Interactive Control of Explicit Musical Features in Generative LSTM-based Systems

Maximos Kaliakatsos-Papakostas
“Athena” Research and Innovation Center
Athens, Greece
maximos@ilsp.gr

Aggelos Gkiokas
“Athena” Research and Innovation Center
Athens, Greece
agkiokas@ilsp.gr

Vassilis Katsouros
“Athena” Research and Innovation Center
Athens, Greece
vsk@ilsp.gr

ABSTRACT

Long Short-Term Memory (LSTM) neural networks have been effectively applied on learning and generating musical sequences, powered by sophisticated musical representations and integrations into other deep learning models. Deep neural networks, alongside LSTM-based systems, learn implicitly: given a sufficiently large amount of data, they transform information into high-level features that, however, do not relate with the high-level features perceived by humans. For instance, such models are able to compose music in the style of the Bach chorales, but they are not able to compose a less rhythmically dense version of them, or a Bach choral that begins with low and ends with high pitches – even more so in an interactive way in real-time. This paper presents an approach to creating such systems. A very basic LSTM-based architecture is developed that can compose music that corresponds to user-provided values of rhythm density and pitch height/register. A small initial dataset is augmented to incorporate more intense variations of these two features and the system learns and generates music that not only reflects the style, but also (and most importantly) reflects the features that are explicitly given as input at each specific time. This system – and future versions that will incorporate more advanced architectures and representation – is suitable for generating music for the features of which are defined in real-time and/or interactively.

1 INTRODUCTION

Artificial Intelligence (AI) and Machine Learning (ML) methods have been extensively explored for generating music. Interesting results have been produced so far in the academic context and a few attempts with debatable success have been made to incorporate such methods in real-world applications – either for the production of AI-music or AI-based tools for assisting/collaborating with musicians. Such methods can be, nonetheless, useful in real-world applications: interacting with AI and ML systems that generate music can be both recreational, and especially educational, for people who are not familiar with playing a musical instrument and are willing to explore musical and/or timbral attribute.

Interactive generation systems have been examined under many aspects; from audio generation systems that help users generate audio mosaics [13] or automatically construct audio content based on target timbres given by the user [21] using adaptive concatenative sound synthesis, to symbolic music systems, which is the focus of the paper at hand, in [19] explored the idea to transform initial “seeds” of music based on desired musical attributes. However, explicit control of many musical parameters is cumbersome and frustrating, since the generated results are often uncontrollable; this fact gave rise to interactive approaches that involved human ratings instead of direct control of parameters.

Rating-based methods often involve Genetic Programming and such methods have been proposed for symbolic music generation [8] or audio [9]. In a rating-based scheme, users listen to samples and rate them, providing fitness evaluations to algorithms that expectedly converge to an estimated “maximum” (a musical result that should be highly-rated). Given the user-fatigue constraints, i.e. the fact that users cannot undergo many rating rounds, genetic algorithms harnessed with Particle Swarm Optimisation have been proposed to accelerate convergence to feature spaces preferred by the user [11]. Cognitively-relevant musical features play an important role in how we understand and classify music qualities and AI/ML methods have been extensively employed for music classification and generation [6]. Real-time interactive systems are also possible by incorporating the notion of feature-based “divergence”; an example is the “evoDrummer”, which produced variations of drum rhythms in real time, according to the desired divergence value given by the user [10].

The advent of new interaction means, e.g. the Leap Motion controller or the Kinect sensor, allow direct interaction with computers, strengthening the value of the real-time control paradigm. Along these lines, gesture-control of compositional processes using the...
Leap Motion controller has been proposed in [17]. These compositional processes, however, incorporated “fixed” patterns based compositions of well-known, idiosyncratic music composers. Another approach using the Kinect sensor has been implemented in the Harmonic Navigator [16]; this system uses an informative visualisation and interaction interface that allows the user to choose the next chords in harmonic progressions proposed by a Markovian probabilistic model trained on a given corpus.

Recurrent Neural networks (RNN) offer a sophisticated “black box” probabilistic model to model sequences through learning from data; since the early 90’s, RNN models have been tested on music composition [18]. Recent developments have allowed experimentation with RNN-based models that employ more sophisticated representations for folk melody generation [20] as well as hybrid deep learning architectures (that involve Restricted Boltzmann Machines) for polyphonic music generation [1].

During the last years, a popular model of RNNs that incorporates variable memory capacities has been extensively tested for music generation, namely, the Long Short-Term Memory [7] (LSTM) network. A typical example case for testing the generative capabilities of models based on the LSTM architecture is to train such networks on very strictly defined styles, as the Bach chorales [14], LSTM-based models have been expanded with impressive results in composing Bach chorales [4], incorporating information about attributes of the musical score that are not deteriorated to notes, i.e. metric structure. Similar techniques have been used for generating drum rhythms [15], given information about the metric structure and the activity of other instruments in a recording. For a more detailed description of music work on deep learning in music, the reader is referred to [2].

LSTM-based models, given a proper representation of music sequences and sufficiently large amounts of training data, are able to implicitly learn the characteristics of musical styles and compose music that very effectively reflects those characteristics. However, these networks, being “black-box” models, are not able to digest explicit information: they are able to internally represent high-level features of music that do not relate, however, with the high-level structures perceived by humans. For instance, the Bach chorales can be rhythmically dense, or at some points higher or lower pitches might have been used by Bach. However, these systems do not have “conscious” access to such information. Is it possible to develop a system that learns a given style, while it can also “consciously” modify its composition according to explicitly-described musical input features?

Aim of this work is not to provide yet another proof that LSTM-based methodologies are indeed able to learn given styles. The goal is to develop and examine a neural network methodology that incorporates explicit information in forms that humans understand. To this end, a minimal neural network implementation (in terms of architecture, data and representation) is developed that allows a preliminary examination on whether direct infusion of desired feature values (as rhythm density and pitch register) in a generative musical LSTM-based system provides satisfactory results in interactive, feature-driven control of the system’s compositions. It should be noted that the study presented herein is different from the approach proposed in [3]. The aforementioned work provided a solution based on neural networks for learning relations/mappings between parameters (whether they are musical and non-musical) without introducing a generative music system that composes music according to given parameters. A more recent version of the system that will be trained with more data and incorporate more features will be integrated into a STEAM (Science, Technology, Engineering, Arts and Mathematics) education platform developed in the context of the iMuSciCA\textsuperscript{3} project, to allow students play music with virtual instruments using feature-driven composition.

In the paper at hand, it is such a system that is proposed; a system that learns and composes music that reflects the characteristics of given styles, while, at the same time, allows users to have explicit and interactive control of some musical features. The methodology, described in Section 2 includes two steps; data augmentation, where modified copies with diverse characteristics of the available data are concatenated to form a unified dataset and the network architecture. Section 3 presents the evaluation of the trained network in artificial scenarios that allow a rigorous examination of the achieved adaptation of the system to the input features. Conclusions are given in Section 4.

### 2 METHODOLOGY

This section describes the methodological steps for selecting, processing and augmenting data and for building a base-level machine learning model for focusing on research questions related with interactive control. Since the goal is not to construct and train a sophisticated system that proves the stylistic adaptability of LSTM-based networks, the applied methodologies related to data and network are kept at the most basic level, focusing on aspects that concern the work-flow that would allow the development of a feature-driven LSTM-based system.

#### 2.1 Data Selection, Processing and Augmentation

The collected dataset was decided to be small with pieces that incorporate specific, consistent and simple musical characteristics. Since data augmentation is among the followed data processing steps, a small dataset size makes it more practical to experiment with many different augmentation setups, considering that training neural networks for many epochs requires considerable processing time. The small dataset with such characteristics is the “Little dragon” trilogy\textsuperscript{4}, which includes three piano pieces suitable for educational purposes. All three pieces are in the C major scale (with simple yet interesting chromatic deviations), in the 4/4 time signature and a tempo of 120 beats per minute, considerable rhythmic diversity and a consistent range of less than 2 octaves.

\textsuperscript{1}RNNs can be characterised as more sophisticated in relation to the Markov models, since their “memory” spans across an arbitrary, not fixed, length; however, the dynamic relations that are developed in the RNN units are not transparent, hence, their characterisation as “black box”.

\textsuperscript{2}Implicit learning is achieved by exposure to many examples without conscious knowledge of the underlying rules; e.g. 3-year-old children can speak correctly without knowing syntax rules through implicit learning of language. *Explicit* learning is performed by consciously applying rules and categorising incoming information; e.g. foreign languages are learned explicitly (while native languages implicitly).

\textsuperscript{3}http://www.imuscica.eu

\textsuperscript{4}http://musescore.com/user/204621/sets/501766 - retrieved May 31 2018.
Interactive Control of Explicit Musical Features in Generative LSTM-based Systems.

Data are straightforwardly represented with a binary matrix that indicates note onsets in each 16th note, while durations (or note ending times) are not considered. The binary matrix of a piece is a matrix \( P \in \{0, 1\}^{128 \times M} \), where \( M \) is the length of the piece in 16ths and each column represents a 16th slice of the piece and includes a binary representation of which respective midi notes (128 in total) begins at this time slice; the row of the notes that have an onset in each 16th (column) are marked with 1 and all others with 0. The initial matrix representation of the dataset, before augmentation, is constructed by concatenating the columns of all matrices (one next to the other) of each piece, resulting in a matrix \( X \in \{0, 1\}^{128 \times N} \), where \( N \) is the sum of all column lengths of each piece in the dataset. Data were collected in musicXML format and were converted to their numerical/matrix format using the Music21 tool. Figure 1 shows the data processing steps after forming the \( X \) matrix, data augmentation and feature extraction follow.

![Figure 1: Data processing work-flow.](image)

During data augmentation, the initial dataset, as represented in the \( X \) matrix, is doubled consecutively according to how many augmentation steps are followed; each augmentation step generates a clone of the data samples with altered characteristics that is appended to the pre-augmentation data samples, doubling their size. In Figure 1, the augmented data is shown as \( A \in \{0, 1\}^{128 \times kN} \), where \( k \) is an integer and depends on the augmentation steps followed. Music features are extracted from the augmented data samples and data augmentation aims to create training data that are more diverse in regards with the considered features.

Since the aim is to examine the ability of LSTM-based networks to interactively adapt to given requested musical features, two features were extracted that are simple to compute and to track by ear: rhythm density (or sparsity) and mean pitch height (or register). Data augmentation should therefore generate data that are more diverse in terms of rhythmic sparsity and pitch register. Two augmentation steps are followed:

1. **Sparsity augmentation:** a copy of the available data is generated and in each bar (16-tuples of columns), columns are randomly selected according to probabilities that constitute the first column most probably to "survive" the deleting process, while other columns (especially these on weaker beats) are more in favor to be deleted. A musical illustration of sparsity augmentation is given in Figure 2 (a).

2. **Register augmentation:** a copy of the data samples is created that is pitch-shifted by some desired amount of semitones. Figure 2 (b) shows a musical illustration of register augmentation by increasing 7 semitones.

The augmented copies generated in this fashion are appended to the end of the available data samples, consecutively doubling their size. For example, if sparsity augmentation is applied on a data sample of length \( N \), then its length is becoming \( 2N \); if register augmentation is followed afterwards, then the \( 2N \) size is doubled to \( 4N \).

After augmenting the data with the aforementioned steps (register augmentation can be applied several times with different pitch shifting values), two features are extracted regarding sparsity and mean register/pitch height. Features are extracted for each 16th time step, on a sliding window of nine sixteenths length – four 16ths before and four after the event of interest. Specifically, the two features under discussion are computed for each column in the augmented matrix inside the corresponding window as follows:

1. **Density (as opposed to sparsity):** the ratio of events in the window that include at least one onset (sum of matrix columns that include a monophonic or a polyphonic event) over the total number of events in the window (sum of events in window).

2. **Register:** the normalised mean value of the highest and the lowest pitch in the window of the event of interest, over the value of the highest pitch in the dataset. If no pitch is present in the window, thus computation of this feature for the window is not possible, the value is inherited from the previous window – if no pitches are included in the beginning window, the value of 0.5 is assigned.

Both feature values are in the range \([0, 1]\). The feature values in the window of each event have the characteristics of step functions; e.g. the sparsity feature takes only values in the form \( i/9 \) (or \( i/\text{window size} \), where \( i \in \{0, 1, \cdots, 9\} \). Since we are interested in providing interactive control of continuous features values, the features were slightly smoothed out by applying a moving average for each feature in a window of length 4 (or the integer part of the half window size).

The feature values extracted from the augmented data are shown in Figure 3. The data samples have been augmented by applying the augmentation steps in the following order: sparsity augmentation, register augmentation 12 semitones down and register augmentation 12 semitones up. The feature values in the aforementioned figure clearly show the feature areas that are formed by applying these augmentation steps. The initial copy of the data takes up to the 848 time-step and since sparsity is applied first, it is evident that the first copy of the initial data (time-steps from 849 to 1696) is an area with smaller density values (upper graph in Figure 3) – the respective register feature values in the lower graph are the same as for the initial copy of the data. Afterwards, when register \(-12\) and \(+12\) augmentations are applied, two new copies (of length 1696) are created leading to the final data sample with length 5088 time-steps. The lower graph shows clearly the register differences of the new copies, while the respective copies in the density feature are identical to the first copy.

---

1http://web.mit.edu/music21/ - retrieved May 31 2018.
concatenate

Figure 2: The two employed data augmentation techniques: (a) sparsity and (b) register.

concatenate

Figure 3: Data augmentation and feature values after applying three steps of data augmentation: (a) sparsity, (b) register −12, and (c) register +12.

2.2 Network functionality, architecture and training

The “vanilla” version of the network presented in this paper is a network that learn compositions implicitly, while at the same time integrates knowledge about the musical characteristics of what it is actually learning, according to the aspects that are reflected by the aforementioned features. The goal is to allow users to interact with this network “explicitly” by providing desired musical characteristics (or target feature values) that will lead the network to compose music that reflects these characteristics. Figure 4 illustrates the desired workflow and the basic architecture employed for the network under discussion.

Generating: Note events are represented as 128-dimensional binary arrays of onsets which are the columns of a binary matrix that represents the entire data; when in “generating” mode, the two desired features for the next event are appended in the 128-dimensional array of the current event, creating a final 130-dimensional input array to the network. The network then generates the next note event (128-dimensional array), which is concatenated with the new user-defined features and fed back as input to recursively compose new music\(^6\). Initially, the network is given a seed sentence of 16 data samples that is taken from the training dataset.

Training: Figure 5 illustrates the training process and how it relates with the data processing steps. The augmented binary matrix (\(A \in \{0, 1\}^{128 \times kN}\)) is vertically concatenated with the feature matrix (\(F \in [0, 1]^{2 \times kN}\)), after the feature matrix has been column-wise shifted by one position to the left to correspond to the features of the next event\(^7\). It should be noted that in Figure 5 the \(f_i\) and \(n_{i-1}\) vectors are illustrated as separate entities for visualisation purposes, however, they are actually concatenated into a single vector. The input matrix (vertical concatenation of \(F\) and \(A\)) is divided into batches of length 320 (corresponding to 20 bars), while the network is unfolded in time for 16 steps (one bar) during training. The cost function is the sigmoid cross entropy (since note representation is binary and not one-hot) and the employed learning algorithm was the Adam optimiser [12].

3 RESULTS

The experiments are designed to test the extent to which the system is able to generate music that corresponds the musical attributes

\(^6\)Typically the network input comprises the last 16 time steps, since it is unfolded in time [5].

\(^7\)This step is most possibly not necessary since the differences in consecutive feature values have been smoothed by the application of the moving averages.
used as input. Ideally, the network should be able to follow the user-provided feature input and compose music that reflects the respective characteristics in real-time. The experimental processes are based on artificial “simulated” scenarios that test the adaptability of the system in pre-defined target feature curves. To this end, the respective feature values described by these curves are given as input to the system at every musical event it composes. After the system generates a composition, the “achieved” features of this composition at every time step are compared with the “target” features that were used as input; high correlations between the target and the achieved features would allow the assumption that the system does indeed have the ability to adapt. A link to an on-line version of the system can be found here: https://athena.imuscica.eu/featureLSTM.

The system is examined by using three input curve scenarios:

- “Slow”: The feature values are changing at a slow rate, including large plateaus of constant feature values, while Figures 6 (a) and (b) illustrate these scenarios (target features), alongside the (achieved) features extracted from the system generations. Target features for 600 time steps (approximately 38 bars) have been used as input, while areas where both features are high, low or any combinations between these two extremes are included.
- "Fast": A shortened version of the “slow” scenario that lasts for 60 time steps (approximately four bars). In this scenario a "burnout" period of 16 steps was used, to allow the system to generate an additional bar after the seed (which is by itself one bar). The “fast” scenario target and achieved features are shown in Figures 6 (c) and (d).
- “Training”: The features that are extracted from the training sequence and used during training are re-imported to the system to generate a new composition. The aim of this scenario is to test the system in “realistic” changes of feature values – and since these features have been extracted by real compositions (training data) suitable for this task. Figures 6 (e) and (f) show the results of this scenario (the target features match the feature values illustrated in Figure 3).

It should be noted that the ranges of the artificial target features were set to [0.3, 0.8] for the density and to [0.2, 0.8] for the register. Regarding density, given that the maximum number of voices is 4, a density below 0.25 would mean silence – which indeed happens in simulations where density values below 0.25 are requested – however, testing the ability of the system to retain the capacity to produce notes close to the lowest possible density is considered more interesting for the presented results. The choice for setting the upper limits to 0.8 for both features and the lowest limit to 0.2 for the register feature are related with the feature values observed in the training set which are within these ranges. Nevertheless, even the system is given feature values that exceed these limits the results are similar.

The illustrations in Figure 6 reveal that the system is indeed able to adapt, to some extent to the feature inputs. This is also numerically evident through the high correlation values between the “target” and the “achieved” feature curves, shown in the “corr.”
Figure 6: Density and register feature adaptation in the “slow” (a-b), “fast” (c-d) and “training” (e-f) feature modulation scenarios.

column in Table 1. This table also examines another hypothesis: that the achieved features should incorporate some lag in comparison to the target features, since it can be assumed that the system has some delay in preparing proper musical responses for a given input.

To examine this hypothesis, the achieved features are shifted backwards one step at a time and their correlation with the target features are measured at each shift. The “opt. lag.” and “opt. corr.” columns in Table 1 show the optimal shifting values for achieving the optimal correlation respectively. In some cases, the optimal correlation is achieved with zero lag, while in the cases where the optimal lag is achieved with a (even a considerable amount) shifting, the gain in correlation is minimal (in most cases around 0.03). Therefore, it can be considered that even without shifting,
Interactive Control of Explicit Musical Features in Generative LSTM-based Systems.

satisfactory results are achieved in real-time, at least as far as the correlation can indicate.

Table 1: Correlations between target and achieved features in all scenarios with no lag ("corr." column) and the optimal lag ("opt. corr." column).

| scenario feature | corr.  | opt. lag | opt. corr. |
|------------------|--------|----------|------------|
| artif. “slow”    | 0.6216 | 0        |            |
| artif. “fast”    | 0.7288 | 1        | 0.7413     |
| training         | 0.7655 | 16       | 0.7658     |

Two additional remarks need to be made regarding the system architecture and the nature of the input features. Regarding the network architecture, the best results were achieved by using 64 units for the LSTM network. Using 256 or 512 produced poor results since the network is choosing to be silent too often. This indicates that, given the small amount of training data, the network was over-specialising and is not able to generalise and provide new solutions to combinations of input feature values and previous note layouts that are not encountered during training.

Regarding the nature of the features, the artificial scenarios are also tested with different levels of noise, examining the hypothesis that the plateaux and steady-gradient changes in the input feature values might not be “realistic” enough – since in training the features appear to be more “noisy”. This hypothesis has been rejected, as the results (not presented due to space constraints) in 100 simulations per noise level (ranging from 0.01 to 0.2) indicate that the achieved results are comparable to the ones produced by the shown artificial input features.

4 CONCLUSIONS

A system based on the Long Short-Term Memory (LSTM) neural networks has been presented that can compose polyphonic music that reflects characteristics of a learned style, while at the same time integrates user-defined features. A data augmentation technique is also proposed that generates copies of the dataset that incorporate the examined features more intensively. Specifically, the dataset is augmented regarding the features of rhythm density and pitch height/register, and the trained system is able to compose music that is more or less rhythmically dense, and with higher or lower pitch registers, according to the feature values requested as input. The results on artificial scenarios show that satisfactory correlations between target features and achieved features (extracted from the artificial generations) are accomplished, showing that interactive, real-time control of these parameters is indeed feasible. Future work will focus on integrating more features like rhythm syncopation or harmonic tension. Additionally, integration of the feature-related layer into more advanced LSTM-based methodologies will be examined, along the application of the data augmentation technique to “benchmark” datasets such as the Bach chorales.

ACKNOWLEDGEMENTS

This work has been done in the context of the ImuScICA project, which received funding from the European Union’s Horizon 2020 research and innovation program with the grant agreement No 731861.

REFERENCES

[1] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. 2012. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. arXiv preprint arXiv:1206.6392 (2012).
[2] Jean-Pierre Briot, Gaetan Hadjeres, and Francois Pachet. 2017. Deep Learning Techniques for Music Generation-A Survey. arXiv preprint arXiv:1709.01620 (2017).
[3] Rebecca Anne Fiebrink. 2011. Real-time human interaction with supervised learning algorithms for music composition and performance. Princeton University.
[4] Gaetan Hadjeres and Francois Pachet. 2016. DeepBach: a Steerable Model for Bach chorales generation. arXiv preprint arXiv:1612.00102 (2016).
[5] S. Haykin. 2008. Neural Networks and Learning Machines (3 ed.). Prentice Hall.
[6] Dorien Herremans, Kenneth Sørensen, and David Martens. 2015. Classification and generation of composer-specific music using global feature models and variable neighborhood search. Computer Music Journal 39, 3 (2015), 71–91.
[7] Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.
[8] Brad Johanson and Ricardo Poli. 2016. GP-music: An interactive genetic programming system for music generation with automated fitness raters. University of Birmingham, Cognitive Science Research Centre.
[9] Maximos A Kaliakatsos-Papakostas, Michael G Epitropakis, Andreas Floros, and Michael N Vrahatis. 2012. Controlling interactive evolution of 8-bit melodies with genetic programming. Soft Computing 16, 12 (2012), 2000–2031.
[10] Maximos A Kaliakatsos-Papakostas, Andreas Floros, and Michael N Vrahatis. 2013. EvoDrummer: Deriving rhythmic patterns through interactive genetic algorithms. In International Conference on Evoloutionary and Biologically Inspired Music and Art. Springer, 25–36.
[11] Maximos A Kaliakatsos-Papakostas, Andreas Floros, and Michael N Vrahatis. 2016. Interactive music composition driven by feature evolution. SpringerPlus 5, 1 (2016), 826.
[12] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014).
[13] Ari Lazier and Perry Cook. 2003. MOSIEVIUS: Feature driven interactive audio mosaicing. In Digital Audio Effects (DAFx).
[14] Feynman Liang. 2016. BachBot: Automatic composition in the style of Bach chorales. Ph.D. Dissertation, University of Cambridge.
[15] Dimos Makris, Maximos A Kaliakatsos-Papakostas, Ioannis Karydis, and Katia Klermanidis. 2017. Combining LSTM and feed forward neural networks for conditional rhythm composition. In International Conference on Engineering Applications of Neural Networks. Springer, 570–582.
[16] Bill Manaris, David Johnson, and Yiorgos Vasilidakis. 2013. Harmonic navigator: A gesture-driven, corpus-based approach to music analysis, composition, and performance. In Proceedings of the 9th AAKAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 67–74.
[17] Marcella Mandanici and Sergio Canazza. 2014. “The hand composer”: gesture-driven music composition machines. In Proc. of 13th Intl. Conf. on Intelligent Autonomous Systems. 15–19.
[18] Michael C Mozer. 1991. Connectionist music composition based on melodic, stylistic and psychophysical constraints. Music and connectionism, 195–211.
[19] Alexander Rigopoulos. 1994. Growing music from seeds: parametric generation and control of seed-based music for interactive composition and performance. Ph.D. Dissertation. Massachusetts Institute of Technology.
[20] Bob Sturm, João Felipe Santos, and Iryna Korshunova. 2015. Folk music style modelling by recurrent neural networks with long short term memory units. In 16th International Society for Music Information Retrieval Conference (ISMIR).
[21] Bob L Sturm. 2006. Adaptive concatenative sound synthesis and its application to micromontage composition. Computer Music Journal 30, 4 (2006), 46–66.