Classification of Monophonic Guitar Notes Across Strings in Noisy Environments

Haruniya Rajan
220256861
Miss Dalia Senvaityte
Big Data Science
Queen Mary University of London

***Abstract*—Music note classification is pivotal in many music information retrieval tasks, including transcription, analysis, and re-synthesis. This paper introduces a novel approach to string-wise musical note classification, utilizing the GuitarSet (Bittner & Bello 2018) data, a rich collection of annotated guitar sound recordings. By employing Convolutional Neural Networks (CNNs), a specialized form of deep learning techniques, the methodology extracts distinctive features such as spectral contrast, chroma, Mel-frequency cepstral coefficients, spectral centroids, spectral rolloff, spectral flatness, zero-crossing rate, and tonnetz. These features are processed, and the pitch data from the GuitarSet (Bittner & Bello 2018) annotations are utilized to classify musical notes. Additionally, the model incorporates Gaussian noise through data augmentation, an experiment aimed at enhancing the model's ability to generalize and reduce overfitting. The model is optimized through hyperparameter tuning, achieving excellent performance in string classification with an accuracy of 80%. However, the note classification presents challenges, reaching only 30% accuracy, indicating the complexity of this task and the need for further refinement. The evaluation metrics include precision, recall, and AUC-ROC, accounting for the imbalance in the dataset. The study's findings underscore the potential of CNNs in music note classification and highlight the disparities in accuracy between string and note classification. The insights gained from this research pave the way for more nuanced models and contribute to various applications within the field of music information retrieval, leveraging the specific characteristics of the GuitarSet data (Bittner & Bello 2018).**

***Index Terms*—Convolutional Neural Networks(CNNs); Audio Signal Processing; Imbalanced Class Distribution; Noise Handling; Deep Learning.**

I. Introduction

Music, as a universal language, transcends the boundaries of culture, geography, and time. It is a form of art and expression that manifests in various styles and genres, each having distinct characteristics. Among the myriad of musical instruments, the guitar stands out for its significant contribution to the creation of music. With its versatility, the guitar can produce a wide array of sounds, making it a popular choice across many musical genres. Understanding the notes played on a guitar can offer invaluable insights into the melody, harmony, and rhythm of a song. However, this task is complex due to the intricate nature of guitar sounds and the multifaceted structure of musical compositions.

In recent years, the rapid development of digital technology and artificial intelligence has opened new avenues for music information retrieval (MIR), a field dedicated to extracting, analyzing, and classifying musical information. Musical note classification, in particular, is pivotal in music transcription, analysis, and synthesis. This paper introduces a novel method for classifying music notes from guitar sounds, utilizing Convolutional Neural Networks (CNNs) and a comprehensive set of audio features, extracted from the GuitarSet data (Bittner & Bello 2018).

The primary aim of this project is to develop an innovative approach for classifying musical notes from guitar sounds, leveraging the capabilities of CNNs and specific audio features. The objectives include extracting relevant features from the GuitarSet data, constructing a CNN model tailored to the unique characteristics of guitar sounds, optimizing the model through hyperparameter tuning, and evaluating its performance using various metrics. The project also seeks to explore the trade-offs between complexity and accuracy, offering insights into the challenges and opportunities in musical note classification.

CNNs have been successfully applied in various domains, including image recognition, natural language processing, and audio signal processing. The architecture of CNNs, inspired by the human visual system, enables the model to capture local dependencies in the data, making it highly effective for handling audio data. The use of multiple convolutional layers allows the model to learn complex patterns, thereby enhancing accuracy.

Unlike traditional methods that rely on plain spectrograms (magnitude/power of short-time Fourier transform), this study extracts eight specific audio features: spectral contrast, chroma short-time Fourier transform, Mel-frequency cepstral coefficients, spectral centroids, spectral rolloff, spectral flatness, zero-crossing rate, and tonnetz. The benefits of using these features include a more nuanced understanding of the audio signal, capturing specific characteristics that contribute to the identification of musical notes. This approach can lead to higher precision in classification. However, the downside is the increased complexity in feature extraction and the potential risk of overfitting if not properly managed. The choice of these features over plain spectrograms reflects a trade-off between complexity and potential accuracy gains.

The remainder of this paper is organized as follows: Section II reviews related work on music note classification. Section III details the methodology, including audio feature extraction,

model construction, and performance evaluation. Section IV presents the experimental results and discusses the findings. Finally, Section V concludes the paper and outlines future research directions.

II. Related Work

Music notes classification has emerged as a vital field of study within the scope of music information retrieval (MIR). The progress in this field has been gradual, evolving from traditional machine learning methodologies to the application of more sophisticated deep learning techniques. In this section, we discuss significant research contributions that have shaped the field of music note classification and how they inspire our current work.

A. GuitarSet Dataset Overview

The present study leverages the comprehensive GuitarSet dataset (Bittner & Bello 2018), an innovative resource that provides high-quality guitar recordings with rich annotations and metadata. Created using a hexaphonic pickup, the dataset offers recordings of individual strings, significantly automating the laborious annotation process. The dataset encompasses 360 excerpts of approximately 30 seconds each, capturing a wide spectrum of musical styles, progressions, and tempi. These excerpts are played by six unique guitarists, allowing for distinct output signals from each string, thereby simplifying the data annotation process (Bittner & Bello 2018). Accompanying every excerpt, the dataset delivers three types of audio recordings and a .jams file that stores 16 specific annotations. These annotations consist of pitch contour and MIDI note annotations for each string, beat position, and tempo annotations. It also includes two unique chord annotations: 'instructed' and 'performed' (Bittner & Bello 2018). The breadth and depth of the GuitarSet dataset make it an apt resource for our study on music note classification. By utilizing this dataset, our research aspires to delve into its wealth of information, building on the significant research contributions that have come before, and make meaningful strides in the domain of music information retrieval.

B. Traditional Machine Learning Methods and Evolution to Deep Learning

1) Early Approaches: In the initial phase of music note classification, methods like k-Nearest Neighbors (k-NN), Decision Trees, and Support Vector Machines (SVMs) were employed (Schölkopf & Smola 2002). A noteworthy study by Poliner and Ellis (2007) applied SVMs for piano note classification and produced encouraging results (Poliner & Ellis 2007). SVMs are binary classifiers defined by a separating hyperplane. For SVMs, the decision function takes the form:

$$f(x) = \text{sign}(w^T x + b)$$

where:

- sign is the sign function, which returns the sign of the input value
- x is the input vector
- w is the normal vector to the hyperplane
- b is the bias term

While these approaches were successful to some extent, they struggled to account for the temporal dependencies of audio data, which is crucial when handling music.

2) Electric Guitar Note Classification: Furthering this line of work, Bello and Pickens (2005) attempted the classification of electric guitar notes using a multi-class SVM approach (Bello & Pickens 2005). The research pointed out that the sound of an electric guitar can be influenced by a range of factors including distortion and other effects. The interactions of these effects with the guitar's natural timbre made the classification problem more complex.

3) Guitar Chord Classification: Humphrey, Bello, and LeCun (2012) used a Convolutional Neural Network (CNN) based approach for automatic chord recognition from audio signals (Humphrey et al. 2012). Their results demonstrated a considerable improvement over previous methods, highlighting the potential of deep learning methods in music information retrieval tasks.

C. Advanced Deep Learning Techniques: CNNs and RNNs

The field of music note classification has seen significant advancements with the introduction of specialized deep learning techniques, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

1) Convolutional Neural Networks (CNNs): Lee et al. (2009) (Lee et al. 2009) were among the pioneers in applying deep learning for music analysis, using a Convolutional Deep Belief Network (CDBN) for genre classification. Unlike traditional CNNs, CDBNs combine the principles of deep belief networks with convolutional layers, allowing for more complex and abstract feature learning. CNNs, on the other hand, have excelled in image classification tasks due to their ability to learn hierarchical patterns directly from the data. When it comes to audio signals, their spectrograms can be treated as 2D images, hence justifying the use of CNNs. The architecture of a typical CNN involves multiple convolutional layers, each followed by a pooling layer, and finally fully connected layers.

In recent years, Choi et al. (2016) (Choi et al. 2016) proposed a CNN architecture for automatic music tagging. Their model was able to infer semantic information from the given audio clips, marking the significance of CNNs in music analysis.

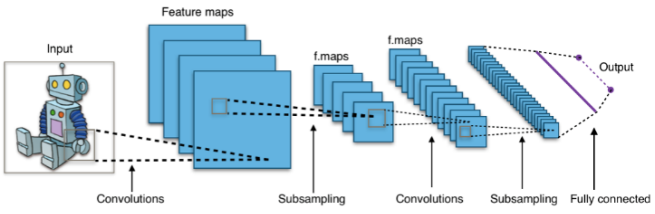

Fig. 1. Typical CNN Architecture

The Fig. 1 illustrates the typical architecture of a Convolutional Neural Network (CNN) (Lee et al. 2009). It consists of multiple convolutional layers, each followed by a subsampling (pooling) layer, and finally fully connected layers. Subsampling, also known as pooling, is a process used in CNNs to reduce the spatial dimensions of the feature maps. By summarizing the information in a local region of the feature map, subsampling helps to make the detection of features invariant to scale and orientation changes. It also reduces the computational complexity of the network.

In recent years, Choi, K., Fazekas, (2016) (Choi et al. 2016) proposed a CNN architecture for automatic music tagging. Their model was able to infer semantic information from the given audio clips, marking the significance of CNNs in music analysis. This approach was further validated by another study by Dieleman and Schrauwen (2016), who explored similar techniques for music tagging and classification (Dieleman & Schrauwen 2014).

2) Recurrent Neural Networks (RNNs): RNNs, designed to handle sequential data, have proven to be effective for music note classification. Thickstun et al. (2017) (Thickstun et al. 2017) trained an RNN on a large dataset of piano recordings, which provided results that outperformed previous traditional machine learning methods. A typical RNN has a memory mechanism that captures the temporal dependencies of sequential data. However, standard RNNs tend to struggle with long-term dependencies due to the vanishing gradient problem. To overcome this, variants of RNNs such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are often used.

D. Music Note Classification for Guitar Sounds

1) Uniqueness of Guitar Sounds: The complexity of guitar sounds arises from the primary sound-producing mechanism of a guitar, which is the vibration of the strings. Various techniques such as plucking, strumming, or fingerpicking can initiate these vibrations, each producing different timbres (Elliott et al. 2013). While these specific techniques may not be represented in the GuitarSet dataset (Bittner & Bello 2018) used in this study, understanding the diversity of guitar sounds is essential for appreciating the challenges of music note classification for guitar sounds.

Moreover, guitarists often use different playing techniques, such as bending, sliding, hammer-ons, and pull-offs, which can affect the nature of the sound. Use of guitar effects like distortion, reverb, or delay can significantly alter the characteristics of the sound produced (Järveläinen et al. 1998).

Another interesting characteristic of guitar music is the use of the same note on different strings. A note played on one string at a higher fret can be the same pitch as a note played on another string at a lower fret. These identical pitches, however, may not sound exactly the same due to differences in string thickness and tension (Howard & Angus 2009).

The Fig. 6 detailed in Appendix, illustrates a guitar fretboard highlighting the various positions for each note, demonstrating how the same fundamental frequency can be played on different strings (Guitar n.d.). This visual representation emphasizes the complexity and versatility of the guitar, where the same pitch can be produced in multiple ways. The ability to play the same note on different strings adds to the richness of the guitar's sound and provides musicians with a wide array of expressive possibilities.

2) Previous works on guitar note classification: There are relatively fewer studies targeting the specific task of music note classification for guitar sounds. Notable among them, Brown and Zhang (2009) developed a system for the automatic recognition of guitar performances, including pitch, onset time, and duration (Brown & Zhang 2009). They used a traditional machine learning method, specifically an SVM, for this task. Despite showing promising results, the system had limitations in dealing with chords and intricate playing techniques such as hammer-ons and pull-offs.

Kehling et al. (2014) proposed an automatic transcription system for electric guitar recordings (Kehling 2014). They used pitch salience functions and Hidden Markov Models (HMM) to recognize pitches, onset times, and other relevant information. Despite the innovative approach, the system faced challenges in identifying notes in guitar solos or other complex performances, such as rapid fingerpicking sequences or heavily distorted power chords.

In a more recent study, Wiggins and Kim (2019) focused on guitar tablature estimation using a Convolutional Neural Network (CNN) (Wiggins & Kim 2019). Their work demonstrates the application of CNNs in understanding guitar music. This approach aligns with the trend towards leveraging deep learning techniques to capture the nuances of musical instruments, particularly the guitar.

Both studies shed light on the complexity of the task and the need for methods that can capture the intricate characteristics of guitar music.

In our research, we aim to advance this line of work by using modern deep learning techniques, particularly focusing on CNNs, and a specially curated dataset of guitar sounds. Our goal is to develop a system that can identify individual notes and classify musical notes across strings. This focus on note classification enables a more targeted approach, allowing for the extraction and analysis of specific characteristics that contribute to the understanding of guitar performances.

In conclusion, the literature on music note classification reveals a progression from traditional machine learning to advanced deep learning techniques, with specific challenges in the realm of guitar sounds. Existing research has shown promise but also highlights gaps in handling the complexity and uniqueness of guitar performances. Our study aims to build on these insights, utilizing the GuitarSet dataset and modern methodologies to contribute to the field. The synthesis of previous works with current techniques aspires to enhance music information retrieval and deepen our understanding of musical expression.

III. Methodology and implementation

A. Audio Signal Visualization and Analysis

The raw audio files are extracted from the comprehensive GuitarSet dataset (Bittner & Bello 2018), which provides high-quality guitar recordings with rich annotations and metadata. These files contain the audio signals that will be subjected to both time-domain and frequency-domain analyses. The initial analysis aims to understand the fundamental characteristics of the audio signals, such as amplitude variations and frequency content.

1) Audio Signal Waveform Analysis: A waveform is a visual representation of an audio signal over time. The height of the waveform at any point in time corresponds to the amplitude of the audio signal at that moment. Larger amplitudes (i.e., taller peaks in the waveform) correspond to louder sounds, while smaller amplitudes (i.e., shorter peaks or valleys) correspond to quieter sounds. The x-axis of the waveform represents time. As you move from left to right along the x-axis, you're moving forward in time through the audio clip. This can show you when certain sounds or events occur within the clip.

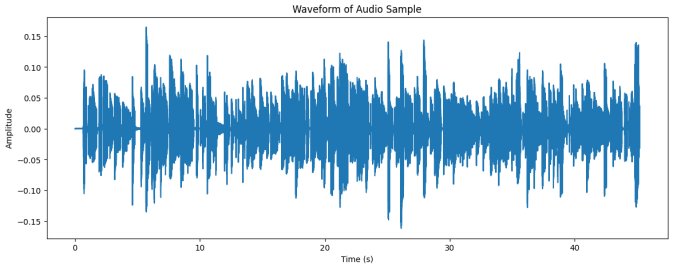

Fig. 2. Waveform of an audio file

The Fig 2 illustrates the waveform of *01_Rock2-85-F_solo_hex_cln.wav*. It's important to note that while waveforms can provide a useful high-level overview of an audio signal, they don't provide any information about the frequency content of the signal. For that reason, we need to use a different kind of visualization, such as a spectrogram.

2) Visual Analysis using Spectogram: A spectrogram is a visual representation of the spectrum of frequencies of sound or other signal as they vary with time which is used to visualize the notes being played over time. It's a 2D plot, with the x-axis representing time, the y-axis representing frequency, and the color or intensity representing the amplitude of the frequency at that time. Each unique sound or voice has a distinctive spectral fingerprint that can be visualized on a spectrogram. Unwanted noise or interference, such as white noise or background noise, often shows up on the spectrogram as random spikes or patterns that don't fit with the rest of the signal.

The Fig 3 illustrates the spectogram visualization of *01_Rock2-85-F_solo_hex_cln.wav* , the brighter colors indicate higher amplitude. The vertical lines indicate sudden increase in amplitude due to either loud sounds or typically where the note is being played. Since we are using

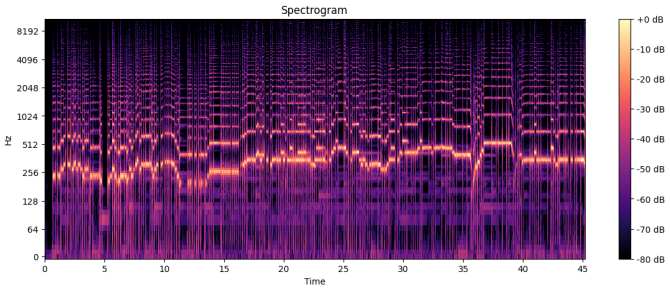

Fig. 3. Waveform of an audio file

the hexaphonic clean version from the GuitarSet (Bittner & Bello 2018), where interference from other strings has been removed, the probability of noise being present is very low, thanks to quiet recording conditions.

B. Audio Data Pre-processing and Feature Engineering

The process begins with the retrieval of audio data from the GuitarSet dataset. The dataset is meticulously processed file-by-file, with each audio file being loaded and analyzed. Alongside the audio data, corresponding annotations are loaded, providing essential information about the musical content.

1) Audio Data Processing: The primary step in the audio data processing is the translation of MIDI note numbers into more comprehensible musical terms. This is achieved through the *midi_to_pitch* function, which converts MIDI note numbers into their corresponding pitch names, translating the numerical representation into a human-readable format of note names and octaves.

Upon obtaining the annotation data, each MIDI note number, along with its start time and end time, is examined. Corresponding audio segments for each MIDI note number are loaded, and each of the six audio channels (representing the six strings of the guitar) is processed individually. If a channel's signal is too short, it's padded to attain the length required for the Fast Fourier Transform (FFT). Following this, a suite of audio features, including spectral features extracted from the spectrogram, is computed using *librosa's* robust feature extraction functions. These features encompass various aspects of the audio signal, capturing both the spectral and temporal characteristics, and form the foundation for further analysis and modeling.

2) Feature Engineering: The feature engineering process begins with pitch contour analysis, providing insights into the pitch's central tendency and variability across the strings. Segmentation of the audio data is performed according to MIDI note annotations, and segments are padded. Various spectral features are extracted, including spectral contrast (a measure of difference in amplitude between peaks and valleys in the sound spectrum), chroma STFT (related to the twelve different pitch classes), MFCCs (Mel-frequency cepstral coefficients, representing the short-term power spectrum), spectral centroids (indicating where the center of mass of the spectrum is located), and others. The Harmonic-to-Noise Ratio

(HNR) is calculated to separate the harmonic and percussive components.

The comprehensive process described here encompasses various aspects of the audio signal, capturing both the spectral and temporal characteristics. These features form the foundation for further analysis and modeling, providing a rich and multifaceted representation of the musical content. By meticulously engineering these features, the study aims to create a robust and informative dataset that can effectively represent the complexity and diversity of guitar sounds.

C. Data Pre-processing and Analysis

This stage involves a series of systematic steps aimed at transforming raw data into a format that can be effectively utilized by predictive models. The process includes feature selection, encoding categorical variables, scaling features, and dealing with class imbalance, among other tasks. Additionally, the analysis part of this stage involves understanding the underlying patterns and characteristics of the data, which includes visualizations, statistical tests, and exploratory analyses.

1) Feature selection: The features are carefully chosen to exclude any non-numerical features and to include those that show the highest correlation with the target variables as illustrated in Fig. 5 in Appendix, namely string and note, with a correlation value over 0.5, and the labels for string and note are separately defined. This selection process ensures that the model is trained on the most pertinent information, enhancing its predictive accuracy and efficiency.

2) Label Encoding: To translate the categorical labels into a format that can be understood by machine learning algorithms, Label Encoding is applied. This transformation converts the string and note labels into numerical values, maintaining the relationship between different classes.

3) Data Splitting: The dataset is partitioned into training and testing subsets, with 80% allocated for training and 20% for testing. This split ensures that the model's performance can be evaluated on unseen data, providing a more realistic assessment of its generalization capability.

4) Outlier Treatment: Outliers can distort the training process, leading to suboptimal performance. The Z-score method is employed to identify and remove outliers from the training data. A threshold of 3 is used to filter out extreme values, resulting in a more robust training set.

5) Standardization: Standardization is performed using the StandardScaler, which normalizes the features to have a mean of 0 and a standard deviation of 1. This scaling process enhances the convergence speed of the model and ensures that all features contribute equally to the learning process.

The pairplot diagram as illustrated in Appendix Fig. 9 reveals both univariate and multivariate distributions of features. The varying scales and distributions of these features can lead to challenges in modeling, as algorithms may give higher importance to features with larger scales. This highlights the need for standardization to ensure that all features contribute equally to the model's performance, allowing for a more accurate and robust analysis.

6) Balancing: Class imbalance visualized in Fig. 7 in Appendix, is addressed using Random Over Sampling (ROS). This technique oversamples the minority class, creating a balanced distribution of classes in the training data. This balance ensures that the model does not become biased towards the majority class and can learn the characteristics of all classes equally. The class distribution after applying ROS (Random Over-Sampling) is shown in Fig. 8 in Appendix.

7) One-Hot Encoding: The labels are further transformed into one-hot encoded format, where each class label is represented as a binary vector. This encoding is essential for multi-class classification tasks, allowing the model to distinguish between multiple classes.

8) Data Augmentation: Data augmentation is implemented by introducing random gaussian noise to the training and testing sets. This artificial expansion of the dataset enhances the model's ability to generalize, reducing the risk of overfitting to the training data.

These preprocessing measures lay the foundation for building a precise and robust model for music note classification, reflecting the complexity and multifaceted nature of the musical content.

Finally, the features are reshaped to fit the input requirements of a Convolutional Neural Network (CNN). This reshaping ensures compatibility with the convolutional layers, enabling the network to learn spatial hierarchies in the data.

D. Building and Tuning the Convolutional Neural Network Model

The process of building and tuning a Convolutional Neural Network (CNN) model for the task of musical note and string classification is a multifaceted endeavor. It involves the careful design of the model's architecture, the selection of appropriate hyperparameters, and the application of techniques to prevent overfitting and create a robust and efficient model tailored to the unique characteristics of the audio data.

1) Model Architecture: The CNN model is constructed through a function named *build_model*, designed to create a flexible and adaptable architecture using *TensorFlow* (Abadi et al. 2016) and *Keras* (Chollet 2015), two popular open-source libraries for deep learning.

The *convolutional layer* serves as the foundation of the model, responsible for detecting local patterns and features within the audio signal. It utilizes filters ranging from 32 to 128, with kernel sizes of either 3 or 5, implemented using *tf.keras.layers.Conv1D* (Abadi et al. 2016). The ReLU activation function is applied to introduce non-linearity, and L1 and L2 regularizers are used to prevent overfitting. If the output of the convolutional layer exceeds a certain dimension, a max pooling layer is introduced, implemented using *tf.keras.layers.MaxPooling1D* (Chollet 2015). This layer reduces the spatial dimensions of the feature maps, preserving the most salient information and reducing computational complexity.

To further mitigate overfitting, a *dropout layer* is added with varying rates from 0.2 to 0.5, implemented using

tf.keras.layers.Dropout (Chollet 2015). This layer randomly sets a fraction of the input units to 0 during each training epoch, promoting a more robust learning process. The *flatten layer* is essential for transitioning from convolutional layers to dense layers, implemented using *tf.keras.layers.Flatten* (Chollet 2015). It reshapes the 2D matrix data into a vector, preparing it for the fully connected layers. Two *dense layers* follow, with units ranging from 32 to 128, implemented using *tf.keras.layers.Dense* (Chollet 2015). These layers interpret the features extracted by the preceding layers, applying ReLU activation and L1 and L2 regularizers for optimal performance. The model concludes with two separate *output layers* for string and note classification, both using softmax activation.

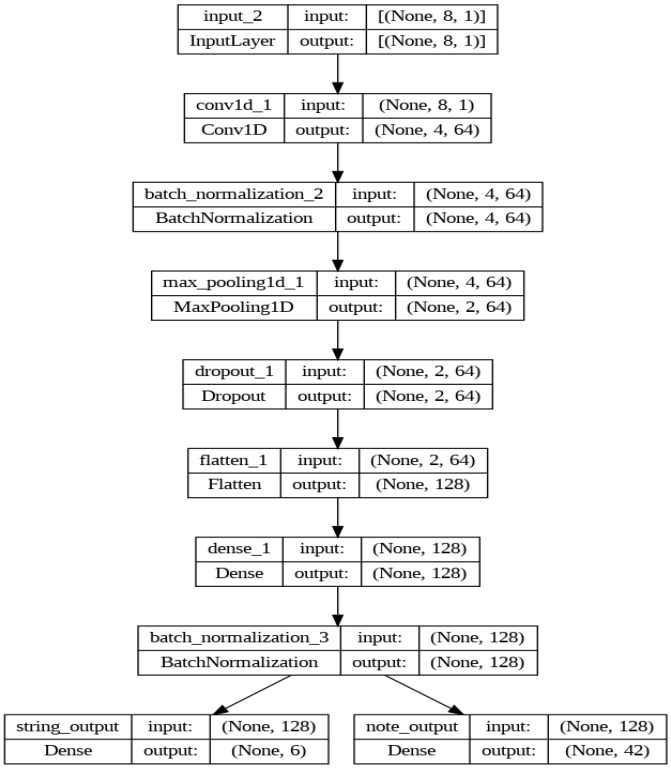

Fig. 4. Model Architecture

The architecture of the model, as illustrated in Fig 4 , is thoughtfully designed to capture the intricate relationships within the audio data, translating them into meaningful musical terms. The flexibility in hyperparameters allows for fine-tuning, ensuring that the model is well-suited to the specific challenges of the task.

2) Hyperparameter Tuning: The model's performance is highly dependent on the selection of appropriate hyperparameters. Utilizing Keras Tuner's *RandomSearch* function (O'Malley et al. 2019), a systematic search is conducted over various combinations of hyperparameters, including the number of filters, kernel size, dropout rate, dense layer units, and learning rate. The optimal values identified through this process are 128 units in the first densely-connected layer and a learning rate of 0.001. This hyperparameter tuning ensures that the model is finely calibrated to achieve the best possible performance.

3) Callbacks and Cross-Validation: Several callbacks are implemented to enhance the training process. *EarlyStopping*, *ModelCheckpoint*, and *LearningRateScheduler* are utilized, all part of the *tf.keras.callbacks* module (Abadi et al. 2016). *EarlyStopping* monitors validation loss and halts training if no improvement is observed after 3 epochs, preventing overfitting. *ModelCheckpoint* saves the best-performing model based on validation loss, preserving the optimal state of the model. *LearningRateScheduler* adjusts the learning rate during training, maintaining the current learning rate for the first 10 epochs and then exponentially decreasing it by a factor of 0.1 for subsequent epochs, facilitating better convergence.

Alongside these callbacks, a 5-fold cross-validation strategy is employed to ensure robustness against data variability. For each fold, the model is trained for 50 epochs, validated, and the hyperparameters are tuned using Keras Tuner's *RandomSearch* (O'Malley et al. 2019) with a maximum of 5 trials. This rigorous validation process, combined with the thoughtful use of callbacks, ensures that the model's performance is consistent across different subsets of the data, providing a solid foundation for the subsequent stages of training and evaluation.

4) Model Training: With the optimal hyperparameters identified, including 128 units in the first densely-connected layer and a learning rate of 0.001, the model is constructed and trained for 10 epochs.The training process is closely monitored, with performance metrics including loss, precision, recall, F1 score, and AUC evaluated for both training and validation sets. The various callbacks, such as *EarlyStopping*, *ModelCheckpoint*, and *LearningRateScheduler*, contribute to efficient and effective training, ensuring that the model is well-suited to the specific challenges of the musical note and string classification task (Abadi et al. 2016).

5) Model Evaluation: The evaluation of the model is a critical step in understanding its performance and identifying areas for potential improvement. Several metrics are used to provide a comprehensive assessment. *Loss* measures the difference between the model's predictions and the true labels, reflecting the overall accuracy. *Precision* quantifies the model's ability to correctly identify only the relevant instances, while *recall* measures the ability to identify all relevant instances. The *F1 score*, the harmonic mean of precision and recall, provides a balanced measure, especially valuable in the context of imbalanced class distribution. The *AUC*, or area under the receiver operating characteristic curve, indicates the model's ability to distinguish between classes, and categorical accuracy measures the accuracy of classification for the categorical labels.

The model's performance metrics are printed for every 2nd epoch, offering insights into how the model evolves throughout training. These metrics are later plotted to visualize changes over time, aiding in the detection of overfitting or underfitting. By examining these metrics, a comprehensive understanding of the model's strengths and weaknesses is obtained, guiding

future iterations and refinements. This methodical approach culminates in a model that is well-suited to the complex and nuanced task at hand.

IV. Results and Discussion

This section presents a comprehensive analysis of the model's performance over the course of the training process, focusing on key metrics such as loss, precision, recall, accuracy, F1 score, and AUC, and the impact of noise. This section is structured to first provide a detailed overview of the training process, followed by an in-depth analysis of the model's performance across different epochs. A summary of the results is presented in Table I, II, III and IV highlighting the model's performance metrics over the final training epoch and the effect of noise on the results.

TABLE I
Training Results for String

Epoch	Loss	Accuracy	Precision	Recall	F1 Score	AUC
10	0.46	0.81	0.82	0.79	0.8	0.975

Table I reveals a steady improvement in all metrics for string classification, with a notable balance between precision and recall, indicating effective learning and classification. The inclusion of Gaussian noise demonstrate the model's robustness to noise, maintaining performance levels.

TABLE II
Validation Results for String

Epoch	Loss	Accuracy	Precision	Recall	F1 Score	AUC
10	0.5	0.79	0.8	0.77	0.78	0.97

Table II demonstrates consistent growth in validation metrics for string classification, reflecting the model's ability to generalize well, with AUC showing enhanced ability to distinguish between classes.

TABLE III
Training Results for Note

Epoch	Loss	Accuracy	Precision	Recall	F1 Score	AUC
10	2.45	0.32	0.76	0.061	0.114	0.95

Table III shows progress in training for note classification, but the low recall and F1 score highlight challenges in identifying positive note instances, suggesting the need for model refinement. The addition of Gaussian noise in data augmentation emphasized the complexity of this task.

TABLE IV
Validation Results for Note

Epoch	Loss	Accuracy	Precision	Recall	F1 Score	AUC
10	2.105	0.3	0.47	0.022	0.043	0.95

Table IV presents validation results for note classification, mirroring training progress, but the low recall and F1 score emphasize the need for further tuning to enhance the model's sensitivity to positive instances.

The trends are further visualized in Appendix, refer to Fig. 10 and 11, illustrating the evolution of each metric for string and note classification over the epochs.

Further, on the basis of the classification report detailed in Table V , Table VI and Fig. 12 in Appendix, we can infer that the string classification model demonstrates a promising performance with an overall accuracy of approximately 78.63%. The precision, recall, and F1-score metrics are consistent across different classes, ranging from 70% to 89%. This consistency indicates a balanced classification ability, where the model is effectively recognizing different strings without significant bias. However, there is still room for improvement, particularly in the fourth and fifth classes, where precision and recall are slightly lower. Enhancing the model's ability to differentiate these specific strings could lead to further improvements in overall performance.

TABLE V
Classification Report for String

String	Precision	Recall	F1 Score
1	0.836649	0.866594	0.851359
2	0.839246	0.834160	0.836695
3	0.814827	0.830143	0.822414
4	0.706092	0.789894	0.745646
5	0.753384	0.701181	0.726346
6	0.897317	0.724963	0.801984
Macro Average	0.807919	0.791156	0.797407

Here, macro average is the arithmetic mean of the individual class related to precision, recall and F1 Score.

TABLE VI
String Accuracy results

Accuracy
0.786252

The note classification model as illustrated in Table VII, VIII and Fig. 13 in Appendix exhibits a more varied performance across different classes, with an overall accuracy of around 29.81%. While some notes, such as F2 and F3, are classified with relatively high precision and recall (around 37% and 49%, respectively), others, such as G♯5 and F♯5, have zero precision and recall. This inconsistency suggests that the model may struggle to differentiate between specific notes, possibly due to similarities in their audio characteristics. The wide range of F1-scores, from 0% to 42%, further emphasizes the model's uneven performance across different notes. Future work could focus on addressing these inconsistencies, possibly through targeted data augmentation, feature engineering, or model architecture adjustments, to enhance the model's ability to accurately classify a broader range of musical notes.

V. Conclusion

In this study, a comprehensive approach to monophonic guitar note and string classification was developed, employing Convolutional Neural Networks (CNNs) and rigorous hyperparameter tuning. While the model demonstrated promising

TABLE VII
CLASSIFICATION REPORT FOR NOTE

Note	Precision	Recall	F1 Score
A2	0.361502	0.339207	0.350000
A3	0.301887	0.339823	0.319734
A4	0.311404	0.250883	0.277886
A♯2	0.236111	0.089947	0.130268
A♯3	0.282443	0.301303	0.291568
A♯4	0.213166	0.303571	0.250460
B2	0.280443	0.319328	0.298625
B3	0.277691	0.310646	0.293245
B4	0.211679	0.162011	0.183544
C3	0.299003	0.338346	0.317460
C4	0.307087	0.254486	0.278323
C5	0.265625	0.220779	0.241135
C♯3	0.359375	0.273810	0.310811
C♯4	0.274603	0.265337	0.269891
C♯5	0.270833	0.240741	0.254902
D3	0.319481	0.355491	0.336525
D4	0.34318	0.326053	0.334400
D5	0.225490	0.425926	0.294872
D♯2	0.052632	0.400000	0.093023
D♯3	0.266010	0.325301	0.292683
D♯4	0.258621	0.231959	0.244565
D♯5	0.080645	0.232558	0.119760
E2	0.311828	0.245763	0.274882
E3	0.389356	0.338200	0.361979
E4	0.347222	0.310835	0.328022
E5	0.117647	0.200000	0.148148
F2	0.374046	0.494949	0.426087
F3	0.383740	0.476768	0.425225
F4	0.353414	0.317690	0.334601
F5	0.052980	0.470588	0.095238
F♯2	0.209524	0.211538	0.210526
F♯3	0.201635	0.191214	0.196286
F♯4	0.251366	0.335766	0.287500
F♯5	0.000000	0.000000	0.000000
G2	0.200000	0.179104	0.188976
G3	0.389344	0.412148	0.400421
G4	0.430070	0.317829	0.365527
G5	0.250000	0.200000	0.222222
G♯2	0.285714	0.457627	0.351792
G♯3	0.394472	0.257377	0.311508
G♯4	0.265896	0.131054	0.175573
G♯5	0.000000	0.000000	0.000000
Macro Average	0.262075	0.282285	0.259243

TABLE VIII
NOTE ACCURACY RESULTS

Accuracy
0.298095

accuracy in string classification (78.63%), the note classification presented challenges, with significant variations in performance across different classes. The analysis revealed potential areas for improvement, particularly in handling the subtleties of musical notes, and suggested exploring different neural network architectures and more sophisticated noise handling techniques. The findings contribute valuable insights to the field of audio signal processing and open exciting avenues for further exploration and innovation.

VI. FUTURE WORK

Building upon both the foundational work of previous researchers and the findings of this study, there are several promising directions for further exploration in the realm of monophonic guitar note classification, especially in noisy environments. The potential extensions of this work are as follows:

A. Exploration of Alternative Model Architectures

Future research could explore alternative neural network architectures, such as recurrent neural networks (RNNs) or transformer models, to enhance note differentiation and robustness to noise. Investigating different architectural designs may lead to improved performance in musical note classification, offering more nuanced insights into the complex relationships within audio data.

B. Music Diversity

While the current study focuses on specific genres such as rock, jazz, and blues, future work could encompass a wider range of musical expressions. Including additional genres will enhance the model's robustness and versatility, reflecting the rich diversity of musical culture.

C. Advanced Noise Handling Techniques

The challenge of noise in audio classification can be addressed through more sophisticated techniques. Future work could explore denoising autoencoders or specialized filtering methods, potentially setting a new standard in noise-robust audio classification. This direction has broad implications, extending beyond guitar note classification to fields like speech recognition and environmental sound analysis.

D. Real-Time Applications

The success in string classification, even under noisy conditions, suggests potential for real-time applications. One intriguing possibility is the development of tools that provide dynamic tuning assistance during live performances, ensuring that the guitar remains in tune even as conditions change. Additionally, educational platforms could be designed to offer instant feedback on playing techniques. Customizing these tools to cater to various skill levels and musical genres could significantly transform the way musicians engage with technology.

E. Collaboration with Music Professionals

Engaging with professional musicians, music educators, and sound engineers could lead to more targeted applications of the technology. Interdisciplinary collaboration may foster innovation and ensure alignment with the real-world needs of the music community.

The avenues outlined above offer a roadmap for future research, promising to advance the field of guitar note classification and contribute to the broader landscape of music technology. The interdisciplinary nature of this work, bridging the gap between music, technology, and education, sets the stage for innovative solutions that resonate with the multifaceted world of musical expression.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C. & ... Ghemawat, S. (2016), 'Tensorflow: Large-scale machine learning on heterogeneous distributed systems', *arXiv preprint arXiv:1603.04467* .

Bello, J. & Pickens, J. (2005), A robust mid-level representation for harmonic content in music signals, *in* 'Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)', pp. 304–311.

Bittner, R. M. & Bello, J. P. (2018), Guitarset: A dataset for guitar transcription, *in* 'Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)', Paris, France.

Brown, J. C. & Zhang, M. (2009), 'Musical frequency tracking using the methods of conventional and 'narrowed' autocorrelation', *Journal of the Acoustical Society of America* **89**(5), 2346–2354.

Choi, K., Fazekas, G. & Sandler, M. (2016), Automatic tagging using deep convolutional neural networks, *in* 'Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)'.

Chollet, F. (2015), 'Keras', https://keras.io.

Dieleman, S. & Schrauwen, B. (2014), End-to-end learning for music audio, *in* 'Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing'.

Elliott, T. M., Hamilton, L. S. & Theunissen, F. E. (2013), 'Acoustic structure of the five perceptual dimensions of timbre in orchestral instrument tones', *J. Acoust. Soc. Am.* **133**(1), 389–404.

Guitar, P. (n.d.), '5 steps to learn the fretboard fast', https://pathfinderguitar.com.au/guitar-blog/5-steps-to-learn-the-fretboard-fast.

Howard, D. M. & Angus, J. (2009), *Acoustics and Psychoacoustics*, 4th edn, Focal Press, Oxford.

Humphrey, E. J., Bello, J. P. & LeCun, Y. (2012), Learning a robust tonnetz space transform for automatic chord recognition, *in* 'Proceedings of the 2012 IEEE International Conference on Audio, Speech and Signal Processing (ICASSP)', pp. 453–456.

Järveläinen, H., Välimäki, V. & Karjalainen, M. (1998), 'Plucked-string models: from the karplus-strong algorithm to digital waveguides and beyond', *Computer Music Journal* **22**(3), 17–32.

Kehling, C. (2014), Automatic tablature transcription, *in* 'Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)'.

Lee, H., Largman, Y., Pham, P. & Ng, A. Y. (2009), Unsupervised feature learning for audio classification using convolutional deep belief networks, *in* 'Advances in neural information processing systems', pp. 1096–1104.

O'Malley, T., Bursztein, E., Long, J., Chollet, F., Jin, H., Invernizzi, L. & others, . (2019), 'Keras tuner', https://github.com/keras-team/keras-tuner.

Poliner, G. E. & Ellis, D. P. (2007), 'A discriminative model for polyphonic piano transcription', *EURASIP Journal on Advances in Signal Processing* **2007**(1).

Schölkopf, B. & Smola, A. J. (2002), *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, MIT press.

Thickstun, Z., Harchaoui, Z. & Kakade, S. M. (2017), Learning features of music from scratch, *in* 'Proceedings of the 31st International Conference on Machine Learning (ICML)', pp. 1969–1977.

Wiggins, J. & Kim, T. (2019), Guitar tablature estimation with a convolutional neural network, *in* 'Proceedings of the 20th International Society for Music Information Retrieval Conference', pp. 33–40.
URL: *https://archives.ismir.net/ismir2019/paper/000033.pdf*

APPENDIX

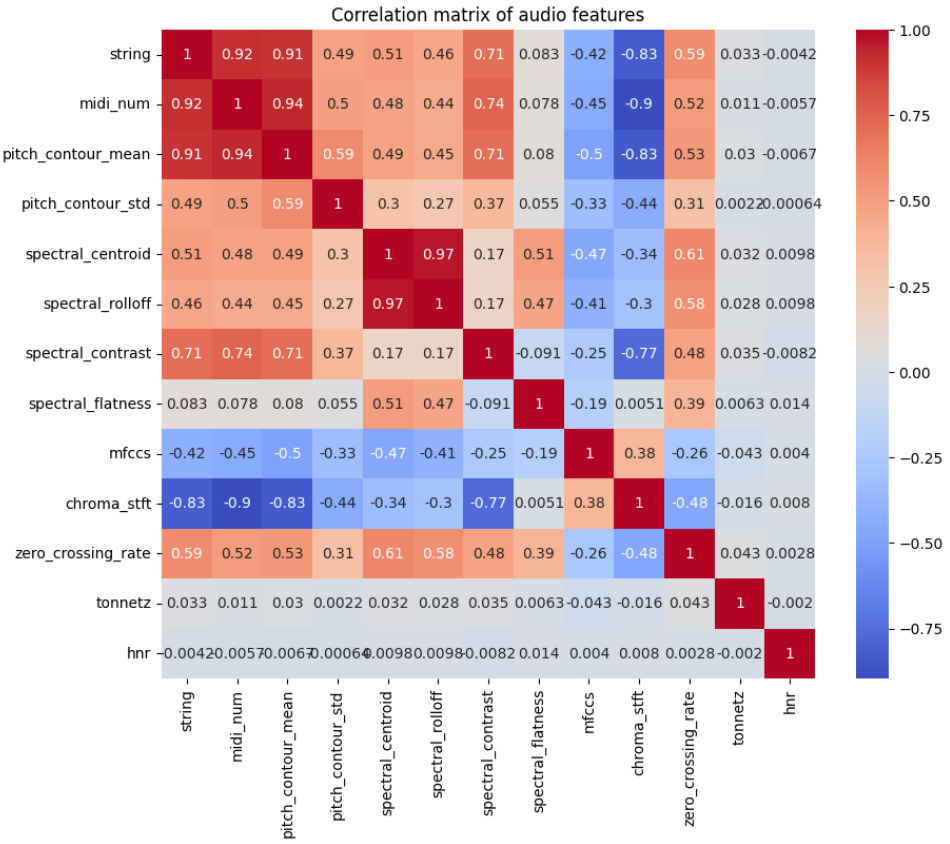

Fig. 5. Correlation heatmap of audio features. Here, we will be selecting the features with a correlation above 0.5 to our target classes, string and note.

Fret Note

Open	1F	2F	3F	4F	5F	6F	7F	8F	9F	10F	11F	12F	13F	14F	15F	16F	17F	18F	19F	20F
E4	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C
B3	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G
G3	G#/Ab	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C	C#/Db	D	D#/Eb
D3	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb
A2	A#/Bb	B	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E	F
E2	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C	C#/Db	D	D#/Eb	E	F	F#/Gb	G	G#/Ab	A	A#/Bb	B	C

Fret Frequency (Hz)

Open	1F	2F	3F	4F	5F	6F	7F	8F	9F	10F	11F	12F	13F	14F	15F	16F	17F	18F	19F	20F
329.63	349.23	369.99	392.00	415.30	440.00	466.16	493.88	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99	830.61	880.00	932.33	987.77	1046.5
246.94	261.63	277.18	193.66	311.13	329.63	349.23	369.99	392.00	415.30	440.00	466.16	493.88	523.25	554.37	587.33	622.25	659.26	698.46	739.99	783.99
196.00	207.65	220.00	233.08	246.94	261.63	277.18	193.66	311.13	329.63	349.23	369.99	392.00	415.30	440.00	466.16	493.88	523.25	554.37	587.33	622.25
146.83	155.56	164.81	174.61	185.00	196.00	207.65	220.00	233.08	246.94	261.63	277.18	193.66	311.13	329.63	349.23	369.99	392.00	415.30	440.00	466.16
110.00	116.54	123.47	130.81	138.59	146.83	155.56	164.81	174.61	185.00	196.00	207.65	220.00	233.08	246.94	261.63	277.18	193.66	311.13	329.63	349.23
82.41	87.3	92.49	97.99	103.83	110.00	116.54	123.47	130.81	138.59	146.83	155.56	164.81	174.61	185.00	196.00	207.65	220.00	233.08	246.94	261.63

Octave Colour Code

2	3	4	5	6

Fig. 6. Guitar Fretboard Illustration: Highlighting the Versatility of Note Positions and the Multiple Ways to Achieve the Same Fundamental Frequency Across Different Strings.

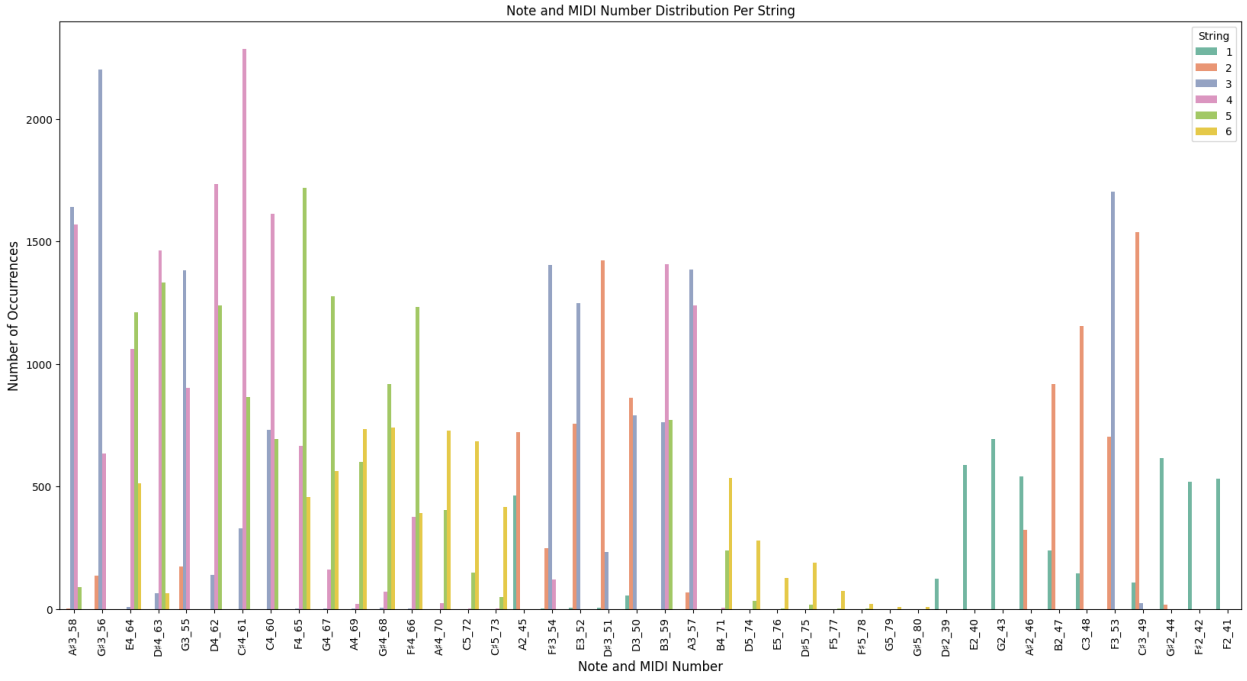

Fig. 7. Class Imbalance Visualization: Demonstrating the Distribution of Notes Before applying Random Over Sampling (ROS) to Achieve Balanced Training Data.

```
Training class counts for y_string:
{1: 28237, 2: 28237, 3: 28237, 4: 28237, 5: 28237, 6: 28237}
Test class counts for y_string:
{1: 922, 2: 1815, 3: 2873, 4: 3008, 5: 2540, 6: 1338}
Training class counts for y_note:
{'A2': 4351, 'A3': 5202, 'A4': 3619, 'A♯2': 4172, 'A♯3': 5156, 'A♯4': 3425, 'B2': 4478, 'B3': 4981, 'B4': 3357, 'C3': 4683, 'C4': 5048, 'C5': 3123, 'C♯3':
4619, 'C♯4': 5059, 'C♯5': 2972, 'D3': 4877, 'D4': 4904, 'D5': 2973, 'D♯2': 3798, 'D♯3': 4788, 'D♯4': 4807, 'D♯5': 2975, 'E2': 3810, 'E3': 4936, 'E4': 4511,
'E5': 2774, 'F2': 3870, 'F3': 4822, 'F4': 4381, 'F5': 2858, 'F♯2': 3814, 'F♯3': 4854, 'F♯4': 4299, 'F♯5': 2774, 'G2': 3767, 'G3': 5074, 'G4': 4068, 'G5': 2774,
'G♯2': 3863, 'G♯3': 4983, 'G♯4': 3823, 'G♯5': 0}
Test class counts for y_note:
{'A2': 227, 'A3': 565, 'A4': 283, 'A♯2': 189, 'A♯3': 614, 'A♯4': 224, 'B2': 238, 'B3': 573, 'B4': 179, 'C3': 266, 'C4': 613, 'C5': 154, 'C♯3': 336, 'C♯4': 652,
'C♯5': 108, 'D3': 346, 'D4': 641, 'D5': 54, 'D♯2': 15, 'D♯3': 332, 'D♯4': 582, 'D♯5': 43, 'E2': 118, 'E3': 411, 'E4': 563, 'E5': 30, 'F2': 99, 'F3': 495, 'F4':
554, 'F5': 17, 'F♯2': 104, 'F♯3': 387, 'F♯4': 411, 'F♯5': 4, 'G2': 134, 'G3': 461, 'G4': 387, 'G5': 5, 'G♯2': 118, 'G♯3': 610, 'G♯4': 351, 'G♯5': 3}
```

Fig. 8. Distribution of Strings and Notes After applying Random Over Sampling (ROS).

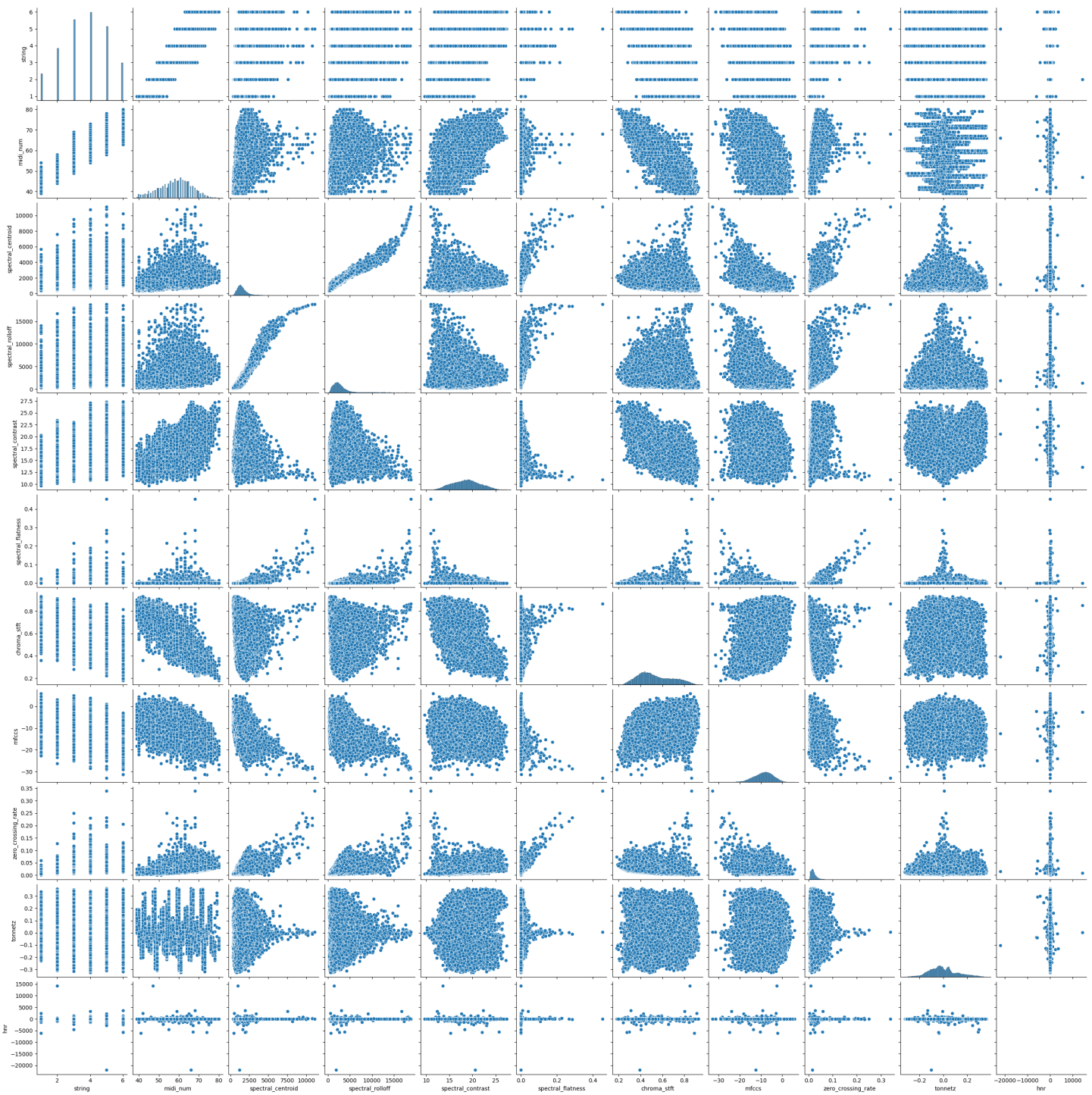

Fig. 9. Pairplot Illustration of Univariate and Multivariate Distributions of Features: A Comprehensive View of Feature Relationships and Scales, Highlighting the Need for Standardization in Modeling.

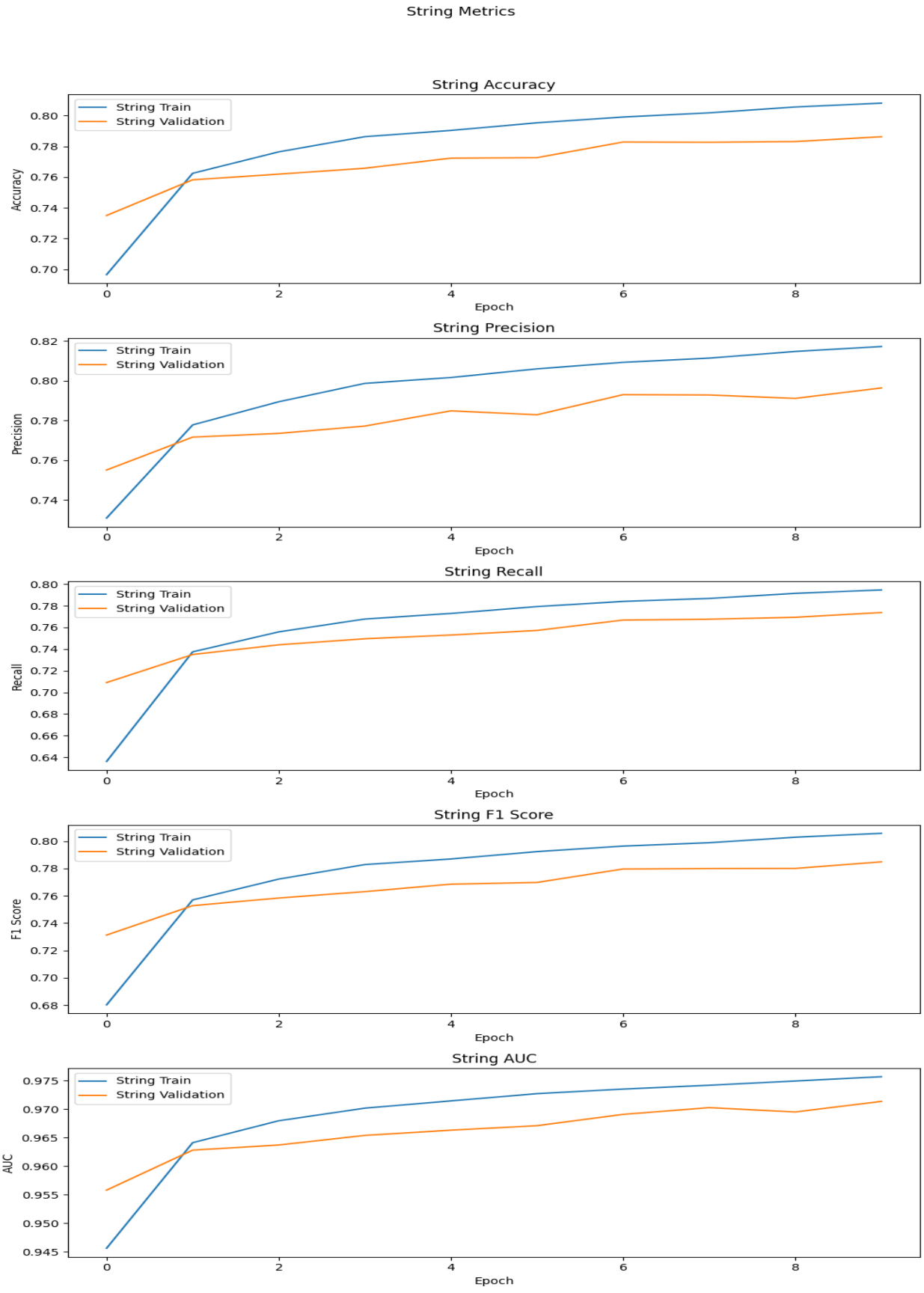

Fig. 10. Training and Validation Metrics Evolution: A visualization of String Classification Performance Across Epochs.

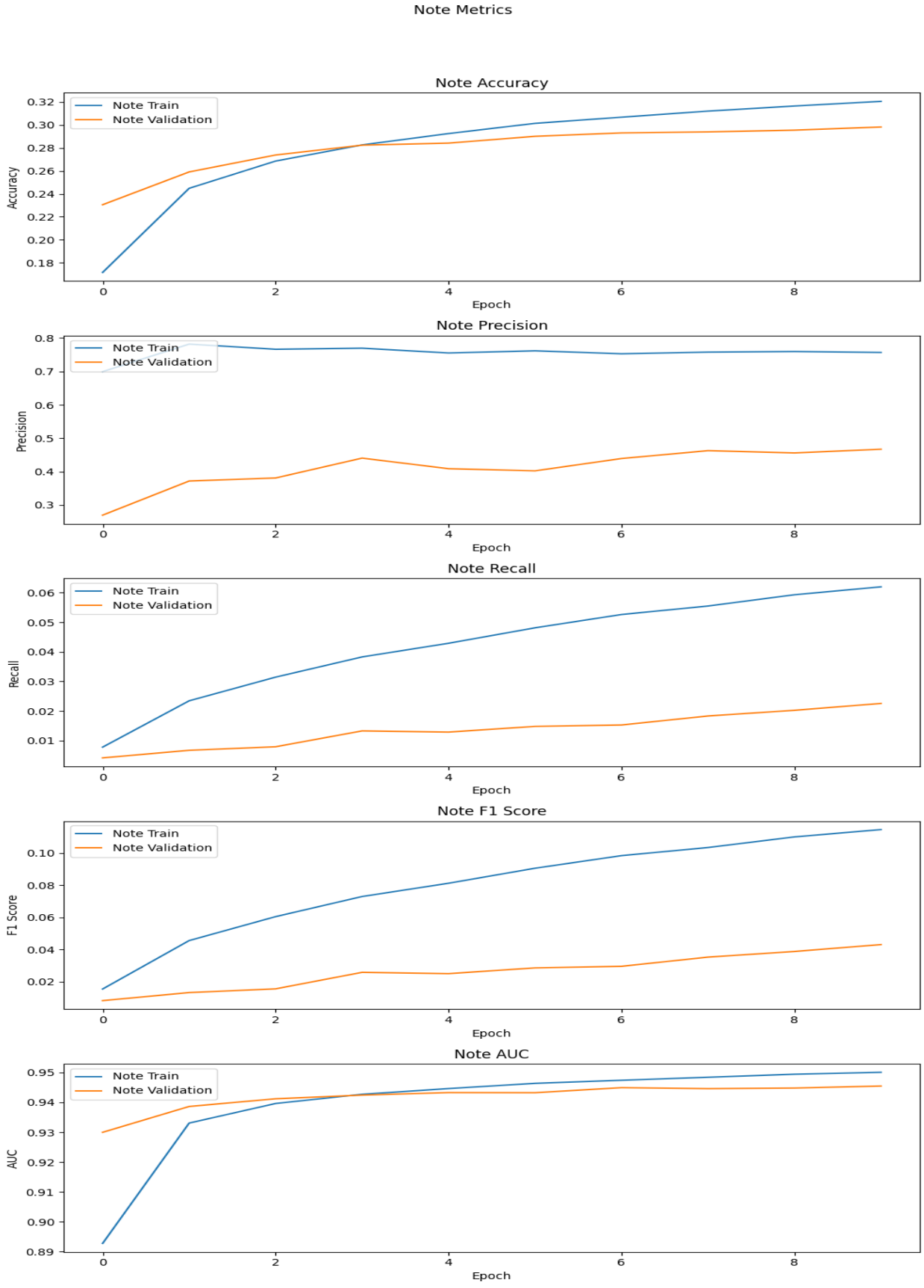

Fig. 11. Training and Validation Metrics Evolution: A visualization of Note Classification Performance Across Epochs.

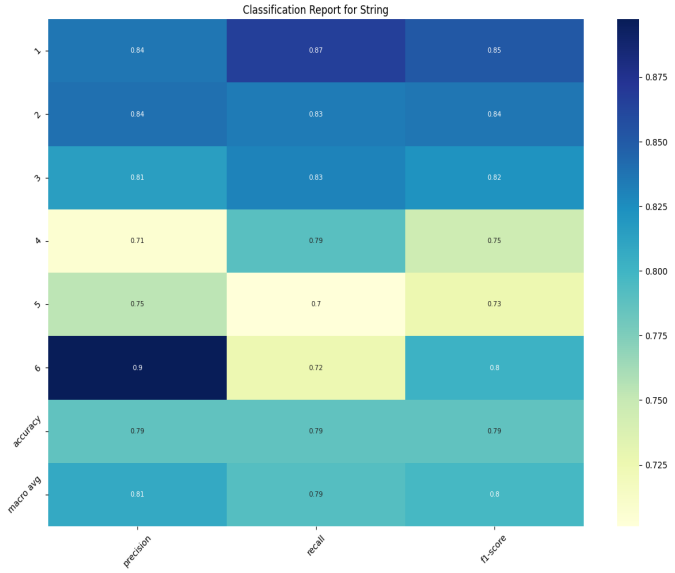

Fig. 12. Classification report of String

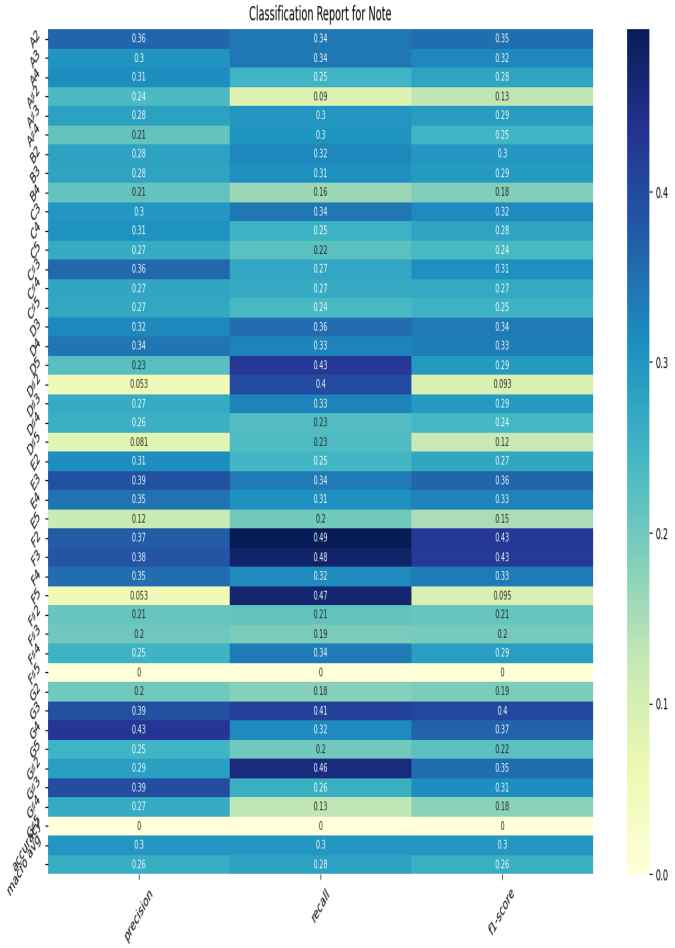

Fig. 13. Classification report of Note