Guiding Co-Creative Musical Agents through Real-Time Flute Instrumental Playing Technique Recognition

Marco Fiorini, Nicolas Brochec

GUIDING CO-CREATIVE MUSICAL AGENTS THROUGH REAL-TIME FLUTE INSTRUMENTAL PLAYING TECHNIQUE RECOGNITION

Marco FIORINI (marco.fiorini@ircam.fr) (0009-0004-0523-2481) [1, *] and
Nicolas BROCHEC (nicolas.brochec@pm.me) (0009-0000-5922-2079) [2, *]

[1] **IRCAM - STMS, CNRS, Sorbonne Université**, Paris, France
[2] **Tokyo University of the Arts**, Tokyo, Japan

[*] *The two authors contributed equally to this work*

ABSTRACT

This paper presents the development of a classifier for real-time flute Instrumental Playing Technique (IPT) recognition. Our classifier, implemented using Convolutional Neural Networks (CNN), shows a solid state-of-the-art accuracy in detecting different flute IPTs, with enough reactivity and steadiness to be accurate in real-time applications. In particular, we use our classifier's output to guide Somax2, a co-creative Corpus-Based Concatenative Synthesis (CBCS) improvisation system developed within the ERC project REACH, of which we are part of. By implementing a new Label Feature in the corpus annotation and in the output selection process of Somax2, we can drive its musical generation through the flute IPT detected by the classifier. Our real-time IPT recognition system offers a new dimension to the Somax2 interaction paradigm, where its artificial agents are now able to responsively engage with the recognized techniques. Contributing to the broader field of human-machine interaction in computer music, our results have potential applications in improvisation, computer-aided composition and new interfaces for musical expression.

1. INTRODUCTION

The paradigm of Corpus-Based Concatenative Synthesis (CBCS) has significantly transformed real-time sonic exploration in live music performances and creative processes. By exploiting extensive sound corpora, musicians can manipulate and concatenate audio segments, leading to the creation of entirely novel music pieces. This process has been successfully exemplified by tools like CataRT [1], now fully integrated into MuBu [2], or FluCoMa [3], presenting comprehensive toolkits for corpus manipulation that offer abundant possibilities for improvisers, sound designers and composers.

Within the domain of computer music, interaction has consistently played a crucial role in performance. Each musician responds to performance gestures, generating a sound influenced not only by the instrument - acoustic or computer - itself but also by the performance space and the performer's interaction with it. In the context of human-machine improvisation, the challenge becomes one of programming an environment with sufficient flexibility to qualify as interactive in response to a performer. Interaction, at any level, signifies a mutually influential process contributing to the ideas of Creative Instrument and Machine Musicianship [4].

The concept of musical agency assumes fundamental importance for this [5]. While the idea of music agents has evolved into a taxonomy of diverse applications [6], in the realms of music improvisation and co-creation, George E. Lewis's Voyager system stands as an early and successful embodiment of this concept [7]. Subsequent innovations such as Omax [8] and contributions from the Raising Co-Creativity in Cyber-Human Musicianship (REACH) project at IRCAM have further refined the notion of co-creativity in cyber-human applications. Introductions of temporal scenarios [9] and high reactivity [10] have breathed new life into the paradigm of co-improvisation and CBCS applications. These systems are, however, based on different models of musical interaction which rarely take into account the timbral aspect of a particular instrument, and can hardly recognise a change in instrumental techniques in real-time and act accordingly.

This work therefore presents an interactive and modular environment in which a particular flute Instrumental Playing Technique (IPT) is identified in real-time and used to guide the co-improvisation of a CBCS-based artificial agent, responding to the technique in question and maintaining stylistic consistency of improvisation thanks to its generative model. We selected the flute as the instrument to focus on because, to our knowledge, no research other than our own in the field of real-time IPT recognition and co-creative improvisation with AI musical agents has yet been conducted on the flute. We chose Somax2 [10] to handle the interactive co-creative part of our research as it is one of the software developed in one of our teams and part of our research area in the REACH project. In addition, although Somax2 possesses efficient responsiveness and immediate interaction capabilities with a real musician, it has no notion of timbre or recognition of instrumental techniques, being tied solely to pitch and chroma descriptors. Our research therefore aims to extend the interaction paradigm provided by its specific model and to explore general pos-

sibilities of integrating real-time IPT recognition into co-improvisation systems with artificial agents.

This paper is structured as follows: Section 2 briefly summarises existing approaches to the detection of ITPs and describes our proposed model for real-time recognition of flute IPTs. Section 4 describes the integration of our model in Somax2, and Section 5 explains the implementation of the proposed system. Section 3 details the results obtained, Section 6 discusses them and Section 7 presents the conclusions.

2. REAL-TIME IPT RECOGNITION

2.1 Existing Systems

Several studies tackle IPT identification based on principal component analysis [11] and onset detection [12]. These methods need a set of audio descriptors designed only to detect one specific IPT and cannot detect other IPTs. Thus, they are not sufficient for our task. Different research focusing on violin [13], electric guitar [14], Chinese bamboo flute [15], drums [16], cello [17], guitar [18] and flute [19] tackle IPT identification as a multi-class classification task using machine learning algorithms. A few of them have suggested using IPT classification for real-time performance [17–19]. To our knowledge, only the study on the cello [17] has tested their system in real-world situations. However, the proposed system has yet to reach sufficient accuracy and is not ready for practical application.

As mentioned above, MuBu [2] and FluCoMa [3] Max/MSP libraries can perform real-time machine listening. They provide several real-time audio signal analysis tools and machine learning algorithms. However, the available tools do not allow a thorough parametrization of the neural networks. Therefore, these tools are not adapted for our real-time IPT recognition task.

2.2 The Flute Playing Techniques

The flute can produce a chromatic scale thanks to the keys, from medium (C3≈261 Hz) to treble pitch (C6≈2093 Hz), and has a wide range of playing techniques. The sound of the flute is produced by the friction of the air on the mouthpiece and different changes in the velocity and characteristics of the blown air allow playing different techniques [20]. Some playing techniques of the flute are based on articulations, such as the *staccato*, *pizzicato*, and *flutter-tongue* techniques. Some are based on the air flux blown by the performer, such as the *ordinario*, *aeolian*, and *whistle tone* techniques. Some are based on fingering, such as the *multiphonics* and *trill* techniques, whereas some involve singing while blowing, *play and sing*, or hitting the keys hard, *key click* [21].

The flute can play many other techniques, but existing study [19] has shown that some should not be utilized as they strongly confuse classifiers because of their similarity to other techniques (i.e. the *aeolian and ordinario*, *harmonics*, *discolored fingering* techniques among others). Additionally, our preliminary experiments show that, despite the very good overall accuracy, the real-time recognition of short-time duration playing techniques, such as the *key click*, *pizzicato*, *staccato*, and *tongue ram* techniques, is very challenging for the classifier. Thus, for the sake of stability when used with Somax2, we decided not to train our classifier on short-time duration playing techniques. The aim of our real-time IPT recognition system is to identify the different flute playing techniques as audio data in real time. Therefore, we use a total of 7 playing techniques, as shown in Table 1.

aeolian	flatterzunge	multiphonics	ordinario
play and sing	trill	whistle tone	

Table 1. Flute playing techniques in 7 categories.

2.3 Datasets

We use the audio files of the *GFDatabase* [22] to train our model. The database includes five types of recordings (A, B, C, D, and E) from seven microphones placed at different distances from the source. This microphone set has recorded 11 flute playing techniques pitch by pitch within the respective register for each flute IPT. Our preliminary experiments show that using the A, B, C, and D recordings brought better performances than using all of them. As mentioned in 2.2, we do not use the *key click*, *pizzicato*, *staccato*, and *tongue ram* techniques.

To thoroughly assess our classifier, we evaluated it on a separate test dataset (heterogeneous datasets). We used the *FullSOL* sound bank [23] for our test dataset because it has the same flute playing techniques as in the training dataset. To ensure that the flute playing techniques in the *FullSOL* sound bank match those in the training dataset, we removed the unused techniques. The *GFDatabase* merged the highly similar playing techniques into the same categories. For example, minor and major second trills fall under the *trill* category. Similarly, playing a whistle tone with a glissando or not is considered performing a *whistle tone*. Playing and singing at the same pitch or not is considered performing a *play-and-sing* technique. Therefore, we applied this same methodology to the flute playing techniques of *FullSOL*.

2.4 Data format

To prepare our data, we follow an existing methodology [19]. We downsampled the audio files sample rate to 24kHz considering 12kHz (the Nyquist frequency) is sufficient to cover most flute harmonics. We trimmed the silence in the audio files because it is irrelevant. We then sliced the audio files into 15-frame-long sequences (≈ 320ms). The sequences are analyzed with a Log-Mel-Spectrogram (LMS) analysis from the Torchaudio library [24]. The LMS is computed on 128 bins, the FFT window is fixed at 2048 samples, and the hop size is set to 512 samples (≈ 21.3ms). The minimum frequency is set to 150Hz because lower frequencies are irrelevant for the flute. Each sound file is sliced into a maximum number of data samples according to the number of frames used. When the sequence length is shorter than 15 frames, we pad the audio sample with zeros. The data is then normalized.

2.5 Data Augmentations

For audio classification problems, data augmentation is frequently used to compensate for the lack of training data, and previous studies have demonstrated a notable increase in accuracy [25]. It consists of increasing the amount of data by applying several transformations to the data we already have. To do so, we utilized these three audio transformations: pitch shift, Gaussian noise addition, and time stretch.

For the pitch shifting, the original recording tuning is shifted randomly within a range of ± 100 Hz compared to the original tuning of 440 Hz. This is based on the assumption that the instrument's tuning can change between performances. Additionally, it would provide pitches out of the tonal scale, which can help the classifier generalize on unseen data. Gaussian noise is added to each file of the original recording. This is based on the assumption that noise is generated when the microphone signal is amplified. For the time stretching, we modify the audio sample length of $\pm 20\%$. This is based on the assumption that playing techniques can be performed at different speeds.

2.6 Model Architecture

To perform the real-time IPT recognition, we chose to implement a deep Convolutional Neural Network (CNN) architecture since several studies have shown its efficacy for instrument-related audio classification tasks [17,18,26,27]. In the field of IPT automatic recognition, existing research proposed several design strategies to build CNN architectures [17]. We first tested the suggested configurations by adapting their capacity to our data. Nevertheless, our preliminary tests have shown poor accuracy results on our dataset. Therefore, we propose augmenting the capacity of the suggested CNN architectures by adding several layers and selecting the hyper-parameters that increase accuracy. Figure 1 shows a schematic representation of our architecture.

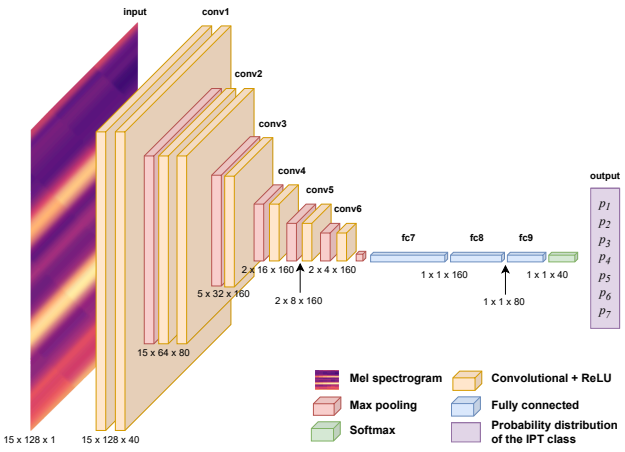

Figure 1. Schematic representation of the proposed architecture.

In our proposed architecture, conv1, conv2, and conv3 use a kernel size of *2x3*, while conv4, conv5, and conv6 use a kernel size of *2x2*. Our preliminary experiments have shown that using a rectangular-shaped kernel size of *2x3* on conv1, conv2, and conv3 led to better accuracy than square-shaped kernel sizes.

We add batch normalization and dropout layers after each convolutional layer. This is because the batch normalization layer fastens the training [28], while the dropout (fixed to 0.25 between each module) prevents overfitting [29]. After the sixth convolutional layer module, we connect three fully connected layers (fc7, fc8, and fc9). We use the identity activation function after these layers because our preliminary experiments have shown better results. Padding is "same" and strides equal to 1 on every convolutional layer.

2.7 Training

The weights of the neural network are initialized with the Xavier Normal Initialization [30]. The training is performed by minimizing the cross-entropy loss function. Mini-batch gradient descent is performed with ADAM optimization with an exponential decay of the learning rate (starting at lr=0.001 with patience of 10 epochs). The classifier is trained for 100 epochs with a V100 GPU machine.

2.8 Measurements

To evaluate the performance of our model, we performed four kinds of measurements. We first measured the accuracy after the training was completed. Then, we computed a confusion matrix to observe how well our classifier identified the playing techniques. These measurements are processed by evaluating the model on successive 15-frame sequences of LMS separated by 15 frames ($\approx$ 320ms).

For a comprehensive assessment of the performances of our model, we conducted a reactivity and steadiness study. For that purpose, we generated a random sequence of two test samples of different playing techniques. The end of the first sample is randomly truncated to create an abrupt change in the IPT. The aim of this test is to determine if the model can detect the change and maintain accurate predictions over time. These measurements are processed by evaluating the model on successive 15-frame sequences of LMS separated by 1 frame ($\approx$ 21.3ms), simulating a real-time situation.

We also measure the total delay time. This was defined by the sum of the audio sample gathering time ($\approx$ 21.3 ms), the audio samples processing time and the time induced by the classifier's prediction [19]. To ensure proper performance, the total delay time should not exceed the time of two frames ($\approx$ 42.6ms). If the system exceeds this limit, the delay will accumulate infinitely. We conducted 10 tests in real-time to measure the audio processing time and classifier-induced prediction time. We then calculated the average and the standard deviation.

3. RESULTS

In this section, we present the results of the evaluation of our model.

3.1 Accuracy and Confusion Matrix

Our model shows a consistent accuracy with a score of 92.56%. The confusion matrix in Figure 2 shows that the model identifies all the playing techniques well. The *trill* technique is the most accurately identified, with an accuracy score of 99.47%. The *multiphonics* technique is the least identified, with an accuracy score of 78.61%.

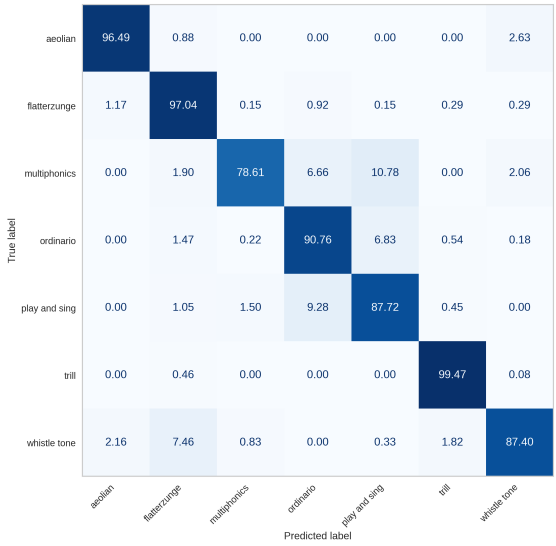

Figure 2. Confusion matrix (%).

3.2 Reactivity and Steadiness Study

We studied how our model reacts and remains steady in its predictions in a real-time situation. The graph in Figure 3 shows that it is reactive to changes and maintains consistency over time. However, two peaks represent misidentifying. The first is between the two techniques, and the second is at the end of the second technique. We think our model may be confused when encountering the first samples of a new technique or when a technique fades out to silence. On this specific test, it had an overall accuracy of 98.66%.

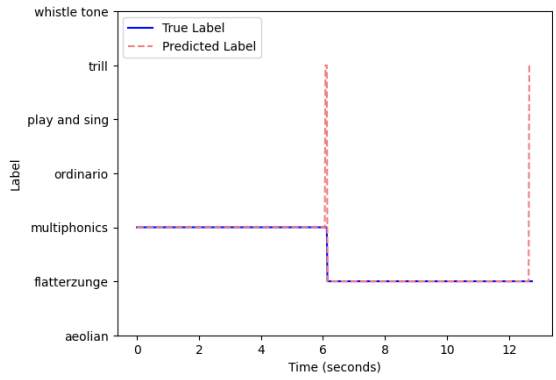

Figure 3. Reactivity and steadiness study: label prediction over time.

3.3 Total Delay Time

To measure the delay time induced by our model and the audio sample processing time, we measured the delay time for each prediction made on the sequence 10 times. We then computed the average and the standard deviation. We found an average of 6.56 ms and a standard deviation of 1.30 ms. With the sample gathering time ($\approx$ 21.3 ms) added, we found a total delay time of 27.86 ms, which is below the limit fixed in 2.8 ($\approx$ 42.6 ms).

4. INTEGRATION WITH SOMAX2

Somax2, developed in the Music Representation team at IRCAM as part of the ERC REACH project, serves to generate stylistically coherent musical interactions, particularly focused on the reactivity aspect of non-idiomatic improvisation. This is achieved through a generative model constructed via statistical learning from a designated corpus in either audio or MIDI format, while interacting with a human improviser. The system, comprehensively detailed in [10], is trained on user-selected musical material to establish a corpus tailored for improvisation purposes. In the following sections, we will give a brief overview of its fundamental concepts and describe the implementation process that resulted in the integration with our real-time IPT recognition system.

4.1 Somax2 Concepts

As described in [10], there are three main processes in the Somax2 system: building and modeling a corpus, influencing, and generating output. Diverging from pure generative methodologies, Somax2 generates improvisational material directly from the original musical data without recourse to an independent model. This genre-agnostic approach involves segmenting the musical corpus into fragments and subjecting each to a multilayer analysis based on various musical features, such as harmony (chroma) and melody (pitch). The resultant multilayer representation forms the foundation of a navigation model, facilitating non-linear exploration of the original data. Thanks to this, Somax2 stands apart from other generative systems by preserving the intrinsic qualities of a specific musical corpus, integrating them directly into the model, and presenting a highly reactive take on generative music tools.

In real-time interactions with a musician, Somax2 performs a similar segmentation and multilayer analysis on the input stream, generating activation peaks in the sequential memory when the analysed input matches the information stored in a specific segment of the corpus. These peaks, resembling probability distributions, signify potential output candidates. When a live input audio or MIDI stream arrives, this is sliced according to either the pitch or the onset and then analysed with pitch and chroma descriptors. The result is a rendition of the input stream in a symbolic domain, called influences, for any of the different layers. Each incoming influence is compared, at a given point in time, with the material stored in the corpus (which was previously built with a similar analysis,

but offline) in order to assess a possible match in the structure of the musical events produced. If this correspondence between internal and external multilayer classification occurs, a trigger peak is generated. At each new time step (after each new event generated) the peaks derived from each layer of the analysis are merged and scaled, depending on whether they are reinforced or not, and at this point the peak with the best score is chosen as the candidate for output.

The interplay of corpus segments, influences and peaks delineates the co-improvisation process. The system's capacity to dynamically balance its internal logic, thanks to self-listening, with external input from the musician results in a coherent yet complex musical output. This co-improvisation process effectively simulates the presence of an active agent in a collaborative musical experience, providing a mixture of coherence and unpredictability, and shows typical aspects of emergence and non-linear dynamics in improvisation, where the apparition of non-linear regimes of structure formation lead to rich musical co-evolution of forms [5].

4.2 Output Selection

Somax2 has up to this point mainly been operating on two musical dimensions when selecting its output: chroma and pitch. In later releases (from Somax 2.5) a number of so called scale actions were introduced to give the user some degree of control over other musical dimensions and various temporal aspects of the generated output. These scale actions operate as gaussian filters by scaling the score of the selected peak according to a parameter $\gamma \in \mathbb{R}_{[0,1]}$. Thanks to this, a peak p, as previously defined in [31], can now be detailed as: $p = \begin{bmatrix} t^{(\mathcal{C})} \\ \gamma y \end{bmatrix}$, where y is a score (height) designated to each peak by the model, $t^{(\mathcal{C})}$ the temporal position of the corresponding slice's onset, and γ can be described as a scale action function operating on a specific trait θ defined in the corpus.

4.2.1 Region Mask

The Region Mask scale action allows the user to specify a region of indices within the corpus that the output should be selected from, i.e. to select only one of multiple sections in a structured corpus. More specifically, the user specifies an interval $[a, b],\ a, b \in \mathbb{Z}_{[1,U]},\ a < b$ where U denotes the size of corpus $\mathcal{C}$, and applies a parameter γ (according to the new peak definition) defined as:

$$\gamma = \begin{cases} 1 & \text{if } a \leq u\left(\mathcal{S}_p^{(C)}\right) \leq b \\ 0 & \text{otherwise.} \end{cases}$$

where $S_p^{(C)}$ denotes the slice in the corpus C corresponding to the time t of the peak p and the function $u(S_p^{(C)})$ returns the index $u \in [1, ..., U]$ of this slice.

In spite of this, the need to have an indefinite number of events corresponding to any instrumental class while being able to efficiently switch from one class to another are missing points in favour of our design idea, detailed in later sections.

4.2.2 General Output Selection Mode

There are other methods in Somax2 to handle the selection of the generated output, but they all rely on filtering and selecting peaks based on the pitch and chroma descriptors the model works with. Since this implementation does not allow working on the detection and handling of different IPTs, we found it necessary to create a label filter that works as a kind of dynamic region filter. The output of this filter we implemented will then make use of the available output selection methods, thus working on the pitch and chroma levels, and the peak selection and decay processes illustrated above.

4.3 Label Feature Formalisation

As previously described in Section 4.2, the output selection modes currently available in Somax2 are not sufficient for integration with our real-time IPT classifier. We have thus defined and implemented a new scale action, called Label Feature, designed to enhance the capabilities of output selection modes in Somax2 when integrated with our system, enabling the user to selectively extract musical segments from a corpus based on the IPT class, resulting in a specific dynamic region masking.

4.3.1 Label Filter Parameter

To enable dynamic region selection based on label classification, we introduce a label filter parameter $\gamma(L)$ defined as:

$$\gamma(L) = \begin{cases} 1 & \text{if } L \in S_L \text{ (filter active)} \\ 0 & \text{otherwise (filter inactive),} \end{cases}$$

where S_L is a user-defined set of labels corresponding to the recognized IPT, determining the criteria for inclusion (and thus providing flexibility to it), and L denotes the classifier output, representing distinct labels corresponding to the recognized IPT, such as staccato, pizzicato, etc.

This filter parameter serves as a binary indicator for inclusion or exclusion of musical slices based on their associated labels. If the detected label L is in the specified set S, the filter is turned on as a switch $\gamma(L) = 1$, allowing slices associated with that label to be included in the output. Otherwise, the switch is off as $\gamma(L) = 0$, excluding slices with labels outside the set S_L.

4.3.2 Label Filter Function

To define a Label Filter, where the label of each slice determines which slices are selected for playback, we can define the set of slices associated with label l as S_l, keeping in mind that L represents the label output by the classifier. The filter can be represented as $F(L) = \bigcup_l S_l$,, where $F(L)$ represents the filtered set of slices based on the classifier output L, and $\cup$ denotes the union operation, which combines all sets S_l corresponding to the labels detected by the classifier.

In particular, giving a set of N possible labels, S_l can be defined for each label l, and the filter function becomes $F(L) = \bigcup_{l=1}^{N} S_l$, where $F(L)$ is the set of all slices that have labels matching the classifier output.

The overall filter function $F(L)$ can then be expressed, considering the filter parameter, as:

$$F(L,\gamma) = \bigcup_{l=1}^{N} \{S_l \mid \gamma(l) = 1\},$$

where S_l represents the set of musical slices associated with label l. In particular, $F(L,\gamma)$ is the union of sets S_l, where each set S_l contains slices from the corpus associated with label l. Only sets S_l for which $\gamma(L) = 1$ are included in the union, ensuring that slices are selected based on the presence of the label in the predefined set S.

At this point, the output slice S_w at time step w is a function of the peak matrix P_w (which contains the peaks combined with previous influences, shifted and decayed corresponding to the time passed since the previous influence according to the regular output selection mode), the corpus C and the history of previous output slices H_w. More specifically, we can define a set $\Xi = \{p_1, ..., p_H\}$ of viable peak candidates with the corresponding set of slices $\xi = \left\{\mathcal{S}_{p_1}^{(C)}, \ldots, \mathcal{S}_{p_H}^{(C)}\right\}$, where $S_{p_h}^{(C)} = F(L,\gamma), \quad \forall \mathcal{S}_{p_h}^{(C)} \in \xi$.

As shown in Figure 4, this formulation allows for the dynamic selection of slices based on the presence of specific N labels l in the predefined set S_l. Thus, the system adapts its output to focus on musical segments characterized by the recognized IPT. This combination of label filter parameter and label filter function results in the new implemented Label Feature.

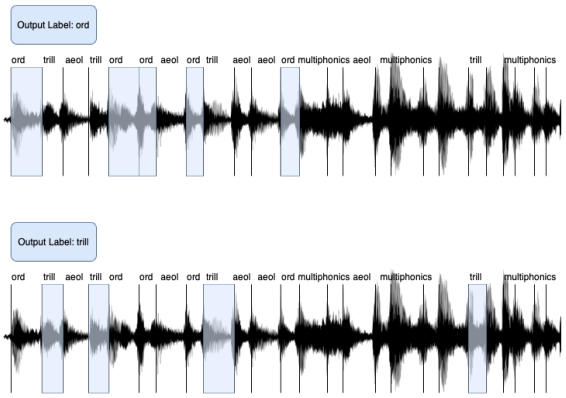

Figure 4. Result of the label filter function on two different IPTs (ord for ordinario, trill) annotated on the same audio corpus. The blue highlights show the output selected by the filter according to the correspondence with the various labels.

5. IMPLEMENTATION

In this section, we describe the implementation of the real-time IPT recognition system with Somax2 improvisation agents, called players. For the real-time IPT recognition system part, we use a Python back-end that runs in the background. The same principle applies for Somax2, where Max/MSP works as a front-end providing audio features, UIs and tools to interact in real-time with the application, but the whole generative core is handled by a Python back-end [10]. Therefore, our implementation ensures architecture consistency and the possibility of integrating our real-time IPT recognition system into the more complex Somax2 architecture.

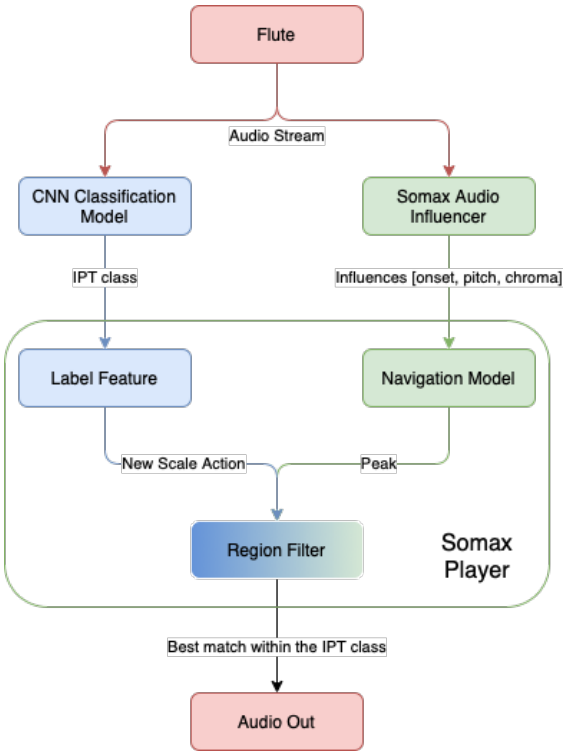

Figure 5. Detailed logic workflow derived from our implementation. The items in red represent the audio elements, those in green are the current modules and objects in the Somax2 model, while the ones in blue represent implementations of our modules.

5.1 Python back-end

On the real-time IPT recognition system side, the Python back-end processes the flute audio stream and runs the model in real time. We describe the processing flow step by step. The incoming audio signal is downsampled to 24 kHz and stored in a 7680 samples-length (15 frames) buffer. The stored samples are analyzed using the LMS before being normalized. Our model predicts the resulting 15 frames-long sequence, and an *argmax* function identifies the class with the highest probability. The class name is sent to Max/MSP through OSC protocol. The last 512 samples are removed, and the buffer is refreshed with the new 512 samples (1 frame). This process repeats. On Somax2 side, since its generative core lies in its Python code, the label feature mentioned in Section 4.3 has been implemented here as a new scale action [10] available to the somax.player.

5.2 Max/MSP front-end

The Max/MSP front-end serves multiple roles: it handles the flute audio stream from the audio interface, sends it to the Python back-end, receives the predicted IPT, and runs Somax2 and its improvisation agents. To send the

audio signal of the flute to Python back-end, we use the BlackHole [1] audio routing device. The predicted IPT is received through the OSC protocol and processed in Somax2 through messages corresponding to the label feature scale action, driving in real-time the selection of the corresponding IPT class. The complete workflow of interaction derived from our implementation can be seen in detail in Figure 5.

6. DISCUSSION

Our model has shown a solid state-of-the-art accuracy score of 92.56%, and the confusion matrix in Figure 2 illustrated its performance by classifying the various flute playing techniques. The reactivity and steadiness study has shown that our model can be accurate in real time. The misidentification issue can be alleviated by implementing a moving average on the probability distribution outputted by the classifier. However, this sacrifices reactivity. The calculation of total delay time showed that our model can be run in real-time without exceeding the limit ($\approx$ 42.6 ms). As mentioned in section 2.2, the short time duration playing techniques such as *key click*, *pizzicato*, *staccato* and *tongue ram* techniques have been removed from the datasets for our experiment. Indeed, our CNN-based classifier model did not provide enough stability for identifying these techniques during preliminary tests. We think adding audio descriptors [32] to Log-Mel-Spectrograms data samples may help the classifier, as they would bring better audio representations.

Future work will include testing and perfecting our system in studio with professional improvisers. We intend to expand the model to other instruments, starting with the clarinet and electric guitar, thus building specific datasets. Thanks to the future conception of a specific Max/MSP object, which would also improve the architectural aspects of the current implementation, the system could also be natively accessible in Max. In this way, user utilisation would also be simplified and a hybrid, multi-level training system could be conceived, in which in addition to providing specific datasets, the user can carry out new training stages on their own database to calibrate the system to their instrument.

7. CONCLUSIONS

This paper presented the application of a state-of-the-art model to recognize flute instrumental playing techniques in real time and to guide the interaction of improvisational artificial agents through this detection, using the co-creative system Somax2.

By exploring playing technique possibilities of a musical instrument, our implementation contributes to the domain of immersive sound, giving new perspectives to work on real-time recognition of IPT and their integration with different human-machine co-creativity frameworks. At the same time, by working on new strategies to engage with sound and music in real time, our research falls within the domain of immersive computing, providing new insights for human-computer interaction in the context of improvised music but potentially extending to computer-assisted-composition scenarios and new interfaces for musical expression.

[1] `https://existential.audio/blackhole/`

Acknowledgments

This research is supported by the European Research Council (ERC) as part of the Raising Co-creativity in Cyber-Human Musicianship (REACH) Project directed by Gérard Assayag, under the European Union's Horizon 2020 research and innovation program (GA #883313). The authors would like to thank Joakim Borg for the continuous Somax2 expertise.

8. REFERENCES

[1] 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, Canada, Sep. 2006, pp. 279–282.

[2] 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 *ICMC*, 2009.

[3] P. A. Tremblay, O. Green, G. Roma, J. Bradbury, T. Moore, J. Hart, and A. Harker, "Fluid corpus manipulation toolbox," Jul. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6834643

[4] R. Rowe, *Interactive Music Systems: Machine Listening and Composing*. Cambridge, MA, USA: MIT Press, 1992.

[5] D. Borgo, *Sync or Swarm: Improvising Music in a Complex Age*. AC Black, 2005.

[6] 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.

[7] G. E. Lewis, "Too many notes: Computers, Complexity and Culture in Voyager," *Leonardo Music Journal*, vol. 10, pp. 33–39, 2000.

[8] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov, "Omax brothers: a dynamic topology of agents for improvization learning," in *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, 2006, pp. 125–132.

[9] J. Nika, K. Déguernel, A. Chemla, E. Vincent, G. Assayag *et al.*, "DYCI2 agents: merging the "free", "reactive", and "scenario-based" music generation paradigms," in *International Computer Music Conference*. Shangai, China, 2017.

[10] G. Assayag, L. Bonnasse-Gahot, and J. Borg, "Cocreative Interaction: Somax2 and the REACH Project," *Computer Music Journal 1-19*, pp. 1–34, Feb. 2024. [Online]. Available: https://doi.org/10.1162/comj_a_00662

[11] M. Malt and M. Gentilucci, "Real time vowel tremolo detection using low level audio descriptors," *arXiv preprint arXiv:1511.07008*, 2015.

[12] M. Malt and E. Jourdan, "Real-time uses of low level sound descriptors as event detection functions using the max/msp zsa. descriptors library," in *Brazilian Symposium on Computer Music*, vol. 12, 2009, pp. 45–56.

[13] L. Su, H.-M. Lin, and Y.-H. Yang, "Sparse modeling of magnitude and phase-derived spectra for playing technique classification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 2122–2132, 2014.

[14] Y.-P. Chen, L. Su, Y.-H. Yang *et al.*, "Electric guitar playing technique detection in real-world recording based on f0 sequence pattern recognition." in *ISMIR*, 2015, pp. 708–714.

[15] C. Wang, E. Benetos, V. Lostanlen, and E. Chew, "Adaptive scattering transforms for playing technique recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 1407–1421, 2022.

[16] P. Herrera, A. Yeterian, and F. Gouyon, "Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques," in *Music and Artificial Intelligence: Second International Conference, ICMAI 2002 Edinburgh, Scotland, UK, September 12–14, 2002 Proceedings*. Springer, 2002, pp. 69–80.

[17] J.-F. Ducher and P. Esling, "Folded cqt rcnn for real-time recognition of instrument playing techniques," in *International Society for Music Information Retrieval*, 2019.

[18] A. Martelloni, A. P. McPherson, and M. Barthet, "Real-time percussive technique recognition and embedding learning for the acoustic guitar," *arXiv preprint arXiv:2307.07426*, 2023.

[19] N. Brochec and T. Tanaka, "Toward real-time recognition of instrumental playing techniques for mixed music: A preliminary analysis," in *International Computer Music Conference (ICMC 2023)*, 2023.

[20] N. H. Fletcher and T. D. Rossing, *The physics of musical instruments*. Springer Science & Business Media, 2012.

[21] C. Levine and C. Mitropoulos-Bott, *The Techniques of Flute Playing I/Die Spieltechnik der Flöte I*. Bärenreiter-Verlag, 2019.

[22] N. Brochec and W. Howie, "GFDatabase: A Database of Flute Playing Techniques," Feb. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.10612396

[23] C. E. Cella, D. Ghisi, V. Lostanlen, F. Lévy, J. Fineberg, and Y. Maresz, "Orchideasol: a dataset of extended instrumental techniques for computer-aided orchestration," *arXiv preprint arXiv:2007.00763*, 2020.

[24] Y.-Y. Yang, M. Hira, Z. Ni, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Mahadeokar, J. Hwang, J. Chen, P. Goldsborough, P. Roy, S. Narenthiran, S. Watanabe, S. Chintala, V. Quenneville-Bélair, and Y. Shi, "Torchaudio: Building blocks for audio and speech processing," *arXiv preprint arXiv:2110.15018*, 2021.

[25] S. Wei, S. Zou, F. Liao *et al.*, "A comparison on data augmentation methods based on deep learning for audio classification," in *Journal of physics: Conference series*, vol. 1453, no. 1. IOP Publishing, 2020, p. 012085.

[26] V. Lostanlen and C.-E. Cella, "Deep convolutional networks on the pitch spiral for musical instrument recognition," *arXiv preprint arXiv:1605.06644*, 2016.

[27] Y. Han, J. Kim, and K. Lee, "Deep convolutional neural networks for predominant instrument recognition in polyphonic music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 208–221, 2016.

[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. pmlr, 2015, pp. 448–456.

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[30] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[31] J. Borg, "Dynamic Classification Models for Human-Machine Improvisation and Composition," Master's thesis, Aalborg University, 2020.

[32] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," *CUIDADO Ist Project Report*, vol. 54, no. 0, pp. 1–25, 2004.