Polyphonic music transcription through dynamic networks and spectral pattern identification*

Antonio Pertusa and José M. Iñesta
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante, Spain
{pertusa,inesta}@dlsi.ua.es

Abstract

The automatic extraction of the notes that were played in a digital musical signal (automatic music transcription) is an open problem. A number of techniques have been applied to solve it without concluding results. This work tries to pose it through the identification of the spectral pattern of a given instrument in the signal spectrogram using time-delay neural networks. We will work in the monotimbrical polyphonic version of the problem: more than one note can sound at the same time but always played by just one instrument. Our purpose is to discover wether a neural network fed only with an spectrogram can detect the notes of a polyphonic music score. In this paper our preliminary but promising results using synthetic instruments are presented.

1 Introduction

Tone perception is a complex phenomenon [5]. Human ear can detect musical tones even in presence of noise. We can also hear a number of simultaneous tones and detect subtle but expressive tonal deviations (vibrato, microtonal intervals, ...). A problem related to this ability in computer science is the automatic score extraction from digitized music or, for short, music transcription.

Music transcription is defined as the act of listening to a piece of music and writing down music notation for the notes that make up the piece [12]. The automatic transcription of monophonic signals (only one note playing simultaneously) is a largely studied problem. Several algorithms have been proposed that are reliable, commercially applicable, and operate in real time. Nevertheless, automatic polyphonic music transcription is an open research problem, because not even the perceptual mechanisms involved in the isolation of different notes and instruments and their insertion in the corresponding musical phrases are clear. This fact causes a lack of computational models to emulate these processes.

*This work has been funded by the Spanish CICYT project TAR; code TIC2000–1703–CO3–02

The first polyphonic transcription system dates back to the 1970s, when Moorer [13] built a system for transcribing two-voice compositions. Recent state-of-the-art in music transcription has been discussed in [9]. There is still not a method to solve completely this problem, but there are mainly two pitch detection method groups: time domain methods and frequency domain methods. Time domain methods are useful for monophonic transcription, but they have shown poorer results for polyphonic transcription than frequency methods.

Most of time domain methods are based on autocorrelation [2], time domain peak and valley measurements, or zero-crossing measurements. Frequency domain methods look for fundamental frequencies, whose harmonics best explain the partials[1] in a signal [7].

In this paper we discusse a frequency domain method based on dynamic neural networks. Connectionist approaches have been used in music for a long time [16, 17] and they have been also used for music transcription [15], recently with polyphonic piano music [11], and they present a good alternative in building transcription systems. In the latter work, Marolt's system (SONIC) tracks groups of partials in piano music with 76 adaptive time-delay neural networks (TDNN) and postprocessing, obtaining good results. TDNN have also been used earlier in speech recognition problems [6, 10].

The spectral pattern of a given sound signal $s(t)$ is the energy distribution that can be found in the constituent partial frequencies of its spectrum, $S(f)$. This pattern is the most important parameter for characterizing the musical timbre. In this work, the music transcription problem is posed through the identification of the spectral pattern of the instrument, using a connectionist dynamic algorithm like time-delay neural networks previously trained with spectrograms computed from polyphonic melodies played by the target instrument. We will only consider tuned sound sources, those that produce a musical pitch, leaving apart those produced by random noise or highly inharmonic sources.

[1] A "partial" is any of the frequencies in a spectrum, being "harmonic" those multiples of a privileged frequency called fundamental that provide the pitch of the sounding note.

In constrast with Marolt's work, we only use one neural network, and our aim is to use this single network to detect notes sounding at each time, their beginnins and ends. Input data of our network are spectrograms without postprocessing, we don't use any kind of auditory model. The purpose of our work is to discover wether a neural network by itself, with the only knowledge of spectra, can detect notes in a polyphonic music score.

We will work only with smooth envelope timbres (i.e. nearly stable timbres from the beginning to the end of each note), and start and end times will be considered the same way as the rest of the sounding note.

To achieve this goal, we need to build input and output pairs formed by the spectra of the sound produced by a source for different times around a given instant t_i. Input is $\{S(f, t_{i+j})\}$ for $j \in [-m, +n]$ for each frequency f, being m and n the number of windows considered before and after the central time, t_i. Output consists in a coding of the set of possible notes $\nu(t_i)$ that are active at that moment in order to produce those spectra.

After the learning phase of the spectral pattern, it is expected that the net will be able to detect the notes in a digitization of a melody produced by that sound source by occurrences of the pattern. It would be desirable a robustness of the method against overlapped patterns (polyphony) and, hopefully, applicable to patterns produced by other instruments of similar timbres.

TDNN are usually considered as non-recurrent dynamic networks [8], although essentially are like static nets traversing temporal series. This kind of nets are able to model systems where the output $y[t]$ has a dependence to a limited time interval in the input $u[t]$:

$$y(t) = F\left[u(t-m), ..., u(t), ..., u(t+n)\right]$$

With this kind of network, time series can be processed as a collection of static input-output patterns, related in short-term as a function of the width of the input window. Due to the absence of feedback, the net architecture is the same as that of a multilayer perceptron and it can be trained by a standard backpropagation algorithm [14]. We have trained the network with different synthetic timbres with promising results.

2 Methodology

2.1 Construction of the input-output pairs

The training set has to be formed by pairs $\{\{S(f, t_{i+j}), j \in [-m, +n]\}, \nu(t_i)\}$. We need to have a set of music scores and synthesize sounds according to the instructions in them in such a way that the correspondence between the spectrum and the set of notes that have produced it is kept at every moment. For this, we have used MIDI files and a software synthesizer developed by the Medialab at MIT named Csound [18, 1].

First we will get into the details of the input data construction and then we will describe the training outputs.

2.1.1 Input data.

From the MIDI sequence, the digital audio file is synthesized and the short-time Fourier transform (STFT) is computed, providing its spectrogram $S(f, t)$. The STFT has been computed used a Hanning window, described at instant τ by this expression:

$$w(\tau) = \frac{1}{2}\left(1 - \cos\frac{2\pi\tau}{N}\right)$$

where $N = 2048$ is the number of samples in the window. Also an overlapping percentage of $S = 50\%$ has been applied in order to keep the spectral information at both ends of the window. With these data, the time resolution for the spectral analysis, $\Delta t = t_{i+1} - t_i$, can be calculated as

$$\Delta t = \frac{SN}{100 f_s}$$

In order to have less frequency bands and less computational load, divisors of the sampling rate can be used, although this limits the useful frequency range. The original sampling rate was $f_s = 44,100$ Hz, nevertheless we have used an operational sampling rate of $44,100/2 = 22,050$ Hz, thus the highest posible frequency is $f_s/2 = 11,025$ Hz. With this value, the equation above yields a value of $\Delta t = 46.4$ milliseconds. This time establish the precision in time for the onset and the end of the notes we try to identify.

The STFT provides 1024 frequencies with a spectral resolution of 10.77 Hz. For our analysis we will transform the spectrum frequencies into b bands in a logarithmic scale of a twelfth of octave (a note) considering bands beginning in frequencies ranging from 50 Hz (for a pitch close to G$\sharp_0$ – G sharp of octave 0 –) to 10,600 Hz (F_8 in pitch), almost eight octaves. This way, we obtain $b = 94$ spectral bands that correspond to 94 notes in that range and they will be provided to each of the 94 neurons in the net input layer.

The amplitude of the bands in the spectra is obtained in dB as attenuations from the maximum amplitude. The dynamic range is 96 dB, provided the 16 bits resolution of the digital audio considered. Before providing the attenuations to the net input they are normalized to the interval $[-1, +1]$, being the -1 value assigned to the maximum attenuation (-96 dB) and $+1$ is assigned to the attenuation of 0 dB. Anyway, in order to remove noise and emphasize the important components in each spectrum, a low level threshold, θ, is applied in such a way that if $S(f_k, t_i) < \theta$ then $S(f_k, t_i) = -1$. See Fig. 1 for a picture of this scheme. This threshold has been empirically established in -45 dB.

Usually, a note onset is not centered in the STFT window, so the bell-shape of the window affects the amplitude if the note starts at this position and some important amount of energy can be lost. A dynamic net as TDNN becomes useful to solve this problem. The overlapping

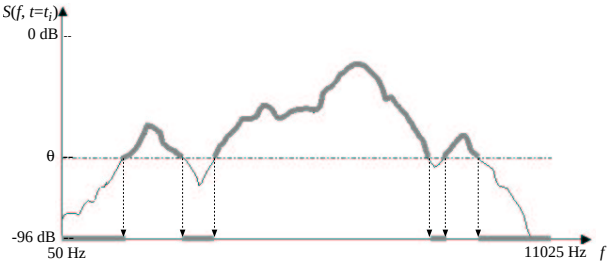

Figure 1. For each spectrum a low level threshold is applied to remove noise.

adjacent positions of the spectrogram will provide this dynamic information to the net. For each window position considered, b new input units are added to the net, so the total number of input neurons will be $b \times (n+m+1)$. See Fig. 2 for a scheme of this architecture.

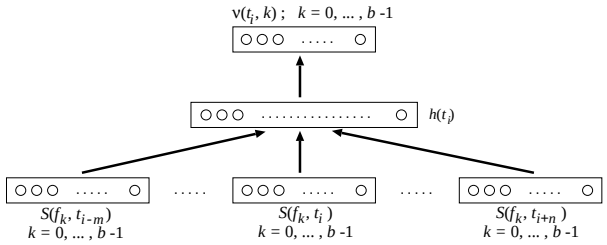

Figure 2. Network architecture and data supplied during training.

2.1.2 Output data.

The net output layer is composed of 94 neurons (see Fig. 2), one by each possible note that can be detected (and the same number of spectral bands at each input). Therefore, we have a symbolic output with as many neurons as notes are in the valid range. We have coded the output in a way that an activation value of $\nu(t_i, k) = 1$ for a particular unit k means that the k-th note is active at that time and 0 means that the note is not active. So we will have 94 component vectors for $\nu(t_i)$. Usually the number of zeros will be much larger than that of ones, because only a few subset of possible notes will be active at each moment.

The series of vectors $\nu(t_i), t_i = 1, 2, ...$ will be named a binary digital piano-roll (BDP). A brief example can be observed in Fig. 3. The vectors at each moment are the training outputs shown to the net during the training phase, while the corresponding spectra are presented to the input.

The BDP is computed from the MIDI file, according to the notes that are active at the times where the windows of the STFT are centered.

```
time          C5            C6            C7
                |             |             |
0.58: ...00000100000000000000001000000...
0.63: ...00000100000000000000001000000...
0.67: ...00000100000000000000000010000...
0.72: ...00000100000000000000000010000...
0.77: ...00000100000000000000100000000...
```

Figure 3. Binary digital piano-roll coding the note activations (1's) at each moment when the spectrogram is computed. Each row represents the activations of the notes for a given time.

2.2 Network parameters

Different parameter values are free in any net. Some of them have a special interest in this work:

Number of spectrum windows at the input ($n+m+1$). The upper limit is conditioned by the fact that it has no sense to have a number of windows so large that spectra from different note subsets appear frequently together at the net input, causing confusion both in training and recognition. Moreover, the computational cost depends on this number. A good contextual information is desirable but not too much.

Activation threshold (α). The output value for the neurons are $y_k(t) \in [-1, +1]$. The final value to decide whether a note has been detected in the spectrogram at time t_i is computed as 1 if $y_k(t_i) > \alpha$ and 0 otherwise. The lower α is the more likely a note is activated. This value controls the sensitivity of the net.

These are the most influent parameters for the obtained results, while others concerning to the training, like weight initialization, number of hidden neurons, initial learning rate, etc have shown to be less important. Different experiments have been carried out varying these parameters and the results presented below are those obtained by the best net in each case. After some initial tests, a number of hide neurons of 100 has proven to be a good choice for that parameter, so all the results presented have been obtained with this number of hidden neurons.

2.3 Success assessment

A measure of the quality of the performance is needed. We will assess that quality at two different levels: 1) considering the detections at each window position t_i of the spectrogram in order to know what happens with the detection at every moment. The output at this level will be named "event detection"; and 2) considering notes as series of consecutive event detections or missings along time. The output at this level will be named "note detection".

At each time t_i, the output neuron activations, $y(t_i)$, are compared to the vector $\nu(t_i)$. A successful detection occurs when $y(t_i, k) = \nu(t_i, k) = 1$ for a given output

neuron k. A false positive is produced when $y(t_i, k) = 1$ and $\nu(t_i, k) = 0$ (something has been detected but it was not actually a note), and a false negative is produced when $y(t_i, k) = 0$ and $\nu(t_i, k) = 1$ (something has been missed). These events are counted over an experiment, and the sums of successes Σ_{OK}, false positives Σ_+, and false negatives Σ_- are computed. Using all these quantities, the success rate in percentage for detection is defined as:

$$\sigma = \frac{100\Sigma_{OK}}{\Sigma_{OK} + \Sigma_- + \Sigma_+}$$

With respect to notes, we have studied the output produced according to the criteria described above and the sequences of event detections have been analysed in such a way that a false positive note is detected when a series of consecutive false positive events is found surrounded by silences. A false negative note is defined as a sequence of false negative events surrounded by silences, and any other sequence of consecutive event detection (without silences inside) is considered as successfully detected notes. The same equation as above is utilized to quantify the note detection success.

3 Results

A number of different polyphonic melodies have been used, containing chords, solos, scales, silences, etc. trying to have different number of notes sounding at different times, in order to have enough variety of situations in the training set. The number of spectrum samples was 31,680.

The training algorithm converges quickly (tens of epochs) and each epoch takes about 15 seconds in a 1.3 GHz PC.

In order to test the ability of the net with waves of different spectral complexity, the experiments have been carried out using different waveshapes for training different nets. The limitations for acoustical acquisition of real data and the need of an exact timing of the emitted notes have conditioned our decision for constructing these sounds using virtual synthesis models.

Sinusoidal waveshape. This is the simplest wave that can be analyzed. All its spectrum energy is concentrated in a single partial. This way, the frequency band of maximum amplitude corresponds to the frequency of the emitted note.

Sawtooth waveshape. This wave contains all the harmonics with amplitudes proportional to $1/p$, being p the number of partial. It presents high complexity because the amplitude of its partials decrease slowly. We have used only the first 10 partials to synthesize this timbre.

Synthesized instrument waveshape. In addition to those artificial waveshapes, a real instrument timbre has been considered: a clarinet, although it has been synthesized through a physical modelling algorithm. Our aim is to obtain results with a complex wave of a sound close to real instruments.

3.1 Network parameter tuning

According to the size of the input context, the best results were obtained with one previous spectrogram window and zero posterior windows. Anyway, these results have not been much better than those obtained with one window at each side, or even 2+1 or 1+2. The detection was consistently worse with 2+2 contexts and larger ones. It was also interesting to observe that the success rate was clearly worse when no context was considered (around 20% less than with the other non-zero contexts tested).

In order to test the influence of the activation threshold, α, some values have been tested. High values (namely, above zero) have shown to be too high and a lot of small good activations were missed. As the value of α was getting low the sensibility and precision of the net increased. Values $\alpha \in [-0.8, -0.7]$ have shown to be the best.

Once these parameters have been tuned and their consistency for different waveshapes tested (within a small range of variation), we have performed a number of cross-validation experiments in order to assess the capability of the net to carry out this task with different timbres.

For training, each data set has been divided into four parts and four sub-experiments have been made with 3/4 of the set for training and 1/4 of the set for test. The presented results are those obtained by averaging the 4 sub-experiments carried out on each data set.

3.2 Recognition results

As expected, the sinusoidal waveshape has provided the best results (around 95% for events and also 95% for notes). Most of the event detection errors have occurred in the onsets and offsets of the notes and the majority of the errors in note detection correspond to missing notes of less than 0.1 s of duration (just one or two window positions in the spectrogram). If we would not consider this very short notes, almost a 100% of success is obtained.

For the sawtooth timbre, the success results are lower due to the higher number of harmonics that the complexity of the wave produces. Around 92% for events and for notes were obtained. Again, most of the misdetections ocurred in very short notes.

For the clarinet waveshape, the results were comparable to those obtained for the synthetic waveshapes, giving values of 92% for events and 91% for notes. All these figures are summarized in the diagonal of tables 1 and 2.

These first results with real timbres suggest that the methodology can be applied to other real instruments at least in the wind family, characterized by being very stable in time. This makes the spectral pattern identification

	sinusoidal	sawtooth	clarinet
sinusoidal	95 %	56 %	65 %
sawtooth	56 %	92 %	72 %
clarinet	56 %	50 %	92 %

Table 1. Event cross-detection results. Rows correspond to the training timbres and columns to the testing timbres.

	sinusoidal	sawtooth	clarinet
sinusoidal	95 %	48 %	55 %
sawtooth	60 %	92 %	65 %
clarinet	60 %	49 %	91 %

Table 2. Note cross-detection results. Rows correspond to the training timbres and columns to the testing timbres.

easier than in other more evolutive timbres, like, for example, percussive instruments.

Errors have been analyzed considering note length, pitch and number of training samples. Errors produced by short notes (less than 0.1 s) constitutes the 31% of total errors. Most of pitch errors correspond to very high (higher than C_7) and very low (lower than C_3) pitches. Our experiments also suggest that increasing the size and variety of the training set could improve the results.

3.3 Changing waveshapes for detection

We have posed the problem of how specific the net weights are for the different timbres considered. For this, spectrograms of a given waveform are presented to a net trained with spectrograms from a different waveform. We have trained and tested nets for the three wave forms. The event and note detection results are displayed in tables 1 and 2.

The success rates range from 48% to 72% so they are clearly worse for this experiment although not catastrophic. It could be said that from two to three out of each four notes have been detected. Probably, if the net were trained with mixed spectrograms from different waveshapes, these results could be improved, but this is yet to be tested.

3.4 Evolution in time for note detection

A graphical study of the net output activations along time has been carried out in order to analyze the kind of errors produced when compared to the desire output (see figure 4). In the plot, the active neurons at each moment can have either a 'o' if they successfully detected an event, or a '+' if the activation corresponded to a false positive event. If there were no activation for a given neuron at a time were the corresponding note was actually sounding, a '-' is displayed, corresponding to a false negative. If there were no notes and no activations, nothing would be displayed.

The example melody of Fig. 4 was neither in the training set nor in the recognition set, and it has been synthesized using the clarinet timbre. For this melody, the event detection rate was 94.3%, the proportion of false positives to the number of events detected was 2.3% and for false negatives was 3.4%. For notes, all of them were detected and just 3 very short false positive notes appeared.

As it is observed, most of the errors were produced in the transition events between silence and sound or viceversa for some notes, due to the time resolution and the excess of energy in a lapse were a note is not coded as sounding in the BDP or the lack of energy were it is already sounding according to the BDP.

4 Discussion and conclusions

This work has tested the feasibility of an approach based on time-delay neural networks for polyphonic monotimbric music transcription. We have applied them to the analysis of the spectrogram of polyphonic melodies of synthetic timbres generated from MIDI files using a physical modelling virtual synthesizer.

The detection success has been about a 94% in average when the test timbre was the same as the one used for training the net. The success rate has decreased (to around 60%) when a net trained with a given waveshape has been tested with spectrograms of different timbres, showing the high specialization of the net.

Errors concentrate in very low-pitched notes, where the spectrogram is computed with less precision and for very high notes, where the higher partials in the note spectrum are cut off by the Nyquist frequency, or, even worse, folded into the useful range, distorting the actual data, due to the aliasing effect caused by cutting down the sampling rate by a factor of two.

As it has been said above, most of the errors are concentrated in the transitions, at both ends of the note activation. This kind of situations can be solved applying a post-processing stage over the net outputs along time. In a music score not every note onset is equally probable at each moment. The onsets and offsets of the notes occur in times that are conditioned by the musical tempo, that determines the position in time for the rhythm beats, so a note in a score is more likely to start in a multiple of the beat duration (quarter note) or some fractions of it (eighth note, sixteenth note, etc.). The procedure that establishes tight temporal constraints to the duration, starting and ending points of the notes is usually named quantization. From the tempo value (that can be extracted from the MIDI file) a set of preferred points in time can be set to assign beginnings and endings of notes. This transformation from STFT timing to musical timing should correct

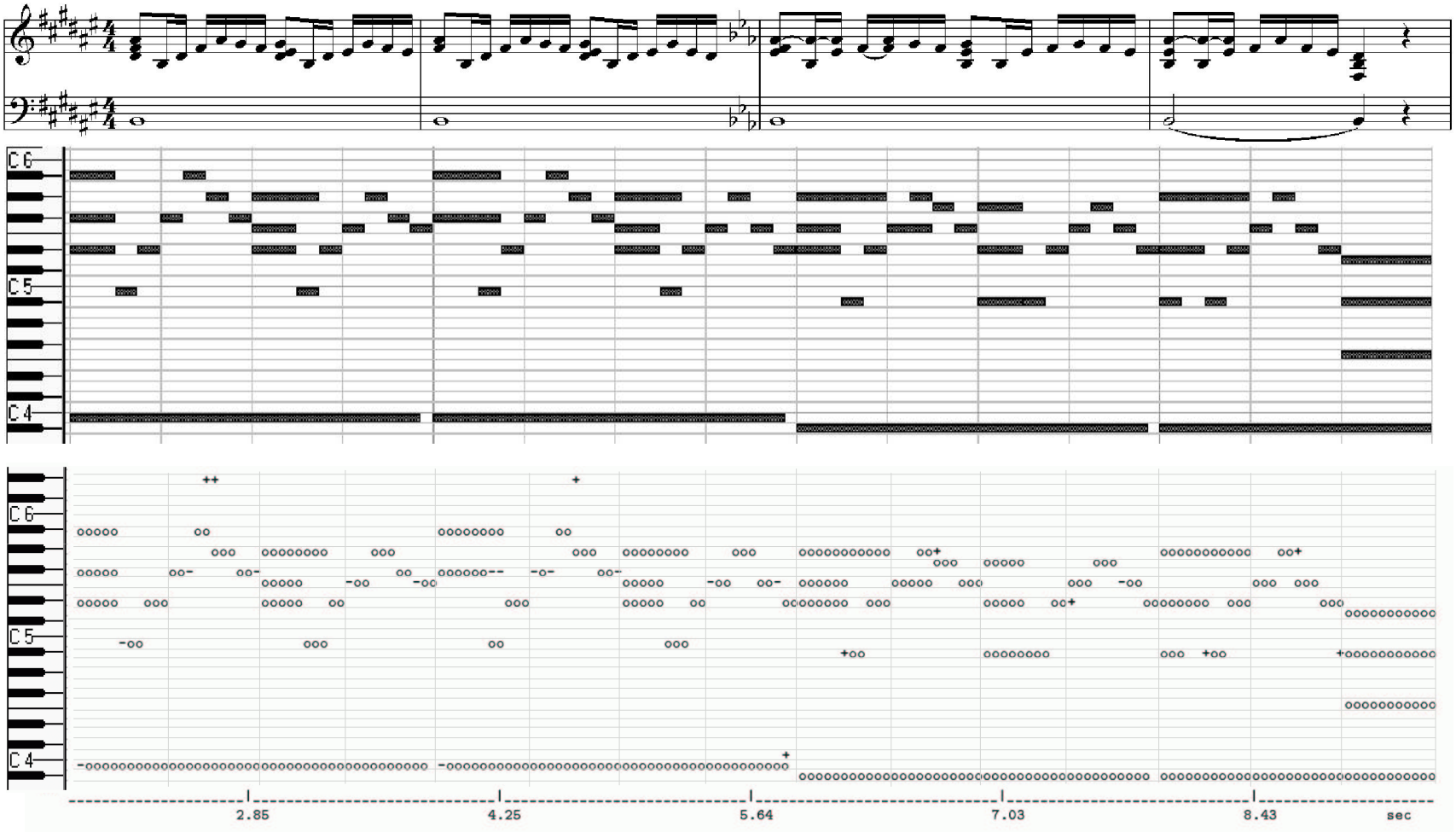

Figure 4. Evolution in time of the note detection for a given melody using the clarinet timbre. Top: the original score; center: the melody as displayed in a sequencer piano-roll; down: the piano-roll obtained from the net output compared with the original piano-roll. Notation: 'o': successfully detected events, '+': false positives, and '-': false negatives.

most of these errors.

False note positives and negatives are harder to prevent and it should be done using a musical model. This is a complex issue. Using stochastic models, a probability can be assigned to each note in order to remove those that are really unlike. For example, in a given melodic line is very unlike that a non-diatonic note two octaves higher or lower than its neighbors appears.

The capability of a net trained with a given timbre for transcribing audio generated by a different, but similar, waveforms should be studied more deeply, but it seems more reasonable to provide the system with a first timbre recognition stage [4, 3], at least at a level of timbric families. Different weight sets could be loaded in the net according to the decision taken by the timbre recognition algorithm before starting the transcription.

References

[1] R. Boulanger. *The CSound Book*. MIT Press, Cambridge, Massachusetts, 1999.

[2] J. Brown and B. Zhang. Musical frequency tracking using the methods of conventional and 'narrowed' autocorrelation. *J. Acous. Soc. Am.*, 89:2346–2354, May 1991.

[3] S. Dubnov and X. Rodet. Timbre characterisation and recognition with combined stationary and temporal features. In *Proc. of International Computer Music Conference, Michigan*, Ann Arbor, MI, USA, 1998.

[4] I. Fujinaga. Machine recognition of timbre using steady-state tone of acoustic musical instruments. In *Proc. of International Computer Music Conference*, pages 207–210, 1998.

[5] D. Hermes. *Pitch Analysis*, chapter Visual Representations of Speech Analysis. John Wiley and Sons, New York, 1992.

[6] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of Neural Computation*. Addison-Wesley, Redwood city, CA, 1991.

[7] W. Hess. *Algorithms and Devices for Pitch Determination of Speech-Signals*. Springer-Verlag, Berlin, 1983.

[8] D. R. Hush and B. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, 1(10):8–39, 1993.

[9] A. Klapuri. Automatic transcription of music, 1998. Master thesis, Tampere University of Technology, Department of Information Technology.

[10] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. In J. W. Shavlik and T. G. Dietterich, editors, *Readings in Machine Learning*, pages 150–170. Kaufmann, San Mateo, CA, 1990.

[11] M. Marolt. Sonic : transcription of polyphonic piano music with neural networks. In *Proceedings of Workshop on Current Research Directions in Computer Music*, November 15-17 2001.

[12] K. Martin. A blackboard system for automatic transcription of simple polyphonic music. Technical Report 385, MIT Media Lab, July 1996.

[13] J. A. Moorer. *On the segmentation and Analysis of Continuous Musical Sound by Digital Computer*. PhD thesis, 1975. Stanford University: Department of Music; Report STAN-M-3.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[15] T. Shuttleworth and R. Wilson. Note recognition in polyphonic music using neural networks. Technical report, University of Warwick, 1993. CS-RR-252.

[16] L. Smith. Applications of connectionism in music research. Technical report, University of West Alabama, 1995. 95/1.

[17] P. M. Todd and D. G. Loy, editors. *Music and Connectionism*. MIT Press, 1991.

[18] B. Vercoe. *The CSound Reference Manual*. MIT Press, Cambridge, Massachusetts, 1991.