Hybrid System for Automatic Music Transcription

Vasco Barros
vasco.barros@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

May 2016

Abstract

The task of automatically transcribing a piece of music is a very challenging one. It implies sound and music perceptiveness which has been proving hard to replicate into machines. There are multiple methods to address sub-problems within this task, achieving successful results. In this thesis a hybrid system for Automatic Music Transcription is proposed, combining two distinct Machine Learning techniques. A state-of-the-art spectrogram factorization technique based on Probabilistic Latent Component Analysis is implemented. This method uses a pre-extracted template library of instruments and their notes to perform the transcription. The template library greatly impacts the transcription process. As such, to automatically determine the correct library size to be used, a Deep Neural Network was trained as a classifier, to identify instruments performing in a sound file. By combining both mentioned techniques, a hybrid transcription system is created that eliminates the need for a manual instrument identification for each considered sound file. This hybrid system proves that combining distinct Machine Learning methods it is possible to improve the transcription process granting it more autonomy. In this case, the proposed system ensures the same transcription accuracy of the Probabilistic Latent Component Analysis method, while adding a higher degree of autonomy in the process, obtained through the automatic instrument identification performed by the trained neural network.

Keywords: Automatic Music Transcription, Machine Learning, Probabilistic Latent Component Analysis, Deep Learning, Convolutional Neural Networks, Hybrid system

1. Introduction

Musical transcription is the process of converting a piece of music into some form of musical notation, which will display the musical notes played across time. Some examples of musical notations are scores, piano-roll representations or rhythmic sequence of chords [19]. Even for those with musical training, listening to a piece of music and manually trying to transcribe it presents itself as a very challenging task. There are several obstacles one may find while performing this task, such as detecting which instrument plays each note or detecting the tempo/beat of each note, but the main challenge is to detect the the note’s pitch.

In the late 70s, audio researchers such as James Moorer, Martin Piszczalski or Bernard Galler, dedicated their research to musical signal analysis. In 1977 Piszczalski and Galler introduced the concept of Automatic Music Transcription (AMT) [19]. This is the process of automatically converting a musical sound signal into its representation as musical notation, through digital analysis of the musical signal. This process has been the target of much research since it was introduced, and nowadays it covers a wide range of subtasks. These sub-tasks are representation of the challenges we face, with the added difficulty of removing human intuition and perception out of the equation. As such, AMT can be viewed as one of the main technology enabling concepts in music signal processing [3].

In the past decades, another field of study has made great achievements in having computers learning from representations of our world, trying to mimic Human learning processes. This field of study — Machine Learning — focus on adding cognitive skills to machines by modelling learning processes [7]. AMT and Machine Learning are intrinsically connected, in the sense that the high-level goal is to make computer perceive and interpret music on its own. As such, many methods developed to address AMT tasks are based on Machine Learning algorithms.

In this thesis the task of automatically transcribing a music signal is addressed. An automatic transcription system is fully designed and implemented. This system will generate a piano-roll representation of notes of one or more instruments performing in a given input file. This system aims to provide support to every musician in their frequent transcription tasks, by automatically registering their
The proposed system will combine distinct methods applied to the AMT process, to generate a fully functional system. This hybrid approach aims to explore the benefit of combining distinct approaches to specific AMT tasks, in order to improve the transcription process. Thus, in this thesis an hybrid system is proposed to automatically transcribe recorded music fragments. This hybrid system will combine two distinct Machine Learning methods, each one addressing a distinct AMT subtask: MultiPitch Estimation and Note Tracking, and Instrument Identification.

1.1. State of the Art
When AMT arose, Piszczalski and Galler focused on transcribing monophonic pieces of music — signals from one instrument source only [19] — but after three decades of research this problem is considered solved [3]. The current challenge resides in automatically transcribing polyphonic music signals — music with several instruments. As such, in polyphonic music we are interested in detecting concurrent pitches, from the same instrument or from multiple distinct instruments. This challenge is referred to as Multi-Pitch Estimation.

A large subset of the state of the art Multi-Pitch Estimation methods focus on two techniques: Non-Negative Matrix Factorization (NMF) and Probabilistic Latent Component Analysis (PLCA). Both these methods can be included in the Spectrogram Factorization group. NMF is a matrix factorization technique where the matrices have no negative values, characteristic that is exploited the factorization process. PLCA takes a probabilistic approach in the spectral factorization task, having achieved state of the art results. In [4] Benetos, Ewert and Weyde propose a PLCA based model for transcribing jointly pitched and unpitched sounds, showing the effectiveness of this technique in regular western music inputs. In [12] an algorithm for Shift-Invariant PLCA is presented, the implemented method could tolerate variations of the spectral envelopes (tuning deviations).

Given a polyphonic music, where multiple instruments play at the same time, the task of identifying which instrument is playing consists on one of the main challenges inside the scope of AMT. Traditional MIR methods focus mainly on two stages: feature extraction and semantic interpretation. Extracting good features is very time consuming, but ultimately it will lead to a good representation of the input signal. These features extractions tend to be task specific and hard to optimize. As such, MIR researchers tend to adopt more powerful semantic interpretation strategies, like Multilayer Perceptrons and Support Vector Machines. In Instrument Identification (and in other main AMT tasks), multiple feature extractions approaches were implemented and perfected in order to achieve better data representations. This is the case of the widely utilized Mel-frequency Cepstral Coefficients [26], which consist on an attempt to define and characterize the timbre of an instrument. Combining these features extractions with the previously mentioned semantic interpreters achieved satisfying results [11].

However, recent studies prove that combining traditional shallow methods with Deep Learning techniques, thus obtaining deeper architectures, allows better high-level representations and, in the end, better results [15]. Deep Learning is a Machine Learning technique, based on Neural Networks, that provides high-level concept learning through multiple layer learning (hence deep). Deep Learning techniques are conquering their space in Instrument Identification, with several methods proving to be more accurate than traditional shallow approaches [13]. Convolutional Neural Networks (CNN) are a specific type of Neural Networks that is widely used in Image Recognition, due to it’s great performance in this task. These Neural Nets exploit the convolution operation properties in order to reduce memory usage and improve performance. Li, Chan, and Chun stated that musical patterns can be captured using CNN due to the similarities between musical data and image data [18]. The authors implemented a CNN for Music Genre Classification. Their implementation required minimal prior knowledge to be constructed and was complemented with the usage of classic features like MFCC.

2. Background
As mentioned above, the proposed method consists on an implementation of two distinct Machine Learning techniques. Both techniques will be addressed in the remainder of this section.

2.1. Multi-Sample Shift Invariant Probabilistic Latent Component Analysis
The Multi-Sample Shift Invariant Probabilistic Latent Component Analysis (MSSIPLCA) model was proposed by Benetos, Ewert and Weyde in [4], and it is based in Probabilistic Latent Component Analysis.

The input to the model is a log-frequency spectrogram, interpreted as a probability distribution across log-frequency $\omega$ and across time $t$ (which is a strong assumption). The log-frequency spectrogram is then represented as $V_{\omega,t}$ and a probability distribution is represented as $P(\omega; t)$. The probability distribution is then decomposed into the known quantity of the frame probability $P(t)$ and into the conditional distribution over log-frequency $P(\omega|t)$.
h bins $P(\omega|t)$:

$$V_{\omega,t} \approx P(\omega, t) = P(t)P(\omega|t)$$ (1)

The conditional distribution over log-frequency bins is then decomposed into two components: a pitched component and an unpitched component. The resulting decomposition is described in the following Equation:

$$P(\omega|t) = P(r = h|t)P_h(\omega|t) + P(r = u|t)P_u(\omega|t)$$ (2)

where $P_h(\omega|t)$ is the spectrogram approximation to the pitched component and $P_u(\omega|t)$ is the spectrogram approximation to the unpitched component. The probability $P(r|t)$ weighs the respective component over time, having $r \in \{h, u\}$ for the pitched and unpitched components respectively.

Considering only the pitched component $P_h(\omega|t)$, as in [2], a latent variable $p \in \{21, \ldots, 108\}$ that represents the pitch using the MIDI scale for pitch, is added to the model. The resulting pitched component is:

$$P_h(\omega|t) = \sum_p P(\omega|p, t)P(p|t)$$ (3)

Additionally, a latent variable for instrument sources $s$ which represents the instrument index; and a latent variable for pitch shifting across log-frequency $f$ the shifting parameter, are also added to the model. Obtaining:

$$P_h(\omega|t) = \sum_{p,s} P_h(\omega|s, p) *_{\omega} P_h(f|p, t)P_h(s|p, t)P_h(p|t)$$ (4)

where $P_h(\omega|s, p)$ represent the spectral templates for a given pitch $p$ and for a specific instrument $s$, $P_h(f|p, t)$ represents the time-varying log-frequency shift per pitch which is convolved with $P_h(\omega|s, p)$ across $\omega$ (operator $*_{\omega}$), $P_h(s|p, t)$ represents the instrument contribution per pitch across time and $P_h(p|t)$ is the pitch activation across time.

Finally, removing the convolution operator in Equation (4), we get the following model for the pitched component:

$$P_h(\omega|t) = \sum_{p,s,f} P_h(\omega - f|s, p)P_h(f|p, t)P_h(s|p, t)P_h(p|t)$$ (5)

In order to reduce the computational effort of the following steps of parameter estimation, the use of pre-extracted and pre-shifted templates is introduced [4, 1]. Thus, with this modification, the proposed model for the pitched component is described as follows:

$$P_h(\omega|t) = \sum_{p,s,f} P_h(\omega|s, p, f)P_h(f|p, t)P_h(s|p, t)P_h(p|t)$$ (6)

where $P_h(\omega|s, p, f)$ are the spectral templates per pitch $p$ and instrument $s$, shifted across log-frequency according to $f$; $P_h(f|p, t)$ represents the time-varying log-frequency per pitch; $P_h(s|p, t)$ represents the instrument contribution per pitch across time and $P_h(p|t)$ is the pitch activation. The time-frequency representation, as in [4], has a spectral resolution of 5 bins per semi-tone, thus having $f \in \{1, \ldots, 5\}$ allowing the templates to by shifted by $\pm \frac{1}{2}$ semi-tones and having the ideal tuning position at $f = 3$.

For the unpitched component, 2 latent variables were added: $d$ which denotes the drum kit component utilized and $z$ which is the index for the templates used for each component. Applying the same process as the pitched component to the unpitched component yields the following decomposition:

$$P_u(\omega|t) = \sum_{a,z} P_u(\omega|d, z)P_u(d|t)P_u(z|d, t)$$ (7)

where $P_u(\omega|d, z)$ denotes the $z$-th spectral template for the drum component $d$, $P_u(d|t)$ represents the drum component activation and $P_u(z|d, t)$ denotes the template contribution per drum component over time. The overall mathematical model is obtained when both components are considered (Equations (6) and (7)).

This mathematical model has several parameters, some of which are fixed and known while the others are unknown. The next step of the proposed MSSIP-LCA method proposed in [4], is to estimate the unknown parameters. As mentioned, the parameters $P_h(\omega|s, p, f)$ and $P_u(\omega|d, z)$, are fixed and known, they correspond to the pre-extracted and pre-shifted templates.

To estimate the unknown parameters, the Expectation-Maximization algorithm (EM) is used [9]. The model’s log-likelihood is defined as:

$$\mathcal{L} = \sum_{\omega,t} V_{\omega,t} \log(P(\omega, t))$$ (8)

EM is divided into two distinct steps. In the Expectation step the contribution of the latent variables is estimated by a weighting function. This process results in the following Equations, for the pitched component and for the unpitched component:

$$P(s, p, f, r = h|\omega, t) = \frac{P(r = h|t)P_h(\omega|s, p, f)P_h(f|p, t)P_h(s|p, t)P_h(p|t)}{P(\omega|t)}$$ (9a)

$$P(d, z, r = u|\omega, t) = \frac{P(r = u|t)P_u(d|t)P_u(z|d, t)}{P(\omega|t)}$$ (9b)

In the Maximization step, the marginals will be re-estimated, but this time with the estimations calculated in the Expectation step. Thus, resulting in the following Equations for the pitched component:
Neural Network (CNN). Convolutional Neural Networks are feed-forward deep neural networks. They are inspired by the architecture of the visual cortex of animals, namely the cat’s visual cortex (as in [14]). CNNs are considered to be among the best pattern recognition systems [5]. This can be seen in the handwritten character recognition task, where in 1998 LeCun et al. developed a benchmark system with state-of-the-art performance [16].

2.2. Convolutional Neural Network

The second technique utilized is a Convolutional Neural Network (CNN). Convolutional Neural Networks are feed-forward deep neural networks. They are inspired by the architecture of the visual cortex of animals, namely the cat’s visual cortex (as in [14]). CNNs are considered to be among the best pattern recognition systems [5]. This can be seen in the handwritten character recognition task, where in 1998 LeCun et al. developed a benchmark system with state-of-the-art performance [16].

Regular neural networks, as the MultiLayer Perceptron, take the input data and, through a series of calculations in the hidden layer, generate an output. These hidden layers consist in several neurons (e.g. the perceptron). These neurons are fully-connected to the previous layer neurons. When the input of the net is an image, it isn’t hard to see that this fully-connected architecture won’t scale properly, with the parameters adding up through the layers. This particularity makes training an arduous and computationally expensive process [23].

In CNNs the inputs are interpreted as images, having 3 dimensions: width, height and depth (with the latter corresponding to the red, green and blue channels when dealing with real images). Thus, the neurons in a CNN will also have these 3 dimensions. The neurons also have the particularity of only being connected with a specific spatial region of the previous layer. This architecture scales well with input images and allows the net to be trained, unlike fully-connected networks where training in this conditions would be very difficult or even impossible [5].

There are three main types of layers when building a CNN: Convolutional layers, Pooling layers and Fully-connected layers. In the following section these fundamental layers will be described alongside with other layer types that can be applied to CNNs. Stacking multiple layers with different type combinations will generate a fully functioning CNN architecture.

Convolutional layers are the fundamental piece of CNNs. They are composed by a set of parameters that consist in a set of weights, often called filters or kernels. As in neural networks, these weights can be updated in a training process to learn different representations of data. A filter has a small width and height compared to the input, but it has the same depth. In the forward pass step of the learning process, each filter is convolved across the width and the height of the input, hence the name convolutional network. As the filter passes through the input image, the filters will be updated in order to be activated when a certain feature arises in a specific spatial location. This process creates an activation map. These activation maps may also be followed by an element-wise activity function.

Each neuron’s output can be interpreted as the result of a neuron analysing a small spatial location [23]. This spatial location is called the receptive field, which dictates the size of the spatial location to be analysed by the neuron. This denotes an important property of CNN: the neurons are locally connected.

The full output of the layer consists in stacked activation maps, creating a 3-dimensional output. The width and the height of this output is given by the convolution operation between the filters and the input. The depth of this output is a chosen quantity. It denotes how many neurons it is desired to analyse the same spatial location. This group of neurons can be interpreted as a depth column.

Each depth column has a spatial location assigned. These spatial locations often overlap, causing different depth columns to analyse the same partial spatial location. This overlap is dictated by the stride quantity. For example, if the stride is set to 1, a new depth column will have as a spatial location, a spatial location 1 spatial unit apart from the previous one. As the convolution operation takes the input data and, through a series of calculations in the hidden layer, generate an output. These hidden layers consist in several neurons (e.g. the perceptron). These neurons are fully-connected to the previous layer neurons. When the input of the net is an image, it isn’t hard to see that this fully-connected architecture won’t scale properly, with the parameters adding up through the layers. This particularity makes training an arduous and computationally expensive process [23].

In CNNs the inputs are interpreted as images, having 3 dimensions: width, height and depth (with the latter corresponding to the red, green and blue channels when dealing with real images). Thus, the neurons in a CNN will also have these 3 dimensions. The neurons also have the particularity of only being connected with a specific spatial region of the previous layer. This architecture scales well with input images and allows the net to be trained, unlike fully-connected networks where training in this conditions would be very difficult or even impossible [5].

There are three main types of layers when building a CNN: Convolutional layers, Pooling layers and Fully-connected layers. In the following section these fundamental layers will be described alongside with other layer types that can be applied to CNNs. Stacking multiple layers with different type combinations will generate a fully functioning CNN architecture.

Convolutional layers are the fundamental piece of CNNs. They are composed by a set of parameters that consist in a set of weights, often called filters or kernels. As in neural networks, these weights can be updated in a training process to learn different representations of data. A filter has a small width and height compared to the input, but it has the same depth. In the forward pass step of the learning process, each filter is convolved across the width and the height of the input, hence the name convolutional network. As the filter passes through the input image, the filters will be updated in order to be activated when a certain feature arises in a specific spatial location. This process creates an activation map. These activation maps may also be followed by an element-wise activity function.

Each neuron’s output can be interpreted as the result of a neuron analysing a small spatial location [23]. This spatial location is called the receptive field, which dictates the size of the spatial location to be analysed by the neuron. This denotes an important property of CNN: the neurons are locally connected.

The full output of the layer consists in stacked activation maps, creating a 3-dimensional output. The width and the height of this output is given by the convolution operation between the filters and the input. The depth of this output is a chosen quantity. It denotes how many neurons it is desired to analyse the same spatial location. This group of neurons can be interpreted as a depth column.

Each depth column has a spatial location assigned. These spatial locations often overlap, causing different depth columns to analyse the same partial spatial location. This overlap is dictated by the stride quantity. For example, if the stride is set to 1, a new depth column will have as a spatial location, a spatial location 1 spatial unit apart from the previous one. As the convolution operation
changes the size of the input image, zero-padding can be triggered to prevent this from happening. Zero-padding consists in adding zeros to the spatial borders of the input. It allows the control of the output dimensions [23].

The previous quantities denote the hyper-parameters of a convolutional layer. A hyper-parameter can be interpreted as a high-level parameter that will influence the model’s performance. The output size can then be computed with the hyper-parameters and the filter information [8], as follows:

\[
O_{h,w} = \frac{I_{h,w} - F_{h,w} + 2P}{S} + 1 \quad (12a)
\]

\[
O_d = K \quad (12b)
\]

where \(O_{h,w}\) denotes the output height and width (that are calculated equally), \(O_d\) denotes the output depth, \(I_{h,w}\) is the input height and width, \(F_{h,w}\) is the filter’s height and width, \(P\) is the amount of zero-padding used, \(S\) is the stride and \(K\) is the number of filters used.

Another important property of convolutional layers is their parameter sharing characteristic. This characteristic is based on the assumption that if it’s useful to calculate a set of features in one position, it is also useful to calculate it in the remaining positions. This means that, for a fixed depth, all neurons share weights and bias, thus reducing the number of parameters and facilitating the learning process. In practice, during the Backpropagation phase each neuron calculates its weight’s gradient, but in the end all gradients will be added up in each existing depth level. Since all the neurons in a depth level share the same weights, then the forward-pass of the layer in each depth level is the convolution between the neuron’s weights (filters) and the input volume. This process results in an activation map, and the set of all activation maps for all depth levels creates the output volume of the layer [23].

A Pooling layer is a layer that is inserted with the intent to achieve spatial invariance by reducing the spacial size of its input. It downsamples the input resolution reducing the computational effort of training the the network and providing overfitting control [20]. Due to it’s success in capturing invariances in image-like data, Max Pooling is the most commonly applied pooling operation [20]. Max pooling applies filters with stride to the input function. The filters define the spatial region where the maximum operator will be used. In Equations (13) the output size of the Pooling layer is computed.

\[
O_{h,w} = \frac{I_{h,w} - F_{h,w}}{S} + 1 \quad (13a)
\]

\[
O_d = 1 \quad (13b)
\]

As mentioned above, the map output is reduced, thus down-sampling the image (Equation (13a)), while it’s depth remains constant (Equation (13b)).

One major concern while dealing with large and complex deep neural networks is overfitting. Overfitting occurs when the network has enough complexity to memorize the training data, losing its generalization capabilities. Overfitting occurrence probability increases with the increasing size of the network [24].

The dropout technique was introduced to address the overfitting concern in artificial neural networks [22]. With this technique overfitting is prevented by temporarily removing some neurons and their connections, in the training process. The dropped units are randomly chosen and each unit is retained with a fixed probability (usually 0.5), independent of other units. As stated in [22], neural networks using the dropout technique can be trained in a similar manner as regular neural networks. The only difference is that the Forward and Backpropagation pass steps are applied to the sample networks instead of the original network.

Fully-connected layers, as the name implies, are layers whose neurons are fully connected to the previous layer neurons. These are standard neural network layers. They are usually employed in the last layers of the CNN architecture, to provide a high-level insight of the input data.

A Loss layer is the last layer of a neural network architecture. It generates the final output, thus the classification. It consists on a Fully-connected layer with an applied loss function. The standard function is the Softmax loss function [17].

\[
f(z)_j = \frac{x^j}{\sum^K e^{x_k}} \quad j = 1, \ldots, K \quad (14)
\]

3. Implementation

The developed program consists on two modules. The first module makes use of the MSSIPLCA technique described in section 2.1 to transcribe automatically the pitch of the input sound. In order to transcribe the pitch of an instrument this technique requires a spectral template of the instrument being played. As such, given an unknown polyphonic input sound several templates must be provided to the module, one for each instrument. To automatically detect which spectral templates must be provided, another module was developed. This second module consists on a Convolutional Neural Network.
Classifier (described in section 2.2 that was previously trained to detect instruments in polyphonic sounds. Thus, these modules combined can receive a unknown polyphonic sound, detect which instruments are being played and through spectral templates transcribe each instrument’s performance.

### 3.1. MSSIPLCA Module

To implement the MSSIPLCA method, the toolbox provided in [4] was used (MSSIPLCA Toolbox). This toolbox contained a demo of an implementation of the aforementioned model which was adapted to consider dynamic template libraries and was parametrized with the desired sample frequency, spectral resolution and audio input size. To calculate the CQT, the Matlab toolbox provided in [21] is used (CQT Toolbox).

Using a digital instrument library, 82 templates were extracted combining 42 different instruments sources with 8 different playing characteristics (e.g. vibrato and staccato) obtaining a total of 17 generic instruments (e.g. for a piano we can have distinct piano sources, generating one generic instrument). An interval of 88 notes was considered, limiting the notes played to the following interval $p \in \{21, \ldots, 108\}$, in MIDI scale values. To extract a pitched template from an audio file, a variant of the PLCA algorithm was used with only one latent variable component, which denotes the pitch. After extracting the pitched template through the process described above the templates where shifted across the log-frequency. Thus, obtaining a pre-extracted and pre-shifted template per pitch and per instrument. After computing all the templates all of them were saved in a matrix, creating a template library.

The log-spectrogram of the input music signal is obtained using CQT from the input signal. In this time-log-frequency representation the relative distance between the harmonics is constant, this can be seen in Figure 1. In this Figure, the log-spectrogram of a piano performing every note individually is displayed.

After applying the MSSIPLCA and estimating the unknown parameters the transcriptions can now be extracted. The transcriptions are extracted as MIDI-scales. The total pitched component transcription and the unpitched component transcription can be extracted as, respectively:

$$P_h(p, t) = P(t)P(r = h|t)P_h(p|t)$$  \hspace{1cm} (15a)
$$P_u(d, t) = P(t)P(r = u|t)P_u(d|t)$$  \hspace{1cm} (15b)

To extract the transcription for each instrument source, the latent variable $s$ should be fixed to the target instrument index $\lambda$, and the the following calculation should be performed:

$$P_h(s, p, t) \rightarrow P(t)P(r = h|t)P_h(p|t)P_h(s = \lambda|p, t)$$  \hspace{1cm} (16)

Performing this calculation yields a piano-roll like matrix, which contains a raw transcription output. In order to obtain a good piano-roll transcription, some post-processing steps are performed. The first step consists in normalizing the raw output. The normalization is performed with the following operation:

$$\frac{P_h(s, p, t)}{\max\left(\sum_{s, p, t} |P_h(s, p, t)|\right)}$$  \hspace{1cm} (17)

After having a normalized raw transcription thresholding is performed. Given a threshold parameter $\sigma$, the transcription is converted into a ones and zeros matrix. If the value of the raw transcription surpasses the $\sigma$ value the output is 1 and 0 otherwise. Lastly, since the minimum duration of a note is defined to be 0.2 seconds, events with duration inferior to 80ms are removed in an effort to eliminate small transcription errors.

Test experiments in this module showed that the library size directly influences the performance of the module. Choosing a large library or too small will result in higher transcription errors. This makes the module directly dependant of the library. In order to grant autonomy to the module and to remove human interaction from this transcription process, an instrument classifier was developed as it will be presented ahead.

### 3.2. Classifier Module

The CNN was developed and trained using the MatConvNet Toolbox [25]. Once again, to compute the
log-spectrogram the CQT Matlab toolbox provided in [21] was used.

A CNN classifier was trained to detect notes of one of three chosen instruments in log-spectrograms of 1.2 seconds. Then this classifier was applied to a musical signal as a windowed function. After a normalization process, and given a classification threshold \( \mu \), an output classification vector is generated. This vector consists of three binary outputs each one corresponding to an instrument, and has a 0 or 1 value whether the instrument is present or not in the input signal.

![Figure 2: Diagram of the implemented CNN's architecture.](image)

The CNN will receive only raw spectrograms as an input. This proved to be a challenging task, as the input data is very complex. The developed net is presented in Figure 2. It has 12 layers: 3 Convolutional layers, 1 Max-pooling layer, 2 Convolutional layers, 1 Max-pooling layer, 1 Convolutional layer, 1 Fully-connected layer, 1 Dropout layer, 1 Fully-connected layer and finally 1 Softmax layer (layers presented from the shallowest to the deepest). All Convolutional and Fully-connected layers were each followed by a ReLU non-linearity. In Figure 2 the feature map sizes can be seen.

The classifier was trained to only classify among three instruments. These instruments were chosen by their digital audio quality, sustain capability and by their distinct sound characteristics. The selected instruments are Bass, Oboe and Violin. The data set was separated into a training data set and into a validation data set, ensuring that in both cases the 3 classes were always equally represented. The training data set contained \( \frac{2}{3} \) of the full data set, and the validation set contained the remainder of the full data set. Also to create a test data set, for each instruments 2 notes from outside the selected range were chosen and suffered the same process mentioned above. This generated a test data set of 60 spectrograms.

The CNN was then submitted to a learning process. It took approximately 6 hours to complete. Then the trained CNN was used to classify the test data set composed of unseen notes from the three instruments considered. The resulting test error was 23.33\%, the classifier correctly classified 46 of the 60 notes contained in the test data set.

As in the previous Module, the log-spectrogram input signal was obtained using the CQT. The input signal was then sampled in 1.2 seconds segments, which were fed to the classifier. For each segment the classifier will output a classification probability for each of the three classes, acting as a windowed function. The overall output is a \( 3 \times N \) matrix where \( N \) is the number of samples. Each of the 3 components of this matrix contains the presence probability over time of the respective instrument.

The output matrix of the classifier’s windowed function like process is then normalized by subtracting the mean classification of each component. This process aims to enhance the occurrence of the highest probability classifications, thus ignoring the average classification which may induce in error. The normalized output is then submitted to a classification threshold \( \mu \).

3.3. Hybrid System
The complete system is composed by both Modules addressed earlier. Given an input signal, its CQT is computed and then is analysed by the classifier in the CNN module. This classifier will segment the log-spectrogram produced, and it will identify one of three instruments in each segment. This will produce a matrix containing the probability of the presence of each instrument in the segments considered acting as a windowed function. This matrix is then normalized by removing the mean probability for each instrument. Then if the obtained value exceeds a classification parameter \( \mu \), the instrument is considered in this input file. The final output of this Module is a binary vector of three values, one for each instrument, determining if an instrument is present (1 valued) or not (0 valued) in the input file. The output of the CNN module and the log-spectrogram of the input file are both received by the MSSIPLCA module. This module will then use the classification vector to determine the size of its template library. The library will only contain templates for the instruments classified as present in the input file. The transcription is the performed using this dynamically set template library.

A diagram of the overall system can be observed in Figure 3. The transcription performed by this system is autonomous, and it only depends on the hyperparameters that affect both modules. After tuning this parameters, it does not need human interaction to perform transcription.

4. Results
To evaluate the performance of the hybrid system a test experiment was conducted. The aim of this experiment is to evaluate the transcription performance when the contribution of both modules.

The data set created for this experiment consists
of 30 random sound files based on random MIDI files. Again, the three levels of polyphony were considered: sound files with 1 instrument, with 2 instruments and with 3 instruments. Each polyphony level represents 1/3 of the dataset. The instruments considered were the 3 instruments that the classifier was trained to identify. The sound files had the duration of 20 seconds and the note’s duration was a random value in [0, 2s, 4s].

4.1. Metrics
In order to evaluate the accuracy of the transcription the following metric was applied. The time interval of each note in the ground truth is inspected in the transcription result, with a tolerance of 40 ms. If a note is present in this time interval, with a duration of over δ(%) of the original interval, then this detected note is considered correctly transcribed. This process detects accurate transcribed notes as well as false negative transcriptions. To detect false positive transcriptions the same process was implemented but this time by inspecting the time interval of each detected note of the transcription result, in the ground truth file. Through a fine-tuning process the parameter δ was fixed at 75% for both metric processes. This value of δ provides a plausible accuracy consideration, as it is not excessively high, ignoring notes not fully transcribed, nor excessively low, considering small transcription errors. To prevent false detections due to temporal synchronization issues, prior to the evaluation, both the ground truth and the transcription are aligned temporally.

To evaluate the overall performance, the sum of false negatives, $\sum FN$ and false positives, $\sum FP$, was divided by the total correct notes, $N$, thus generating an error measure, $\varepsilon$.

$$\varepsilon = \frac{\sum FN + \sum FP}{N}$$  \hspace{1cm} (18)

This designed formula for measuring error (Equation 18) consists on a simple arithmetic computation which takes into account both types of errors, false negatives and false positives, and provides a ratio of the total errors occurred versus the number of existing notes to be detected in the input music piece.

4.2. Experiment Results
The experiment ran for the 30 files with different values of $\mu \in [0, 0.05]$, (with $\mu$ denoting the classification threshold parameter introduced in the CNN Module). The overall result can be observed in Figures 4 and 5. Inspecting the graph in Figure 5 that plots the mean classification error against the different $\mu$ values, it can be seen that the best result was obtained for $\mu = 0.02$, corresponding to an error of $\varepsilon = 0.433$.

Figure 3: Diagram of the proposed hybrid system.

Figure 4: Transcription error for the 3 polyphony levels considered.

Figure 5: Mean Transcription error obtained for all polyphony levels.
probability values, which will lead to consider instruments that have been misclassified and have a residual probability value. Considering these instruments, will add to the final transcription falsely detected notes (transcription errors). Due to the chosen metric, these falsely detected notes will be considered errors (false positives) as well as the remainder errors generated during the transcription of the instruments that actually are present in the music piece, (with high probability values). Thus, the ratio between all the errors considered versus the number of existing notes can achieve values superior to 1.

Setting a high value of $\mu$ provides a conservative classifier, that will only consider instruments with high probability values. Once again, inspecting the graph in Figure 4, it can be seen that high value of $\mu$ will discard instruments that although having a lower than $\mu$ probability value, are present in the sound file increasing the false negative transcription notes as all the notes performed by the discarded instrument are not transcribed. These disregarded notes plus the transcription errors of the considered instruments cause an error of over 100%.

Using an implementation of the state-of-the-art MSSIPLCA algorithm proves that transcribing polyphonic signals, even when the instruments are priorly known, is a very challenging task. In the performance evaluation of the algorithm the average of the transcription error obtained is 35.90%, always considering that the instruments being played are known. In the proposed hybrid system no prior information regarding which instrument is playing was used. Instead the CNN module detects which instruments are playing, choosing the corresponding instrument from it’s instrument library. Thus, this improvement while adding autonomy to the system also adds uncertainty. The best overall result was obtained for $\mu = 0.02$, which can be seen in Figure 5. The obtained $\epsilon = 0.433$ error, although not as low as the best average error obtained — 35.90% — allows the system to detect which instruments are playing among it’s template library, increasing the average error by 7.4%.

5. Conclusions

The developed system consists in a hybrid implementation of two distinct methods. The first method implemented is a state-of-the-art spectrogram factorization technique developed by Benetos et al. [4], named Multi Sample Shift Invariant Probabilistic Latent Component Analysis. This method uses a pre-extracted template library (of instruments and their notes) to perform Multi-Pitch Detection as well as Note Tracking. After evaluating the performance of the aforementioned module, it was found that the size of the template library considered in the transcription process would impact the resulting transcription. To address this issue and to automatically select the appropriate templates, a classifier was designed to perform instrument identification.

The designed classifier is a Convolutional Neural Network, a Machine Learning technique. A CNN was designed with 12 layers and it was successfully trained to identify individual notes of 3 distinct instruments. The proposed system was then assembled, using a module to perform Multi-Pitch Detection and Note Tracking (MSSIPLCA module), but this time with a template library defined by the classification output of another module containing the developed CNN (CNN module). The system’s overall result is a transcription error of approximately 40%. Using only an implementation of the state-of-the-art MSSIPLCA algorithm, with prior information regarding which instrument is present in the considered music piece, a mean transcription error of approximately 35% was achieved, showing the difficulty of transcribing polyphonic music signals. The proposed module removes the need of this prior information regarding which instruments are playing, while increasing the average transcription error by a small percentage (¡10%).

Thus, the proposed hybrid system successfully performs an automatic transcription of a given input file. It achieves a transcription error approximately similar to the transcription error presented by Benetos et al. method [4]. Although it does not particularly improve the transcription error of this method, it additionally performs Instrument Identification via a CNN. With this new task considered, the hybrid system now combines two distinct methods in order to improve the transcription process. The system can now decide on its own the instruments to be considered, providing a more automatic transcription process.

References


