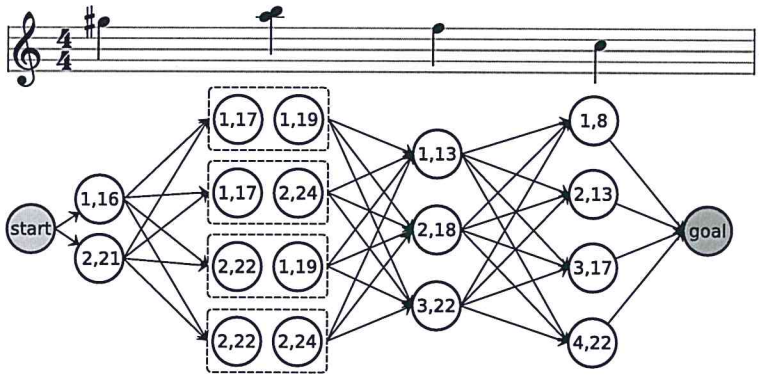

Figure 6. A directed acyclic graph of string and fret candidates for a note and chord followed by two more notes. Weights have been omitted for clarity. The notation for each node is (string number, fret number).

The edge weights between nodes in the graph indicate the biomechanical difficulty of transitioning between fretting-hand positions. Three biomechanical complexity factors are aggregated to form each edge weight: the fret-wise distance required to transition between notes or chords, the fret-wise finger span required to perform chords, and a penalty of one if the fretting hand surpasses the seventh fret. The value of this penalty and fret threshold number were determined through subjective analysis of the resulting tablature arrangements. In the event that a note is followed by a chord, the fret-wise distance is calculated by the expression

$$\left| f - \frac{\max(g) - \min(g)}{2} \right|, \tag{4}$$

[2] `http://musescore.org`

such that $f \in \mathbb{N}$ is the fret number used to perform the note and g is a vector of fret numbers used to perform each note in the chord. For more detailed information regarding the formulation of this graph, please refer to the conference proceeding of Burlet and Fujinaga (2013) or thesis of Burlet (2013).

TRANSCRIPTION EVALUATION

The polyphonic transcription algorithm described in this paper is evaluated on a new dataset of synthesized guitar recordings. Before processing these guitar recordings, the number of pitches k and maximum polyphony p of the instrument must first be calculated in order to construct the DBN. Knowing that the input instrument is a guitar with six strings, the pitch estimation algorithm considers the $k = 51$ pitches from *C2–D6*, which spans the lowest note capable of being produced by a guitar in *Drop C* tuning to the highest note capable of being produced by a 22-fret guitar in *Standard* tuning. Though a guitar with six strings is only capable of producing six notes simultaneously, a chord transition may occur within a frame and so the maximum polyphony may increase above this bound. This is a technical side effect of a sliding-window analysis of the audio signal. Therefore, the maximum frame-wise polyphony is calculated from the training dataset using the equation

$$p = \max_i \left(\left(Y^{(pitch)} \mathbb{1} \right)_i \right) + 1, \tag{5}$$

where $\mathbb{1}$ is a vector of ones. The addition of one to the maximum polyphony is to accommodate silence where no pitches sound in an analysis frame.

The experiments outlined in this section will evaluate the accuracy of pitch estimates output by the DBN across each audio analysis frame as well as the accuracy of note events output by the entire polyphonic transcription algorithm. A formal evaluation of the guitar tablature arrangement algorithm used in this work has already been conducted (Burlet and Fujinaga, 2013).

Ground-truth Dataset

Ideally, the note pitch estimation algorithm proposed in this work should be trained and tested using recordings of acoustic or electric guitars that are subsequently hand-annotated with the note events being performed. In practice, however, it would be expensive to fund the compilation of such a dataset and there is a risk of annotation error. Unlike polyphonic piano transcription datasets that are often created using a mechanically controlled piano, such as a Yamaha Disklavier, to generate acoustic recordings that are time aligned with note events in a MIDI file, mechanized guitars are not widely available. Therefore, the most feasible course of action for compiling a polyphonic guitar transcription dataset is to synthesize a set of ground-truth note events using an acoustic model of a guitar.

Using the methodology proposed by Burlet and Fujinaga (2013), a ground-truth dataset of 45 synthesized acoustic guitar recordings paired with MIDI note-event annotations was compiled. The dataset was created by harvesting the abundance of crowdsourced guitar transcriptions uploaded to `www.ultimate-guitar.com` as tablature files that are manipulated by the *Guitar Pro* desktop application.[3] The transcriptions in the ground-truth dataset were selected by searching for the keyword "acoustic", filtering results to those that have been rated by the community as 5 out of 5 stars, and selecting those that received the most numbers of ratings and views. The dataset consists of songs by artists ranging from The Beatles, Eric Clapton, and Neil Young to Led Zeppelin, Metallica, and Radiohead.

Each Guitar Pro file was manually preprocessed to remove extraneous instrument tracks other than guitar, remove repeated bars, trim recurring musical passages, and remove note ornamentations such as dead notes, palm muting, harmonics, pitch bends, and vibrato. The guitar model for note synthesis was set to a Martin & Co. acoustic guitar with steel strings. Finally, each Guitar Pro file is synthesized as a WAV file and also exported as a MIDI file, which captures the note events occurring in the guitar track. The MIDI files in the ground-truth dataset are publicly available on `archive.org`. [4] The amount of time required to manually preprocess a Guitar Pro tablature transcription and convert it into the necessary data format ranges from 30 minutes to 2.5 hours, depending on the complexity of the musical passage.

In total the dataset consists of approximately 104 minutes of audio, an average tempo of 101 beats per minute, 44436 notes, and an average polyphony of 2.34. The average polyphony is calculated by dividing

[3] `www.guitar-pro.com`

[4] `https://archive.org/details/DeepLearningIsolatedGuitarTranscriptions`

the number of note events by the number of chords plus the number of individual notes. The distribution of note pitches in the dataset is displayed in Figure 7.

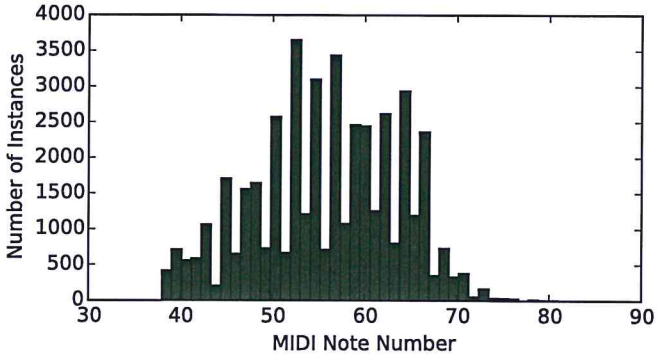

Figure 7. Distribution of note pitches in the ground-truth dataset.

Algorithm Parameter Selection

Before training and evaluating the described polyphonic transcription algorithm on the ground-truth dataset, several preliminary experiments were conducted to select reasonable parameters for the algorithm. Each preliminary experiment involved the following variables: audio sampling rate (Hz), window size (samples), sliding window hop size (samples), number of network hidden layers, number of nodes per hidden layer, and input features: either the power spectrum or Mel frequency cepstral coefficient (MFCC) features, which are often used in the field of speech recognition (Hinton et al., 2012). Each preliminary experiment selected one independent variable, while the other variables remained controlled. The dependent variable was the standard information retrieval metric of f-measure, which gauges the accuracy of the pitch estimates produced by the DBN over all audio analysis frames. For these preliminary experiments, the ground-truth dataset was partitioned into two sets, such that roughly 80% of the guitar recordings are allocated for training and 20% are allocated for model validation.

GRID SEARCH? OR NO? THESE WON'T BE INDEP.

The results of the preliminary experiments with the proposed transcription system revealed that a sampling rate of 22050 Hz, a window size of 1024 samples, a hop size of 768 samples, a network structure of 400 nodes in the first hidden layer followed by 300 nodes in the penultimate layer, and power spectrum input features yielded optimal results. For network pretraining, 400 epochs were conducted with a learning rate of 0.05 using 1-step contrastive divergence with a batch size of 1000 training instances. For network fine-tuning, 30000 epochs were conducted with a learning rate of 0.05 and a batch size of 1000 training instances. The convergence threshold, which ceases training if the value of the objective function between epochs does not fluctuate more than the threshold, is set to $1E-18$ for both pretraining and fine-tuning. These algorithm parameters are used in the experiments detailed in the following sections. The experiments conducted in this paper were run on a machine with an Intel® Core™ i7 3.07 GHz quad core CPU, 24 GB of RAM, and an Nvidia GeForce GTX 970 GPU with 1664 CUDA cores. For more details regarding these preliminary experiments, consult the thesis of Burlet (2015).

Frame-level Pitch Estimation Evaluation

MIGHT BE USEFUL TO DISTINGUISH BETWEEN PARAMS (MODEL) & HYPERPARAMS (SYSTEM)

5-fold cross validation is used to split the songs in the compiled ground-truth dataset into 5 sets of training and testing partitions. For each fold, the transcription algorithm is trained using the parameters detailed in the previous section. After training, the frame-level pitch estimates computed by the DBN are evaluated for each fold using the following standard multi-label learning metrics (Zhang and Zhou, 2014): precision (p), recall (r), f-measure (f), one error, and hamming loss. The precision calculates the number of correct pitch estimates divided by the number of pitches the algorithm predicts are present across the audio analysis frames. The recall calculates the number of correct pitch estimates divided by the number of ground-truth pitches that are active across the audio analysis frames. The f-measure refers to the balanced f-score, which is the harmonic mean of precision and recall. The one error provides insight into the number of audio analysis frames where the predominant pitch is estimated incorrectly. The hamming

OF THE ESTIMATION?

	r_{poly}	p	r	f	ONE ERROR	HAMMING LOSS
BEFORE HMM	0.55	0.73	0.70	0.71	0.13	0.03
AFTER HMM	0.55	0.81	0.74	**0.77**	0.13	0.02

Table 1. 5-fold cross validation results of the frame-level pitch estimation evaluation metrics: r_{poly} denotes the polyphony recall, p denotes precision, r denotes recall, and f denotes f-measure.

loss provides insight into the number of false positive and false negative pitch estimates across the audio analysis frames. In addition, the frame-level polyphony recall (r_{poly}) is calculated to evaluate the accuracy of polyphony estimates made by the DBN.

Using the ground-truth dataset, pretraining the DBN took an average of 172 minutes while fine-tuning took an average of 246 minutes across each fold. After training, the network weights are saved so that they can be reused for future transcriptions. The DBN took an average of 0.26 seconds across each fold to yield pitch estimates for the songs in the test partitions. The results of the DBN pitch estimation algorithm are averaged across the 5 folds and presented in Table 1. After HMM frame smoothing the results substantially improve with a precision of 0.81, a recall of 0.74, and an f-measure of 0.77. Figure 5 provides visual evidence of the positive impact of HMM frame smoothing on the frame-level DBN pitch estimates, showing the removal of several spurious note events.

The results reveal that the 55% polyphony estimation accuracy likely hinders the frame-level f-measure of the pitch estimation algorithm. Investigating further, when using the ground-truth polyphony for each audio analysis frame, an f-measure of 0.76 is noted before HMM smoothing. The 5% increase in f-measure reveals that the polyphony estimates are close to their ground-truth value. With respect to the one error, the results reveal that the DBN's belief of the predominant pitch—the label with the highest probability—is incorrect in only 13% of the analysis frames.

Note Event Evaluation

Although evaluating the pitch estimates made by the algorithm for each audio analysis frame provides vital insight into the performance of the algorithm, we can continue with an evaluation of the final note events output by the algorithm. After HMM smoothing the frame-level pitch estimates computed by the DBN, onset quantization is performed and a MIDI file, which encodes the pitch, onset time, and duration of note events, is written. An evaluation procedure similar to the music information retrieval evaluation exchange (MIREX) note tracking task, a yearly competition that evaluates polyphonic transcription algorithms developed by different research institutions on the same dataset, is conducted using the metrics of precision, recall, and f-measure.[5] Relative to a ground-truth note event, an estimate is considered correct if its onset time is within 250ms and its pitch is equivalent. The accuracy of note offset times are not considered because offset times exhibit less perceptual importance than note onset times (Costantini et al., 2009). A ground-truth note event can only be associated with a single note event estimate.

These metrics of precision, recall, and f-measure are calculated on the test partition within each of the 5 folds used for cross validation. Table 2 presents the results of the polyphonic transcription algorithm averaged across each fold. The result of this evaluation is an average f-measure of 0.67 when considering note octave errors and an average f-measure of 0.69 when disregarding note octave errors. Octave errors occur when the algorithm predicts the correct note pitch name but incorrectly predicts the note octave number. An approximately 2% increase in f-measure when disregarding octave errors provides evidence that the transcription algorithm does not often mislabel the octave number of note events, which is often a problem with digital signal processing transcription algorithms (Maher and Beauchamp, 1994). Note that the frame-level pitch estimation f-measure of 0.77, presented in Table 1, does not translate to an equivalently high f-measure for note events because onset time is considered in the evaluation criteria as well as pitch.

Another interesting property of the transcription algorithm is its conservativeness: the precision of the note events transcribed by the algorithm is 0.81 while the recall is 0.60, meaning that the algorithm favours false negatives over false positives. In other words, the transcription algorithm includes a note event in

[5] http://www.music-ir.org/mirex

	DBN Transcription			
	Precision	Recall	f-measure	Runtime (s)
Octave Errors	0.81	0.60	**0.67**	**48.33**
No Octave Errors	0.84	0.63	0.69	–

	Zhou et al. (2009)			
	Precision	Recall	f-measure	Runtime (s)
Octave Errors	0.70	0.50	**0.56**	**293.52**
No Octave Errors	0.78	0.56	0.62	–

Table 2. 5-fold cross validation results of the precision, recall, and f-measure evaluation of note events transcribed using the DBN transcription algorithm compared to the Zhou et al. (2009) transcription algorithm. The first row includes octave errors while the second row excludes octave errors.

the final transcription only if it is quite certain of the note's correctness, even if this hinders the recall of the algorithm. Another cause of the high precision and low recall is that when several guitar strums occur quickly in succession, the implemented transcription algorithm often transcribes only the first chord and prescribes it a long duration. This is likely a result of the temporally "coarse" window size of 1024 samples or a product of the HMM frame-smoothing algorithm, which may extend the length of notes causing them to "bleed" into each other. A remedy for this issue is to lower the window size to increase temporal resolution; however, this has an undesirable side-effect of lowering the frequency resolution of the DFT which is undesirable. A subjective, aural analysis of the guitar transcriptions reflects these results: the predominant pitches and temporal structures of notes occurring in the input guitar recordings are more or less maintained.

HOW MUCH IS DUE TO THE ONSET DETECTION? IS THAT ALWAYS ON, OR...?

Additionally, the guitar recordings in the test set of each fold are transcribed by a state-of-the-art, digital signal processing polyphonic transcription algorithm developed by Zhou et al. (2009), which was evaluated in the 2008 MIREX and received an f-measure of 0.76 on a dataset of 30 synthesized and real piano recordings (Zhou and Reiss, 2008). The Zhou et al. (2009) polyphonic transcription algorithm processes audio signals at a sampling rate of 44100 Hz. A window size of 441 samples and a hop size of 441 samples is set by the authors for optimal transcription performance (Zhou and Reiss, 2008).

The transcription algorithm described in this paper resulted in an 11% increase, or a 20% relative increase, in f-measure compared to the transcription algorithm developed by Zhou et al. (2009) when evaluated on the same dataset, and further, performed these transcriptions in a sixth of the time. This result emphasizes a lucrative property of neural networks: after training, feeding the features forward through the network is accomplished in a small amount of time.

PERHAPS, BUT NOT REALLY THE WORD YOU WANT HERE :)

With a precision of 0.70 and a recall of 0.50 when considering octave errors, the Zhou et al. (2009) transcription algorithm also exhibits a significantly higher precision than recall; in this way, it is similar to the transcription algorithm described in this paper. When disregarding octave errors, the f-measure of the Zhou et al. (2009) transcription algorithm increases by approximately 6%. Therefore, this state-of-the-art digital signal processing transcription algorithm makes three times the number of note octave errors as the transcription algorithm described in this paper.

DISCUSSION

Considering the results of the experiments outlined in the previous section, there are several benefits of using the developed transcription algorithm. As previously mentioned, the accuracy of transcriptions generated by the algorithm surpasses the current state of the art and makes less octave errors. Moreover, the developed polyphonic transcription algorithm can generate transcriptions for full-length guitar recordings in the order of seconds, rather than minutes or hours. Given the speed of transcription, the proposed polyphonic transcription algorithm could be adapted for real-time transcription applications, where live performances of guitar are automatically transcribed. This could be accomplished by buffering the input guitar signal into analysis frames as it is performed. Another benefit of this algorithm is that the trained network weights can be saved to disk such that future transcriptions do not require retraining the model.

As well, the size of the model is relatively small (less than 12MB) and so the network weights can fit on a portable device or microcontroller. Feeding forward audio features through the DBN is a computationally inexpensive task and could also operate on a portable device or microcontroller. Finally, the developed polyphonic transcription algorithm could easily be adapted to accommodate the transcription of other instruments. All that is required is a set of audio files that have accompanying MIDI annotations for supervised training.

On the other hand, there are several detriments of the transcription algorithm. First, the amount of time required to properly train the model is substantial and varies depending on several parameters such as the audio sampling rate, window size, hop size, and network structure. To make training time reasonable, the computations should be outsourced to a GPU that is capable of performing many calculations in parallel. Using a GPU with less CUDA cores, or just a CPU, significantly increases the amount of time required to train the model. After training ceases, either by reaching the set number of training epochs or when the objective function stops fluctuating, it is not guaranteed that the resulting network weights are optimal because the training algorithm may have settled at a local minima of the objective function. As a consequence of the amount of time required to train the pitch estimation algorithm, it is difficult to search for good combinations of algorithm parameters. Another arguable detriment of the transcription algorithm is that the underlying DBN pitch estimation algorithm is essentially a *black box*. After training, it is difficult to ascertain how the model reaches a solution. This issue is exacerbated as the depth of the network increases. Finally, it is possible to overfit the model to the training dataset. When running the fine-tuning process for another 30000 epochs, the f-measure of the transcription algorithm began to deteriorate due to overfitting. To mitigate against overfitting, the learning rate could be dampened as the number of training epochs increase. Another solution involves the creation of a validation dataset, such that the fine-tuning process stops when the f-measure of the algorithm begins to decrease on the guitar recordings in the validation dataset. The method used in this paper is *early stopping*, where the fine-tuning process is limited to a certain number of epochs instead of allowing the training procedure to continue indefinitely.

OR DROPOUT, REGULARIZATION, DENOISING, DATA AUGMENTATION...

CONCLUSION

When applied to the problem of polyphonic guitar transcription, deep belief networks outperform state-of-the-art transcription algorithms. Moreover, the developed transcription algorithm is fast: the transcription of a full-length guitar recording occurs in the order of seconds and is therefore suitable for real-time guitar transcription. As well, the algorithm is adaptable for the transcription of other instruments, such as the bass guitar or piano, as long as the pitch range of the instrument is provided and MIDI annotated audio recordings are available for training.

The polyphonic transcription algorithm described in this paper is capable of forming discriminative, latent audio features that are suitable for quickly transcribing guitar recordings. The algorithm workflow consists of audio signal preprocessing, feature extraction, a novel pitch estimation algorithm that uses deep learning and multi-label learning techniques, frame smoothing, and onset quantization. The generated note event transcriptions are digitally encoded as a MIDI file, that is processed further to create a *MusicXML* file that encodes the corresponding guitar tablature notation.

An evaluation of the frame-level pitch estimates generated by the deep belief network on a dataset of synthesized guitar recordings resulted in an f-measure of 0.77 after frame smoothing. An evaluation of the note events output by the entire transcription algorithm resulted in an f-measure of 0.67, which is 11% higher than the f-measure reported by a state-of-the-art, single-instrument transcription algorithm (Zhou et al., 2009) on the same dataset.

The results of this work encourage the use of deep architectures such as belief networks to form alternative representations of industry-standard audio features for the purposes of instrument transcription. Moreover, this work demonstrates the effectiveness of multi-label learning for pitch estimation, specifically when an upper bound on polyphony exists.

Future Work

There are several directions of future work to improve the accuracy of transcriptions. First, there are substantial variations in the distribution of pitches across songs, and so the compilation of more training data is expected to improve the accuracy of frame-level pitch estimates made by the DBN. Second, alternate methods could be explored to raise the accuracy of frame-level polyphony estimates, such as

training a separate classifier for predicting polyphony on potentially different audio features. Third, an alternate frame-smoothing algorithm that jointly considers the probabilities of other pitch estimates across analysis frames could further increase pitch estimation f-measure relative to the HMM method proposed by Poliner and Ellis (2007), which smooths the estimates of one pitch across the audio analysis frames. Finally, it would be beneficial to investigate whether the latent audio features derived for transcribing one instrument are transferable to the transcription of other instruments.

In the end, the big picture is a self-sufficient guitar tablature transcription algorithm that is capable of feeding itself data to improve its transcriptions. There are many guitarists that share manual tablature transcriptions online that would personally benefit from having an automated system capable of generating transcriptions that are almost correct and can subsequently be corrected manually. There is incentive to manually correct the output transcriptions because this method is potentially faster than performing a transcription from scratch, depending on the quality of the automated transcription and the difficulty of the song. The result is a crowdsourcing model that is capable of producing large ground-truth datasets for polyphonic transcription that can then be used to further improve the polyphonic transcription algorithm. Not only would this improve the accuracy of the developed polyphonic transcription algorithm, but it would also provide a centralized repository of ground-truth guitar transcriptions for MIR researchers to train and test future state-of-the-art transcription algorithms.

THIS COULD USE SOME WORK.

ACKNOWLEDGMENTS

Special thanks are owed to Ruohua Zhou and Joshua Reiss for the opensource implementation of their transcription algorithm evaluated in this work, as well as the individuals who uploaded manual tablature transcriptions to `www.ultimate-guitar.com`.

REFERENCES

Bello, J. P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., and Sandler, M. B. (2005). A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047.

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., and Klapuri, A. (2013). Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434.

Benetos, E., Dixon, S., Giannoulis, D., Kirchoff, H., and Klapuri, A. (2012). Automatic music transcription: Breaking the glass ceiling. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 1002–1007, Porto, Portugal.

Benetos, E. and Weyde, T. (2013). Explicit duration hidden Markov models for multiple-instrument polyphonic music transcription. In *Proceedings of the International Conference on Music Information Retrieval*, pages 269–274, Curitiba, Brazil.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*, pages 3–10, Austin, TX.

Boulanger-Lewandowski, N. (2014). *Modeling High-Dimensional Audio Sequences with Recurrent Neural Networks*. PhD thesis, Université de Montréal.

Burlet, G. (2013). Automatic guitar tablature transcription online. Master's thesis, McGill University.

Burlet, G. (2015). Guitar tablature transcription using a deep belief network. Master's thesis, McGill University.

Burlet, G. and Fujinaga, I. (2013). Robotaba guitar tablature transcription framework. In *Proceedings of the International Society for Music Information Retrieval*, pages 421–426, Curitiba, Brazil.

Costantini, G., Perfetti, R., and Todisco, M. (2009). Event based transcription system for polyphonic piano music. *Signal Processing*, 89(9):1798–1811.

Dessein, A., Cont, A., and Lemaitre, G. (2010). Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *Proceedings of the International Society for Music Information Retrieval Conference*, Utrecht, Netherlands.

Dixon, S. (2006). Onset detection revisited. In *Proceedings of the International Conference on Digital Audio Effects*, pages 133–137, Montréal, QC.

Hainsworth, S. W. and Macleod, M. D. (2003). The automated music transcription problem. Technical report, Department of Engineering, University of Cambridge.

Heijink, H. and Meulenbroek, R. G. J. (2002). On the complexity of classical guitar playing: Functional adaptations to task constraints. *Journal of Motor Behavior*, 34(4):339–351.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434.

Hinton, G. E., Osindero, S., and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Huang, X., Acero, A., and Hon, H. (2001). *Spoken Language Processing: A guide to theory, algorithm, and system development*. Prentice Hall, Upper Saddle River, NJ.

Humphrey, E., Bello, J., and LeCun, Y. (2012). Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the International Society for Music Information Retrieval*, pages 403–408, Porto, Portugal.

Humphrey, E., Bello, J., and LeCun, Y. (2013). Feature learning and deep architectures: New directions for music informatics. *Journal of Intelligent Systems*, 41(3):461–481.

Klapuri, A. (2004). Automatic music transcription as we know it today. *Journal of New Music Research*, 33(3):269–282.

Klapuri, A. (2005). A perceptually motivated multiple-F0 estimation method. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 291–294, New Paltz, NY.

Klapuri, A. (2006). Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 216–221, Victoria, BC.

Lee, H., Grosse, R., Ranganath, R., and Ng, A. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the International Conference on Machine Learning*, pages 609–616, Montréal, QC.

Maher, R. C. and Beauchamp, J. W. (1994). Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America*, 95(4):2254–2263.

Marolt, M. (2004). A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Transactions on Multimedia*, 6(3):439–449.

Martin, K. D. (1996). A blackboard system for automatic transcription of simple polyphonic music. Technical Report 385, Massachusetts Institute of Technology.

Moorer, J. A. (1975). *On the segmentation and analysis of continuous musical sound by digital computer*. PhD thesis, Department of Music, Stanford University, Stanford, CA.

Nam, J., Ngiam, J., Lee, H., and Slaney, M. (2011). A classification-based polyphonic piano transcription approach using learned feature representations. In *Proceedings of the International Society for Music Information Retrieval*, pages 175–180, Miami, FL.

Poliner, G. E. and Ellis, D. P. W. (2007). Improving generalization for polyphonic piano transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 86–89, New Paltz, NY.

Radicioni, D. P. and Lombardo, V. (2005). Computational modeling of chord fingering for string instruments. In *Proceedings of the International Conference of the Cognitive Science Society*, pages 1791–1796, Stresa, Italy.

Radisavljevic, A. and Driessen, P. (2004). Path difference learning for guitar fingering problem. In *Proceedings of the International Computer Music Conference*, Miami, FL.

Raphael, C. (2002). Automatic transcription of piano music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 1–5, Paris, France.

Ryynänen, M. and Klapuri, A. (2005). Polyphonic music transcription using note event modeling. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 319–322, New Paltz, NY.

Smaragdis, P. and Brown, J. C. (2003). Non-negative matrix factorization for polyphonic music transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and*

Acoustics, pages 177–180, New Paltz, NY.

Tang, L., Rajan, S., and Narayanan, V. K. (2009). Large scale multi-label classification via metalabeler. In *Proceedings of the International Conference on World Wide Web*, pages 211–220, New York, NY.

Tuohy, D. R. and Potter, W. D. (2005). A genetic algorithm for the automatic generation of playable guitar tablature. In *Proceedings of the International Computer Music Conference*, pages 499–502, Barcelona, Spain.

Tuohy, D. R. and Potter, W. D. (2006). An evolved neural network/HC hybrid for tablature creation in GA-based guitar arranging. In *Proceedings of the International Computer Music Conference*, pages 576–579, New Orleans, LA.

Tzanetakis, G. and Cook, P. (2000). MARSYAS: A framework for audio analysis. *Organised Sound*, 4(3):169–175.

Utgoff, P. E. and Stracuzzi, D. J. (2002). Many-layered learning. *Neural Computation*, 14:2497–2539.

Yeh, C., Roebel, A., and Rodet, X. (2010). Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1116–1126.

Zhang, M. and Zhou, Z. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837.

Zhou, R. and Reiss, J. D. (2008). A real-time polyphonic music transcription system. In the *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2008/F0_zhou.pdf`.

Zhou, R., Reiss, J. D., Mattavelli, M., and Zoia, G. (2009). A computationally efficient method for polyphonic pitch estimation. *EURASIP Journal on Advances in Signal Processing*, 2009(729494):1–11.