TINNBRE: A TIMBRE-BASED MUSICAL AGENT

Andrea BOLZONI (https://orcid.org/0000-0003-0370-9122) [1], **Balandino DI DONATO** (https://orcid.org/0000-0001-6993-2445) [2], and **Robin LANEY** (https://orcid.org/0000-0002-9319-8209) [1]

[1] **The Open University**, Milton Keynes, United Kingdom
[2] **Edinburgh Napier University**, Edinburgh, United Kingdom

ABSTRACT

In this paper, we present tiNNbre, a generative music prototype-system that reacts to timbre gestures. By timbre gesture we mean a sonic (as opposed to body) gesture that mainly conveys artistic meaning through timbre rather than other sonic properties. The system is designed and developed to be used in free improvisation and composition. Our prototype is powered by a neural network trained using a supervised learning approach on a set of sonic gestures representing the stimulus (or input) and a correspondent set of sonic gestures representing the reactions (or output). This model is then used to explore and generate new musical material. Here, we present an informal evaluation of our system, based on two use cases. In the first, the system is trained using MFCC analysis, while the second uses Constant-Q Transform spectrum. Participants were asked to rate our system's generated audio, particularly in respect of timbre. Results showed that although tiNNbre has been tested offline, it fosters timbre-rich interaction with the sonic materials and the co-creation process. Broadly speaking, musicians preferred the results obtained using the second system other than for purely percussive gestures.

1. INTRODUCTION

In the last few decades, several computer-aided composition tools and improvising systems have been researched and produced to support musicians' creativity [1]. In the field of Music and AI, we are particularly interested in exploring the use of musical agents that can learn, from an existing dataset of sonic materials or an input signal, how to react to timbre gestures and generate an outcome that is effective in respect of the training data. Our approach is not focused on a specific piece, performer or performance, but aims to define a general framework where musicians can freely explore and interact with the reactions yielded by the system.

In this work, we adopt the concept of "musical agents" as *"artificial agents that tackle musical creative tasks, partially or completely"* [2]. Systems built on this principle can learn how to react to a sonic stimulus, either off-line or in real-time, through the analyses of a symbolic corpus, which is a representation of sonic corpus through a series of descriptors, for example, MIDI notation or meta-data. With this in mind, we prototyped tiNNbre utilising Deep Learning for its capability to extract complex features, in particular for mapping inputs given to a system to an expected output. Drawing on recent employments of Deep Learning in audio synthesis, we propose two different system prototypes that reacts to timbre stimulus by generating new audio materials.

With this approach we aim to create a system that supports musicians in free improvisation and electronic music composition, placing timbre at the centre of the music-making process. To satisfy our musical aims, our first objective is to prototype tiNNbre as a system that can learn from a sound corpus how to react to timbre gestures, and establish a sense of interaction between the AI and the musician. The system has been tested remotely off-line, but will soon be coded to be used in real-time during music performance. In particular, the contribution will be the ability of the system to convey and interact through complex timbre-based perceptive features, thus providing new ways to engage creative and collaborative experiences between human and machine.

The following sections include a review of the relevant literature, a description of the system, and an evaluation of our prototype through a short survey we ran with five musicians.

2. BACKGROUND

2.1 Co-creative music systems

The Continuator [3] is a Markov-based system able to learn a musician's style from a MIDI dataset and play in the same style. Furthermore, it can learn from a musician while playing and continue an unfinished phrase in the same style. OMAX [4] is a system based on Factor Oracle, a model that represents sub-phrases in a sequence of symbols. It can play in "free-mode" and "beat-mode". In "free-mode", no musical structure is expected. The musician plays freely, the phrases are analysed and organised as sequences of symbols by the Factor Oracle model, and the system plays back phrases re-built from the database. The "beat-mode" is based on a metric/harmonic structure, where the style extracted by the human musician is put into relation to the structure, and the phrases played back by the system are put back into context with the time step the structure was played in. Improtek [5] builds upon the gen-

eration model presented in OMAX [4], and introduces the concept of "scenario". In the form of long-term constraint, the scenario defines a musical context, e.g. a harmonic progression for a jazz improvisation explored while performing with a musician. VirtualBand [6] is a system that can learn, from a corpus of MIR features, how a human musician reacts to another human musician in a particular musical style. The models obtained are used to control virtual musical agents that interact, in real-time, with human musicians.

2.2 Timbre and timbre-oriented co-creative music systems

Timbre is defined as the feature that lets us distinguish two sounds with the same pitch and intensity [7]. We can distinguish two different acoustic instruments that play the same note at the same intensity. After a comprehensive survey of AI methods in algorithmic composition, Fernandez and Vico state that *"algorithmic composition automates (to varying degrees) the various tasks associated with music composition, such as the generation of melodies or rhythms, harmonization, counterpoint and orchestration"* [1]. Nevertheless, some music styles – such as free-improvisation, acousmatic and contemporary electronic music – use timbre as a key sonic feature in their narrative. In this context, we embrace Heller's definition of timbre: *"timbre is a kind of executive summary of the distribution of amplitudes of the various partials in a complex tone"* [8].

Hsu [9] speaks about the relevance of timbre in free improvisation and his experience with the saxophonist John Butcher, known for his innovative work with saxophone timbre. Hsu created a system driven by timbral feature envelopes of an input sound, which are then mapped into parameters of different sound synthesis and processing, such as filtered noise, waveguide-based clarinets, and metallic-sounding comb filters. Yee-King proposes a musical agent that employs additive synthesis and frequency modulation synthesis to adapt the timbre of the musical agent to the timbre of the human musician [10]. Other systems are based on sound-corpus processing, and concatenative synthesis. CataRT [11] is a real time sound synthesis system that lets us explore, and re-synthesise a sound corpus through its audio descriptor space. Similarly, EarGram [12] rearranges an audio corpus with concatenative synthesis, but through generative strategies.

2.3 Deep Neural Networks and audio synthesis

Recently, many different generative audio systems using Deep Learning have been proven to be very effective. Wavenet [13] is a deep neural network used for text-to-speech and music generation. It generates the audio waveform directly. DDSP [14] employs an autoencoder architecture, introducing components that improve the capability of this architecture to synthesize audio. Tatar et al. [15] propose a new audio synthesis method using Deep Learning, called Latent Timbre Synthesis (LTS). They aim to provide composers with a tool that interpolates and extrapolates the timbre of different sounds, acting on the latent space of a Variational Autoencoder.

We aim to explore the possibilities given by this body of works, particularly focusing on Latent Timbre Synthesis [15], in the context of a co-creative musical agent.

3. METHODOLOGY

Before we outline our methodology, we define some terminology. We use *dataset* to refer to all features extracted from sonic material used to train the system through a supervised learning approach [16]. Our *datasets* are made of input audio analysis features, which we call *stimulus*; and, output audio analysis features, which we call *reactions*. Likewise, when using the model, we use *stimulus* to refer to the input data and *reactions* to refer to the output generated by the system. Furthermore, since Godøy defines a sound-related gesture as movement in sound, a trajectory shape in time and space [17], similarly, we define a timbre gesture as a sonic event that is characterized by a trajectory shape in time and timbral space.

The system design consisted of three phases: system development, testing, and user evaluation. During development, different pair of analyses techniques and neural network architectures were tested. Two of them have been chosen, and in the testing phase hyperparameters have been set to optimize performance. Each test aimed to evaluate, both on the aesthetic and the numerical sides, the *stimulus/reactions* mapping realised through our system. Finally, a user evaluation has been performed. The evaluation criteria were the effectiveness of the two different techniques to yield timbre-based *reactions* to *stimulus* (see sections 4.1 and 4.2). Effectiveness is used to refer to the system's capability to vary the *reaction* at different *stimulus* changes and the sonic quality.

Each dataset is made of five pairs of *stimulus/reactions*: A, B, C, D, and E. For the pairs A, B, and C, *stimulus* and *reactions* are timbre gestures, in D the *stimulus* is a timbre gesture but the *reaction* is silence, and in E *stimulus* and *reaction* are both silence. Each gesture, and the silences, last 3 minutes. They have been recorded separately, and then concatenated in two audio files of 15 minutes each, one for the input and one for the output. The two audio files were sliced in analyses windows to extract the audio features. We used 3-minute recordings to give musicians the opportunity to explore the chosen timbral gesture in depth, thus producing useful data for training the model.

The audio files are mono, with a 44100 Hz sampling rate and 16 bit depth. The dimension of the dataset is calculated as in Equation 1, where n is the dimension of the dataset, f_s is the sampling rate in Hz, l_t is the length in seconds of the dataset, and h the dimension in samples of the hop size used in feature extraction.

$$n = \frac{f_s \cdot l_t}{h} \tag{1}$$

The stimulus and the reactions are then stored in two matrices of dimension $m \cdot n$, where m is the length of the feature vector, and n is the dimension of the dataset.

Given the *stimulus* matrix X made of n vectors x, and the *reaction* matrix Y made of n vectors y, the mapping

between stimulus and reactions is as in Equation 2:

$$y_t = f(x_t) \quad \texttt{for} \quad t = 1, \ldots, n \tag{2}$$

In other words, each *vector* at time *t* in the input is mapped to the *vector* at the same time *t* in the output, where *t* represents the onset of a feature extraction window. Thus, the system doesn't take account of history.

4. THE SYSTEM

4.1 Version 1

The first version of tiNNbre was built using MuBu toolbox [18] in Max/MSP [1] for feature extraction and audio synthesis, and Tensorflow [19] framework in Python to train the neural network model. Data between the two pieces of software were shared via Open Sound Control (OSC). An overview of the system architecture is shown in Figure 1.

With MuBu, we extract the Mel-frequency cepstrum (MFCC) features from the stimulus and reaction audio materials. We choose MFCC features for their efficacy in representing the power spectrum of sounds [20]. 13 MFCC coefficients are extracted with Fast Fourier Transform (FFT) with window size of 1024 samples, and hop size of 512 samples.

A neural network is then trained using these data. The neural network has an architecture of an input layer of 13 neurons, three dense layers of 512 neurons, and an output layer of 13 neurons. The *reaction* MFCCs are also classified in a multi-dimensional binary search tree, using the Max/MSP external *mubu.knn* [18].

At runtime, we extract MFCCs from the audio input and send them to the model. Each MFCC vector yielded by the model is fed to *mubu.knn* that supplies the marker of the closest MFCC vector in the *reaction* corpus. Then, the concatenative synthesis external *mubu.concat~* plays the audio sample corresponding to the marker. To be consistent with the window and hop size of the FFT used to extract MFCCs, the grains have length of 23 ms, and period of 12 ms.

This architecture has been chosen with a trial and error approach. We tried less and more layers, and less and more neurons for each layers. Even if the results of configurations similar to the one chosen weren't significantly worse, the chosen one gave the best results.

4.2 Version 2: Spectral analysis and re-synthesis

The second version has been developed in Python with the Librosa library [21] for audio analyses and re-synthesis, and the Tensorflow framework for generating the neural network model (see Figure 2).

Here, we aim to train a model with audio spectrum data. As proposed in [15], we extract from the input and audio corpus the Constant-Q Transform (CQT) [22], a wavelet transform with a constant ratio between the centre frequencies of the bins and their bandwidth. As suggested by Tatar et al. [15], we used a hop-size of 128 samples, 48 bins per

[1] https://cycling74.com

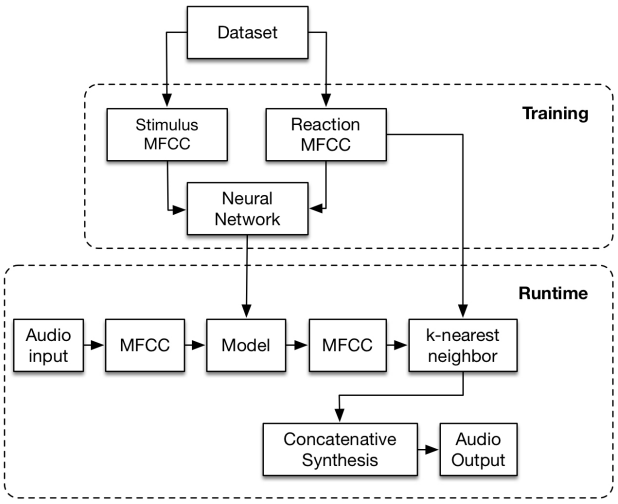

Figure 1. Architecture tiNNbre version 1

octave for 8 octaves, and a minimum frequency of 32.7 Hz. Therefore, each vector is made of 384 coefficients. We found, although this solution considers a spectrum range from 32.7 Hz to 8371.2 Hz, thus losing anything above 8371.2 Hz, this to be the optimal configuration. Using a hop-size of 256 samples, 24 bins per octave for 9 octaves, and a minimum frequency of 32.7 Hz was a better configuration to reproduce the attack of percussive sounds. However, it produced heavy sonic artefacts when re-synthesized after going through the neural network pipeline.

The neural network architecture is structured in an autoencoder manner, and it is inspired by one of the architectures used in [15]: an input layer of 384 neurons, five dense layers of respectively 2048, 512, 256, 512 and 2048 neurons, and an output layer of 384 neurons.

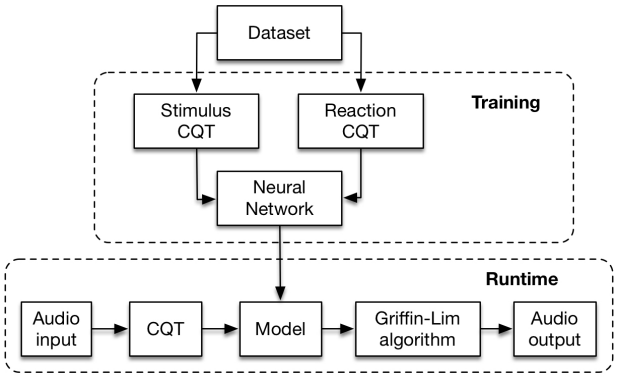

Figure 2. Architecture tiNNbre version 2

At runtime, we extract the CQT from the audio input and send it to the neural network. The CQT data yielded by the network is then synthesised through the Fast Griffin-Lim algorithm [23] implemented in Librosa.

5. EVALUATION

We ran a comparative survey between the two versions with five musicians. These were all professionally trained

musicians. Each musician was required to provide the sound material for the training of the model and a pre-recorded improvisation/composition employing all the four *stimulus* A-B-C-D [2]. After training the models with the sound material provided, and generating audio with the two different implementations of tiNNbre using the pre-recorded improvisation/composition provided, participants were asked to answer the following questions, rating them using a 1-5 Likert scale:

1. Overall, how do you find the system-generated output?
2. To what degree does the sound generated as the output varies at a different input sonic material?
3. As a musician, to what extent do you experience an interaction between the human musician and the system?
4. How likely will you use this system in your music-making process?

And, they were asked the following open-ended question:

- What are your thoughts on this music extract produced by the system considering timbrical features of the sound?

Every musician rated two outcomes of the system, one for each version for different musicians. Materials rated by each musician were presented in such a way to avoid that musicians could be biased by the awareness of the sound material provided to train the system. The evaluation order for the audio generated with tiNNbre v1 was:

- Musician 1 rated dataset 2 (Musician 2's material)
- Musician 2 rated dataset 3 (Musician 3's material)
- Musician 3 rated dataset 4 (Musician 4's material)
- Musician 4 rated dataset 5 (Musician 5's material)
- Musician 5 rated dataset 1 (Musician 1's material)

And, the one generated with tiNNbre v2:

- Musician 1 rated dataset 3 (Musician 3's material)
- Musician 2 rated dataset 4 (Musician 4's material)
- Musician 3 rated dataset 5 (Musician 5's material)
- Musician 4 rated dataset 1 (Musician 1's material)
- Musician 5 rated dataset 2 (Musician 2's material)

5.1 Dataset creation, and audio evaluation

Due to Covid-19 restrictions in 2021 in Italy, the production of audio materials by participants and the rating were not conducted in a controlled laboratory environment. Generated audio materials and listening activities were done with musicians already owned instruments and equipment. We acknowledge that the results described below might differ from those we aim to collect in a future formal study; yet, we believe that current observations are worth reporting to the community.

[2] Link to all the datasets and generated audio: https://cutt.ly/WbFR6Wt. Stimulus on the left channel, reactions on the right channel.

Musician 1 provided sonic materials recorded with a Nord Piano keyboard as a stimulus and a Nord Piano keyboard and Moog Mother 32 synthesizer as reactions. Musician 2 played a drum set both as stimulus and as reactions. Musician 3 played a tenor saxophone for both stimulus and reactions. Musician 4 employed different iPad apps that emulate synthesizers for both stimulus and reactions. Musician 5 played double-bass for both stimulus and reactions.

After we collected the materials from musicians, we generated new materials with tiNNbre v1 and v2. Subsequently, musicians were given the audio files and the survey. Audio materials were labelled "Listening 1" and "Listening 2" for files generated with tiNNbre v1 and v2, respectively. However, to avoid bias, participants were not made aware of which version of the system generated which audio.

5.2 Results

The neural networks for tiNNbre v1 and v2 have been trained with 50 epochs, an Adam optimizer [24], and learning rate 10^{-4}. tiNNbre v1 has been trained with batch size 20, v2 with batch size 100. On a Mac Mini, M1 chip, and 8 GB of RAM, the training of v1 takes 2 minutes and 50 seconds, while the training of v2 takes about 20 minutes. Considering that we could use much bigger datasets in future works, we tried different configurations of v2 architecture, with the same number of layers but fewer neurons, for faster training of the network. Unfortunately, all the different configurations produced sonic artefacts, in particular high pitched noises.

The output sonic material of tiNNbre v1 has been generated in 'delayed realtime', reproducing the musicians' improvisation audio files in Max/MSP. The output of v2 has been generated offline. With batch size 100, it takes about 7.7 seconds to generate 60 seconds of audio output.

We report the loss and accuracy metrics of both the training set and the test set in Table 1 and 2, respectively referring to tiNNbre v1 and v2, where loss measures the mean squared error between the the predicted outputs and the expected outputs, and the accuracy measures how many predicted outputs match the expected outputs.

Tables 3 and 4, report rating values of audio materials generated using tiNNbre v1 and v2 in relation to the dataset (D) used to generate the audio and the rating musician (RM). The tables also show the average (AVG) of those values. The last column on the right shows the AVG rate for each question, while the last row the AVG rate for each dataset. The cell on the bottom right shows the average rate of all the rates.

6. DISCUSSION

Results from the loss and accuracy metrics evaluation show that tiNNbre v1 has higher accuracy results, while v2 has lower loss results. However, the datasets used in the v1 have a different value range from the datasets used in the v2: in v1 it is made of MFCC features, in v2 of CQT features; thus, two different types of data. Since the loss met-

	D1	D2	D3	D4	D5
TR Loss	$2.3e^{-2}$	$1.6e^{-2}$	$1.5e^{-2}$	$2.1e^{-2}$	$3.3e^{-2}$
TR Acc.	0.62	0.40	0.41	0.64	0.84
TS Loss	$2.5e^{-2}$	$1.8e^{-2}$	$1.6e^{-2}$	$2.0e^{-2}$	$3.6e^{-2}$
TS Acc.	0.51	0.32	0.57	0.83	0.89

Table 1. Loss and accuracy metrics, tiNNbre v1. Legend: D = dataset; TR = Training; TS = Test.

	D1	D2	D3	D4	D5
TR Loss	$3.5e^{-4}$	$1.2e^{-4}$	$4e^{-4}$	$1.3e^{-2}$	$3e^{-4}$
TR Acc.	0.40	0.31	0.15	0.42	0.50
TS Loss	3.83^{-4}	$1.4e^{-4}$	$4e^{-4}$	$1.3e^{-2}$	$4e^{-4}$
TS Acc.	0.40	0.31	0.15	0.43	0.50

Table 2. Loss and accuracy metrics, tiNNbre v2. Legend: D = dataset; TR = Training; TS = Test.

ric measures the mean squared error, it should not be used to compare the two versions of the system. Instead, we can compare the accuracy performance between the two different versions. Accuracy does not seem to be related to the evaluation made by musicians: v1 has a higher accuracy than v2 (see Tables 1 and 2); v2 received a higher rating by musicians (see Tables 3 and 4). Furthermore, it seems that none of the evaluation metrics is related to musicians' evaluations. Despite good user evaluation, in general accuracy turned out to be low. It could be related to the difficulty of the neural network to exactly reproduce the vectors used in training.

Comparing the overall ratings, we can see that musicians preferred materials generated with tiNNbre v2. Another relevant aspect is that all audio materials generated with v2 were rated higher than v1, except those generated with dataset 2. Dataset 2 is made of percussive sounds, thus confirming the issues encountered in the development, where we looked for CQT parameters that could better resynthesise the attack of percussive sounds.

The answers to the open-ended question raised doubts from musicians about the effectiveness of the sounds generated by the system with the datasets used to train the neural network. For example, Musician 3 wondered if – where the system produced silence with tiNNbre v1 trained with dataset 4 – *"the reaction was intentionally a silence or if it is caused by the fact that, below a certain dynamic threshold, the system does not yield any reaction"*. About the sounds generated with tiNNbre v2 trained with dataset 4, Musician 2 commented on his perception of the middle section, in which there is no reaction by the system, but he decided to give a high rating anyway, assuming that the silence was the correct reaction. We know that silence matched the expected reaction, but we understand that it could have been confusing for a musician during the evaluation process. However, at this stage, we didn't want them to be conditioned by knowing the sonic materials used to generate the datasets, and we consider these answers valuable for future evaluation methodologies we will employ.

Musicians commented that reactions produced with the second tiNNbre v2 were more effective than v1. Musician

Q	D1	D2	D3	D4	D5	AVG
1	3	4	3	3	3	3.2
2	3	4	3	4	3	3.4
3	2	5	3	2	3	3
4	2	4	3	3	3	3
AVG	2.5	4.25	3	3	3	3.15

Table 3. Musicians' evaluations, tiNNbre v1. Legend: Q = Question, D = dataset, and RM = rating musician.

Q	D1	D2	D3	D4	D5	AVG
1	4	4	4	4	4	4
2	4	4	5	5	3	4.2
3	4	3	5	5	3	4
4	5	3	4	4	3	3.8
AVG.	4.25	3.5	4.5	4.5	3.25	4

Table 4. Musicians' evaluations, tiNNbre v2. Legend: Q = Question, D = dataset, and RM = rating musician.

4 – about the sounds generated with tiNNbre v2 trained with dataset 1 – found the stimulus and reactions more consistent, and perceived more sensitivity and adaptability of the sonic material proposed as output. Regarding the sounds generated with tiNNbre v2 trained with dataset 3, Musician 1, comparing it to the other track, wrote: *"I seem to grasp more autonomy between the question and the answer, without however losing coherence"*. Musician 2, about the sounds generated with tiNNbre v2 trained with dataset 4, found the reaction of the system very effective, the sounds and timbres being well blended and enriching the composition.

7. CONCLUSION AND FUTURE WORK

We presented two prototypes of tiNNbre, a timbre-based neural network musical agent, to foster the co-creative music improvisation and composition process. An evaluation of the two versions of the system showed musicians' preference towards the second version, which uses spectral analyses and re-synthesis and an autoencoder neural network. For this reason, this implementation will be the subject of the future development of this system. However, the second version proved to be less effective with percussive sounds. We will look for better performing spectral analyses and resynthesis for percussive sounds, test different loss functions, and formally evaluate the impact of different architectures in both versions of the system.

In the future, we plan to conduct a study in a laboratory setting so to gather more reliable data. As the system does not currently take into consideration the development of sound gestures over time, as shown in Equation 2, we will explore a different implementation of tiNNbre using sequence modelling, so to test the efficacy of the system in generating an outcome that takes into account a temporal sequence.

Acknowledgments

We thank all the musicians that took part to this research,

for their time, experience, and insightful comments.

8. REFERENCES

[1] J. D. Fernández and F. Vico, "Ai methods in algorithmic composition: A comprehensive survey," *J. Artif. Int. Res.*, vol. 48, no. 1, p. 513–582, Oct. 2013.

[2] K. Tatar and P. Pasquier, "Musical agents: A typology and state of the art towards musical metacreation," *Journal of New Music Research*, vol. 48, no. 1, pp. 56–105, 2019. [Online]. Available: https://doi.org/10.1080/09298215.2018.1511736

[3] F. Pachet, "The continuator: Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[4] G. Assayag, G. Bloch, and M. Chemillier, "Omax-ofon," in *Sound and Music Computing (SMC) 2006*, 2006, pp. 1–1.

[5] J. Nika, M. Chemillier, and G. Assayag, "Improtek: Introducing scenarios into human-computer music improvisation," *Comput. Entertain.*, vol. 14, no. 2, Jan. 2017. [Online]. Available: https://doi.org/10.1145/3022635

[6] J. Moreira, P. Roy, and F. Pachet, "Virtualband: Interacting with stylistically consistent agents." in *ISMIR*, 2013, pp. 341–346.

[7] M. Campbell, "Timbre (i)," Grove Music Online, 2021. [Online]. Available: https://doi.org/10.1093/gmo/9781561592630.article.27973

[8] E. J. Heller, *Why you hear what you hear: an experiential approach to sound, music, and psychoacoustics*. Princeton University Press, 2013.

[9] W. Hsu, "Strategies for managing timbre and interaction in automatic improvisation systems," *Leonardo Music Journal*, vol. 20, pp. 33–39, 2010. [Online]. Available: http://www.jstor.org/stable/40926371

[10] M. J. Yee-King, "An automated music improviser using a genetic algorithm driven synthesis engine," in *Applications of Evolutionary Computing*, M. Giacobini, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 567–576.

[11] D. Schwarz, G. Beller, B. Verbrugghe, and S. Britton, "Real-time corpus-based concatenative synthesis with catart," in *9th International Conference on Digital Audio Effects (DAFx)*, Montreal, Quebec, Canada, 2006, pp. 279–282.

[12] G. Bernardes, C. Guedes, and B. Pennycook, "Eargram: An application for interactive exploration of concatenative sound synthesis in pure data," in *From Sounds to Music and Emotions*, M. Aramaki, M. Barthet, R. Kronland-Martinet, and S. Ystad, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 110–129.

[13] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[14] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "Ddsp: Differentiable digital signal processing," *arXiv preprint arXiv:2001.04643*, 2020.

[15] K. Tatar, D. Bisig, and P. Pasquier, "Introducing latent timbre synthesis," *arXiv preprint arXiv:2006.00408*, 2020.

[16] R. Fiebrink and P. R. Cook, "The wekinator: a system for real-time, interactive machine learning in music," in *Proc. of The Eleventh International Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht)*, vol. 3, 2010.

[17] R. I. Godøy, "Gestural affordances of musical sound," in *Musical Gestures: Sound, Movement, and Meaning*. Routledge, 2010, pp. 103–125.

[18] N. Schnell, A. Röbel, D. Schwarz, G. Peeters, R. Borghesi *et al.*, "MuBu and friends–assembling tools for content based real-time interactive audio processing in Max/MSP," in *Proc. of the International Computer Music Conference*, ser. ICMC, Montreal, Quebec, Canada, 2009, pp. 426–430.

[19] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. Savannah, GA: USENIX Association, Nov. 2016, pp. 265–283.

[20] H. Eghbal-Zadeh, M. Schedl, and G. Widmer, "Timbral modeling for music artist recognition using i-vectors," in *2015 23rd European Signal Processing Conference (EUSIPCO)*. IEEE, 2015, pp. 1286–1290.

[21] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. of the 14th python in science conference*, ser. SciPy 2015, vol. 8, Austin, Texas, USA, 2015, pp. 18–25.

[22] C. Schörkhuber and A. Klapuri, "Constant-q transform toolbox for music processing," in *7th Sound and Music Computing Conference, Barcelona, Spain*, 2010, pp. 3–64.

[23] N. Perraudin, P. Balazs, and P. L. Søndergaard, "A fast griffin-lim algorithm," in *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2013, pp. 1–4.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.