GROOVE - Generate Rhythm on our Virtual Engine

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

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

1.1. Sampling Theorem

The Sampling Theorem is a fundamental concept in signal processing, stating that a continuous signal can be fully reconstructed from its discrete samples if it is band-limited and sampled at a rate greater than twice its highest frequency component. This minimum rate is known as the Nyquist rate. The theorem ensures accurate digitization of analog signals without loss of information, which is critical in applications like audio processing and communications. Violating the Nyquist criterion leads to aliasing, where different signals become indistinguishable.

Let $x(t)$ be a continuous-time signal that is band-limited, i.e., its Fourier transform $X(f) = 0$ for all $|f| > f_{\text{max}}$. Then $x(t)$ can be perfectly reconstructed from its samples $x(nT)$, taken at uniform intervals T, if the sampling frequency $f_s = \frac{1}{T}$ satisfies:

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

1.2. Fourier Transform

Since pitch and musical notes are fundamentally related to frequency, transforming the time-domain audio signal into the frequency domain is essential for meaningful comparison. The continuous-time Fourier Transform of a signal x(t)x(t) is defined as:

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

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

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

Using the DFT, both teacher and student audio can be converted into comparable spectral representations. This is crucial for detecting frequency mismatches, quantifying pitch accuracy, and assessing note-wise alignment. Without the Fourier domain representation, subtle pitch errors—especially in unaligned or variable-length singing—would be hard to detect using raw time-domain features.

1.3. Filtering

Here, **band-pass filters** will be primarily used to isolate specific frequency bands corresponding to individual Indian musical notes. Since notes like Sa, Re, Ga, etc., have well-defined fundamental frequencies (based on the chosen tonic), narrow band-pass filters allow for precise detection by capturing only the energy around each note's range while excluding irrelevant frequencies. A filter bank of these band-pass filters, each tuned to a specific note, enables systematic pitch tracking and classification.

Additionally, a **low-pass filter** will be used at the preprocessing stage to remove high-frequency noise, ensuring smoother pitch contours. A **high-pass filter** may also be applied to eliminate low-frequency interference like microphone rumble or room hum, further improving signal clarity before note analysis.

2. Week 2

2.1. Linear Regression

Linear regression models the relationship between a student's pitch accuracy and a single influencing factor, such as frequency deviation from the teacher's note. It helps predict pitch error and identify trends in singing performance, enabling targeted feedback and correction in Indian classical vocal training through data-driven analysis.

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

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

2.2. Logistic Regression

In the GROOVE project, logistic regression is used to classify audio inputs into binary labels: 0 for no error and 1 for error. It analyzes features such as pitch deviation, note duration, and frequency stability to determine note accuracy. Trained on labeled singing data, the model learns decision boundaries that distinguish correct from incorrect renditions. Its simplicity, interpretability, and low computational cost make it especially suitable for real-time or asynchronous evaluation settings. Despite being a linear model, logistic regression performs effectively when paired with well-engineered features, offering a fast and reliable method for error detection in Indian classical singing assessment.

2.3. Evaluation Metrics in Machine Learning

2.3.1 Precision

It is important when the cost of false positives is high. It measures the proportion of positive predictions that are actually correct. In tasks like medical diagnosis or error detection in singing (as in GROOVE), high precision ensures the system doesn't wrongly label correct actions as mistakes, avoiding unnecessary feedback.

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

2.3.2 Recall

It's critical when the cost of false negatives is high. It evaluates how many actual positive cases were correctly identified. For example, in singing assessment, high recall ensures that the system catches most pitch or timing errors, preventing missed feedback that could hinder learning progress.

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

2.3.3 F1 Score

It balances both precision and recall, making it ideal for imbalanced datasets where neither metric alone provides a full picture. In GROOVE, where detecting subtle singing errors is more important than raw count accuracy, the F1 score ensures the model is both sensitive and precise in identifying mistakes.

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

Area Under Curve (AUC), Accuracy, and Logarithmic Loss (Log Loss) are key metrics for evaluating classification models. AUC measures a model's ability to distinguish between classes across all thresholds, with a higher AUC indicating better separability between correct and incorrect notes—especially useful for confidence-based decision-making or when class distributions are uneven. Accuracy, while simple and intuitive, reflects the overall proportion of correct predictions but can be misleading in imbalanced datasets where it may overlook consistent errors. Log Loss complements these by quantifying the uncertainty in predictions, heavily penalizing confident but incorrect outputs. It is especially valuable when probabilistic confidence in note classification is required, beyond just binary labels.

3. Week 3

3.1. CNNs

3.1.1 Data Preparation

Input data (such as images or spectrograms) is preprocessed—normalized and reshaped—before being passed through convolutional layers that extract local patterns. Pooling layers (like max-pooling) reduce dimensionality and retain important features. Non-linear activation functions such as the sigmoid are applied to introduce non-linearity:

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

3.1.2 Loss Function

After the final output layer, a loss function quantifies the difference between predicted and actual values. For binary classification tasks, like labelling error as 1, and correct as 0, binary cross-entropy is commonly used:

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

3.1.3 Optimizer

Optimizers like Adam adjust the model's weights based on gradients to minimize the loss function over time, improving predictions with each epoch.

3.1.4 Backpropagation

It efficiently computes how much each parameter (weight and bias) in the network contributed to the final prediction error. This is done by applying the **chain rule** of calculus across each layer of the network. Gradients of the loss function are propagated from the output layer back to the input, enabling precise weight updates. For any parameter www, the update is derived from:

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

These gradients are then used to update parameters.

3.2. Deep Learning on AI Models

Deep learning in AI models like ChatGPT relies on transformer architectures that process input data using layers of attention and feedforward networks. At its core, ChatGPT uses self-attention to weigh the relevance of each word in a sequence relative to others, capturing context and dependencies. The attention mechanism is computed as:

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

where Q, K, and V are the query, key, and value matrices, and dk is the dimension of the keys. Stacked layers of such computations, trained on vast text datasets via backpropagation and optimization (e.g., Adam), enable the model to generate coherent, context-aware responses. The final output is generated by selecting tokens with the highest predicted probability from a learned distribution over vocabulary.

Softmax: converts a vector of scores into probabilities by exponentiating each value and normalizing them, ensuring all outputs sum to 1 for classification tasks.

4. Week 4

4.1. Study of Audio Signal Processing for ML

In this week, we studied a structured series of lectures focused on audio signal processing techniques used in modern machine learning workflows. The goal was to understand how raw audio signals—such as speech or music—can be transformed into representations suitable for automated analysis and modeling. We explored both classical signal processing methods and deep learning-based approaches.

4.2. Speech Recognition using CNN's

To enhance our understanding of model architectures specific to speech tasks, we watched Keith Galli's tutorial on **speech recognition with CNNs in Keras/TensorFlow**. This video demonstrates how convolutional neural networks, typically used in image tasks, can be applied effectively to raw audio or spectrogram input. It walks through a hands-on pipeline: loading audio, generating mel-spectrogram features, building a CNN model, and training it for speech classification. The tutorial also covers important implementation practices, such as data normalization, model evaluation metrics, and error analysis. Overall, it provided practical insights that informed our design decisions for the acoustic feature extractor and classification layers used in our system.

4.3. EEG Signal Processing with Python (Introductory survey)

We also reviewed the "EEG Signal Processing" video series from PyCon Canada, which, although focused on neural (brainwave) signals rather than audio, offered valuable transferable techniques. Key takeaways include methods for preprocessing biosignals in Python—such as filtering, windowing, and Fourier-based spectral analysis—using libraries like NumPy and SciPy. These signal-processing foundations helped reinforce our comprehension of time-frequency transformations, noise suppression, and pipeline structuring, principles that directly apply to our audio front-end module.

5. BCI-KER Challenge

5.1. 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.

5.2. 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.

5.3. Approach

Since the problem is over a decade old there have been multiple solutions proposed to the solution using various Machine Learning models but none until now(atleast to my knowledge) have used deep learning using multi layer perceptron(reffered to as MLP from hereon). In this approach I have used a MLP to discriminate between the positive and negative feedback to allow for automatic error correction as discussed in the original paper$^{(1)}.Previousapproachesusedvariousothermodelsoracombination$

5.4. 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)

5.5. 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

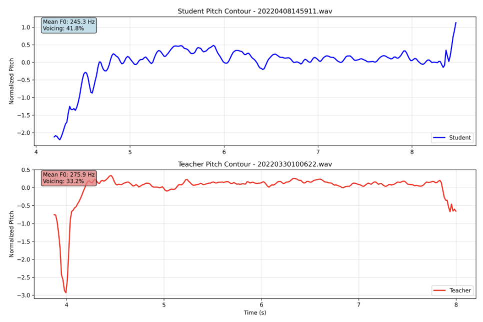

Figure 1. Normalized Pitch Contours

6. GROOVE

6.1. Proposed Pipeline

6.1.1 Normalization of raw audio pairs

Preprocessing begins with noise reduction using bandpass filters to isolate the vocal frequency range. Normalizing both student and teacher audio ensures consistency in amplitude and scale, reducing the impact of recording conditions and enabling fair comparison. For normalization, we used **semitone conversion** as it reflects human pitch perception :

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

6.1.2 Match the pitch contours

Pitch contour extraction captures the frequency progression of singing over time. Aligning the contours of student and teacher audio reveals how closely the student followed the intended melody. This step is vital for note-wise comparison and tonal analysis. Refer Figure 3.

6.1.3 Cost Matrix

We calculate a frame-by-frame cost matrix using a logarithmic frequency distance to mimic human pitch perception. This matrix highlights deviations across time and frequency and forms the basis for time-alignment analysis. Refer Figure 4.

6.1.4 Best Dynamic Time Warping path

In the context of singing analysis, the teacher and student may render the same musical phrase with different timing. DTW calculates the optimal path through the cost matrix (constructed using pitch distance), minimizing the total alignment cost while allowing non-linear warping in time. This makes it especially useful for asynchronous learning scenarios, where strict time alignment is not guaranteed.

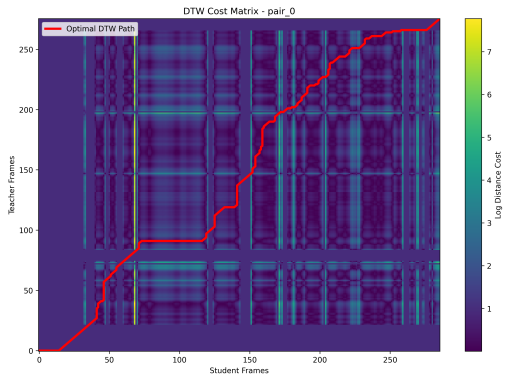

Figure 2. DTW path through cost matrix

The DTW path ensures that each note in the student's contour is compared to the most appropriate reference point in the teacher's contour, leading to more accurate error detection and feedback.

6.1.5 Aggregate for Cost

Aggregating average and maximum mismatch values summarizes the overall pitch accuracy and detects the most off-pitch segments. The resulting plots provide intuitive visual feedback for learning and assessment.

7. Refrences

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