NANYANG TECHNOLOGICAL UNIVERSITY

INTERACTIVE MUSIC VISUALIZATION

Agnes Tan Ziqi

College of Computing and Data Science

2024/25

NANYANG TECHNOLOGICAL UNIVERSITY

CCDS24-0355

INTERACTIVE MUSIC VISUALIZATION

Submitted in Partial Fulfilment of the Requirements
for the Degree of Bachelor of Engineering in Computer Science
of the Nanyang Technological University

by

Agnes Tan Ziqi

College of Computing and Data Science

2024/25

Abstract

Our project aims to revolutionize how violinists learn and perfect their craft through an interactive feedback system that analyzes performances and provides actionable visual feedback so that users can immediately see the mistakes made. The system leverages an Audio Spectrogram Transformer (AST) model, pre-trained on ImageNet [1] and refined using a carefully selected dataset of 1,400 annotated violin audio samples. The system works by converting audio input into spectrograms (visual representations of sound) and analyzing them to detect specific performance elements that need improvements.

Early testing shows promising results: our model can identify issues like pitch inaccuracies, improper bowing techniques, and other technical problems as the music is played. This creates a powerful learning tool that benefits both students seeking to improve and instructors looking to provide more precise guidance. This technology bridges traditional music instruction with modern technology. By providing data-driven insights during practice sessions, violinists can develop a deeper understanding of their playing habits and make targeted improvements. Rather than replacing conventional teaching methods, our system enhances them with objective analysis that might otherwise take years of experience to develop.

Acknowledgments

I extend my heartfelt thanks to A/P Alexei Sourin, my project advisor, for providing crucial direction and unwavering support during this project. His knowledge and motivation have been key in guiding the course and the success of this endeavour.

I am also profoundly grateful to Wang HanQin for his devoted mentorship, technical guidance, and support in tackling obstacles throughout the project's development and implementation stages. His patience and readiness to impart his expertise have significantly enhanced my educational journey.

Contents

List of Tables

List of Figures

Introduction

Interactive music feedback systems offer the potential to fundamentally transform and revolutionize how musicians refine and enhance their skills. Feedback is vital in helping musicians grow and develop, and failing to provide guidance on subsequent steps is akin to offering no feedback at all [2]. Traditional feedback sources, such as educators, mentors, or peers, are indeed beneficial but often entail a significant investment of time, carry a degree of subjectivity [3], and can be costly. Moreover, there is a lack of automated systems capable of providing real-time, actionable feedback tailored to individual performances. Transitioning from learner to teacher or peer, and recognizing various feedback types that enhance learning, is essential [3]. The main difficulty lies in the lack of an adaptable and accurate system for musicians to obtain constructive, actionable feedback on their performances. This project seeks to develop a system for analyzing specifically violin performances and offer visual feedback to help musicians enhance their technique. By leveraging deep learning, the system addresses common performance issues such as poor pitch, uneven tone, and improper bowing methods. The system employs an Audio Spectrogram Transformer (AST) model [4] that is trained using deep learning techniques. The AST model is developed utilizing an open-source dataset composed of approximately 1,400 violin audio samples, which are annotated with multi-label classifications. The model begins with ImageNet pre-training [5] and is subsequently fine-tuned for analyzing violin performances, capitalizing on the organized dataset and crafted features. Training the system with these datasets enables it to identify performance patterns, spot errors, and provide immediate corrections. This system blends traditional music education with technology, bridging human expertise and automated feedback for a scalable solution suited to musicians and educators.

Literature Review

1 Music Performance

1.1 Automatic Assessment of Tone Quality in Violin Music Performance

Giraldo presents a machine learning approach to evaluate the timbre quality in violin performances, filling a gap in music education technologies that typically focus on pitch and timing accuracy. The research methodology consists of three main stages: data acquisition, offline machine learning modeling, and user-defined machine learning modeling [6]. Data acquisition involved defining tone qualities through expert consultation, recording violin tones, and performing perceptual tests to assess correlations in perceived tone quality [7]. The offline machine learning modeling process involved extracting audio features such as pitch, spectral, and energy descriptors, and then selecting the most relevant features. Machine learning models were trained to classify tone qualities, including linear regression, support vector machines (SVM), and artificial neural networks (ANNs). Improve the accuracy of the tone quality classification. Under User-Defined Machine Learning Modeling, a real-time feedback system was developed, allowing users to train their tone quality models using their recordings and receive immediate feedback, allowing for personalized model creation and instant evaluation. This methodology used is depicted in Figure 1 below.

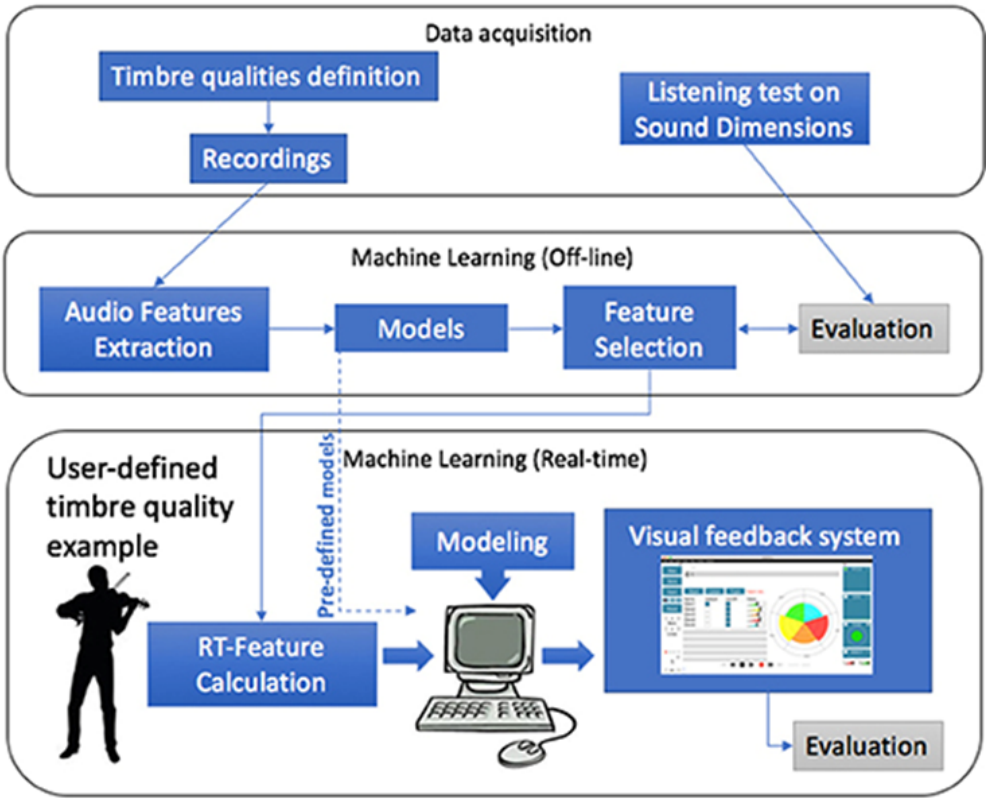

Figure 1: Overall framework for automatic tone assessment using machine learning. Adapted from [6].

The study found acceptable reliability in listener ratings and significant correlations between certain qualities of the tone. The best performance in classifying the tone qualities was achieved using ANNs. The real-time system demonstrated high accuracy in classifying user-defined tone qualities, although some accuracy loss was observed in cross-validation tests among different performers.

The real-time feedback system provides an immediate assessment of tone quality during practice, enhancing the learning experience by offering insights into a performer's playing. In summary, this research presents a significant advancement in the assessment of violin performance quality, particularly in timbre, and offers a practical tool for music education.

1.2 Deep Learning-Based Music Quality Analysis Model

The research article by Jing Jing presents a deep learning-based model for music quality analysis, integrating shallow and deep learning techniques to enhance recognition accuracy. This paper proposes a speech emotion recognition model that employs a decision fusion method that combines features extracted by the PCANET network with traditional shallow learning methods, specifically using SVM [8]. This creates an effective decision-making layer, and the goal is to take advantage of the strengths of both approaches to improve music quality recognition and robustness, addressing the limitations of each method through a fusion process that optimizes performance [9]. An overview of such fusion is shown in Figure 2.

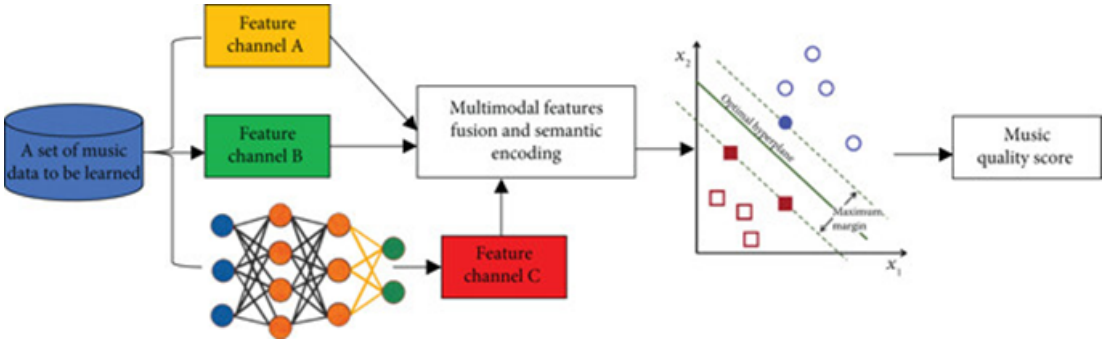

Figure 2: The flowchart of our proposed music quality evaluation. Adapted from [9].

The study highlights the importance of feature extraction, which has been a focal point of research for decades, involving various acoustic characteristics essential for effective recognition of music quality. These features can be categorized into: Prosodic Features, Frequency Domain Features, and Sound Quality Features [10]. Experimental results demonstrate improved recognition rates using this hybrid approach compared to traditional methods, showcasing its effectiveness in music quality evaluation and potential applications in artificial intelligence and human-computer interaction systems. By integrating various methodologies, the proposed model ensures accurate assessments across different conditions, improving the understanding and recognition of music quality.

1.3 Analysis of Piano Performance Characteristics by Deep Learning and Artificial Intelligence and Its Application in Piano Teaching

This study explores the integration of Deep Learning (DL) and Artificial Intelligence (AI) in modern piano teaching, specifically aimed at preschool children. It emphasizes enhancing the teaching environment and the functions of intelligent pianos, proposing innovative methods to improve the quality of piano education. It talks about the intelligent piano that connects to devices via the Internet, facilitating online learning and enhancing user engagement through interactive features and games [11]. A novel method utilizing Convolution Neural Networks (CNN) is introduced to detect piano note onsets by transforming the piano music signal from a time-domain waveform to a frequency distribution over time, achieving stability after 80,000 iterations [12]. For further details on the CNN model, please refer to Appendix 1. A questionnaire was conducted on 40 preschool children aged 4 to 6 years and parents to gather feedback on teaching effectiveness, interest levels, and learning outcomes when the intelligent piano teaching method was used to teach 'Jingle Bells'. The system used is depicted in Figure 3. More than 80% of the surveyees expressed positive feedback on this method in all 3 aspects.

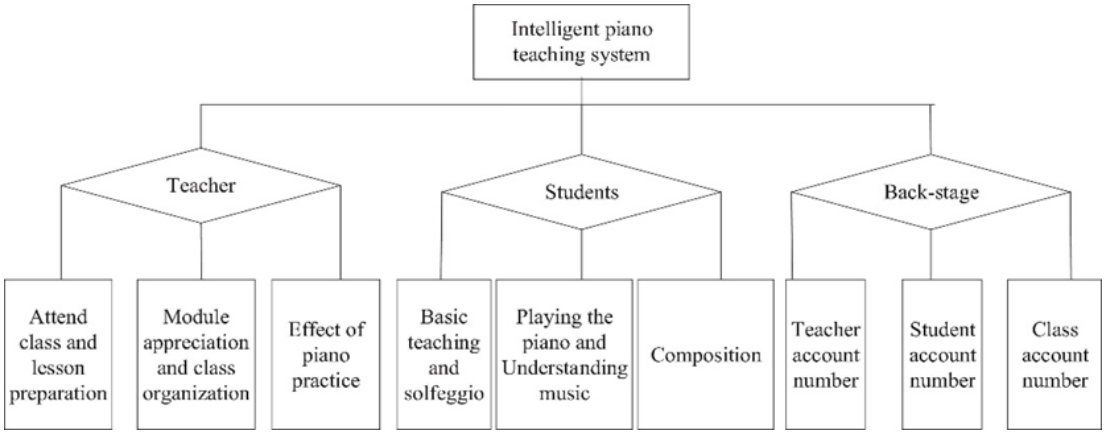

Figure 3: Intelligent piano teaching system. Adapted from [12]

The intelligent piano teaching method effectively combines DL and AI, optimizing piano performance and greatly enhancing preschool children's interest in learning. It significantly contributes to piano education, facilitating not only skill acquisition, but also fostering a positive learning environment for preschool children. This intelligent teaching model offers a promising avenue for improving music education through technology that surpasses traditional methods.

2 Identification and Classification

2.1 Musical Instrument Identification Using Deep Learning Approach

This paper details a novel method for automatically identifying multiple musical instruments within audio excerpts. The approach utilizes a set of individual convolutional neural networks (CNNs), one for each instrument, rather than a single unified model. The limitations of traditional spectral and cepstral analyses are discussed, particularly their struggles with percussive sounds and nonharmonic effects, as well as the impact of articulation techniques on frequency shifts. The Slakh data set [13], which contains 2100 audio tracks with aligned MIDI files and separate instrument stems, was used. Four instruments were selected, bass, drums, guitar, and piano, and the audio tracks were divided into 4-second excerpts. The data set was then divided into training, validation and evaluation sets, with class weighting applied to balance the representation of each instrument. An example of spectrograms of selected instruments and the prepared mix is presented in Figure 4.

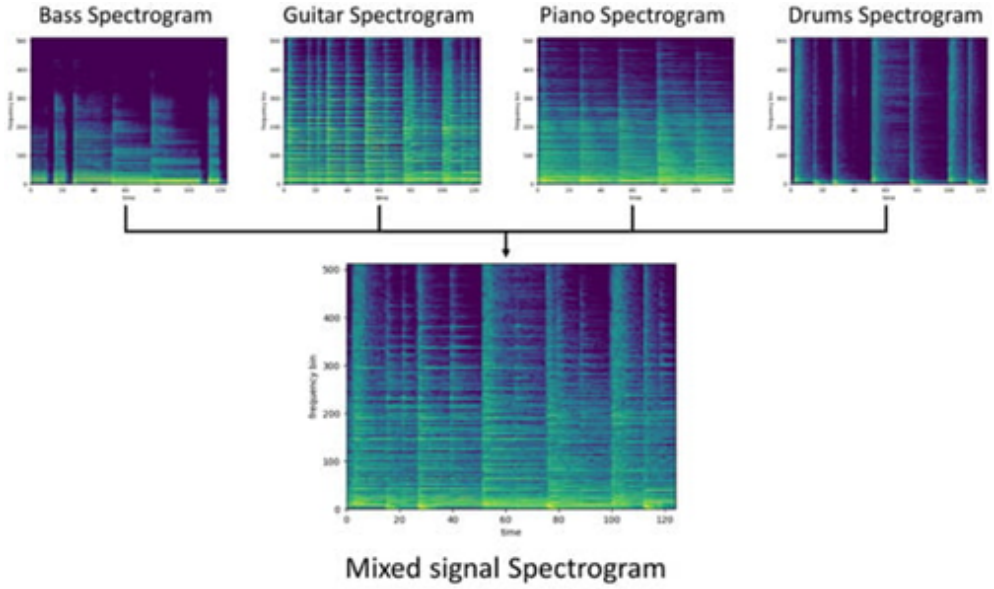

Figure 4: Example of spectrograms of selected instruments and the prepared mix. Adapted from [14].

The proposed neural network was implemented using the Keras framework and functional API [15]. It uses MFCCs as input features, followed by a series of 2D convolutional layers, max pooling, batch normalization, and dense layers [14]. The authors analyse the performance for each instrument individually, noting that drums were most easily identified, while bass, guitar, and piano presented more challenges. A confusion matrix is presented in Table 1 to further illustrate the model's performance.

Table 1: Confusion matrix (in percentage points). Adapted from [14].

	Ground Truth Instrument [%]			
Predicted Instrument	**Bass**	**Guitar**	**Piano**	**Drums**
Bass	81	8	7	0
Guitar	4	69	13	0
Piano	5	12	77	0
Drums	3	7	6	82

Their multi-CNN approach is not only flexible, allowing for extension of the model

with more instruments by adding new sub models in the architecture proposed, but it is also superior in terms of performance compared to existing methods.

2.2 Music and Instrument Classification using Deep Learning Techniques

Haidar-Ahmad presents a multi-class classifier designed to identify musical instruments in audio streams. The model employs a Convolutional Neural Network (CNN) that processes audio streams to produce mel-spectrograms and classifies them into four categories: “Piano”, “Drums”, “Flute”, or “Other”. The field of music classification has evolved from traditional algorithms like Support Vector Machines (SVMs) and Gaussian mixture models to deep learning techniques that automate feature extraction. The transition to deep learning has shown promising results, as researchers utilize CNN architectures similar to AlexNet - a convolutional neural network that is 8 layers deep [16] and VGGNet - Visual Geometry Group, Department of Science and Engineering of Oxford University [17], achieving significant improvements in classification accuracy. The dataset used here is AudioSet by Google [18] which provides human-labeled audio clips from YouTube. It consists of diverse audio streams, and poses additional processing challenges due to background noise and multiple instruments. he project narrows its focus to 9,600 samples and employs a pre-processing pipeline to standardize and normalize the audio data, ultimately converting it into mel-spectrograms. The model developed is a feed-forward CNN, using mel-spectrograms as input. It comprises 7 layers with varying filter sizes and a max-pooling mechanism. The model’s architecture is optimized with He initialization and uses cross-entropy as the loss function. The evaluation metrics, including precision, recall, and F1-score, yielded an average precision of 70%, recall of 65%, and F1-score of 64%. A confusion matrix revealed that the model struggled most with differentiating ”Drums” from ”Other.” As seen in Table 2 [19].

Table 2: Confusion Matrix for the 4 Classes. Adapted from [20].

Actual Predicted	Piano	Drums	Flute	Other
Piano	134	2	40	24
Drums	19	74	2	105
Flute	30	0	160	10
Other	41	5	6	148

3 Models for Music Tagging and Feedback

3.1 A real-time system for measuring sound goodness in instrumental sounds

This study notes that the assessment of musical performance quality is inherently subjective [21] and varies widely building on previous research by Knight et al. [7], it aims to classify isolated musical notes based on performance quality, focusing on five key sound attributes: Dynamic Stability, Pitch Stability, Timbre Stability, Timbre Richness, Attack Clarity. By focusing on these attributes, the study introduces an innovative system that extends beyond traditional tuners by evaluating the overall sound quality of musical performances in real-time. The proposed system assigns a quantitative score to each note, comparing it against a set of predefined reference sounds to provide an objective measure of performance quality. The system consists of two primary components: the learning process and the real-time application. The learning process involves multiple stages, including data collection, segmentation, feature selection, and score mapping, ensuring that the model can accurately assess performance quality across different musical contexts. Meanwhile, the real-time component segments input notes compute features and generate goodness scores as presented in Figure 5 [21].

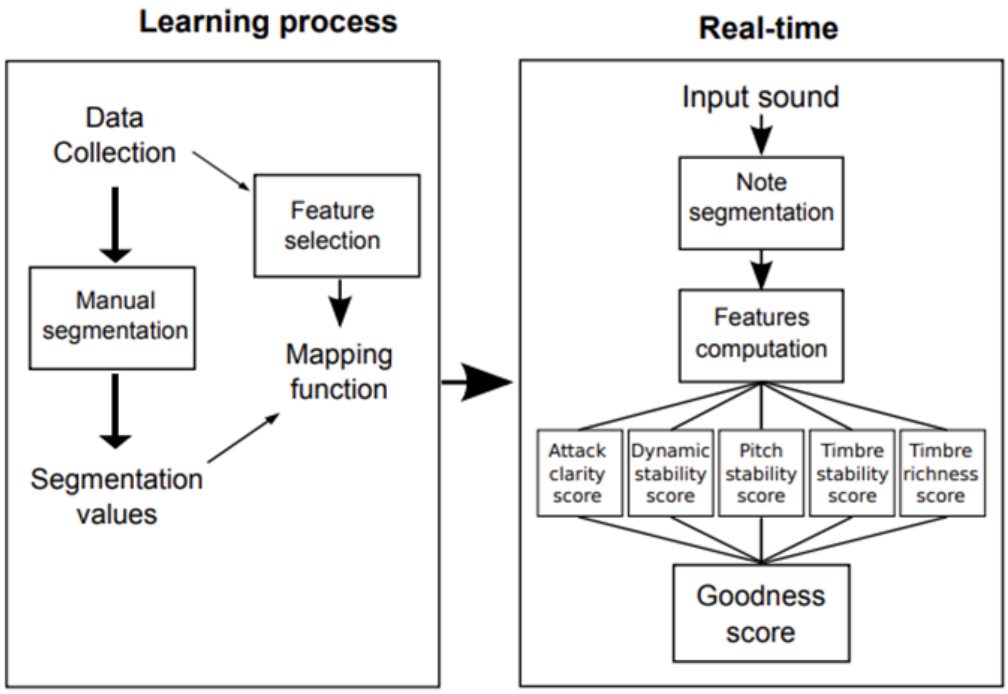

Figure 5: System Diagram. Adapted from [21].

Its development involved recording professional musicians, annotating the recordings with quality ratings from music teachers, and applying machine learning to identify audio features that reflect performance quality. The training dataset comprises recordings of single notes from three instruments: clarinet, flute, and trumpet. Each instrument has six classes of sound, including five intentionally poorly played classes and one well-played class. The dataset was meticulously recorded using professional microphones and included a variety of semitones. Using the Essentia library, different audio descriptors are selected for each sound attribute, with almost 200 features analyzed. A machine learning approach identifies the most discriminative feature for each attribute, allowing for precise classification. The overall goodness score is then calculated as the average of the five sound attribute scores. Musicians played notes while using the prototype system, grading each sound attribute. The evaluation confirmed that the system's scores align closely with musician criteria, with mean absolute errors indicating a promising level of accuracy.

3.2 Evaluation of CNN-based Automatic Music Tagging Models

Recent advances in deep learning have significantly improved content-based automatic music tagging systems. Various architectures based on CNNs and Fully Convolutional Network (FCN) [22] have been proposed (see 6), achieving state-of-the-art results in the multi-label binary classification task of music tagging. However, differences in experimental setups hinder direct comparisons among these architectures. To address this, the paper presents a consistent evaluation of various music tagging models across three datasets—MagnaTagATune [23], Million Song Dataset [24], and MTG-Jamendo [25] — using common evaluation metrics (ROC-AUC and PR-AUC). The models are also assessed for their generalization capabilities against perturbations like time stretch and pitch shift.

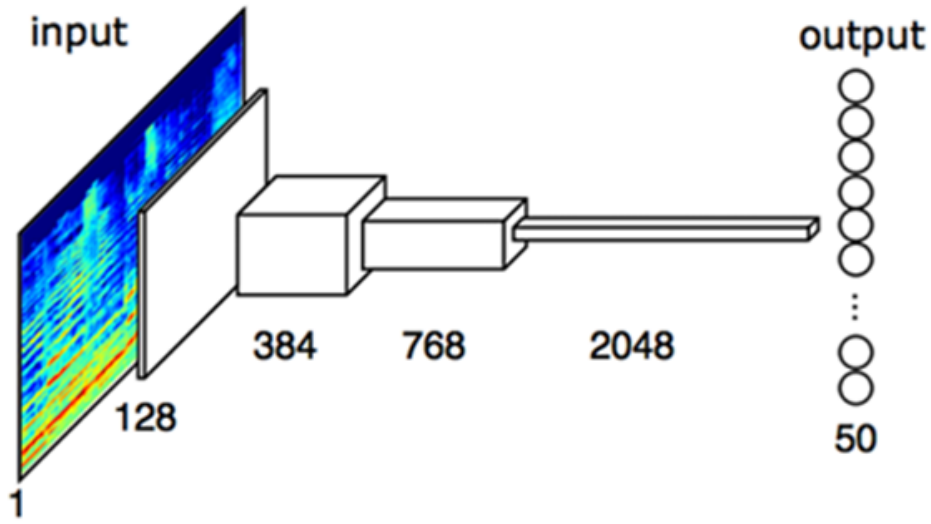

Figure 6: A block diagram of the proposed 4-layer architecture, FCN-4. Adapted from [22]

Automatic music tagging aims to predict relevant tags for songs that carry semantic music information, which can be utilized in applications such as music recommendation and retrieval. The field has shifted from hand-crafted feature extraction methods to data-driven approaches using deep learning, mainly CNNs, which have revolutionized performance in music tagging tasks. Despite their success, direct comparisons are complicated by inconsistent experimental setups reported by different researchers. This

paper aims to provide a clearer comparison of the architectures by standardizing the evaluation process. Music tags encompass various semantic information like genres, moods, and instruments. The challenge is that relevant characteristics may not be present throughout an entire song, leading to a multiple instance problem where only specific instances are responsible for a tag's association. 2 approaches were taken to tackle this: Song-Level training where the model rained on full songs to produce song-level predictions, and Chunk-Level Training where the model is trained on short audio chunks, generating chunk-level predictions that are aggregated for evaluation. The evaluation uses both the area under the receiver operating characteristic curve (ROC-AUC) and the area under the precision-recall curve (PR-AUC), with the latter being more informative for skewed datasets [26]. The performance results indicate that models trained with short audio excerpts generally outperform those trained with longer segments.

3.3 AST: Audio Spectrogram Transformer

The Audio Spectrogram Transformer (AST) is introduced as a revolutionary model for audio classification. Unlike traditional models that rely heavily on convolutional neural networks (CNNs), AST operates purely on self-attention mechanisms [4]. This paper demonstrates that AST can achieve state-of-the-art results across multiple audio classification benchmarks. CNNs have been the primary architecture used in audio classification [27] with its inductive biases such as spatial locality and translation equivariance. Recently, hybrid models combining CNNs with self-attention mechanisms have shown improved performance to the point where the necessity of CNNs for audio classification is questioned, leading to the development of the AST capable of capturing long-range global context. AST is evaluated on several classification tasks and datasets, which include: AudioSet [28], ESC-50 [29], and Speech Commands [30] and achieved outstanding results of 0.485 mAP, 95.6% accuracy and 98.1% accuracy, respectively, on each dataset. AST also supports variable-length inputs without architectural changes. Compared to CNN models, AST has a simpler architecture with fewer parameters and converges faster during training [4].

Figure 7 illustrates the proposed Audio Spectrogram Transformer (AST) architecture. The AST model processes audio waveforms by converting them into 128-dimensional log Mel filterbank spectrograms of size 128 × 100t. These spectrograms are then divided into overlapping 16×16 patches, flattened into 1D embeddings, and enriched with positional embeddings to preserve spatial structure. A Transformer encoder with 768 embedding dimensions, 12 layers, and 12 attention heads is then applied. The classification output is derived from the [CLS] token, which represents the spectrogram's features [4]. The fallback of this model is that it needs more data to train on and only starts to outperform CNNs when data is over 14 million for image classification tasks [31].

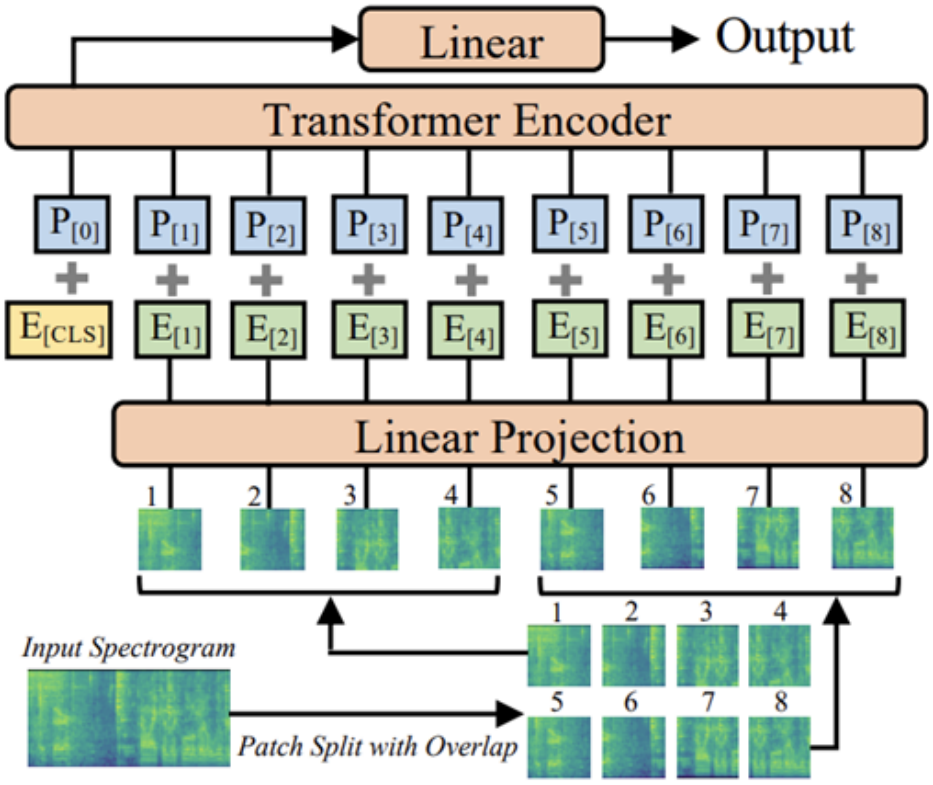

Figure 7: The proposed audio spectrogram transformer (AST) architecture. Adapted from [3].

Audio datasets do not typically have such capacity and thus drove the application of cross-modality transfer learning to AST since images and audio spectrograms have similar formats. The paper therefore also highlights the benefits of transferring knowl-

edge from Vision Transformers (ViT) [32] pre-trained on ImageNet [1] to AST. This is done by averaging weights of the ViT patch embedding layer for the single-channel input of the AST, normalizing the input spectrogram and adapting positional embeddings through a cutting and bilinear interpolation method. This was then tested on AudioSet [28] that consists of over 2 million 10-second audio clips labelled with sounds from 527 categories. The model employs techniques such as data augmentation, model aggregation, and binary cross-entropy loss during training. The importance of ImageNet pre-training shows significant performance improvement, especially with smaller datasets. The AST represents a significant advancement in audio classification, demonstrating the potential of purely attention-based models in processing audio data.

Hypothesis and Plan

This project hypothesises that by leveraging Audio Spectrum Transmitter (AST) features and Deep Learning techniques, it is possible to accurately classify violin audio samples into multiple categories and provide actionable visual feedback to improve violin performance.

Specifically, the project aims to achieve an F1-score of at least 0.6 and demonstrate the utility of visual feedback in enhancing the learning process for violinists.

The R&D plan involves:

1. **Data Acquisition**: Sourcing a comprehensive dataset of annotated violin audio samples covering various playing techniques, quality levels, and performance characteristics.
2. **Model Development**: Implementing a deep learning model leveraging AST architecture, optimized specifically for violin audio classification.
3. **Evaluation**: Rigorously testing the model using metrics such as F1-score, with a target threshold of 0.6 or higher for production deployment.
4. **Visual Feedback System and Web Application**: Designing an intuitive visual feedback interface that translates model predictions into actionable insights for violinists, integrated into an interactive webpage that allows users to:
    - Upload violin audio samples
    - Receive predictions from the model
    - View visual feedback on their performance

The system will be optimized to make the learning process more interactive and intuitive, providing valuable insights into areas for improvement in a violinist's technique, such as dynamics, pitch accuracy, and tonal quality.

Project Design and Methodology

1 Train Dataset Preparation

The dataset used to train the AST model was downloaded from Zenodo and is meant to organize the sounds effortlessly. It is organized in four different tables: sounds, takes, packs and ratings [33]. The table of interest here would be 'sounds'. Refer to Table 3 below for the original columns.

Table 3: Columns from 'sounds' table

cid	name	type
0	id	INTEGER
1	instrument	VARCHAR(8,0)
2	note	VARCHAR(2,0)
3	octave	INTEGER
4	dynamics	VARCHAR
5	recorded_at	DATETIME
6	location	VARCHAR
7	player	VARCHAR
8	bow_velocity	VARCHAR(6,0)
9	bridge_position	VARCHAR(10,0)
10	string	INTEGER
11	csv_file	INTEGER
12	csv_id	INTEGER
13	pack_filename	VARCHAR
14	pack_id	INTEGER
15	attack	INTEGER
16	decay	INTEGER
17	sustain	INTEGER
18	release	INTEGER
19	offset	INTEGER
20	reference	BOOLEAN
21	klass	VARCHAR
22	comments	TEXT
23	semitone	INTEGER
24	pitch_reference	FLOAT

This dataset contains audio samples from 12 different instruments but the one we are looking at would be violin. The columns 'pack_id' and pack_filename' tells us which

row describes which audio file. ‘pack_id’ links to another table: ‘packs’ which tells us the folder name where the audio file resides. A SQL query is executed to create and update a new column called ‘folder’ in the ‘sounds’ table by setting it to the corresponding name from ‘packs’ table, where the ‘pack_id’ in ‘sounds’ matches the ‘id’ in ‘packs’. If any of these columns (‘folder’ and ‘pack_filename) is NULL, there is no telling which audio the row is referring to and is thus not useful. Hence, I start by removing rows with such NULL values (2 rows deleted). In ‘sounds’ table, the column ‘klass’ is a user-generated tags of the tonal qualities of the sound [33]. This column is the essence of the dataset where we need our model to be able to predict such classes. Thus, we also need to remove rows with empty ‘klass’ since they will not be useful as well (222 rows deleted).

Since our focus is on violin, I filtered out rows with ‘instrument’ equal to ‘violin’ into another table named ‘violinSounds’. This table will be the primary table we will work within this project. This table consists of 1383 rows and 20 different classes (Refer to Appendix 3 - Classes). Some audios have multiple classes sewn into 1. For example, ’bad-richness bad-timbre bad-pitch’ is considered as a type of ‘klass’ in the dataset but it is in fact 3 classes: ’bad-richness’, ’bad-timbre’ and ’bad-pitch’. Thus, the relationship between audio and ‘klass’ is one-is-to-many.

In this case, it is a Multi-label classification problem in deep learning. In the dataset pre-processing, I need to follow the labelling techniques of Multi-label classification. For example, suppose there are 20 ’klasses’, where ’bad-richness’, ’bad-timbre’ and ’bad-pitch’ are ‘klass’ 0, 1, and 2. For an audio sample that belongs to these three classes, it should be labelled as a 20-dimensional vector (1, 1, 1, 0, 0, ..., 0). Another new column ‘labelvector’ is added to show this information.

The cleaned dataset is eventually split into 80-20 for training and testing. The training dataset consists of 1106 audio samples and the remaining 277 will be used as a test dataset.

2 AST Model

2.1 Rationale for AST Selection

Audio Spectrum Transformer (AST) was chosen as the foundational technology for this project due to its superior performance in audio classification tasks. Unlike traditional Convolutional Neural Networks (CNNs) that may miss temporal dependencies in audio signals, AST effectively captures both spectral and temporal information through its transformer architecture. AST treats audio spectrograms as image-like patches and processes them with self-attention mechanisms, allowing it to recognize complex patterns in violin performances that might be missed by conventional methods. Additionally, AST has demonstrated state-of-the-art results on audio classification benchmarks, making it particularly suitable for the nuanced task of distinguishing between different violin playing techniques and quality indicators.

2.2 Environment Setup

The coding environment is initialized by importing essential libraries for data manipulation, audio processing, and machine learning. Refer to Appendix 2 for a list of libraries used for this project. The model is configured to run on a CUDA-capable GPU with a CPU as a backup if one is not available. To ensure reproducibility, a fixed random seed is set, making the operations in the code deterministic, ensuring others can replicate results.

2.3 Dataset Class

A custom PyTorch Dataset class, ViolinAudioDataset is defined to handle the loading and processing of the violin audio data from a JSON file and apply the necessary transformation to prepare the data for training. Here, the labels are converted from a comma-separated string into a list of integers, which are then used to create a multi-hot encoding vector (a vector of zeros and ones, where each position corresponds to a class label). The audio file is loaded using librosa, and a Mel spectrogram is computed from the audio signal, which is then converted to a decibel scale. If the Mel spectrogram has

fewer time steps than 688, it is padded with zeros.

The dataset was randomly split into training and testing subsets, with 80% of the audio samples allocated to the training set and the remaining 20% reserved for testing. This random split ensures that the model is trained on a diverse set of samples while allowing for an unbiased evaluation of its performance on unseen data.

2.4 Model Architecture

A pre-trained model from the AST library to process spectrogram data is loaded after downloading it into a local directory. This model is primarily designed for audio classification tasks and can be applied to various datasets like AudioSet, ESC-50, and other audio classification challenges. In our case, we would like to train it on our dataset [33]. The input spectrogram is set to 688 time frames as obtained from the longest audio in the entire dataset. This process involves using the 'librosa' library to load audio files and compute their Mel spectrograms and determine the maximum number of time steps across all of them which works out to be 688.

```
model = ASTModel(label_dim=num_classes, fstride=10, tstride=10, input_fdim=128, input_tdim=688,
                 audioset_pretrain=False)
```

Figure 8: An instance of the AST Model created

An instance of the AST model is created (refer to Figure 8) set to predict 20 (label_dim) classes for each input sample.

- **'fstride' = 10**: Each frame of the spectrogram will be 10 frequency bins apart.
- **tstride = 10**: Each frame of the spectrogram will be spaced 10 time steps apart.
- **'input_fdim' = 128**: The input to the model will have 128 frequency bins, which is the typical value for Mel spectrograms.
- **'input_tdim' = 688**: The input to the model will have 688 time frames.
- **'audioset_pretrain' = False**: The model will not be pre-trained on the AudioSet dataset and will be trained from scratch with our own dataset – Good-sounds dataset.

This configuration will allow the AST model to process the spectrogram of the audio and predict a set of labels corresponding to the 20 categories.

2.5 Training Process

Before the training process, a few variables are declared:

```
num_classes = 20
num_samples_per_class = [1383 for i in range(20)]
num_positives_per_class = [215, 74, 37, 36, 36, 107, 225, 365, 135, 148, 170, 51, 37, 37, 37, 37, 74, 253, 105, 25]
scaling_factor = 3
```

Figure 9: Variables needed to calculate weights

- **'num_samples_per_class'**: The total number of samples for each class, represented by an array of 20 elements, all with value 1106 (total number of audio samples in the training dataset).
- **'num_positives_per_class'**: The number of samples that are classified under each class, represented by an array of 20 elements.
- **'scaling_factor'**: A scaling factor of 2 is used to adjust the weights to account for the class imbalance since more 0s are present in each label.

To address the class imbalance issue in the dataset, it is crucial to scale the loss for positive samples (1s) relative to their rarity. In datasets where certain classes have a much smaller number of positive samples compared to others, these under-represented classes can often be overlooked during model training, leading to biased predictions. To mitigate this, higher weights are assigned to the classes with fewer positive samples. The below formula is used for the calculation of the weights for each class:

$$\text{pos\_weight}[i] = \frac{\text{num\_positives}[i]}{\text{num\_samples}[i] \times 2} \tag{1}$$

The batch size is set to 1 so each update will be based on a single sample. (Batch size is 1 as the GPU used can only load 1 to 2 pieces of 10-second music). The model will be trained for 5 epochs as well. The Adam Optimizer is used with a learning rate of 0.00001. Pytorch's 'nn.BCEWithLogitsLoss' function is employed

as the loss function for binary classification tasks, which is suitable for multi-label classification problems. It combines a sigmoid activation with binary cross-entropy loss and 'pos_weight' calculated before is passed to account for class imbalance.

These parameter values were determined through extensive experimentation, where multiple trial runs with different variations were conducted. The final combination was selected based on its performance, yielding the best results in terms of evaluation metrics. This iterative process ensured that the chosen parameters provided the most optimal balance between model performance and computational efficiency.

```
# Training loop
accumulation_steps = 2  # Number of steps to accumulate gradients before updating weights

for epoch in range(num_epochs):
    model.train()
    print(f"Beginning Epoch {epoch + 1}/{num_epochs}")
    total_loss = 0
    optimizer.zero_grad() # Clear gradients at the start of each epoch

    for batch_idx, (mel_spectrogram, labels) in enumerate(train_loader):
        mel_spectrogram, labels = mel_spectrogram.to(device), labels.to(device)

        outputs = model(mel_spectrogram)  # Explicit call to forward with debugging
        loss = criterion(outputs, labels)
        loss = loss / accumulation_steps  # Scale loss for accumulation
        loss.backward()  # Backward pass (accumulate gradients)

        # optimizer updates weights only after 4 steps
        if (batch_idx + 1) % accumulation_steps == 0:
            optimizer.step()
            optimizer.zero_grad()  # Clear gradients after the step

        total_loss += loss.item() * accumulation_steps  # Undo scaling for correct total loss

    # Handle leftover gradients if the dataset size is not divisible by accumulation_steps
    if (batch_idx + 1) % accumulation_steps != 0:
        optimizer.step()
        optimizer.zero_grad()

    print(f"Finished Epoch {epoch + 1}/{num_epochs}, Loss: {total_loss / len(train_loader)}")
```

Figure 10: Training loop

A training loop shown in Figure 10 is implemented. Gradient accumulation is incorporated in this training loop to simulate a larger batch size without increasing memory usage. This is particularly useful when working with memory-intensive models like AST or other complex neural networks. To simulate a larger effective gradient accumulation allows to accumulate gradients over multiple iterations (each with a batch

size of 1). After accumulating the gradients for a certain number of steps, it performs the weight update. This is essentially equivalent to having a larger batch size without requiring more memory. The accumulation step is set to 2, meaning gradients will be accumulated and averaged over 2 batches before updating the model's weights, mimicking the behaviour of using a larger batch size of 2. For each epoch, the model is set to training mode and batches of Mel spectrograms and their corresponding labels are loaded, and the model performs a forward pass to generate predictions.

The values of each hyperparameter are selected after several trials of training the model with different values and combinations and evaluating its metrics. The binary cross-entropy loss with logits is computed between the predictions and the true labels, incorporating the class-specific weights to handle the label imbalance. The loss is also divided by accumulation steps to ensure that after accumulation, the gradient magnitudes remain consistent with normal training. Gradients are zeroed before each backward pass, which computes and accumulates gradients for the model parameters. The optimizer updates the model's parameters based on these gradients, and the batch loss is accumulated to calculate the average loss for the epoch. After completing all batches, the average loss is printed, measuring training progress.

2.6 Evaluation Metrics and Results

The Binary Cross-Entropy (BCE) Loss reflects the difference between the predicted outputs and the true labels, quantifying how well the model's predictions align with the ground truth. BCE Loss values indicate better performance, as they signify a closer match between the predicted probabilities and the actual labels. During training the loss values progressively decreased across the epochs, demonstrating the model's ability to learn patterns and improve during training. This steady decline in loss demonstrated that the model effectively adjusted its weights and biases through backpropagation and optimization, leading to improved prediction accuracy over time. The BCE loss is summarized in the table below:

Table 4: Average BCE Loss for each epoch

Epoch	Average Loss
1	0.7457695109314988
2	0.5753985324615165
3	0.4952367063456806
4	0.4231891400815673
5	0.3416443977801375

To evaluate the model's performance, the model was tested on the test dataset, which consists of 20

The model demonstrates a moderate overall performance, achieving an accuracy of 23.1%. Notably, it exhibits strong recall at 82.25%, indicating its effectiveness in identifying true positive cases. Precision is slightly lower at 52.15%, suggesting room for improvement in reducing false positives. The F1-score of 61.36% reflects a balanced trade-off between precision and recall, showcasing the model's reliability in scenarios where both metrics are critical.

Low accuracy in this context reflects the challenge of class imbalance rather than model inefficiency. Since the model is likely predicting the majority class more often, accuracy is skewed. However, metrics like recall (82.25%) and F1-score (61.36%) suggest that the model is effective in identifying key positive cases, making it more suitable for imbalanced classification tasks where accuracy alone isn't a reliable performance indicator.

Evaluation on External Violin Audio Samples

1 Evaluation Methodology

The violin audio clips used in the testing were sourced from YouTube, where specific search terms related to the desired class categories were used. Specific keywords like ”violin crescendo sound,” ”violin vibrato,” and ”violin bad pitch” were used to find relevant YouTube videos. The titles and descriptions of these videos were used as the basis for the initial classification, and the goal is for the model to later label each clip based on these keywords.

To evaluate the model's performance, these samples were preprocessed and passed through the trained model to predict corresponding labels. The pretrained AST model was initialized with the same architecture and hyperparameters used during training. Each audio sample was preprocessed into Mel spectrogram representations, ensuring consistency with the model's input requirements, and converted into PyTorch tensors. The preprocessed audio samples were fed into the model, which generated output logits. These logits were converted to probabilities using the sigmoid activation function, which outputs values between 0 and 1. A threshold of 0.5 was then applied: if the probability exceeded 0.5, the label was classified as present (1), and if it was below 0.5, the label was classified as absent (0). This thresholding process helps determine the final classification for each label. The model's predictions were then compared against a predefined set of the previously mentioned 20 possible classes to assess its performance. This evaluation provided insights into the model's practical applicability

and performance when dealing with new, unseen data.

To gain qualitative insights, the predicted labels were manually reviewed on the back end to assess their alignment with the audio characteristics. Audio playback functionality was incorporated to allow for auditory verification of the samples during analysis as shown below.

Figure 11: Screenshot of model testing with audio playback in Python

2 Visualizing Model Predictions with an Interactive Web Page

To provide an interactive way of analysing violin audio classification results, a Flask-based web application was developed. The application allows users to upload audio files, processes them using a pre-trained model, and dynamically displays classification results. This system enhances usability by providing real-time feedback, structured visualisation, and segmented audio analysis.

The back end is implemented in Flask and utilizes Flask-SocketIO for real-time updates. Key functionalities include:

- File Upload Handling: Users upload audio files, which are stored on the server for processing.
- Audio Playback: A built-in audio player allows playback control (play, pause, seek).
- Audio Segment and Processing

- Uploaded audio is divided into 1, 2, or 3-second chunks depending on its length.
- Each chunk is converted into a Mel spectrogram using Librosa and processed through the AST model developed.

- Prediction Generation and Live Updates
  - Model outputs multi-label prediction for each audio chunk.
  - Results are sent to the front end in real-time using WebSockets.

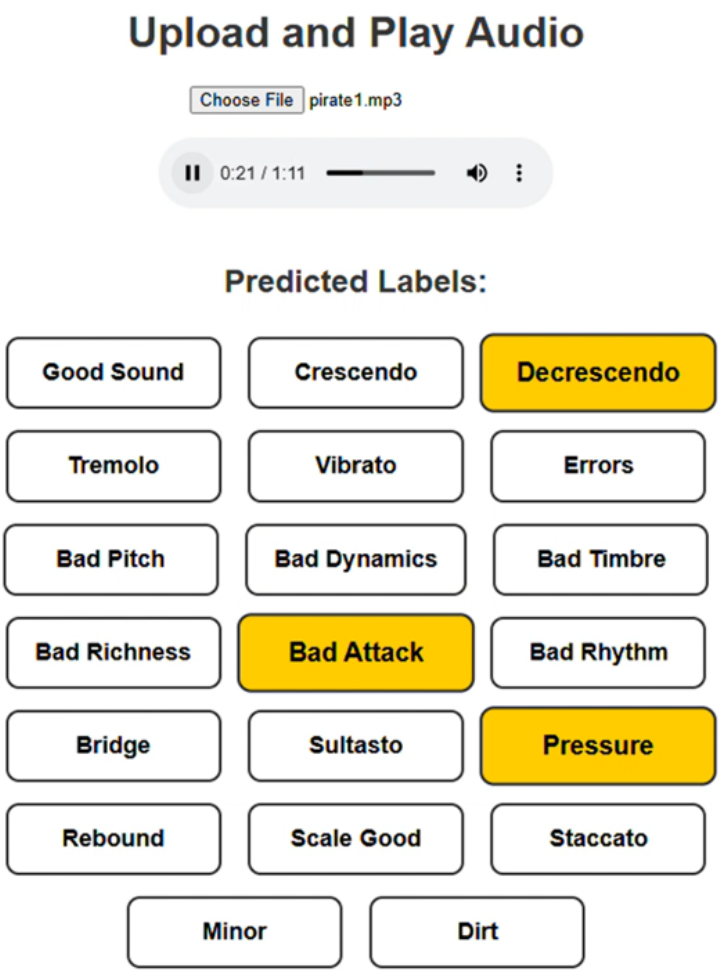

Figure 12: Model Prediction Visualization

As the audio plays, the model processes the recording and displays predicted labels corresponding to detected labels. These labels are displayed in a grid format, with each

box representing a possible classification where active labels (i.e., those predicted with high confidence for the current time segment) are highlighted in yellow, while inactive ones remain uncoloured.

This provides an interactive and intuitive way to showcase the model's functionality, making it easily understandable even for non-technical users.

Challenges and Future Considerations

1 Misleading Classifications Due to Broad Audio Labelling

Our initial testing shows that the model's results are generally logical but would require further training with a more robust dataset to improve accuracy. During testing, certain sounds clearly belonged to specific categories, yet the model failed to predict them correctly. This suggests that while the model has learned meaningful patterns, it still struggles with certain cases, likely due to limitations in data representation and training diversity.

One key limitation was the dataset quality and labelling of the audio. For example, a 5-second audio sample was labelled as 'bad-attack', yet only the first half-second exhibited the defining characteristic of this class, while the remaining audio was normal. As seen in Figure 13 below, the evenly spaced horizontal lines suggest a harmonic sound. However, the bright region at the beginning (left side) suggests an initial transient or attack phase which is what caused the audio to be labelled as 'bad-attack'.

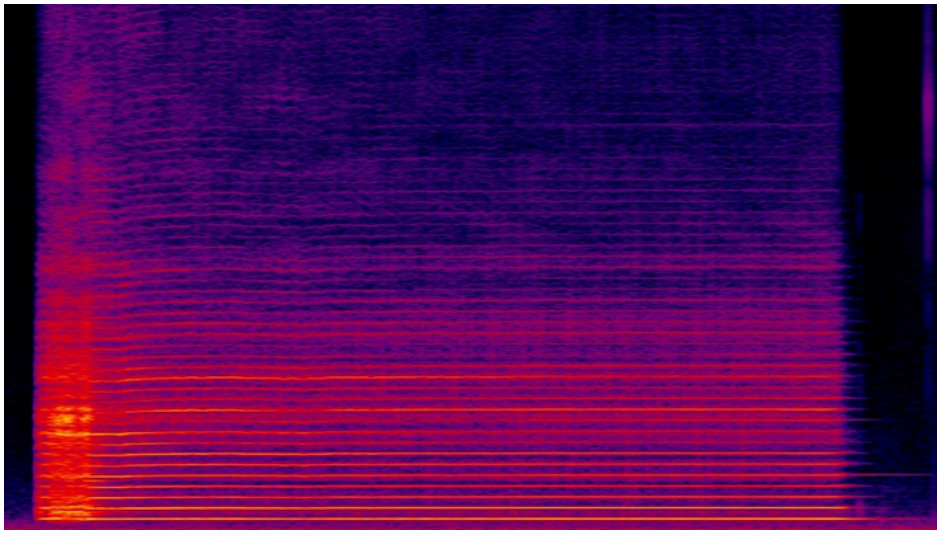

Figure 13: Spectrogram Image of the 5-second audio sample

Because the entire sample carried the bad-attack label, the model could learn misleading associations, incorrectly generalizing normal audio as part of the bad-attack category. This kind of labelling inconsistency can lead to misclassification, reducing the model's reliability, especially when dealing with real-world data. A more precise dataset with properly segmented and labelled audio would enhance the model's performance.

2 Live Audio Input

The initial plan was to use live audio as input, with the model predicting classes dynamically as the audio played in real-time, highlighting areas for improvement at the moment. This approach would have allowed for immediate feedback and real-time visualisation of detected classes. However, due to the complexity of implementation, time constraints and the model's limitations, this approach was not viable.

One of the main obstacles was the model's inference speed. When we tried implementing this real-time approach, our model simply couldn't keep up. The computational requirements for processing audio segments, generating Mel spectrograms, and making predictions were too heavy to maintain real-time performance. The model could not predict fast enough to provide results instantaneously, leading to noticeable delays. These delays would have disrupted the interactive experience, making real-time

classification impractical.

We had to make a practical compromise and work around this limitation. An alternative approach was adopted. Instead of processing the audio as it plays, our current system analyses the entire recording first and then displays the results during playback. This ensured smoother visualization and removed the need for real-time computation, albeit at the cost of losing immediate classification feedback. Live audio input remains a promising goal for future development. As processing power improves and algorithm optimization continues, we're confident that true real-time analysis will become possible, enabling instantaneous feedback and a fully interactive experience.

Conclusion

This project successfully developed an interactive music feedback system that analyses violin performances and provides actionable visual feedback. By leveraging an Audio Spectrogram Transformer (AST) model pre-trained on ImageNet and fine-tuned on a curated dataset of 1,400 violin audio samples, the system effectively identified performance issues such as poor pitch accuracy and improper bowing techniques. The model demonstrated a moderate overall performance, achieving an F1-score of 61.36%, with strong recall (82.25%) but lower precision (52.15%). While the accuracy of 23.1% was affected by class imbalance, the model's ability to identify positive cases reliably suggests its suitability for imbalanced classification tasks.

Although real-time audio classification was initially intended, technical constraints led to an alternative approach in which audio recordings were preprocessed before playback, ensuring smoother visualization. Future improvements will focus on optimizing inference speed to enable real-time predictions, enhancing model precision, and expanding the dataset to improve generalization.

Overall, this system bridges the gap between traditional music education and advanced deep learning techniques, providing violinists with valuable insights to refine their skills. With further refinements, it holds the potential to become an essential tool for musicians and educators, offering an interactive and data-driven approach to performance assessment.

Appendices

Appendix 1 - Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) works by processing input data (such as images) through multiple layers to automatically learn and detect features. They contain many layers with each learning to identify different aspects of an image. Through training, CNNs adjust the filters and weights to improve their ability to recognize complex patterns, making them especially effective for tasks like image classification, object detection, and more. They can also be quite effective for classifying audio, time series, and signal data [34].

Three of the most common layers are convolution, activation or ReLU, and pooling.

- **Convolution**: Applies a series of convolutional filters to the input images, each designed to extract specific features from the images.
- **Rectified Linear Unit (ReLU)**: Speeds up and enhances training by converting negative values to zero and keeping positive values, ensuring only activated features are passed to the next layer.
- **Pooling**: Simplifies the output by performing nonlinear downsampling, reducing the number of parameters that the network needs to learn.

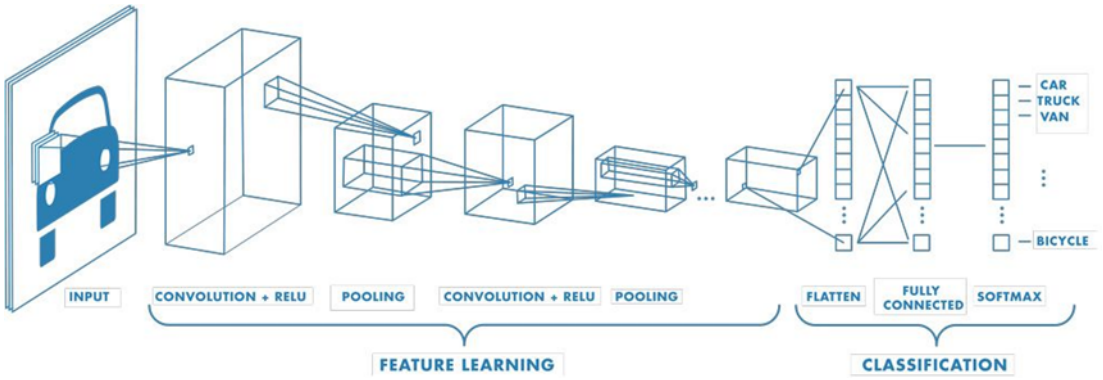

Figure 14: Example of a network with many convolutional layers. Filters are applied to each training image at different resolutions, and the output of each convolved image is used as the input to the next layer [34].

Appendix 2 – Libraries Used

Table 5: List of Python Libraries Used

No.	Library	Description
1	sqlite3	SQLite database interaction
2	pandas	Data manipulation and analysis
3	json	JSON data handling
4	torch	PyTorch deep learning framework
4.1	torch.optim	Optimization algorithms for PyTorch
4.2	torch.utils.data	Utilities for handling datasets in PyTorch
5	wget	File download utility
6	timm	Pretrained deep learning models for PyTorch
7	librosa	Audio processing and analysis
8	numpy	Numerical computing library
9	random	Random number generation
10	os	Operating system interface
11	sklearn.metrics	Performance metrics for machine learning

Appendix 3 - Classes

Below is a comprehensive list of the different ”klass” types used in the analysis:

Table 6: Classification of Sound Characteristics

Category	**Class Name**
Good Sound	good-sound
Dynamics	crescendo
	decrescendo
	tremolo
Pitch	vibrato
	bad-pitch
Errors	errors
Bad Characteristics	bad-dynamics
	bad-timbre
	bad-richness
	bad-attack
	bad-rhythm
Techniques	bridge
	sultasto
	pressure
	rebound
Scale Characteristics	scale-good
	staccato
	minor 2
	dirt

Bibliography

[1] Jia Deng et al. *ImageNet: A large-scale hierarchical image database*. IEEE, June 2009. DOI: 10.1109/cvpr.2009.5206848. URL: http://dx.doi.org/10.1109/CVPR.2009.5206848.

[2] John Hattie and Shirley Clarke. *Visible Learning: Feedback*. Routledge, Aug. 2018.

[3] Gary E. McPherson, Jennifer Blackwell, and John Hattie. "Feedback in Music Performance Teaching". In: *Frontiers in Psychology* 13 (June 2022). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2022.891025. URL: http://dx.doi.org/10.3389/fpsyg.2022.891025.

[4] Yuan Gong, Yu-An Chung, and James R. Glass. "AST: Audio Spectrogram Transformer". In: *CoRR* abs/2104.01778 (2021). arXiv: 2104.01778. URL: https://arxiv.org/abs/2104.01778.

[5] Yuan Gong. *AST Model Implementation in Python*. GitHub repository. 2024. URL: https://github.com/YuanGongND/ast/blob/master/src/models/ast_models.py.

[6] Sergio Giraldo et al. *Automatic Assessment of Tone Quality in Violin Music Performance*. Vol. 10. Frontiers Media SA, Mar. 2019. DOI: 10.3389/fpsyg.2019.00334. URL: http://dx.doi.org/10.3389/fpsyg.2019.00334.

[7] T. Knight and F. Upham. "The Potential for Automatic Assessment of Trumpet Tone Quality". In: (2011). URL: https://archives.ismir.net/ismir2011/paper/000102.pdf.

[8] Yichu Han. "Research on the application of personalized recommendation". In: *Applied and Computational Engineering* 43.1 (Feb. 2024), pp. 155–159.

ISSN: 2755-273X. DOI: 10.54254/2755-2721/43/20230825. URL: http://dx.doi.org/10.54254/2755-2721/43/20230825.

[9] Jing Jing. "Deep Learning-Based Music Quality Analysis Model". In: *Applied Bionics and Biomechanics* 2022 (June 2022). Ed. by Ye Liu, pp. 1–6. ISSN: 1176-2322. DOI: 10.1155/2022/6213115. URL: http://dx.doi.org/10.1155/2022/6213115.

[10] L. Chunlan. *Modern Electronic Technology*. SYNERGY PUBLISHING PTE. LTD, 2018.

[11] Siyu Ren et al. "Digitalization and energy: How does internet development affect China's energy consumption?" In: *Energy Economics* 98 (2021), p. 105220. ISSN: 0140-9883. DOI: https://doi.org/10.1016/j.eneco.2021.105220. URL: https://www.sciencedirect.com/science/article/pii/S0140988321001250.

[12] Weiyan Li. "Analysis of Piano Performance Characteristics by Deep Learning and Artificial Intelligence and Its Application in Piano Teaching". In: *Frontiers in Psychology* 12 (Jan. 2022). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2021.751406. URL: http://dx.doi.org/10.3389/fpsyg.2021.751406.

[13] Ethan Manilow et al. "Cutting Music Source Separation Some Slakh: A Dataset to Study the Impact of Training Data Quality and Quantity". In: *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2019.

[14] Maciej Blaszke and Bożena Kostek. "Musical Instrument Identification Using Deep Learning Approach". In: *Sensors* 22.8 (Apr. 2022), p. 3033. ISSN: 1424-8220. DOI: 10.3390/s22083033. URL: http://dx.doi.org/10.3390/s22083033.

[15] Keras Team. *Keras documentation: The Functional API*. 2019. URL: %5Curl%7Bhttps://keras.io/guides/functional_api/%7D.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Asso-

ciates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[17] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556. URL: https://arxiv.org/abs/1409.1556.

[18] Google. *AudioSet*. Accessed: 2025-03-15. 2017. URL: https://research.google.com/audioset.

[19] Lara Haidar-Ahmad. "Music and Instrument Classification using Deep Learning Technics". In: 2019. URL: https://api.semanticscholar.org/CorpusID:215780803.

[20] John M. Geringer and Clifford K. Madsen. "Musicians' Ratings of Good versus Bad Vocal and String Performances". In: *Journal of Research in Music Education* 46.4 (1998), pp. 522–534. DOI: 10.2307/3345348. eprint: https://doi.org/10.2307/3345348. URL: https://doi.org/10.2307/3345348.

[21] O. Romaní Picas et al. "A real-time system for measuring sound goodness in instrumental sounds". In: *AES 138th Convention*. Warsaw, Poland, May 2015. URL: http://hdl.handle.net/10230/32131.

[22] Keunwoo Choi, George Fazekas, and Mark B. Sandler. "Automatic tagging using deep convolutional neural networks". In: *CoRR* abs/1606.00298 (2016). arXiv: 1606.00298. URL: http://arxiv.org/abs/1606.00298.

[23] Daniel. *The MagnaTagATune Dataset – City University MIRG — mirg.city.ac.uk*. https://mirg.city.ac.uk/codeapps/the-magnatagatune-dataset. 2013.

[24] *GitHub - saturnMars/bigDataTechnologies: In this project, we suggest a simple Big Data system to process and analyse the Million Song Dataset (MSD) using Spark through the handy Databricks platform in its community Edition which is available for free. — github.com*. https://github.com/saturnMars/bigDataTechnologies. 2020.

[25] Dmitry Bogdanov et al. *MTG-Jamendo Dataset (1.0)*. Machine Learning for Music Discovery Workshop, International Conference on Machine Learning

(ML4MD, ICML2019), Long Beach, CA, United States. 2019. DOI: 10.5281/zenodo.3826813. URL: https://doi.org/10.5281/zenodo.3826813.

[26] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 233–240. ISBN: 1595933832. DOI: 10.1145/1143844.1143874. URL: https://doi.org/10.1145/1143844.1143874.

[27] Yann LeCun and Yoshua Bengio. "Convolutional networks for images, speech, and time series". In: *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258. ISBN: 0262511029.

[28] Jort F. Gemmeke et al. "Audio Set: An ontology and human-labeled dataset for audio events". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017, pp. 776–780. DOI: 10.1109/ICASSP.2017.7952261.

[29] Karol J. Piczak. "ESC: Dataset for Environmental Sound Classification". In: *Proceedings of the 23rd ACM International Conference on Multimedia*. MM '15. Brisbane, Australia: Association for Computing Machinery, 2015, pp. 1015–1018. ISBN: 9781450334594. DOI: 10.1145/2733373.2806390. URL: https://doi.org/10.1145/2733373.2806390.

[30] Pete Warden. "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". In: *CoRR* abs/1804.03209 (2018). arXiv: 1804.03209. URL: http://arxiv.org/abs/1804.03209.

[31] Alexey Dosovitskiy et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[32] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention". In: *CoRR* abs/2012.12877 (2020). arXiv: 2012.12877. URL: https://arxiv.org/abs/2012.12877.

[33] Oriol Romani Picas et al. *Good-sounds dataset*. Version 1.1. Zenodo, June 2017. DOI: 10.5281/zenodo.4588740. URL: https://doi.org/10.5281/zenodo.4588740.

[34] MathWorks. *What Is a Convolutional Neural Network?* Online; accessed March 15, 2025. 2024. URL: https://www.mathworks.com/discovery/convolutional-neural-network.html.