Audio Engineering Society

Conference Paper 6

Presented at the AES International Conference on Machine Learning and Artificial Intelligence for Audio 2025 September 8–10, London, UK

This paper was peer-reviewed as a complete manuscript for presentation at this conference. This paper is available in the AES E-Library (http://www.aes.org/e-lib), all rights reserved. Reproduction of this paper, or any portion thereof, is not permitted without direct permission from the Journal of the Audio Engineering Society.

Flute Tone Quality Classification: a Machine-Learning-Based Evaluation Tool

Nikita Sane[1] and Jonathan Abel[2]

[1]*Henry M. Gunn High School*
[2]*CCRMA, Stanford University*

Correspondence should be addressed to Nikita Sane (nikitassane@gmail.com)

ABSTRACT

Music classification models have been used to accurately place audio recordings into different classes based on genre, instrument, and era, among other features. In this work, we introduce a machine learning algorithm that classifies flute recordings based on their tone quality. Flute tone quality is subjective, and its underlying parameters currently lack clear definitions in terms of objective audio signal processing features. Here we consider two quality descriptors, "focus" and "brightness," and present a largely "black box" algorithm to categorize recorded flute notes into quality descriptor classes. The model was built by first gathering and labeling a set of roughly 5000 high-quality flute note recordings derived from the Freesound database according to three levels each of focus and brightness. Then a pre-processing script that performs spectral analysis was developed. Finally, a convolutional neural network was trained, and its hyper-parameters, including the learning rate, batch size, and dropout rates, adjusted. An overall classification rate of 77% was achieved, with brightness and focus being correctly classified at rates of 83% and 86%, respectively.

1 Introduction

1.1 Background

Flute playing can be split into six major modalities: rhythm, articulation, dynamics, tone quality, intonation, and musicality [1]. Of these, the most subjective measures are musicality and tone quality (both go beyond the notes and symbols on the sheet music). In music auditions and competitions, judges make subjective, but still informed, decisions that may be subconsciously biased. To curb this bias and supplement decision-making, we propose creating an objective technology-based tool that rates a player's flute audition. We focused specifically on labeling tone quality as it is a playing modality that is highly nuanced.

We anticipate that our technology tool will be beneficial for not only audition judges but also young players, especially those without access to quality flute instruction. Learning what different tone qualities sound like is a crucial step in the early stages of learning the flute. With a tool that provides feedback on an audio recording, young players can develop their musical ear to hear when their tone is optimal.

Flutists typically change their tone quality by manipulating air speed, air direction, and the shape formed inside the mouth. Slower air speeds often result in

less bright tones. Larger mouth cavities result in fuller sounds. The specific manipulation of the embouchure (lip and mouth shape) depends on which register the flutist plays in.

Classification problems in the field of machine learning for audio analysis often use a convolutional neural network (CNN) fed with Mel Frequency Cepstral Coefficient data for each sample [2]. We used the basic model architecture of a music genre classifier [3] to begin classifying flute tone quality.

1.2 Related Work

Nykänen and Johansson [4] establish a connection between embouchure/mouth shapes and tone quality manipulation in saxophones, and this relationship is present for flute players too [5].

In the current field of music information retrieval, a more implicit approach is often used to analyze the spectral envelopes of different flute tone qualities. For instance, Clements and Yorita [6] explored the correlation between harmonic content and perceived flute tone quality, suggesting that specific spectral profiles might be indicative of certain tonal characteristics. They used descriptors such as "rich" or "dark" identified by expert flutists.

Advancements in machine learning have introduced more sophisticated approaches to this domain. Hernandez-Olivan and Beltran [7] developed a deep learning model employing a multi-head attention mechanism to classify timbres of orchestral instruments, achieving 62% classification accuracies across twenty instrument classes.

Similarly, recent studies by Agarwal and Greer [8] have used spectrogram-based deep learning and large language models to assess recorded flute performances. Their model is trained on labeled audition data and predicts category-specific scores (such as tone, intonation, and rhythm) that closely align with human adjudicators' assessments. CNNs specifically have been applied to optimizing flute playing and teaching systems [9]. By analyzing audio recordings, these models aim to strengthen students' musical literacy and performance skills.

Beyond the flute, Giraldo and Ramirez [10] have developed a system for users to train their own machine learning models that evaluate the tone quality of violin performances. Using a dataset of annotated violin recordings, they correlate different perceptual audio descriptors to build a real-time feedback system.

While the domains of instrument classification and performance assessment have seen significant progress, the subjective nature of tone quality, especially for instruments such as the flute, presents ongoing challenges. Our work builds upon these foundations. By using a labeled dataset and focusing on perceptual descriptors such as "brightness" and "focus," we aim to provide a more nuanced and objective evaluation tool for flutists and educators.

2 Methods

2.1 Dataset Labeling Process

We used the Freesound library [11] to train the proposed machine learning model. The library is a collaborative database of over 600,000 sounds uploaded by musicians, researchers, and audio engineers. The data we downloaded consisted of two packs labeled “Flute scales” and “Flute scales (played badly)” containing 76 and 281 audio files respectively. These packs were recorded by the Music Technology Group at the University of Pompeu Fabra in the context of their goodsounds.org project [12]. All of the audio files ranged from 10 to 30 seconds in length and were sampled at 48 kHz.

Note that we use “clips” and “recordings” interchangeably in this section. To label the data, ten flutists of high-intermediate to pre-professional level were selected. The 357 total audio clips were split into ten groups of thirty recordings each by simple random sampling. Then, the ten groups were further split into five pairs, and ten to twelve additional audio clips were given to both raters of each pair allowing for some overlap between assigned recordings. A total of 300 clips were be rated by one flutist, and fifty-seven clips were rated by two. Figure 1 shows a visual representation of this process.

We selected two descriptors of flute tone on which to rate each clip: “brightness” and “focus.” Brighter tones have dominant upper overtones and are rich in harmonics. Musicians often control the brightness of their tone to fit a certain part of the music. Focused tones are characterized by a stronger sinusoidal component of the sounds compared to the residual noise-like component. Musicians almost always strive for focused, fuller

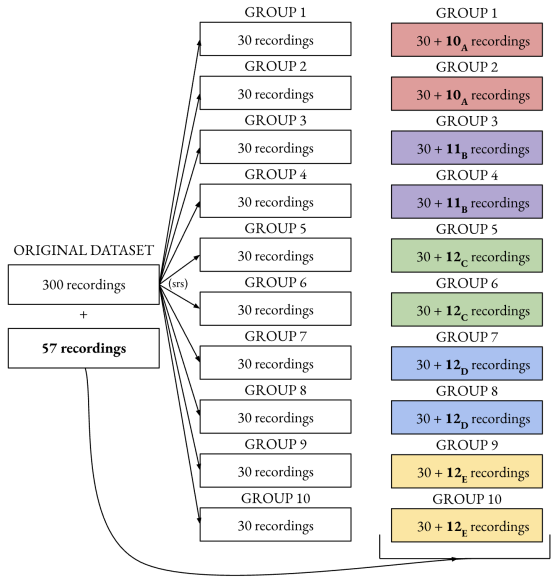


Fig. 1: Data-separation process used to assign audio clips to raters.

tones. Below are some words that are closely related to each of our selected descriptors that are sometimes used to describe flute tone quality in the music world.

- Bright: resonant, brilliant (antonyms: dark, dull)
- Focused: full, rich, clear, supported, centered (antonyms: hollow, airy, weak, thin)

Each participating flutist was assigned one group of recordings to rate. Each descriptor was rated on a scale of 1 to 3. For example, a clip rated "1_bright_3_focus" had a dark and centered tone. We selected these two descriptors because they are sufficiently independent of one another: a tone that is bright could be unfocused or full, and a tone that is full could be brilliant or dull.

Once each clip was rated, it was labeled accordingly. The clips that were rated by two flutists were used to ensure that the ratings were at least fairly consistent across raters. Table 1 shows the distribution of final labels after rating for the audio dataset. From the distribution of labels, we see it was common to rate brightness and focus as a "2" perhaps because the rating scale lacked granularity.

Table 1 also reveals that the dataset was imbalanced across different rating categories. We used class weighting to address this imbalance. Each label's respective weight was computed to be inversely proportional to its frequency in the dataset.

Table 1: Number of audio clips categorized into each label and their respective class weights.

Label	# of Audio Clips	Class Weight
1_bright_1_focus	51	0.778
1_bright_2_focus	53	0.747
1_bright_3_focus	23	1.718
2_bright_1_focus	37	1.071
2_bright_2_focus	74	0.536
2_bright_3_focus	45	0.881
3_bright_1_focus	14	2.825
3_bright_2_focus	31	1.280
3_bright_3_focus	29	1.364

2.2 Preprocessing

An essential stage of data collection involved cleaning the data. Analyzing the spectrograms of various audio clips, we identified that the power spectrum was highly unstable around the onset of notes and during changes in scale degrees. Thus, we constructed a preprocessing script in Python that split each audio clip at the note changes to create several sub-samples that only contained one pitch.

To write this script, we used Librosa [13] and NumPy [14]. Librosa is a Python library with functions to perform fast fourier transforms, onset detection, spectrogram analysis, etc. on audio files. NumPy is a Python library that supports computations with matrices and large arrays.

The first step of preprocessing was to identify the fundamental frequencies at several time indices along the recording to track when note changes occurred. So, we extracted frequencies and their corresponding magnitudes along the input file using `librosa.piptrack`, a pitch-tracking method. Within the resulting pitches array, we set a lower threshold at 240 Hz for fundamental frequencies, ignoring data outside the modern flute range (246 Hz–2093 Hz). Within the magnitudes array, we set a lower threshold

of 0.6, as the fundamental frequency is the strongest harmonic heard in the power spectrum. Out of the remaining frequencies in each time index, we set the lowest non-zero frequency present as the fundamental. The output matrices of `librosa.piptrack` were thus effectively flattened into a 1D array of fundamental frequencies with each index representing a time frame.

The next step of preprocessing was separating each recording into individual notes. This was achieved by noting the time indices where the fundamental frequency changed by greater than 20 Hz within the length of the audio. We called this new list "pitch_change_times."

Each range of audio data between adjacent time indices in our pitch_change_times list represented a single note within a scale being played in each audio recording. We call these sections of the recordings "samples." From the original dataset of 357 rated recordings, we extracted over 5000 labeled samples; each recording featured 16 notes of an ascending and descending one-octave scale.

Once the data was labeled and processed, we extracted discrete features from each audio sample. We used two approaches, one based on note MFCCs and partial amplitudes, and the other using note spectra. The following two sections describe the features computed as inputs for our machine learning model.

2.3 Method 1: Mel-Frequency Cepstral Coefficients (MFCCs) and Partial Amplitudes

Mel-Frequency Cepstral Coefficients (MFCCs) [15] are compact representations of the audio signal's spectral properties, obtained by applying a Discrete Cosine Transform (DCT) to the log-scaled Mel spectrogram, making them useful for speech and audio analysis. Equation (1) is used by the `librosa.feature.mfcc` function.

$$MFCC[m] = \sum_{k=1}^{K} S[k] \cdot \cos\left(m \cdot \left(k - \frac{1}{2}\right) \frac{\pi}{K}\right) \quad (1)$$

Here, $S[k]$ is the log-magnitude Mel spectrogram; K is the number of Mel filterbanks; m is the cepstral coefficient index. The DCT is applied to decorrelate the Mel spectrogram features.

We computed the first thirteen MFCCs for each sample using the `librosa.feature.mfcc` function and saved them along with their original rated labels.

Along with MFCC data, we selected values from the power spectrum for each audio clip, targeting each audio descriptor separately. We noted that a substantial portion of the sense of focus or brightness came from the steady-state middle portion of the note, away from the note onset and note release. Accordingly, in our classifier, we first isolated the middle 500 ms of each sample. We performed a Short-Time Fourier Transform (STFT) on the note segment using the `librosa.stft` function. We used a 2048-point-long Hann window [16], a 4096 bin DFT size, and a 512 sample hop length. Given that the sampling rate of the recordings was 48 kHz, we pulled 44 STFT frames for each sample.

We formed a periodogram spectral estimate for each note by averaging the STFT power over time,

$$X_{RMS}(k) = \sqrt{\frac{1}{44} \sum_{t=1}^{44} |X_k^{(t)}|^2} \quad (2)$$

In (2), $X_k^{(t)}$ is the magnitude of the DFT at frequency bin k and time frame t.

As an example, in Figure 2, two clips are shown, featuring a G4 to G5 major scale, articulated on every note, and with different focus ratings. As can be seen, the recording with lower rated focus has visible sub harmonics, especially in the higher note ranges, and has energy at odd integer multiples of half the fundamental frequency. To capture this observation, we computed the subharmonic frequencies $0.5f_0$, $1.5f_0$, $2.5f_0$, and $3.5f_0$ where f_0 is the fundamental frequency of a given sample as computed during preprocessing. Then, in the root mean square spectrum, we extracted the amplitudes of the nearest frequency bins to the subharmonics. The amplitudes were converted to decibels (dB) and normalized to the frequency of the fundamental before being added to the input data for each sample.

In Figure 3, two clips are shown featuring the same G4 to G5 major scale, articulated on every note, and with different brightness ratings. Note that the recording with higher rated brightness has a higher amplitude in the upper harmonic range for each note. To capture this observation, we computed the amplitudes of the frequency bins closest to the first seven harmonics in the root mean square spectrum of a given sample ($2f_0$, $3f_0$, etc.). These computed amplitudes were converted to dB, normalized to the fundamental frequency amplitude, and added to the input data for each sample.

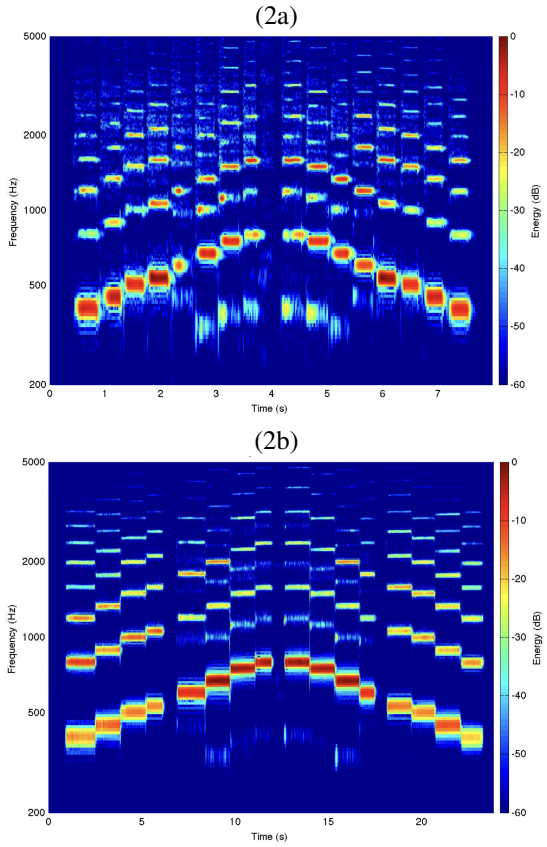


Fig. 2: Two spectrograms for different audio clips. (2a) was computed from a recording rated 2_bright_1_focus, and (2b) was computed from a recording rated 2_bright_3_focus.

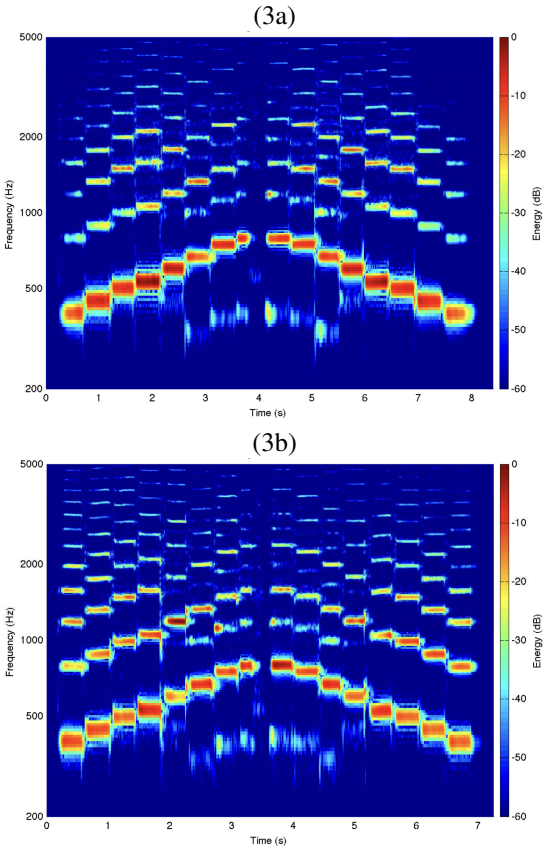


Fig. 3: Two spectrograms for different audio clips. (3a) was computed from a recording rated 1_bright_2_focus, and (3b) was computed from a recording rated 3_bright_2_focus.

The final input prepared for our model using this input data was a `.json` file containing a dictionary with three keys: "mapping," "mfcc + power spectrum data," and "labels." A visual representation of our dictionary is provided in Figure 4.

2.4 Method 2: Full Spectral Data

Method 1 of extracting data to input yielded an average classification rate of 71.9%. Another form of spectral data we tested inputting into the model was the entire root mean square spectrum for each sample instead of selected values. The entire spectrum would provide a more nuanced description of the full sound profile of each sample. For these spectra, we used frequency bins between 80 Hz and 8 kHz so as to capture the fundamental and subharmonic frequencies for all of the tones used, while avoiding any low-frequency noise, and to include high-frequency signal energy that was plainly visible in the note spectra. The spectra were normalized and clipped to range from -60 dB to 0 dB.

The final input prepared for our model using Method 2 was a `.json` file containing a dictionary with three keys: "mapping," "full power spectra," and "labels."

2.5 Basic Model Architecture

Before building our convolutional neural network (CNN), we tested a more simple k-means clustering [17] machine learning model with the Method 1 input data. It did not achieve notable clustering patterns.

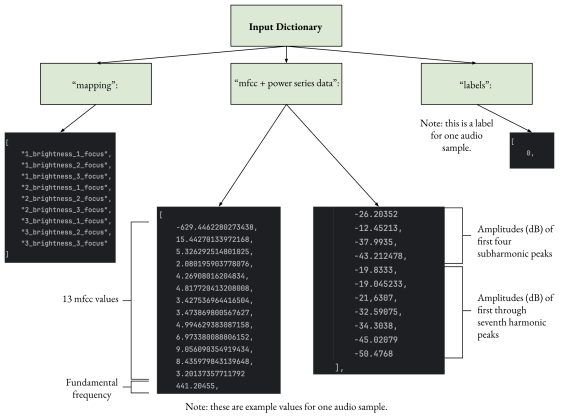
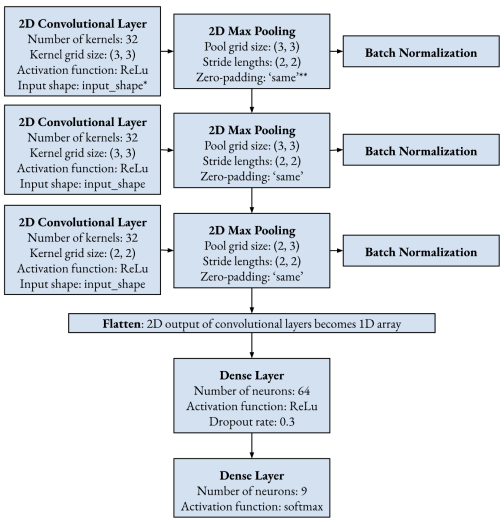


Fig. 4: Visual representation of the input dictionary using Method 1.



*input_shape is the shape of the training set of input data.
**'same' is a command in Keras that adds padding to match the input size to the kernel size.

Fig. 5: Visual representation of the basic model architecture.

We used TensorFlow [18]—a software library for machine learning functions—and Keras [19]– a higher-level API for neural networks– to develop our Python model.

A common machine learning model used for genre classification problems is a CNN with three convolution layers and two dense, fully-connected layers. Thus, we began with the same architecture for our model before fine-tuning to improve convergence. For the first four layers, we used the Rectified Linear Unit (ReLU) activation function [20] largely due to its efficient convergence rate in stochastic gradient descent. ReLU returns the input if it is greater than zero, and zero otherwise according to (3) below.

$$f(x) = \max(0, x) \tag{3}$$

We use the softmax activation function in our last layer—a dense layer of nine neurons. The softmax activation function converts a vector of raw scores into a probability distribution by exponentiating each value and normalizing it so that the sum of all outputs equals 1 according to (4) below. Each of the nine neurons in the final layer represented one tone rating label.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{4}$$

In the fourth layer, we used a dropout rate of 0.3 to avoid over-fitting in the training process. A visual representation of the basic model architecture is provided in Figure 5.

For the test-train-validation split, 55% of the input data was training data, 25% was testing data, and 20% was validation data by random splitting performed in TensorFlow.

3 Training Details

We will discuss the effects on train and test set evaluation accuracy from varying the following hyperparameters: optimizer learning rate, number of epochs, batch size, and dropout rate. We varied one hyperparameter at a time and started testing from the following settings: 0.001 learning rate, 30 epochs, batch size of 32, and 0.3 dropout rate. Input data Method 2 of providing full spectra for each sample yielded overall higher classification rates, so we tested hyperparameters using this method.

3.1 Learning Rate

Figure 6 in Appendix A shows that learning rates of 0.01 and 0.001 result in overfitting as well as unstable convergence. Thus, we chose a learning rate of 0.0001.

3.2 Number of Epochs

Figure 7 in Appendix A shows that 15 and 50 epochs result in unstable learning curves and overfitting respectively. We selected 30 epochs for our model.

3.3 Batch Size

Figure 8 in Appendix A shows that a batch size of 32 offers the least overfitting and a proper learning curve.

3.4 Dropout Rate

Figure 9 in Appendix A shows that a dropout rate of 0.3 improves both train and test set accuracy substantially while avoiding overfitting.

4 Results

4.1 Data Note Splitting Strategy Outcome

As described in the Methods section, we expanded our data from 357 audio clips to 5000 samples by rating the clips before splitting the recordings into single-note samples. This strategy helped reduce noisy data from the onsets and tapers of articulated notes as well as breathing in the recordings. It was an effective way to isolate the pure tone of each note within each recording.

4.2 Final Model Parameters, Accuracy Outcomes

Our final model used the keras.Adam [21] optimizer which follows a stochastic gradient descent approach for efficient convergence. The optimizer used a learning rate of 0.0001 and the sparse categorical cross entropy loss function [22] that varies according to (5). Our main motivation for using sparse categorical cross entropy over categorical cross entropy was that our target labels were integers rather than one-hot encoded vectors.

$$L = \frac{-1}{m}\sum_{i=1}^{m} y_i \cdot log(\hat{y}_i) \tag{5}$$

In (5), each example has a true label vector y_i and a predicted probability distribution vector $\hat{y}_i$. We used 30 epochs and a batch size of 32 when training the model. The dense layer of 64 neurons used a dropout rate of 0.3. Using the outlined hyperparameters and model architecture, we achieved an average accuracy rate of 0.77 on the test set over ten trials.

Table 2: Test set classification rates for individual and combined descriptors using data input method 1 (mfcc + selected power spectrum data).

Descriptor	Average Classification Rate
Brightness	0.823
Focus	0.787
Brightness and Focus	0.719

Table 3: Test set classification rates for individual and combined descriptors using data input method 2 (full power spectrum data).

Descriptor	Average Classification Rate
Brightness	0.834
Focus	0.865
Brightness and Focus	0.772

5 Discussion

Our study aimed to develop a machine-learning-based tool for classifying flute tone quality, focusing on the descriptors of brightness and focus. The results of our model indicate that flute tone quality classification is a feasible problem for machine learning but presents challenges due to the subjective nature of tone perception and the limited size of labeled data available.

The classification rates for brightness and focus individually as well as the overall classification rates for the test set are shown in Table 2 and Table 3 as an average over ten full training cycles. Note that brightness and focus were treated as binary classification tasks each. It is interesting that the errors were highly correlated between classifying focus and brightness for both data input methods. While the second input method was more computationally demanding, providing full spectral data for each sample yielded the highest average test accuracy of 0.77.

Our results suggest that while the model successfully captured some aspects of flute tone quality, there is room for improvement. One contributing factor to the lower accuracy could be the granularity of our labeling system. Our three-level scale for brightness and focus

may not have been fine-grained enough to capture the full range of tone variations perceived by expert flutists. Future iterations of this study could explore a finer rating scale or continuous values for labeling to improve precision.

Another key factor influencing model performance was the size and diversity of the dataset. While segmentation increased the number of training samples, it did not increase the variety of recorded environments, microphones, or playing styles represented in the dataset. This limitation may have led to overfitting on the specific acoustic characteristics of our dataset. A more diverse dataset, including recordings from a wider range of flutists and recording conditions, would likely improve generalizability.

In terms of data processing methods, exploring other, simpler audio features such as spectral flatness could improve the model's ability to capture focus and brightness. Additionally, inputting data from note onsets may provide the model with important information on the articulation style and its effects on tone quality.

Hyperparameter tuning had a significant impact on the final model's accuracy. Our tests indicated that an optimizer learning rate of 0.0001, batch size of 32, and dropout rate of 0.3 provided the best balance between convergence stability and generalization. However, further optimization using automated hyperparameter search methods, such as Bayesian optimization [23], could refine the model further. In conclusion, our machine-learning-based flute tone quality classifier demonstrates promise but requires further refinement.

Future work could focus on expanding the dataset, improving the granularity of labels, and incorporating additional spectral and temporal features. With continued development, this tool could serve as a valuable supplement for musicians seeking objective feedback on their tone quality and for audition judges aiming to reduce bias in their evaluations.

References

[1] Toff, N., *The Flute Book: a Complete Guide for Students and Performers*, Oxford University Press, 2012.

[2] Rawat, P., Bajaj, M., Vats, S., and Sharma, V., "A Comprehensive Study Based on MFCC and Spectrogram for Audio Classification," *Journal of Information and Optimization Sciences*, 44(6), pp. 1057–1074, 2023.

[3] Ghildiyal, A., Singh, K., and Sharma, S., "Music Genre Classification Using Machine Learning," in *2020 4th International Conference on Electronics, Communication and Aerospace technology (ICECA)*, pp. 1368–1372, IEEE, 2020.

[4] Nykänen, A. and Johansson, Ö., "Development of a language for specifying saxophone timbre," in *Stockholm Music Acoustics Conference: 06/08/2003-09/08/2003*, volume 2, pp. 647–650, Kungl. Tekniska Högskolan, 2003.

[5] Hakala, R., "Changing my sound: Developing the Tone Quality on the Flute with Embouchure Change," *Gothenburg University Library*, 2023.

[6] Yorita, R. and Clements, J., "Using spectral analysis to evaluate flute tone quality," in *Proceedings of Meetings on Acoustics*, volume 23, AIP Publishing, 2015.

[7] Hernandez-Olivan, C. and Beltran, J. R., "Timbre classification of musical instruments with a deep learning multi-head attention-based model," *arXiv preprint arXiv:2107.06231*, 2021.

[8] Agarwal, M. and Greer, R., "Spectrogram-Based Deep Learning for Flute Audition Assessment and Intelligent Feedback," in *2023 IEEE International Symposium on Multimedia (ISM)*, pp. 238–242, IEEE, 2023.

[9] Xu, C., "Analysis and Optimization of Flute Playing and Teaching System Based on Convolutional Neural Network," *Mathematical Problems in Engineering*, 2022(1), p. 1846863, 2022.

[10] Giraldo, S., Waddell, G., Nou, I., Ortega, A., Mayor, O., Perez, A., Williamon, A., and Ramirez, R., "Automatic assessment of tone quality in violin music performance," *Frontiers in Psychology*, 10, p. 334, 2019.

[11] Font, F., Roma, G., and Serra, X., "Freesound technical demo," in *Proceedings of the 21st ACM international conference on Multimedia*, pp. 411–412, 2013.

[12] Bandiera, G., Picas, O. R., Tokuda, H., Hariya, W., Oishi, K., and Serra, X., "Good-sounds. org: A Framework to Explore Goodness in Instrumental Sounds." in *ISMIR*, pp. 414–419, 2016.

[13] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., and Nieto, O., "librosa: Audio and music signal analysis in python." *SciPy*, 2015, pp. 18–24, 2015.

[14] Harris, C. R., Millman, K. J., Van Der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al., "Array programming with NumPy," *Nature*, 585(7825), pp. 357–362, 2020.

[15] Davis, S. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, 28(4), pp. 357–366, 1980.

[16] Testa, A., Gallo, D., and Langella, R., "On the processing of harmonics and interharmonics: Using Hanning window in standard framework," *IEEE Transactions on Power Delivery*, 19(1), pp. 28–34, 2004.

[17] Hartigan, J. A. and Wong, M. A., "Algorithm AS 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, 28(1), pp. 100–108, 1979.

[18] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al., "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pp. 265–283, 2016.

[19] Gulli, A. and Pal, S., *Deep learning with Keras*, Packt Publishing Ltd, 2017.

[20] Agarap, A. F., "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

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

[22] Jadon, S., "A survey of loss functions for semantic segmentation," in *2020 IEEE conference on computational intelligence in bioinformatics and computational biology (CIBCB)*, pp. 1–7, IEEE, 2020.

[23] Snoek, J., Larochelle, H., and Adams, R. P., "Practical bayesian optimization of machine learning algorithms," *Advances in neural information processing systems*, 25, 2012.

A Training Details

The following figures outline our hyperparameter testing stage. They track train and test set evaluation accuracy and error over each epoch of training as we manipulate the learning rate, number of epochs, dropout rate, and batch size. Note that we input data from preprocessing Method 2 for training.

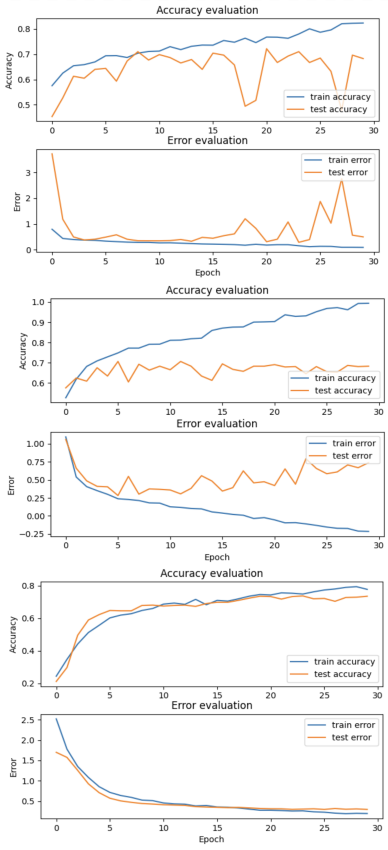


Fig. 6: From top to bottom, the learning rates used were 0.01, 0.001, 0.0001.

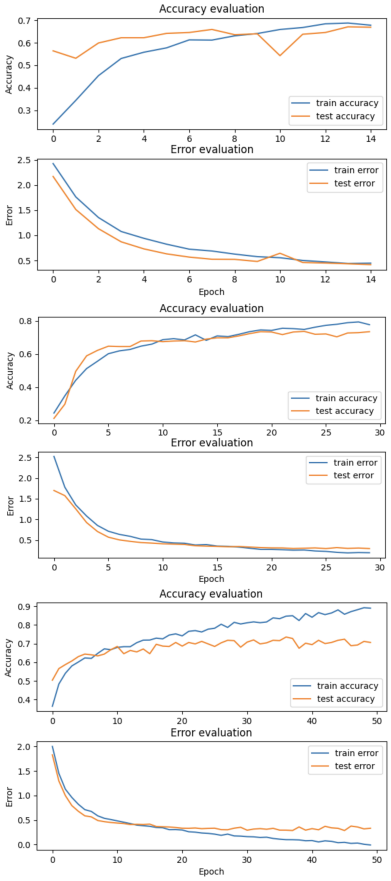
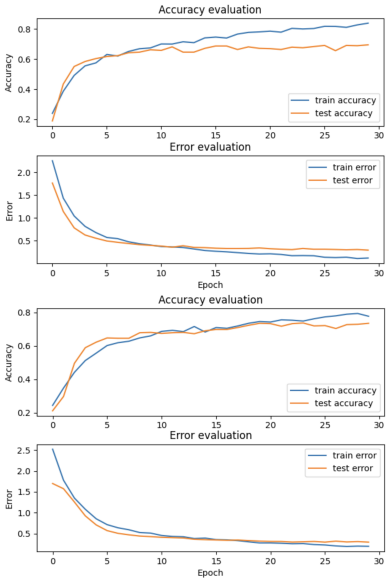


Fig. 7: From top to bottom, the number of epochs used were 15, 30, 50.



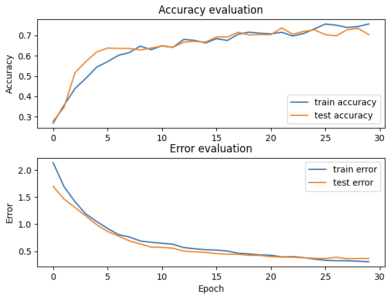


Fig. 8: From top to bottom, the batch sizes used were 16, 32, 64.

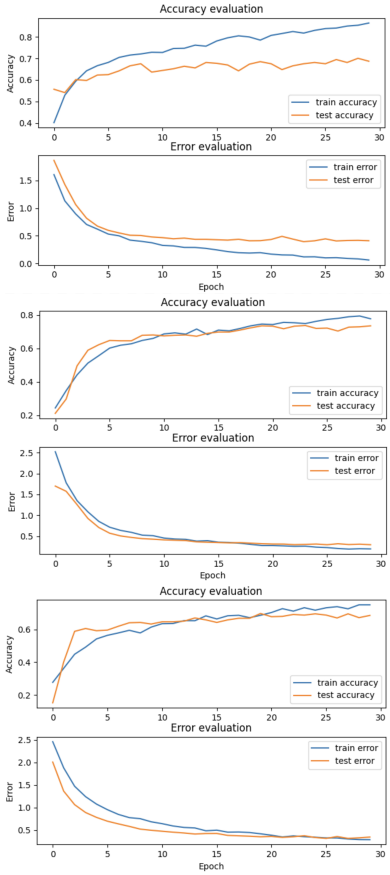


Fig. 9: From top to bottom, the dropout rates used were 0.2, 0.3, 0.5.