GROOVE - Generate Rhythm on our Virtual Engine.

Mentors: Kshitij Bhardwaj, Ashish Upadhyay, Soham Panchal.
Mentees: Abhinav Mishra, Aashvi Agarwal.

Keywords: Asynchronous Singing Assessment, Indian Classical Music, Pitch Contour Extraction, Frequency Normalization, Dynamic Time Warping (DTW), Note-wise Distance Matrix, Audio Feedback System, Virtual Music Pedagogy.

Abstract

This project presents a structured and scalable framework for asynchronous singing assessment, specifically tailored to Indian classical music, using a signal processing and machine learning pipeline. The system operates within a virtual teacher-student architecture, where students learn through pre-recorded sessions of Indian classical vocal exercises, and their responses are analyzed either in real-time or post-hoc. It incorporates pitch contour extraction, frequency normalization, and temporal alignment to detect deviations in both pitch and timing. By modeling tonal accuracy and using techniques such as dynamic time warping and note-wise distance matrices, the system accurately identifies mismatches between the teacher's and student's renditions. The pipeline enables objective, culturally relevant feedback, providing a practical tool for Indian classical music training in virtual settings.

1. Week 1–2

1.1. Fundamentals of Sampling and Fourier Analysis

Understanding musical pitch requires converting time-domain audio signals to the frequency domain. The **Sampling Theorem** states that a band-limited signal can be perfectly reconstructed if sampled above twice its highest frequency, known as the Nyquist rate:

$$f_s > 2f_{\text{max}}$$

The continuous-time Fourier Transform is defined as:

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j2\pi ft}\, dt$$

For digital audio, the Discrete Fourier Transform (DFT) is used:

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

This spectral representation enables pitch comparison between teacher and student performances.

1.2. Filtering Techniques in Audio Preprocessing

Band-pass filters isolate frequencies corresponding to specific musical notes (e.g., Sa, Re, Ga), enabling pitch-specific energy analysis. A filter bank targeting these note frequencies allows accurate pitch tracking.

Low-pass and **high-pass filters** are used in preprocessing to eliminate high-frequency noise and low-frequency interference (e.g., room hum), improving signal clarity.

1.3. Regression Models for Singing Assessment

Linear regression models the relationship between pitch accuracy and frequency deviation, helping identify trends in vocal performance:

$$\text{SS}_{\text{mean}} = \sum_{i=1}^{n}(y_i - \bar{y})^2 \qquad \text{Var}_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Logistic regression classifies notes as correct (0) or incorrect (1) using features like pitch deviation and duration. Its efficiency and interpretability make it ideal for real-time singing error detection.

1.4. Evaluation Metrics in Classification

1.4.1 Precision

Measures how many predicted positives are actually correct. Important in singing assessment to avoid giving incorrect feedback.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

1.4.2 Recall

Reflects how many actual positives were detected. Ensures most pitch or timing errors are caught.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

1.4.3 F1 Score and Other Metrics

The F1 score balances precision and recall:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Additional metrics:

- **AUC** evaluates class separability across thresholds.
- **Accuracy** gives overall correctness but can be misleading on imbalanced data.
- **Log Loss** penalizes confident wrong predictions—useful for probabilistic error detection.

2. Week 3–4

2.1. Convolutional Neural Networks (CNNs)

CNNs are widely used in tasks involving spatial or temporal patterns, such as image classification or spectrogram analysis in audio. They consist of layers that extract hierarchical features from input data like mel-spectrograms. The general pipeline includes convolutional filters, pooling, non-linear activation functions, and training using loss functions and optimizers.

2.1.1 Data Preparation

Input spectrograms are normalized and reshaped before passing through convolutional layers. These layers detect local patterns, while pooling layers (e.g., max-pooling) reduce spatial dimensions and retain dominant features. Nonlinear activation functions like the sigmoid introduce necessary complexity:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

2.1.2 Loss Function

In binary classification tasks—such as detecting correct (0) or incorrect (1) singing notes—the **binary cross-entropy** loss is typically used:

$$\mathcal{L} = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

2.1.3 Backpropagation and Optimization

The model is trained using optimizers like **Adam**, which adjust weights based on gradients calculated by backpropagation. Using the chain rule, gradients are propagated from the output to input layers:

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w}$$

These gradients guide parameter updates across training epochs.

2.2. Transformer-Based Deep Learning Models

Advanced AI systems like ChatGPT use **transformers**, which rely on self-attention mechanisms to capture long-range dependencies in data sequences. Each token in an input sequence is compared against others using:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V$$

Here, Q, K, and V are the query, key, and value matrices, and d_k is the key dimensionality. The softmax ensures attention weights sum to 1, enabling context-aware weighting. These attention layers, combined with feedforward networks and trained via backpropagation, allow transformers to generate coherent outputs token by token.

Softmax: Converts a vector of scores into a probability distribution:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

2.3. Audio Processing Foundations for ML

We studied how raw audio can be preprocessed into structured inputs for ML models. Using transformations like Fourier and mel-spectrograms, we extract time–frequency representations suitable for neural networks. Concepts such as windowing, noise reduction, and frequency-domain filtering were reinforced through both theoretical discussion and practical exploration using Python libraries (NumPy, SciPy).

Additionally, EEG signal processing videos provided insights into biosignal analysis, with overlapping techniques like filtering and spectral decomposition that apply directly to audio pipelines.

2.4. CNNs for Speech Recognition

We explored CNN-based speech classification via Keith Galli's tutorial using Keras/TensorFlow. The pipeline included:

- Loading and preprocessing audio into mel-spectrograms
- Building and training a CNN for speech classification

- Evaluating model performance using metrics and error analysis

This hands-on example reinforced our understanding of audio classification architectures and informed design choices for our own pitch and feedback classification system in GROOVE.

3. BCI-KER Challenge

3.1. Problem Description

As humans think, we produce brain waves. These brain waves can be mapped to actual intentions. In this competition, you are given the brain wave data of people with the goal of spelling a word by only paying attention to visual stimuli. The goal of the competition is to detect errors during the spelling task, given the subject's brain waves.

3.2. The Setup

The "P300-Speller" is a well-known brain-computer interface (BCI) paradigm which uses Electroencephalography (EEG) and the so-called P300 response evoked by rare and attended stimuli in order to select items displayed on a computer screen. In this experiment, each subject was presented with letters and numbers (36 possible items displayed on a matrix) to spell words. Each item of a word is selected one at a time, by flashing screen items in group and in random order. The selected item is the one for which the online algorithm could most likely recognize the typical target response.

The goal of this challenge is to determine when the selected item is not the correct one by analyzing the brain signals after the subject received feedback.

3.3. Approach

Since the problem is over a decade old, there have been multiple solutions proposed using various Machine Learning models, but none until now (at least to my knowledge) have used deep learning with a multi-layer perceptron (referred to as MLP from hereon).

In this approach, I have used an MLP to discriminate between the positive and negative feedback to allow for automatic error correction, as discussed in the original paper(1). Previous approaches used various other models or a combination of them (SVM, Logistic Regression, etc.); however, we believe that these models are inherently at a disadvantage because there are a few features (this is an intuition — the relevant features will be updated here once enough evidence is acquired) which they are unable to take into account, and which the DL model is able to benefit from without explicitly stating them, due to its architecture.

3.4. Feature Selection

The features for the model which the MLP takes as input are listed below -

1. 11200 EEG signals which are taken in the duration from 300ms after the feedback(the amount of time it takes for the brain to register the feedback) to 1300ms after the feedback(which is the total duration for which the feedback is present on the screen). With a sampling rate of 200 Hz and 56 EEG channels that amounts to total of 11200 EEG data points for each feedback event. 2. 200 EOG signals which correspond to the eye movement of the subject in response to the feedback. Sampling rate of 200Hz. 3. 1 Time Stamp corresponding to the time of the feedback event from the beginning.(Meta Feature) 4. 1 Session Number of the subject(Meta Feature) 5. 1 Word Number(Meta Feature) 6. 1 Character Number in the word(Meta Feature)

3.5. MLP Model Parameters

The libraries used in the overall model were Scikit-learn, Pandas, Numpy, os The MLP model was prepared using the sklearn neural network library with the parameters as follows-

1. hidden_layer_sizes = (11404,1000,1000,2)
2. activation = 'relu'
3. solver = 'adam'
4. alpha = 3e-5
5. batch_size = 200
6. learning_rate = 'adaptive'
7. learning_rate_init = 0.01
8. max_iter = 200
9. shuffle = True
10. random_state = 42
11. tol = 1e-4
12. verbose = True
13. warm_start = True
14. early_stopping = True
15. validation_fraction = 0.2
16. beta_1 = 0.9
17. beta_2 = 0.999
18. epsilon = 1e-8
19. n_iter_no_change = 10

4. GROOVE

4.1. Proposed Pipeline

Our pipeline for error detection in singing performance builds upon core audio signal processing principles while refining the feedback mechanism for greater granularity and relevance. The system primarily compares a student's vocal performance with a reference (teacher's) rendition by analyzing their melodic contours.

4.1.1 Segmentation and Note Identification

The first step involves segmenting both teacher and student audio clips into timeframes corresponding to individual notes, such as *Sa*, *Re*, *Ga*, *Ma*, etc. This segmentation allows us to perform localized analysis on note-level granularity, which is crucial for identifying specific musical errors. Each note window is then passed forward in the pipeline for comparative processing.

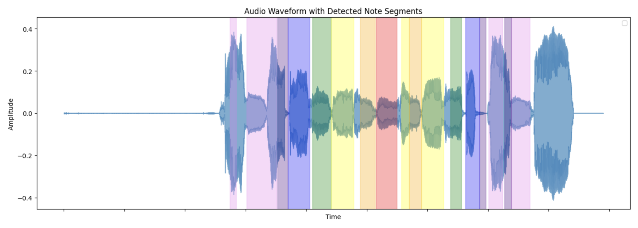

Figure 1. Timewise Segmentation of notes

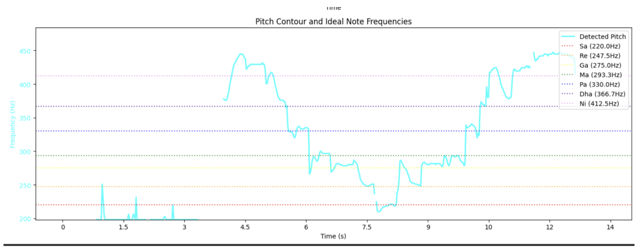

Figure 2. Pitch detection

4.1.2 Pitch Normalization

Once segments are extracted, pitch normalization is applied. This step ensures that both teacher and student pitch contours are compared on a common scale, accounting for differences in vocal range and recording conditions. Normalization is crucial for enabling fair and perceptually meaningful comparisons.

Several pitch normalization methods were explored:

- **Z-Score Normalization:** Normalizes the pitch contour by subtracting the mean and dividing by the standard deviation. This method standardizes data across speakers but may discard perceptual information.
- **Semitone Conversion:** Converts pitch values from Hz to semitone scale using the formula:

$$n = 12 \cdot \log_2 \left(\frac{f}{f_0} \right)$$

  where f_0 is a reference frequency. This method was preferred because it aligns closely with how humans perceive pitch intervals.
- **Mean-Centered Hz:** Subtracts the mean pitch value from each pitch point in the contour. Useful when absolute pitch is less relevant but tonal variance needs preservation.
- **Proportion of Range (POR):** Normalizes pitch based on the speaker's pitch range. It maps each frequency as a proportion between their minimum and maximum observed values.

Among these, semitone conversion was chosen for its perceptual alignment and ability to preserve musical structure across different vocalists.

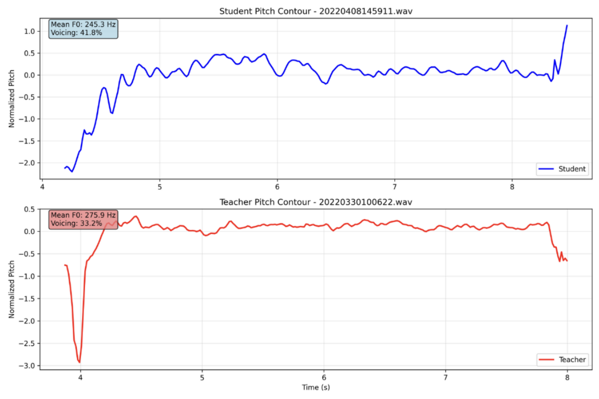

Figure 3. Normalized Pitch Contours of Teacher and Student

4.1.3 Pitch Contour Matching and Visualization

With normalized pitch data, we extract pitch contours for both the teacher and student. Matching these contours visually and numerically helps identify deviations across time. Previously, we used a cost matrix computed via logarithmic pitch distances, followed by Dynamic Time Warping (DTW) to align the sequences. This method remains useful and is still part of our experimental pipeline.

It is worth noting that the resulting DTW path may not always begin at the origin (0,0). This occurs when the student and teacher do not start singing at the exact same time, or when silence/padding is present at the beginning of either recording. DTW accommodates such temporal offsets by initiating the alignment from the most suitable matching frame, rather than forcing strict synchronization from the start.

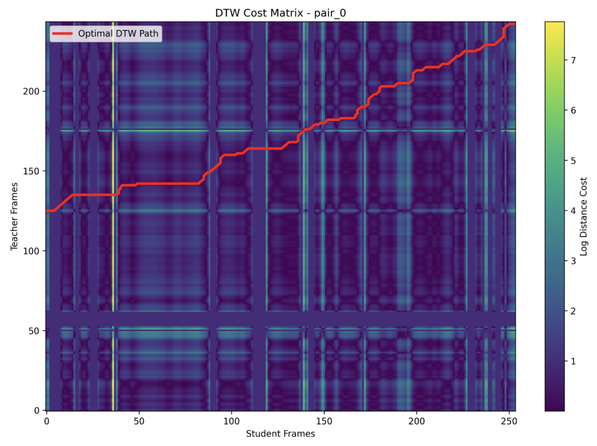

Figure 4. DTW path through cost matrix based on semitone difference

4.1.4 Note-wise Mistake Detection

Building upon the DTW framework, we now additionally focus on **note-specific mistake identification**. Instead of summarizing pitch error across the entire phrase, we visualize pitch mismatch over time to identify the exact point of deviation. This method is particularly useful for pinpointing which specific note (e.g., *Re* or *Ma*) was sung off-pitch.

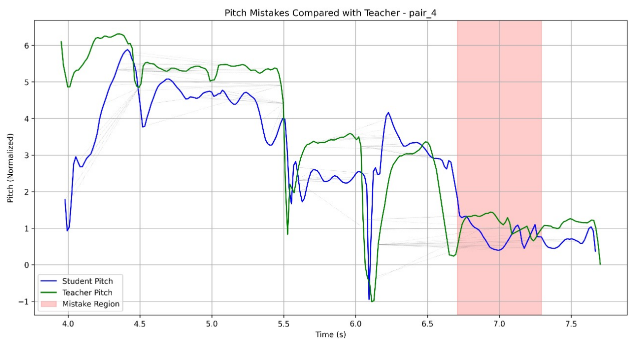

Figure 5. Pitch Mistakes

4.1.5 Error Aggregation and Feedback

Finally, we compute both average and maximum pitch deviation across segments. These metrics are aggregated and plotted to offer both high-level performance summaries and detailed, note-specific feedback. The visualizations enable learners to identify not just whether they went off-pitch, but *when* and *by how much*, making the feedback actionable and intuitive.

5. References

1. https://onlinelibrary.wiley.com/doi/10.1155/2012/578295
2. https://github.com/alexandrebarachant/bci-challenge-ner-2015/blob/master/README.md
3. https://www.techrxiv.org/users/681419/articles/682248-automatic-detection-and-analysis-of-singing-mistakes-for-music-pedagogy