Toward Real-Time Recognition of Instrumental Playing Techniques for Mixed Music: A Preliminary Analysis

Nicolas Brochec, Tsubasa Tanaka

Toward Real-Time Recognition of Instrumental Playing Techniques for Mixed Music: A Preliminary Analysis

Nicolas BROCHEC
Tokyo University of the Arts
nicolas.brochec@pm.me

Tsubasa TANAKA
Tokyo University of the Arts
tanaka.tsubasa@ms.geidai.ac.jp

ABSTRACT

In contemporary mixed music, real-time digital sound processing is often applied to live instrumental performances. However, switching between sound effects often relies on manual computer operation or the performer's foot pedal, burdening the operator. This research aims to develop a system that automatically classifies instrument playing techniques and switches sound effects according to the result of the classification, reducing the burden on the operator and expanding creative possibilities in contemporary mixed music. To realize such a system, the classification accuracy of existing research is not sufficient. In this study, we focused on the flute and tested various input data formats to improve the accuracy of the classification. The results show that using a number of frames of around 15 with the Log-Mel-Spectrogram (LMS) data format improves accuracy. Furthermore, we have measured the computational times of some classification algorithms to assure that the system can actually be used in real time. We found that Multi-Layer Perceptron (MLP) with LMS data format was the best choice among them because it had high accuracy and was fast enough, while other algorithms had concerns about computational speed.

1. INTRODUCTION

In the field of contemporary mixed music (*musique mixte*), real-time digital acoustic effects were applied to live instrumental performances circa 1980. This advancement was made possible thanks to increasing computer calculation speeds. Nevertheless, sound effects are usually fixed for each section of a piece, and switching between them is often done by computer operation or the performer's foot pedal, which places a burden on the operator. This is where automated score follower systems such as Antescofo [1] have been developed. These systems track the performer's temporal position on the score and automatically switch effects, reducing the burden on the operator. However, score followers require the score input information for each piece of music in order to estimate the performer's position, which adds an extra burden. Furthermore, instrumental techniques used in contemporary mixed music are nowadays various, and Antescofo, except for some, is not capable of identifying them all. This research proposes developing a system that automatically identifies instrument playing techniques (IPT) and switches sound effects in real time according to them. This would reduce the burden on the operator and the cost of music score input. If put to practical use, the creative possibilities of mixed music would expand to make better use of the real-time nature of sound effects.

There are some existing studies that tackle IPT identification based on principal component analysis [2] and onset detection [3]. These approaches require the design of a set of audio descriptors that detect only one IPT and cannot detect other IPTs. Hence, they are not sufficient for our task. Other research focusing on violin [4], cello [5, 6], electric guitar [7], Chinese bamboo flute [8], and drums [9] tackle IPT identification as a multiclass classification task using machine learning algorithms. To our knowledge, only the studies on cello [5, 6] have suggested using IPT classification for real-time performance. For this purpose, they attempted to discriminate various cello playing techniques using convolutional neural networks (CNN). Nonetheless, the proposed system has not yet reached sufficient accuracy and is not yet ready for practical application.

We think that one of the ways to improve the accuracy of IPT classification is to test different data dimensionalities and measure how they impact accuracy. In [5, 6], a number of frames of 60 have been employed as the data unit length. We did preliminary experiments with the flute instead of the cello, with different numbers of frames. It seemed that a smaller number of frames increased accuracy.

Therefore, the objective of this study is to find the most preferable data dimensionality to enhance the global accuracy of classifiers. In addition, we measure the maximum delay time to evaluate the real-time performance of the system. We selected the flute as the instrument for the experiments. This is because the first author of this paper is actually planning to compose a piece for flute that he is familiar with using this real-time classification system. To our knowledge, no research other than our own in the field of IPT classification has yet been conducted on the flute.

2. FLUTE PLAYING TECHNIQUES

There is a wide range of playing techniques for each type of instrument. In the case of the flute, different techniques such as *legato* and *staccato* articulations, *vibrato* and *flut-*

1	aeolian	11	play-and-sing
2	aeolian-and-ordinario	12	play-and-sing-unison
3	discolored-fingering	13	sforzando
4	flatterzunge	14	staccato
5	fortepiano	15	tongue-ram
6	harmonic-fingering	16	trill-major-second-up
7	key-click	17	trill-minor-second-up
8	multiphonics	18	whistle-tone
9	ordinario	19	whistle-tone-sweep
10	pizzicato		

Table 1. Flute playing techniques on 19 categories

ter, *slaps*, *key percussion* and other special techniques can be performed. The sound of the flute is mainly produced by the friction of air blown into the mouthpiece. Differences in the velocity and characteristics of the blown air cause changes in technique [10]. For example, the velocity of the air causes changes in volume and octave, and the adjustment of the airflow by the tongue produces differences in attack, such as *staccato*, *slap* and *flutter*. The flute can produce a chromatic scale thanks to the keys, from medium (C3 $\approx$ 261Hz) to treble pitch (C6 $\approx$ 2093Hz). Special techniques such as multiphonics and quarter tones can also be achieved by unusual fingering [11, 12]. The aim of this research is to automatically identify these various techniques as audio data in real time.

3. TRAINING DATASET

In this study, we perform an experiment that aims to investigate which data format is most adequate for flute playing technique classification. For this purpose, we create multiple training datasets with different data formats and test them on a set of algorithms.

3.1 Sound Bank

To form our training datasets, we use FullSOL [13], an existing sound bank of instrument solo recordings that include various IPTs of classical music instruments. This sound bank includes 27 different flute techniques, totaling 689MB and 1261 audio files for approximately 2 hours of sound.

3.2 Techniques Selection

Some techniques recorded in the FullSOL sound bank are not suitable for this task. For instance, transitional techniques (i.e. from ordinario sound to aeolian sound) could cause confusion for the classifiers, given their evolution over time. Such techniques will not be used in this study. Also, similar techniques that have been separated for audio sequencer needs (i.e *note-lasting* and *ordinario* techniques) are merged into one category. This study is then a 19-class problem and uses 388MB of data, comprising 837 audio files for approximately 1 hour of sound. The 19 categories of flute playing techniques are shown in the Table 1.

n_frames	60	45	30	15	10	5
LMS	7680	5760	3840	1920	1280	640
FBA	128	128	128	128	128	128

Table 2. Data dimensionality for one sample

3.3 Data Formats

Following the methodology in [5, 14], we downsampled the audio file sample rate to 24kHz considering 12kHz (the Nyquist frequency) is sufficient to cover most flute harmonics. Additionally, reducing data dimensionality will decrease the prediction time. We trimmed the silence in the audio file because it is irrevelant. We then sliced the audio file into 60-frame-long sequences ($\sim$1.28s). The sequences are analyzed with a Log-Mel-Spectrogram (LMS) analysis from the Librosa library [15] using the same parameters as in [5, 6]. The LMS is computed on 128 bins, the FFT window is fixed at 2048 samples, and the hop size is set to 512 samples ($\sim$21.3ms). This is because the sensation of simultaneity between two different sound stimuli begins when the time gap between them is less than 20ms [16]. The data is then normalized.

Each sound file is sliced into a maximum number of data samples according to the number of frames used. In cases of lack of length, as for *pizzicato*, *staccato* and *key clicks*, sample arrays are padded with zeros.

In order to investigate which data format is most suitable for our task, we propose reducing the data dimensionality using two methods. The first method consists of reducing the number of total frames from 60 frames to 5 frames. The second method consists of computing the average of each frequency bin (FBA) of a Log-Mel-Spectrogram over the number of frames. Table 2 shows the data dimensionality for both methods according to the number of frames.

4. METHODS OF CLASSIFICATION

4.1 Algorithms

To compare different algorithms, we chose three different types of them from the Scikit-learn library [17]. We have tested several parameter configurations and selected those that enhanced accuracy. Support Vector Machine Polynomial Kernel is set to default parameters, and the k-Nearest Neighbours algorithm is set to 1 neighbour (instead of 5). Multi-Layer Perceptron is set with two hidden layers of 400 neurons each because the default configuration didn't have sufficient layers to process large data dimensionalities.

4.2 Data Augmentation

To compensate for the lack of training data, data augmentation is often effective for audio classification tasks, and existing research has shown a substantial rise in accuracy [18]. The method consists of increasing the amount of data by making some modifications to the data we already have. For our experiments, we use three types of data augmentation methods: pitch shift, reverb addition, and noise addition, similar to those used in existing research [5, 6]. For the pitch shifting, the original recording tuning is shifted

n_frames	60	45	30	15	10	5
n_samples	10336	13832	21177	43165	65295	131937

Table 3. Total n_samples according to frames number

randomly within a range of ±20 Hz compared to the original tuning of 440 Hz. This is based on the assumption that the tuning of the instrument can change between performances. For reverb addition, a reverb of a fixed length is applied to the original sound source. This is based on the assumption that reverberation differs depending on the performance environment. For noise addition, white noise is added randomly to each file of the original recording. This is based on the assumption that noise is generated when a microphone is used to capture the sound of a musical performance. This data augmentation increases the number of training data points by a factor of four compared to the original data. As shown in the Table 3, the total training data samples are shown according to the number of frames after performing data augmentation.

5. EXPERIMENTS

We perform two experiments to find the best data format for accuracy score and prediction time. We first test our set of algorithms with different numbers of frames using the LMS and FBA data formats. Then, we measure the total prediction time for each data format, number of frames, and algorithm.

5.1 Test Data

In our experiments, we use *train_test_split* function from the Scikit-learn library to select 25% of our data for the test dataset. Because the amount of data for each class is not balanced, we use the parameter *stratify* of this function to select approximately the same percentage of samples from each target class as the complete dataset.

5.2 Accuracy Measure

As the train and test datasets are balanced, they may not vary much with different splits of train and test data. That is why, for SVM and KNN algorithms, we measure the accuracy only once. For MLP algorithm accuracy, we measure accuracy for each of the 500 epochs of training and then apply a moving average to the prediction score for each iteration. Computing the moving average helps filter out fluctuations in the prediction scores and represents overall performance rather than performance on specific tests.

5.3 Total Delay Time Calculation

The total delay time is composed of three different types of time. First, there is the time required to gather enough samples to process the spectrogram analysis. Since our system aims to make a prediction at each frame, this time is equal to the hop size value of the spectrogram analysis at 24kHz (512 samples $\approx$ 21.3ms).

Second, there is the time required to compute the spectrogram analysis. As the results may depend on CPU background task load, this was computed 10,000 times on the same audio sample, with any other application opened. From the 10,000 measurements, we computed the average, the standard deviation, and the maximum for a better understanding of the time required to compute the spectrogram analysis.

Third, there is the prediction time induced by the algorithm. We measured the prediction time on the first 500 test data samples and computed the average, the standard deviation, and the maximum.

The prediction time induced by each of the classification algorithms (SVM, kNN, MLP) must not exceed 21.3ms assuming that the predictions are done sequentially. If the time for a prediction for one frame is over 21.3ms, the next prediction is delayed for this, and the delays of the frames accumulate unlimitedly.Therefore, summing the time for gathering samples and the limit of the prediction time, the upper limit of the total maximum delay time for our system is two frames long, or 42.6 ms.

The time measurements are highly dependent on computer calculation speed. For these experiments, we used an i7-8700K CPU with 32.0 GB of RAM on a 64-bit Windows 10 operating system PC.

6. RESULTS

6.1 Accuracy

6.1.1 LMS Data Format

Results on Table 4 indicate that SVM is the best algorithm for our 19-class classification task. When using a frame number of 15, the accuracy reaches 77.50%.

n_frames	60	45	30	15	10	5
SVM	74,68	71,00	72,95	**77,50**	73,00	67,59
kNN	67,69	69,40	69,80	66,04	66,42	65,15
MLP	71,54	72,68	73,58	77,17	75,73	70,91

Table 4. Accuracy (%) on 19 classes, LMS data format

6.1.2 FBA Data Format

Results on Table 5 show that for the number of frames 60, 45, and 30, the MLP algorithm achieves higher accuracies with the FBA data format compared to the LMS data format. When using a frame number of 30, the accuracy reaches 75.40%.

n_frames	60	45	30	15	10	5
SVM	69,12	65,96	66,23	57,84	62,29	60,77
kNN	61,05	63,35	63,07	61,61	63,49	58,55
MLP	74,02	74,55	**75,40**	72,35	72,58	69,05

Table 5. Accuracy (%) on 19 classes, FBA data format

6.2 Confusion Matrices and Case Analysis

We computed confusion matrices to understand how well our set of algorithms identified different techniques. We provide confusion matrices for LMS and FBA data formats, with the number of frames corresponding to the best accuracy score for each algorithm.

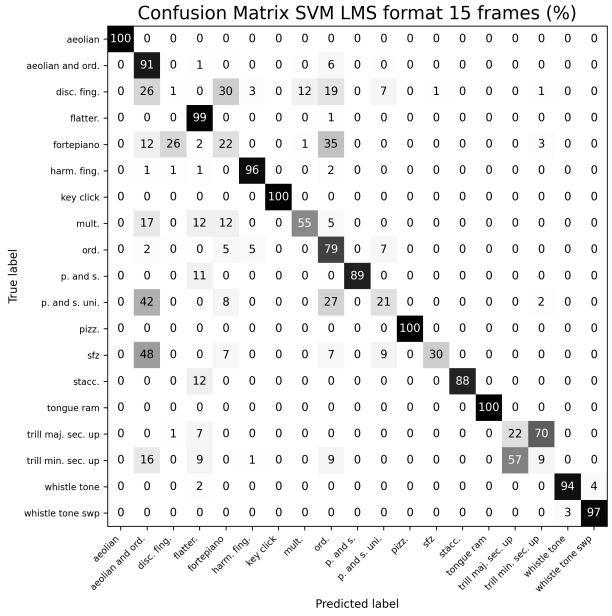

Figure 1. Confusion matrix, SVM, LMS data format, 15 frames

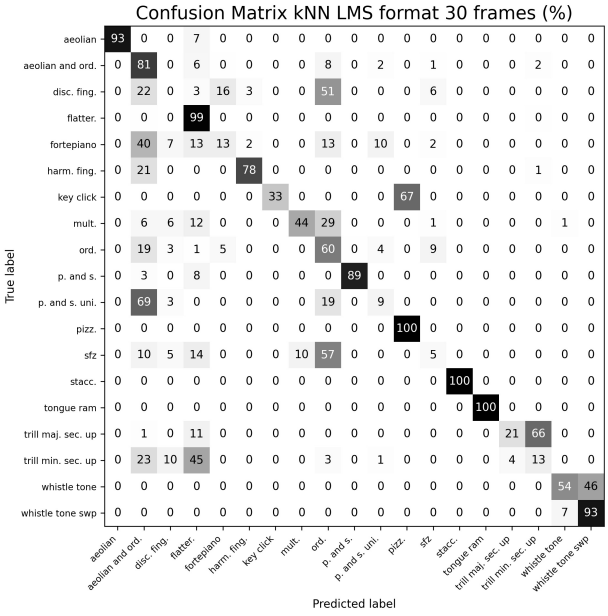

Figure 2. Confusion matrix, kNN, LMS data format, 30 frames

6.2.1 LMS Data format

We can see in Figures 1, 2, and 3, that the majority of the techniques are well identified by SVM and MLP. In the case of kNN, we can see a lack of identification even though some techniques of identification reach high accuracies (*aeolian*, *flatterzunge*, *play and sing unison*, *pizzicato*, *staccato*, *tongue ram*, *whistle tone sweep*). This may be attributed to the fact that some techniques are too similar. We can see for the three algorithms that *trill major second up* and *trill minor second up* techniques are misidentified. Also, *sforzando* is misidentified with *aeolian and ordinario* for SVM and *ordinario* for kNN. MLP is performing better than SVM and kNN, with fewer errors. All three algorithms do not accurately identify the *discolored fingering* technique. The *discolored fingering* technique consists of playing a note *ordinario* with an alternative fingering. This results in a soft detuning of the note. Because we are using pitch shifting as a data augmentation technique, it may be difficult for the algorithms to distinguish the augmented data of *ordinario*, *fortepiano*, and *aeolian and ordinario* techniques from the original data of the *discolored fingering* technique.

6.2.2 FBA Data format

Figures 4, 5 show that a lot of techniques are misidentified. The notably low accuracy score may be attributed to the small dimensionality of the data. For SVM, all short-length techniques, such as *key click*, *pizzicato*, *staccato*, *tongue ram* have been predicted as *staccato*. This may be attributed to data scarcity. Indeed, in the case of the *staccato* technique, there are in our dataset only 38 audio files, each approximately 400ms in length. Only a limited amount of training data can be generated from a single audio file (400ms $\approx$ 19 frames). Figure 6 shows that the performance of MLP is always better than that of SVM and kNN. Nevertheless, similar techniques such as *trill major second up* and *trill minor second up*, *whistle tone* and *whistle tone sweep* are misidentified.

6.3 Total Delay Time

To ensure that our system can be used for real-time performance, we measured the total delay time, the sum of the time required to gather enough samples to process the spectrogram analysis, the time required to compute the spectrogram analysis, and the time induced by the algorithms of predictions (see Section 5.3).

6.3.1 Log-Mel-Spectrogram Computation Time

Table 6 and Table 7 show the computation times for LMS and FBA data formats. The computation time of LMS data format is comparable to that of the FBA data format. Given that the maximum time is approximately 16.5μs, we can conclude that the Log-Mel-Spectrogram analysis computation time is negligible. Therefore, it will not be included in the total delay time calculation.

n_frames	60			45			30		
	mean	sd	max	mean	sd	max	mean	sd	max
LMS	7,3	0,4	12,8	6,2	0,4	16,5	4,7	0,3	9,6
FBA	7,3	0,4	13,9	6,2	0,4	16,1	4,7	0,3	9,7

Table 6. Log-Mel-Spectrogram calculation time (μs), computed 10,000 times.

n_frames	15			10			5		
	mean	sd	max	mean	sd	max	mean	sd	max
LMS	3,7	0,3	7,5	3,2	0,3	6,5	2,8	0,2	7,4
FBA	3,6	0,3	7,2	3,3	0,3	6,5	2,8	0,2	5,3

Table 7. Log-Mel-Spectrogram calculation time (μs), computed 10,000 times.

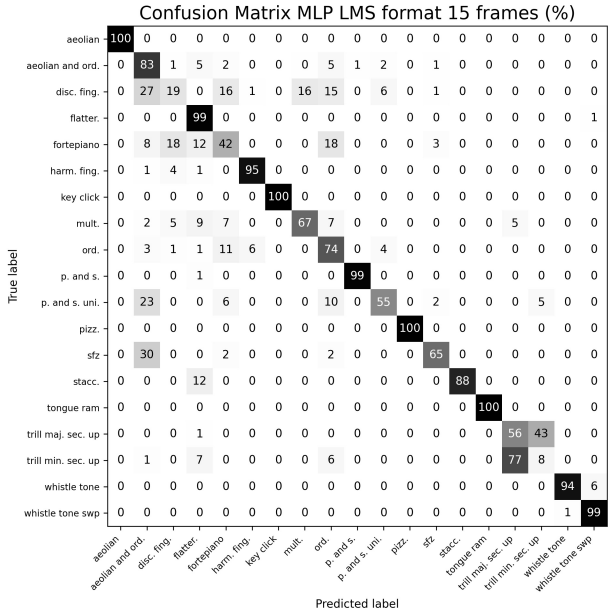

Figure 3. Confusion matrix, MLP, LMS data format, 15 frames

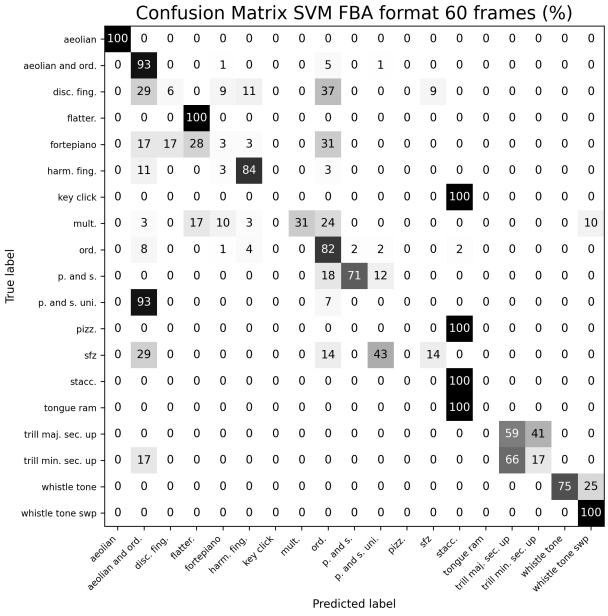

Figure 4. Confusion matrix, SVM, FBA data format, 60 frames

6.3.2 Algorithm's Prediction Time

Tables 9 to 14 show the prediction time induced by the algorithms according to the number of frames. The results show that prediction with FBA data format (Table 11, Table 12, Table 13) is performed much faster than LMS data format (Table 8, Table 9, Table 10) particularly for SVM and kNN algorithms. For kNN, the calculation time is related to the amount of data, whereas for SVM, the calculation is slower because the LMS data format has a wider data dimensionality than the FBA data format. Concerning the MLP algorithm, there is not much difference between the two data formats (time difference less than 2.0ms) even though prediction time with the FBA data format is still 50% quicker than the LMS data format. In summary, the FBA data format hastened prediction times significantly for SVM and kNN algorithms, taking only approximately a tenth of the time required by the LMS data format.

n_frames	60			45		
	mean	sd	max	mean	sd	max
SVM	17,62	2,09	25,71	15,79	2,10	23,63
kNN	60,57	2,48	70,61	60,50	2,17	68,67
MLP	1,49	0,32	2,58	1,22	0,23	1,88

Table 8. Prediction time (ms), LMS data format

n_frames	30			15		
	mean	sd	max	mean	sd	max
SVM	13,80	1,936	21,15	10,90	2,44	19,92
kNN	64,86	2,91	87,50	66,49	2,90	83,17
MLP	0,37	0,09	1,27	0,48	0,10	1,03

Table 9. Prediction time (ms), LMS data format

n_frames	10			5		
	mean	sd	max	mean	sd	max
SVM	8,60	1,56	13,80	6,90	1,40	10,35
kNN	66,49	2,90	83,17	67,16	2,68	84,56
MLP	0,48	0,10	1,03	0,80	0,09	1,15

Table 10. Prediction time (ms), LMS data format

n_frames	60			45		
	mean	sd	max	mean	sd	max
SVM	0,27	0,05	0,64	0,32	0,06	0,67
kNN	1,79	0,3	3,01	2,17	0,33	3,19
MLP	0,27	0,05	1,01	0,27	0,06	1,03

Table 11. Prediction time (ms), FBA data format

n_frames	30			15		
	mean	sd	max	mean	sd	max
SVM	0,41	0,11	0,89	0,67	0,18	1,44
kNN	2,83	0,41	4,07	4,91	0,60	3,39
MLP	0,28	0,06	1,05	0,32	0,04	0,72

Table 12. Prediction time (ms), FBA data format

n_frames	10			5		
	mean	sd	max	mean	sd	max
SVM	1,27	0,35	4,15	2,35	0,43	3,92
kNN	7,50	0,84	11,96	12,73	1,10	16,50
MLP	0,33	0,04	0,54	0,39	0,05	0,79

Table 13. Prediction time (ms), FBA data format

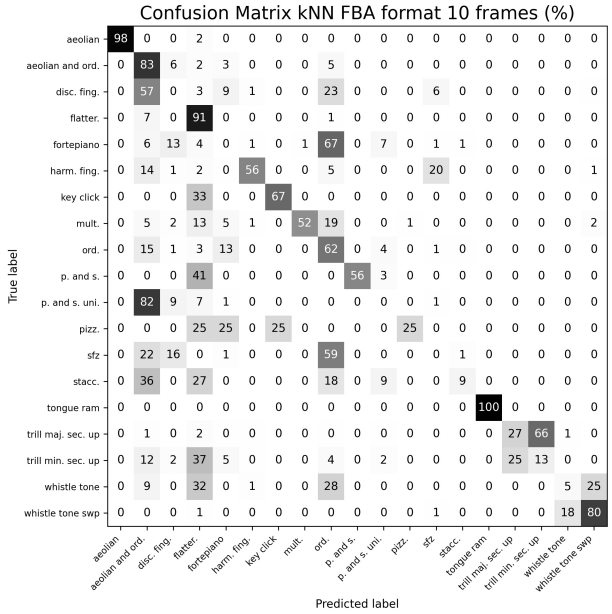

Figure 5. Confusion matrix, kNN, FBA data format, 10 frames

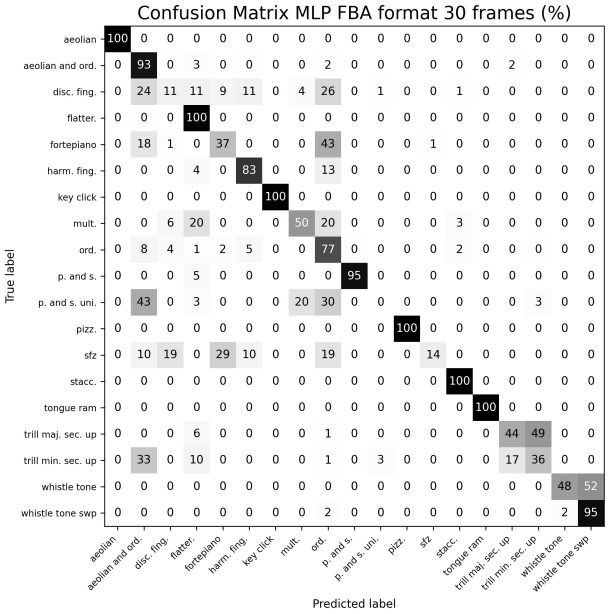

Figure 6. Confusion matrix, MLP, FBA data format, 30 frames

n_frames	60	45	30	15	10	5
SVM	47,01	44,93	42,45	41,22	35,10	31,65
kNN	91,91	89,97	108,8	104,47	104,47	105,86
MLP	23,88	23,18	22,57	22,33	22,33	22,45

Table 14. Total maximum delay time (ms), LMS data format

n_frames	60	45	30	15	10	5
SVM	21,94	21,97	22,19	22,74	25,45	25,22
kNN	24,31	24,49	25,37	24,69	33,26	37,80
MLP	22,31	22,33	22,35	22,02	21,84	22,09

Table 15. Total maximum delay time (ms), FBA data format

6.3.3 Total Maximum Delay Time

To represent worst-case scenarios, we calculated the total maximum delay time. As outlined in Section 5.3, the time to gather audio data is a fixed duration of approximately 21.3ms. This time is subsequently added to the algorithm's maximum prediction time. As mentioned in Section 6.3.1, the Log-Mel-Spectrogram analysis computation time is negligible and not included in the total maximum delay time calculation. A total maximum delay time near 21.3ms and below 42.6ms is necessary.

Table 14 and Table 15 show that MLP is much quicker to perform prediction than SVM and kNN on both LMS and FBA data formats (∼22ms). With the LMS data format, SVM is usable only with a number of frames below 30. kNN is usable with the FBA data format (37.8ms at most), but unusable with the LMS data format (89.97ms at least).

7. DISCUSSION

The best accuracy we found overall was 77.50% with the SVM algorithm and LMS data format with 15-frame-long samples and a total maximum delay time of 41.22ms. For the MLP algorithm, we found that the FBA data format yields a little higher accuracy scores compared to the LMS data format for the numbers of frames 60, 45, and 30. For SVM and kNN algorithms, tables show a drop in accuracy for the FBA data format. LMS is a higher-dimensional format than FBA. With the LMS data format, when the data dimensionality becomes too large, performance drops. On the other hand, since FBA data format is an average of frames, the data dimensionality is always small, but the accuracy tends to decrease if the number of frames is too small.

We want to design a system capable of making a prediction at every frame (1 frame ≈ 21.3ms) for real-time responsiveness. We found that when using the LMS data format, the total maximum delay time of SVM (31.65ms at least) is near the maximum delay time upper limit (42.3ms) and exceeds it when using the number of frames of 60 and 45 (47.01ms, 44.93ms). For kNN, when using the LMS data format, we found that the total maximum delay time (89.97ms at least) exceeded the maximum delay time upper limit for any number of frames. When using the FBA data format, we can see an improvement in the total maximum delay time for SVM (25.45ms at most) and kNN (37.80ms at most), making them usable but not as fast as MLP. In the case of MLP, the total maximum delay time on both data formats is small (∼22ms) and close to the 20ms threshold, which is the minimum duration required for perceiving the difference of timings [16]. That is why, MLP is the best choice considering both accuracy score and prediction time criteria.

In our experiments, we used three simple machine learning algorithms. For MLP, the prediction time does not change significantly between LMS and FBA data formats because it does not depend on the input dimension or the amount of training data (see Table 3). MLP has another advantage in that accuracy could be further increased by improving the neural network architecture, tuning hyperparameters, and using pre-trained models without impacting the prediction time very much.

Additionally, the confusion matrices revealed that the *staccato* and *key click* techniques were poorly identified. While other techniques, such as *sforzando* and *pizzicato*, demonstrated better identification, they still did not match the accuracies found with other techniques, such as *ordinario*. From this observation, we concluded that shorter-length techniques are less identifiable than others. As previously explained in Section 6.2.2, only a limited amount of data can be extracted from the short-length audio files. To enhance the classification of these techniques, we had better increase the data volume by using multiple sound banks and improving data augmentation methods.

We found that similar techniques, such as *trill major second up* and *trill minor second up*, and *whistle tone* and *whistle tone sweep* techniques, are misidentified. From this observation, we can guess that similar techniques are difficult to discriminate for the classification algorithms. To enhance the classification of these techniques, several similar techniques should be combined into one class in our future study.

The confusion matrices showed that the *discolored fingering* technique is not well recognized by the three algorithms with both LMS and FBA data formats. This may be caused by the similarity between the augmented data of other techniques and the original data of the *discolored fingering* technique (see Section 6.2.1 and Section 6.2.2). We consider that the *discolored fingering* technique should not be included in our future study.

8. CONCLUSIONS

In this article, we presented a preliminary analysis of real-time recognition of flute IPT for mixed music. Our study explored various algorithms and data formats to find the preferable data dimensionality to improve classification accuracy and measured the maximum delay time for real-time performance.

In our experiment, we found that reducing the number of frames around 15 improved accuracy and that SVM has the best accuracy with LMS data format. Concerning the prediction time, we made the following observations. The prediction time of MLP is smaller with the FBA data format than the LMS data format. When using the LMS data format, the total maximum delay time of kNN exceeds the maximum delay time upper limit, making kNN unusable with this data format. In the case of SVM, its total maximum prediction time is below but near the maximum delay time upper limit, making the SVM usable but not fully secured. When using the FBA data format, the total maximum delay time of kNN and SVM are below the maximum delay upper limit, making them usable but still slow compared to the MLP total maximum delay time. Therefore, based on both the accuracy score and delay time criteria, we conclude that MLP is the best choice as a comprehensive judgment for our task.

From the observation of the confusion matrices, we found that short-time techniques, such as *staccato*, *key click*, *sforzando*, and *pizzicato*, are less accurately identified when using the SVM and kNN algorithms. Even if MLP better identifies these techniques, accuracy still needs to be improved. The possible cause of this drop in accuracy could be due to training data scarcity, as the audio files for these techniques are short in length. Because of this, we will augment data quantity in our further study. Additionally, we found that techniques, such as *trill major second up* and *trill minor second up*, and *whistle tone* and *whistle tone sweep* techniques, are not well identified. The possible cause of this drop in accuracy could be the similarity of the techniques. To address this problem, we will combine similar techniques into the same class in our further study. We also found that the *discolored fingering* technique is poorly identified. The low accuracy scores may be caused by the similarity between the augmented data of other techniques and the original data of the *discolored fingering* technique. To handle this concern, we will not include this technique in our future study.

This study has some limitations. The train and test datasets originated from the same sound bank (homogeneous datasets). Consequently, the results in this experiment do not accurately represent real-time mixed music performance situations where the audio data from the circumstance of the live performance differs from that of training data. To deal with that, future research should evaluate the accuracy with different sound banks (heterogeneous datasets). Also, each data sample in our study only included delimited audio samples of one technique, where the first frame starts from the attack and the last ends at the release. This differs from real-time situations where the system analyzes audio on-the-fly where two techniques may be played successively. To address this concern, we should test the system in real time with a flutist. Given the time-critical nature of the real-time classification, Python may not be an optimal choice for ensuring time efficiency. Thus, we will consider using alternative high-performance languages such as C++ in our future research.

Acknowledgments

This research is related to the ERC Reach Project (GA #883313).

9. REFERENCES

[1] J.-M. Echeveste, "Un langage de programmation pour composer l'interaction musicale: la gestion du temps et des événements dans Antescofo," Ph.D. dissertation, Université Pierre et Marie Curie-Paris VI, 2015.

[2] 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.

[3] M. Malt and M. Gentilucci, "Real Time Vowel Tremolo Detection Using Low Level Audio Descriptors," *arXiv preprint arXiv:1511.07008*, 2015.

[4] 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.

[5] J.-F. Ducher and P. Esling, "APPRENTISSAGE PROFOND POUR LA RECONNAISSANCE EN TEMPS REEL DES MODES DE JEU INSTRUMENTAUX," in *Journées d'Informatique Musicale*, 2019.

[6] ——, "FOLDED CQT RCNN FOR REAL-TIME RECOGNITION OF INSTRUMENT PLAYING TECHNIQUES," in *International Society for Music Information Retrieval*, Delft, Netherlands, Nov. 2019. [Online]. Available: https://hal.science/hal-02472560

[7] 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.

[8] 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.

[9] 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.

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

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

[12] P.-Y. Artaud and G. Geay, *Flûtes au présent: traité des techniques contemporaines sur les flûtes traversières à l'usage des compositeurs et des flutistes*. Editions Jobert & Editions musicales transatlantiques, 1980.

[13] 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.

[14] 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, jan 2017. [Online]. Available: https://doi.org/10.1109%2Ftaslp.2016.2632307

[15] 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 *Proceedings of the 14th python in science conference*, vol. 8, 2015, pp. 18–25.

[16] I. J. Hirsh and C. E. Sherrick Jr, "Perceived order in different sense modalities." *Journal of experimental psychology*, vol. 62, no. 5, p. 423, 1961.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[18] 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.