Scientific Research Publishing

Open Journal of Social Sciences, 2025, 13(9), 278-310
https://www.scirp.org/journal/jss
ISSN Online: 2327-5960
ISSN Print: 2327-5952

Multi-Instrument Detection in Polyphonic Music with Cultural Instruments

Sovathanak Meas, Rezza Moieni

Cultural Infusion Pty Ltd., Melbourne, Australia
Email: msovathanak@gmail.com, Rezza.M@Culturalinfusion.org.au

How to cite this paper: Meas, S., & Moieni, R. (2025). Multi-Instrument Detection in Polyphonic Music with Cultural Instruments. *Open Journal of Social Sciences, 13,* 278-310.
https://doi.org/10.4236/jss.2025.139017

Received: June 30, 2025
Accepted: September 19, 2025
Published: September 22, 2025

Abstract

The study adapts several machine-learning and deep-learning architectures to recognize 63 traditional instruments in weakly labelled, polyphonic audio synthesized from the proprietary Sound Infusion collection. Ten thousand 5s clips were algorithmically generated, features such as Mel-spectrograms, MFCCs, and VGGish embeddings were extracted, and six models were evaluated. The re-implemented Han et al. Convolutional Neural Network (CNN) attained the best result (micro F1 = 0.55; macro F1 = 0.50), approaching published performance on mainstream Instrument Recognition in Musical Audio Signals (IRMAS) data. Results highlight data scarcity and class imbalance as key obstacles for culturally diverse MIR.

Keywords

Convolutional Neural Network, Multi-Instrument Detection, Cultural Instruments, Deep Learning, Multi-Label Classification

1. Introduction

In recent years, music has become more accessible than ever for the 67.9% of the human population with internet access. With advances in technology, music can be digitalised and accessed on the internet without the need to visit the local store to purchase physical copies such as CDs or DVDs. However, this convenient access to music is not universal since roughly 32% of the world still does not have access to the internet (Destatis, 2025). But accessibility to music leads to ease of access to different types of music created from both well-known and cultural instruments that are unique to each culture and their respective history. Despite the growing popularity and excitement for cultural instruments, such as the Chinese hulusi, these instruments do not experience the same level of representation as more commonly used instruments in popular music. Even when the opportunity

arises to hear an under-represented instrument, it is often not as clearly distinguishable to listeners as common European instruments, such as the piano or violin, are. The ability to distinguish the sounds of many traditional instruments is diminishing as the number of practitioners dwindles (Vaiedelich & Fritz, 2017).

The human ear is capable of recognizing and identifying musical instruments in music, but the more instruments that form part of a composition, the more challenging this becomes. There is an abundance of research dedicated to tackling the problem of distinguishing musical instruments in musical compositions in the field of Musical Information Retrieval (MIR). However, most of the success is on single note or isolated note instrument recordings. In recent years, there has been an increase in the number of papers addressing the problem of multi-instrument detection, which all relied on datasets of better-known instruments, such as the piano or the guitar.

There are only a few available algorithms that can detect and identify musical instruments using a large dataset. Kailewang and Moieni (2022) proposed a method that can detect a single instrument with high precision when not combined with other instruments or vocals. Their objective was to introduce this innovative technology to the market, aiming to address an increasing demand for learning about such instruments while promoting and protecting cultural diversity.

Section 2 will discuss previous research achievements for single and 'predominant' instrument detection and the lack of multi-instrument detection for cultural and modern instruments (piano, violin, etc.). Model architecture will be described in Section 3, followed by data collection methods, data transformations, and data augmentation in Section 4. Section 5 will give an overview and discussion of each model's performance. The results will be analysed in Section 6, followed by areas of improvement in Section 7. This paper will conclude in Section 8.

2. History of Study

The growth of technology has been remarkable, with calculators that help solve difficult mathematical problems in a matter of seconds, computer vision software capable of recognizing patterns at a level that the human eye cannot see, and programs that can identify the title of a piece of music from a few seconds of audio. Yet, these innovations in audiovisual recognition software have overlooked instruments from diverse cultural backgrounds. Cultural instruments often mean a lack of data. Sometimes the practice of these instruments is lost in time and only scant recordings remain. This emphasizes the important role of instrument detection software that can recognize these instruments for the protection and promotion of cultural diversity.

Musical instruments all have many similarities, especially when we compare the guitar with the ukulele, or the cello with the violin. In the field of computer vision, there are countless research papers dedicated to developing object detection software that can identify and recognise musical instruments from images. Dewi,

Chen, and Christanto (2023) used the YOLO (You Only Look Once) models, such as YOLOv5 and YOLOv7, to identify musical instruments and achieved an average accuracy of around 85%. These are advanced object detection models that have been popularised in recent years due to their speed and accuracy. The YOLO models are Convolutional Neural Network (CNN) based models that process an entire image in a single pass, which makes them computationally efficient in object detection tasks (Kundu, 2023). Human eyes can identify and recognise similar objects just from a glance; however, this technology can be used to help aspiring musicians identify the differences between the instruments and learn, and it also benefits individuals with visual impairment.

Shazam is one of the most popular applications for music recognition. Its main function is to identify music in the user's environment while minimizing false positives. Shazam's database contains approximately 1.8 million songs and tracks, which it uses for music recognition (Wang, 2003). The performance of Shazam is dependent on the signal-to-noise ratio and the length of the audio sample, but it is able to achieve close to perfection when the two variables are optimal. The software is currently widely used by iPhone users; the Shazam app is integrated with the phone and can be activated at any time for music identification.

In the case of instrument detection, there are no real-world applications comparable to Shazam that can determine what instruments are present in an audio sample. Even so, MIR is a growing field of research. MIR involves processing audio content to understand or categorise music (Kaminskas & Ricci, 2012). It involves extracting features like individual notes from the instruments in music, which can then be used to identify music genres or analyse the styles of a particular artist. Numerous researchers in the past were mainly focused on musical instrument detection on isolated or solo recordings of a particular instrument.

Single instrument detection has achieved significant success, as evidenced by Li, Qian, and Wang (2015), where they began to incorporate deep learning models such as CNNs in place of traditional machine learning methods. CNNs are deep learning models that are especially good at processing and analyzing spatial patterns, such as images or audio spectrograms, like those that previous researchers have used in MIR. Success in identifying musical instruments using CNNs over traditional methods such as random forest or logistic regression has shown improvement across the board, except for precision. CNNs process an image by using filters to detect patterns in the image, such as an instrument's spectrogram, enabling the model to learn about abstract features that cannot be discerned by the human eye. Li, Qian, and Wang's (2015) CNN trained on audio was able to achieve a macro average F1 score of 0.6433 and an accuracy of 82.74%, in comparison to their traditional machine learning method, random forest, which was only able to achieve an F1 score of 0.4471 and 82.13% accuracy.

Traditional machine learning methods have performed exceptionally well, mainly the XGBoost model proposed by Liu et al. (2022), which addresses the single instrument detection problem and achieved a result of 97.65%, outperform-

ing every other classification model they tested, including support vector machine (SVM), logistic regression, etc. They achieved this by incorporating various features such as the root mean square energy, zero crossing rate, spectral bandwidth, harmonic signals, spectral rolloff, spectral centroid, and Mel-frequency cepstral coefficients (MFCC).

Recently, higher quality data has become more publicly available. Datasets that are curated for 'predominant' instrument detection' aim to detect the main instrument in a music recording, while 'multiple instruments detection' aims to detect all instruments in a particular recording. These popular datasets are used in conjunction with the ever more powerful deep learning models such as CNN and Recurrent Neural Network (RNN) for improved performance. A paper by Hing and Settle (2021) utilises the IRMAS dataset, where a CNN model architecture is constructed using the transfer learning technique, and they were able to achieve a high F1 score when classifying voice audio, 0.86, whereas the performance for classifying instruments was only between 0.47 and 0.69.

Another paper by Han et al. (2017) also addresses the problem of 'predominant' instrument detection; however, they constructed their CNN model to detect multiple 'predominant' instruments. This model proposed by Han et al. (2017) was regarded as the state-of-the-art model, able to achieve a result of 0.50 in macro and 0.65 in micro F1 score. This model was then used by multiple papers as a baseline for further improvements.

Reghunath and Rajan (2021) made use of the IRMAS dataset and found promising results with the use of CNN, RNN, and modified two CNN models with LSTM and GRU at the end of the architecture (C-LSTM and C-GRU). They achieved promising results with CRNN as it outperforms the standalone CNN and RNN architecture in 'predominant' instrument recognition. Their results show a significant improvement over that of the state-of-the-art model by Han et al. (2017), where they were able to achieve up to 9.09% in macro F1 score and a 7.81% improvement on the micro F1 score.

A recent paper by Zhong et al. (2023) tackled the 'predominant' instrument detection problem by approaching it from a different perspective. According to Zhong et al. (2023), their instrument recognition model contains a learnable front-end layer and a CNN-based feature extraction layer followed by a learnable pooling layer. Instead of the traditional method of providing the audio recording for feature extraction, Zhong et al. (2023) augmented their monophonic data and used it to pre-train their model, then finetuned it for 'predominant' instrument recognition with multi-instrument recordings. Zhong et al. (2023) achieved up to 0.674 in micro F1 score and 0.584 in macro F1 score. However, the focus of the authors' research was on datasets containing common instruments such as piano, violin, etc.

It is evident that single instrument detection and 'predominant' instrument detection problems have been addressed with models proposed by previous research with good performances. However, the difficulty remains for multi-instrument

detection. A paper by Lei (2022) addressed this problem by proposing a two-level classification model based on CNN, but instead of extracting features such as Mel spectrograms and MFCCs similar to previous papers, they use the "Constant Q Transform" (CQT) matrix as input. The first-level classification will use this feature as input, and its result will be a rough classification which will then be passed on to the second-level classification. The final classification level contains residual networks of the same architecture trained to identify instruments. Lei (2022) used a combination of well-annotated datasets which consists of Medley DB, MIXING SECRETS, and Bach10 to create variety in the data for training. As a result, they were able to achieve an average accuracy of 85% across all instruments.

Chen et al. (2024) addressed the multi-instrument detection problem by constructing a binary classifier for each instrument with a CNN configuration. Their binary classifier is trained to identify whether or not the given sample contains the specified instrument. Chen et al. (2024) utilised a One-vs-All (OvA) model by using CNN to extract features from the input spectrograms. Chen et al. (2024) model's performance was measured using accuracy, and their model was capable of achieving an accuracy of up to 53% with a 6-instrument recording dataset. For duo instrument accuracy, they achieved up to 77%, followed by 71% for trio instrument recordings, 62% for 4-instrument recordings, and 58% for 5-instrument recordings.

The success and attention that MIR has received in recent years in music instrument detection rest on its ability to detect well-known musical instruments like the piano, guitar, and violin. Cultural instruments are missing due to limited access around the world and insufficient data or lack of strong-labelled data; for instance, Lei (2022) utilizes available datasets to combine them into one dataset for optimal training and testing, which is evident based on the strong result, and these available datasets are not fully inclusive. While previous research certainly contains developed algorithms and applications such as Lei (2022) with their exceptional model with 85% average accuracy, and Chen et al. (2024) with their multi-instrument recognition model with capabilities to detect up to 6 instruments with above 50% accuracy, these algorithms and models were based on the datasets containing common instruments. This paper focused on musical instrument recognition for cultural instruments.

A paper by Kailewang and Moieni (2022) tackleded this problem by shifting the focus from common musical instruments to cultural musical instruments to support cultural diversity. They constructed a bidirectional RNN, a fully connected layer network with an attention layer, and a CNN with an attention layer and found very little success due to model overfitting.

According to Kailewang and Moieni (2022), when users encounter a cultural instrument that sparks their interest, they can record the sounds of the instrument and upload it to their website for identification. However, the authors' model was not able to perform when given the test set. The SVM model, which is the best model out of the two, resulted in an F1 score of 0.55. F1 is a measurement that

balances both precision and recall to determine the model's performance. A precision of 0.71 means 71% of the predicted instruments were correctly labelled, and a recall of 0.47 means the model missed 53% of the actual instruments that are present. Kailewang and Moieni (2022) deep learning models were overfitted due to the limited dataset as well as low variety in the audio samples of the instruments.

This paper will make use of precision, recall, and F1 score as the means to measure and evaluate the models. Given that there is practically no focus on cultural instruments, this paper aims to tackle this problem with a dataset similar to the one Kailewang and Moieni (2022) used in their paper.

3. Data

3.1. Data Collection

There is a clear lack of publicly available data for indigenous musical instruments around the world. The proposed method was trained on the Sound Infusion dataset, which consists of hundreds of different types of cultural instruments with samples of varying lengths. However, in this paper, we decided to only extract the same 63 unique instruments (a subset of the entire Sound Infusion dataset), excluding the vocals, that were used in the paper by Kailewang and Moieni (2022) to show a comparison to the authors' approach compared to ours. The songs that were compiled using the available instruments will also follow the same structure as that of Kailewang and Moieni (2022), where there were 2 to 5 instruments. We did not include vocals in the training dataset.

Instead of manually creating the music by playing the instrument together and recording the result with an external program, we generatedd the music more efficiently by using Python and its librosa library. To achieve this, we separatedd audio files for each of the instruments. Sound Infusion provided these, along with multiple variations of each instrument. Most instruments have between 5 and 15 unique recordings, though a few have only one, while others have more than 25. There were factors that could affect the strength of the model in detecting certain instruments, like those with few variations. Low variation createdd an issue with overfitting, where training data and testing data are the same. These instruments are Cameroon-DrumsetBikutsi, Cameroon-ShakerBikutsi, China-GongsTuned-SoftMallet, China-GongsTunedWoodMallet, and China-SmallErhuPlectrum. Hence, the performance of the model on these instruments is invalid (see **Table 1**).

3.2. Data Generation

Kailewang and Moieni (2022) manually generated datasets by recording each song manually using Sound Infusion's individual instrument dataset, which represents approximately 4300 sample songs. This is an inefficient and time-consuming process, so instead of manually recording the songs, we created a simple algorithm using Python to automate the song creation and data labelling process. Our audio

Table 1. Instrument recording train and test split.

	Training	Testing
Armenia-Duduk	3	1
Bali-Gamelan Ensemble	7	2
Bolivia-Charango	7	3
Bolivia-Moseno	15	5
Bolivia-Roncoro Chords	4	1
Bolivia-Roncoro Solo	4	1
Brazil-Afuche Cabasa	4	1
Brazil-Agogo	7	2
Brazil-Bass Guitar Bossa	4	2
Brazil-Berimbau	4	1
Brazil-Bongos	4	1
Brazil-BongosCowbell	4	1
Brazil-Cabasa	3	1
Brazil-Claves	6	2
Brazil-Cuica	4	2
Brazil-EggShaker	5	2
Brazil-Guitar	4	1
Brazil-Pandeiro	4	1
Brazil-PercussionSet	6	2
Brazil-RainStick	4	1
Brazil-Surdo	7	3
BurkinaFaso-BaraDrum	8	2
BurkinaFaso-BassLine	22	8
Cameroon-Congas	1	1
Cameroon-Djembe	1	1
Cameroon-DrumsetBikutsi	1	same as train
Cameroon-PercussionSetBikutsi	1	1
Cameroon-ShakerBikutsi	1	same as train
China-Bawu	4	2
China-BeijingOperaGongs	4	1
China-BianzhongBells	8	2
China-BigErhuPlectrum	6	2
China-CeylonGuitar	6	2
China-ChauGongs	3	1
China-Dizi	4	1
China-Dongxiao	4	1

Continued

China-Erhu	7	2
China-FengGong	5	2
China-Gaohu	8	2
China-GongsTunedMetalMallet	1	1
China-GongsTunedSoftMallet	1	same as train
China-GongsTunedWoodmallet	1	same as train
China-Hulusi	4	2
China-JinghuOperaViolin	4	1
China-KouXian	4	1
China-Pipa	4	2
China-ShanghaiBabyPiano	8	2
China-Sheng	4	2
China-SmallErhuPlectrum	1	same as train
China-WuhanTamTam	3	1
China-Xiao	6	2
China-YangQin	4	1
Congo-Bongos	8	2
Congo-Sanzas	8	2
Cuba-Guiro	3	1
Cuba-Triangle	2	1
Egypt-Fiddle	5	2
Germany-CrumhornAlto	4	2
Germany-CrumhornBass	4	2
Germany-CrumhornConsortium	4	2
Germany-CrumhornSoprano	4	2
Germany-CrumhornTenor	4	2
Germany-Gemshorn	5	2

files were recorded in mp3 format with a Constant Bit Rate and a sampling rate of 48,000 Hz.

```
FUNCTION create_folders(folder_names, location):
    FOR each name in folder_names:
        CREATE directory at (location + '/' + name) IF NOT EXISTS
    END FOR
RETURN
```

Figure 1. Pseudocode to create folders.

```
FUNCTION unique_generator(num_songs): SET songs = Empty List
    WHILE LENGTH(songs) < num_songs:
        SET instrument_amt = Random number between 2 and 5
        IF instrument_amt < 3:
            SET sel_type = Randomly select an instrument type
            SET sel_instr = Randomly select instrument_amt instruments from sel_type
            SET selected_instruments_list = List(sel_instr)
        ELSE:
            SET amt = instrument_amt

            WHILE amt > 0:
                SELECT one instrument from each category: aerophone, chordophone, idiophone, membranophone

                IF instrument_amt == 5:
                    SET rand_type = Randomly select an instrument type
                    SET extra_instr = Randomly select one instrument from rand_type
                    ADD extra_instr to selected_instruments_list
                END IF
                amt = amt - 1
            END WHILE
        END IF
        WHILE LENGTH(selected_instruments_list) < 5:
            ADD 'NA' to selected_instruments_list
        END WHILE

        IF selected_instruments_list NOT IN songs:
            APPEND selected_instruments_list TO songs
        END IF
    END WHILE
CONVERT songs TO DataFrame
ASSIGN column names ['instrument1', 'instrument2', 'instrument3', 'instrument4', 'instrument5']
EXPORT DataFrame TO CSV
RETURN DataFrame
```

Figure 2. Pseudocode to generate a CSV file with random song combinations.

We separated the musical instruments into 4 categories: Membranophone, Chordophone, Aerophone, and Idiophone, as Kailewang and Moieni (2022) did. We created folders for each of the instruments using the code in **Figure 1** (above) for the training instruments and testing instruments, to solve the overfitting problem Kailewang and Moieni (2022) experienced with their CNN and RNN models. We then populated the folders with all the sample recordings of their respective instruments from Sound Infusion's dataset. For the folders with multiple sample recordings, we reserved approximately 80% of the samples for training, while using the remainder as input for testing. For instance, instruments with 5 samples are split such that 4 samples are for training and 1 for testing. **Table 1** shows a clear representation of our data split strategy; we prioritized including more training data than testing for the purpose of scaling this technology in the future. As mentioned previously, we clearly see that a few of the instrument's recordings (Cameroon-DrumsetBikutsi, Cameroon-ShakerBikutsi, China-GongsTunedSoftMallet, China-GongsTunedWoodMallet, and China-SmallErhuPlectrum) were used in both the training and testing data.

Figure 2 shows the unique generator function, which is responsible for generating instrument combinations and exporting these combinations into a Comma Separated Values (CSV) file. This function uses the random library in Python to decide the number of instruments present and which instrument name to include (based on the Membranophone, Chordophone, Aerophone, and Idiophone). The function generates a certain number of songs based on user input. For this paper, we specified the function to create a total of 10,000 unique instrument combinations from both training and testing samples and exported them into a CSV file. The data generation process involved two steps: 1) to generate 8,000 random audio samples for training data; and 2) to generate 2,000 samples for testing data.

The function starts by randomly picking a number between 2 and 5. This number decides how many instruments will be included in the song. If the number of instruments (let's call this n) is 3 or fewer, the function will randomly select n instruments to include in the song. However, when the function detects that n equals 4, it will pick one from each of the 4 categories to add the name of the instrument to the song. If n is 5, it will pick one instrument from each of the 4 categories, then add one random instrument. The CSV file has 5 columns to indicate the 5 instruments present in each song (each row) and for songs with fewer than 5 instruments, the missing instruments are replaced with "NA."

```
FUNCTION data_generator(song_lst):
    instrument_lst = Extract instrument names from song_lst
    chosen_instrument = Randomly audio sample from '/instrument/path/' + first instrument name in instrument_lst
    song = Load audio from chosen_instrument
    FOR instrument in (1, LENGTH(instrument_lst)):
        IF instrument IS NOT 'NA':
            chosen_instrument = Randomly select file from '/instrument path/' + instrument
            audio = Load audio from chosen_instrument
        IF audio.duration > song.duration:
            song = Overlay audio on song
        ELSE:
            song = Overlay song on audio
        END IF
    END IF
END FOR
EXPORT song AS MP3 to 'path/to/save' with filename based on instrument_lst
```

Figure 3. Pseudocode to create the dataset.

Once the song combination CSV file and instrument folders are created, the CSV file is passed to the data generator function in Figure 3, which reads each row to determine the instruments used in each song. The function then locates the corresponding song folder and overlays the selected instruments using the PyDub library to generate the final track. It exports the completed composition as an mp3 file to the specified path. Each exported file is named to reflect the instruments used in the song, with instrument names separated by underscores. For example, a song featuring only the Armenian Duduk and Chinese Bawu will be named "Armenia-Duduk_China-Bawu_NA_NA_NA.mp3."

```
FUNCTION data_labelling(path):
    FOR each song IN list of files in path:
        SET instrument_name = ""
        Create textfile to write at "/label/path" + song + ".txt"
        FOR i FROM 0 TO LENGTH(song) - 1:
            IF song[i] IS NOT "_":
                APPEND song[i] TO instrument_name
            ELSE:
                IF instrument_name IS "NA":
                    SET instrument_name = ""
                    CONTINUE
                WRITE instrument_name TO file f
                SET instrument_name = ""
    END FOR
END FUNCTION
```

Figure 4. Pseudocode for data labeling function.

This naming convention is essential as it simplifies the labeling process for the dataset. The data labeling function in **Figure 4** processes the folder containing the exported dataset by reading the filenames, extracting the instrument names (excluding the "NA" entries), and saving them into a text file. This labeling process is similar to that used in the OpenMIC-2018 dataset (Humphrey et al., 2018); however, our dataset is weakly labeled, meaning it does not include timestamps for the instrument appearances.

3.3. Data Preprocessing

We extracted various audio features from the files to train our models, including the Mel Spectrogram, Mel MFCCs, VGGish embeddings, Spectral Bandwidth, Spectral Rolloff, Spectral Centroid, Harmonic signals, Root Mean Square (RMS) energy, and Zero Crossing Rate. We used Python's librosa library to extract the musical features from all 10,000 audio files. Our recordings for all the songs were of varying lengths, from 5 seconds to 15 seconds, so we trimmed all the audio in both the training and testing data to limit the feature extraction to just 5 seconds of audio when all instruments are playing. We only trimmed the audio to 5 seconds when we extracted the Mel spectrogram, VGGish embeddings, and MFCCs. For the machine learning methods, we extracted audio features and used the mean and variance of each as input to the model. This approach allowed the model to incorporate a wider range of features while minimising data complexity. However, it introduced limitations to the model, as summarising features in this way leads to a loss of temporal information: the model can only rely on mean and variance to differentiate between the instruments. The resulting input for machine learning methods is a 1×52 dimensional feature vector.

3.3.1. Mel Spectrogram

The Mel spectrogram was a widely used feature in previous studies, including Reghunath and Rajeev (2021), Mukhedkar (2020), and Han et al. (2017). For our dataset, we've set the FFT window size (*n_fft*) to 1024, the hop length to 512, and the number of Mel bands (*n_mel*) to 128, using a Hann window function. Additionally, to better replicate human auditory perception—which is logarithmic—we apply a logarithmic scale to the Mel spectrogram, following the approach of Kailewang and Moieni (2022). The resulting output dimension for each audio sample is 128×216.

3.3.2. Mel Frequency Cepstral Coefficients (MFCCs)

In addition to the Mel spectrogram, we also extracted MFCCs from the data. We followed the same feature extraction parameters used by Kailewang and Moieni (2022), setting the frame length to 2048, hop length to 160, and using a Hann Window. To compute the frequency spectrum, we've set *n_fft* to 512 and use 40 Mel filters. For each audio sample, we extracted 24 MFCCs. However, unlike Kailewang and Moieni (2022), who used only VGGish embeddings, our deep learning models took both MFCCs and Mel spectrograms as input. The MFCC feature has a shape of 24×2756 for each audio.

In addition, we extracted a second form of MFCC that follows the same feature extraction process by Liu et al. (2022), where the MFCCs are compressed by calculating the mean and variance of each coefficient. For this extraction, we set the number of MFCC (*n_mfcc*) to 20, while keeping all other parameters at their default values in the librosa library. We then used these compressed features as input for the traditional machine learning methods.

3.3.3. VGGish

We also adopted an approach similar to that used in the OpenMIC-2018 dataset (Humphrey et al., 2018) by leveraging resources from the developers of AudioSet and VGGish. First, we trimmed the generated audio files to 5 seconds and transformed this raw audio into a 128-dimensional vector every 0.96 seconds using a fixed window size. To account for variations in audio length, we standardized each audio sample to 5 seconds, resulting in a 5×128 matrix of VGGish embeddings. This method was applied in previous research on polyphonic music instrument detection by Mukhedkar (2020), which reported promising results on datasets of well-represented instruments such as MUSDB18.

3.3.4. Spectral Bandwidth

We also extracted spectral bandwidth, defined by Kailewang and Moieni (2022) as the weighted average frequency of the signal at each frame. This feature was extracted using Python, and its mean and variance were used as input for the traditional machine learning methods.

3.3.5. Spectral Rolloff

The spectral rolloff was set to 85%, following the approach of Kailewang and Moieni (2022) and Liu et al. (2022). We extracted this feature and use its mean and variance as input. Additionally, we adopted Liu et al.'s (2022) method of separating audio signals into harmonic and percussive components. In this paper, we focused only on the harmonic signals, such as those produced by the Armenian Duduk, since our dataset contains a limited variety of percussive instruments. This separation was performed using the Harmonic-Percussive Source Separation (HPSS) algorithm, the preprocessing technique used by Liu et al. (2022). We then computed the mean and variance of the harmonic component for use in our machine learning methods.

3.3.6. Spectral Centroid

The spectral centroid was extracted from the raw audio files. Put simply, this feature reflects the brightness of the timbre. Timbre, as described by Liu et al. (2022), refers to the weighted average frequency based on the energy within a given frequency range. For instance, the timbre varies depending on whether the audio contains more high or low frequencies.

3.3.7. Zero Crossing Rate

The zero crossing rate refers to the number of times a particular waveform of an

audio dips below or above zero.

3.3.8. Root Mean Square (RMS) Energy

Lastly, we extracted the Root Mean Square (RMS) energy, a commonly used measure for calculating the amplitude envelope of an audio signal. We included this feature as part of our input data, as it was also used by Liu et al. (2022) in their XGBoost model.

4. Model Architecture

4.1. Machine Learning Methods

4.1.1. Random Forest

This is our baseline random forest model implemented using traditional machine learning methods. Because random forest alone cannot handle multi-label data, we wrap it with scikit-learn's MultiOutputClassifier, enabling multi-label classification. This model served as a benchmark for comparison with the other machine learning method, XGBoost. We've set the number of estimators to 100 and the maximum tree depth to 6.

4.1.2. XGBoost

As noted by Liu et al. (2022), this model performs exceptionally well for detecting single instruments in audio. In our work, we adapted it to handle multi-label data for multi-instrument detection. However, like the random forest model, XGBoost cannot natively handle multi-label classification. Therefore, we wrapped the XGBoost model with scikit-learn's MultiOutputClassifier to enable multi-label support, similar to our approach with the random forest model. We used the same training parameters as Liu et al. (2022) with 100 estimators, a learning rate of 0.05, a maximum depth of 6, a subsample ratio of 0.8, and a minimum child weight of 1. Additionally, we used 'logloss' as the evaluation metric.

4.2. Deep Learning Methods

Unlike traditional machine learning methods, modern deep learning methods require additional considerations such as batch size, optimiser, learning rate, loss function, and the number of epochs. For our training, we set the batch size to 64. We used the Adam optimiser, with a learning rate of 0.001, following Han et al. (2017). However, unlike Han et al., who used the categorical cross-entropy loss function, we used binary cross-entropy loss, since this is a multi-label problem. Training loss will be used to gauge when to stop the training. We set the training to halt after 5 epochs when training loss does not improve.

4.2.1. bi-LSTM

The bi-LSTM is our baseline model among deep learning methods. This model is based on the architecture from Mukhedkar (2020) (shown in **Figure 5**). However, the architecture was reconfigured to handle multi-label input with 63 unique classes by using a sigmoid activation function for the output layer. Our recorded songs

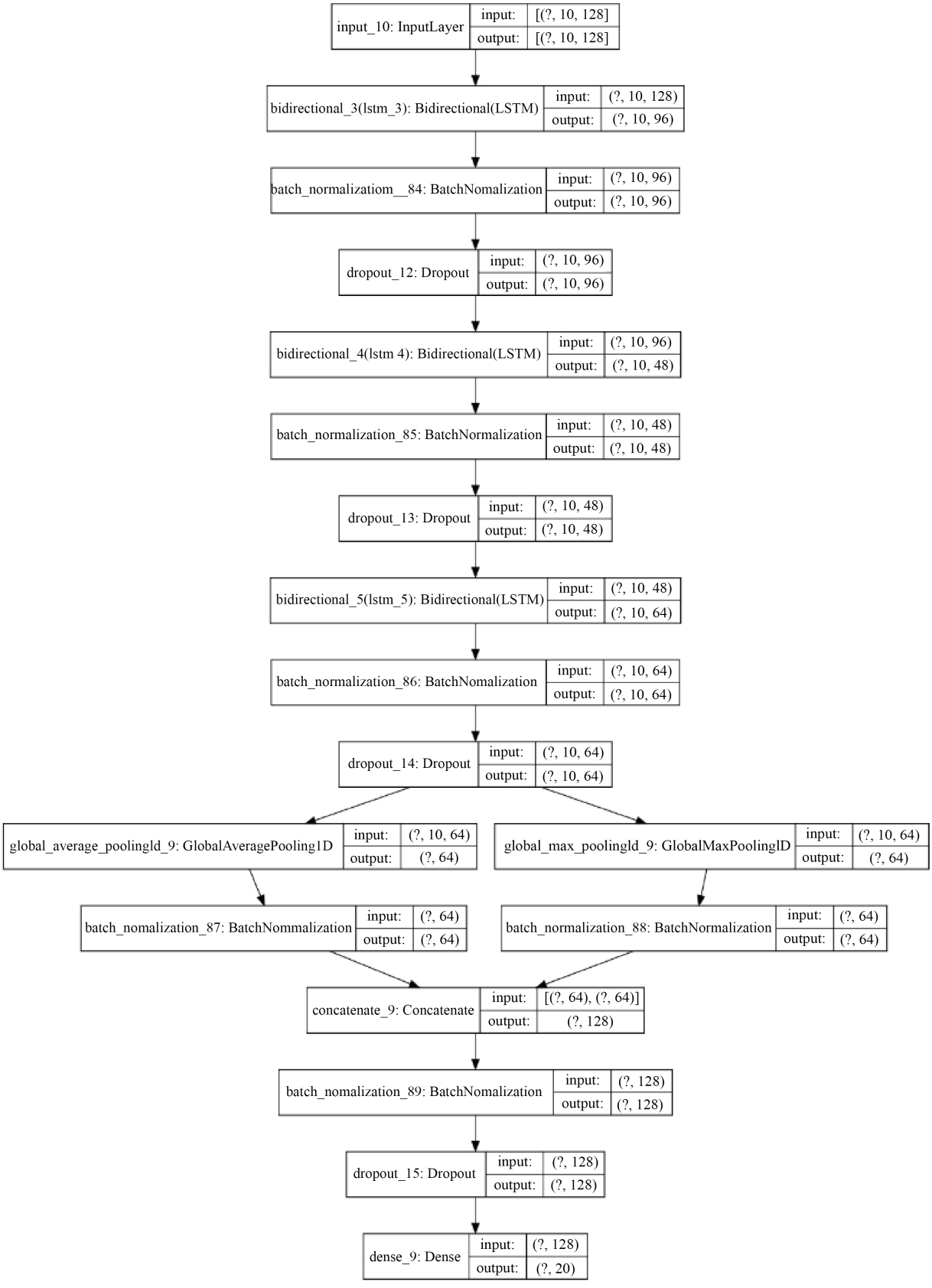

Figure 5. bi-LSTM model architecture (Mukhedkar, 2020).

are considered as time series data, which can also be approached as a standard sequence learning problem, making RNN models a natural choice (Mukhedkar, 2020). Accordingly, we used the "Long Short-Term Memory" (LSTM) architecture as applied by Mukhedkar (2020). The model takes extracted VGGish embeddings as input. However, the main drawback of the RNN model is its inability to capture frequency domain invariances. The training parameters are shown in **Table 2**.

Table 2. Deep learning models' hyper-parameters.

Models	Epochs	Batch size	Early-stop patience (training loss)	Learning rate	Optimiser	Loss function
CRNN	100	64	3	0.005	Adam	Binary Cross Entropy
Han's CNN	100	64	3	0.005	Adam	Binary Cross Entropy
bi-LSTM	100	64	3	0.005	Adam	Binary Cross Entropy

4.2.2. C-RNN

Reghunath and Rajan (2021) designed a model architecture that combines the benefits of CNNs and RNNs for 'predominant' instrument detection. The proposed CRNN architecture is shown in **Figure 6**. We adapted this model for our multi-label dataset, and we replaced the softmax activation function at the output layer with a sigmoid function. According to Reghunath and Rajan (2021), CNNs are unable to retain temporal context information, whereas RNNs can. Therefore, the model architecture includes convolutional layers followed by two bidirectional LSTM units. Training parameters are shown in **Table 2**.

*	Mel-spectrogram-CNN	Modgdgram-CNN	*	Mel-spectrogram-CRNN	Modgdgram-CRNN
×4	2 × Conv2D (3×3), d_i	Conv2D (3×3), f_i	×3	2 × Conv2D (3×3), d_i	Conv2D (3×3), f_i
	Leaky ReLU (α = 0.33)	ReLU		Leaky ReLU (α = 0.33)	ReLU
	3×3 Max-pooling,stride (3,3)			Batch Normalization	
	Dropout (0.25)			2×2 Max-pooling,stride(2,2)	
	Global Max-pooling			Flatten (1024)	
	Dense (1024)	Dense (512)		2 × Bidirectional LSTM / GRU (32 units)	
	Dropout (0.5)			Flatten (1024)	
	Dense (11), Softmax Activation			Dense (512)	
				Batch Normalization, Dropout (0.5)	
				Dense (11), Softmax Activation	

Figure 6. C-RNN model architecture by Reghunath and Rajan (2021).

4.2.3. Han's CNN

We also usedd the state-of-the-art CNN model developed by Han et al. (2017).

Their architecture was specifically designed to handle multi-class and multi-label data; however, their input dataset contained 11 unique instruments, whereas ours has 63. Therefore, we only modified the number of output classes for the sigmoid activation function. Additionally, we used basic ReLU as opposed to Leaky ReLU since our dataset differs slightly from that of the IRMAS. A previous study by Xu et al. (2015) has shown instances where ReLU outperforms Leaky ReLU depending on the dataset. Although Han et al.'s (2017) model was primarily developed for 'predominant' instrument detection, since it is capable of identifying multiple well-known musical instruments in an audio file, we can evaluate its performance in detecting all instruments in the audio file. The original model architecture is shown in **Figure 7**. The training parameters for this model are shown in **Table 2**.

Input size	Description
1 × 43 × 128	mel-spectrogram
32 × 45 × 130	3 × 3 convolution, 32 filters
32 × 47 × 132	3 × 3 convolution, 32 filters
32 × 15 × 44	3 × 3 max-pooling
32 × 15 × 44	dropout (0.25)
64 × 17 × 46	3 × 3 convolution, 64 filters
64 × 19 × 48	3 × 3 convolution, 64 filters
64 × 6 × 16	3 × 3 max-pooling
64 × 6 × 16	dropout (0.25)
128 × 8 × 18	3 × 3 convolution, 128 filters
128 × 10 × 20	3 × 3 convolution, 128 filters
128 × 3 × 6	3 × 3 max-pooling
128 × 3 × 6	dropout (0.25)
256 × 5 × 8	3 × 3 convolution, 256 filters
256 × 7 × 10	3 × 3 convolution, 256 filters
256 × 1 × 1	global max-pooling
1024	flattened and fully connected
1024	dropout (0.50)
11	sigmoid

Figure 7. Han et al. (2017) Model architecture.

5. Results and Discussion

For all our deep learning models, the sigmoid output threshold for a particular instrument to be present in a song is when it is greater than or equal to 0.5. For instance, if the Armenian Duduk's sigmoid output is 0.5, then the instrument is considered to be present in the song, and vice versa if it is less than 0.5. For our results, we focused on the micro and macro average of the F1 scores, recall, and precision as in previous research papers such as Han et al. (2017). These metrics were calculated using Scikit Learn's library "*classification_report*", and we used the Pandas library to format and export the results into CSV files.

The random forest model performance, measured by micro and macro F1 scores, was only 0.225 and 0.19, respectively. Although the model achieved a high micro-average precision of 0.946, further observation revealed that the model as-

sumed many of the instruments were present in almost every audio file, as the micro average of recall was only 0.128. The F1 score, precision, and recall at a class level (see Table 3), clearly reveal that the random forest model assumes most of the instruments are present in the audio files, since almost all of them have a precision score of 1, while many of the instruments' recall is close to 0. For example, the Brazil rainstick has a precision score of 1 but a recall of only 0.015, which shows that the model assumed that the rainstick was present in many audio samples when it was not. Some instruments were completely undetected; for instance, German-Gemshorn's precision, recall, and F1 score were all 0. Many other instruments suffered similar issues, as shown in Table 3, for instance the China-Xiao.

Table 3. Random forest performance.

	precision	recall	f1-score	support
Armenia-Duduk	0.75	0.069	0.126	131
Bali-Gamelan Ensemble	0	0	0	115
Bolivia-Charango	1	0.118	0.211	119
Bolivia-Moseno	1	0.223	0.365	130
Bolivia-Roncoro Chords	1	0.016	0.032	122
Bolivia-Roncoro Solo	0	0	0	132
Brazil-Afuche Cabasa	0.625	0.094	0.164	106
Brazil-Agogo	0.667	0.017	0.034	115
Brazil-Bass Guitar Bossa	1	0.259	0.412	108
Brazil-Berimbau	0.333	0.008	0.016	119
Brazil-Bongos	0	0	0	104
Brazil-BongosCowbell	0	0	0	86
Brazil-Cabasa	1	0.129	0.228	101
Brazil-Claves	1	0.036	0.07	111
Brazil-Cuica	0	0	0	94
Brazil-EggShaker	1	0.039	0.075	103
Brazil-Guitar	1	0.351	0.52	94
Brazil-Pandeiro	1	0.01	0.02	101
Brazil-PercussionSet	0	0	0	91
Brazil-RainStick	1	0.015	0.03	130
Brazil-Surdo	0.989	0.701	0.82	127
BurkinaFaso-BaraDrum	0	0	0	89
BurkinaFaso-BassLine	1	0.061	0.115	115
Cameroon-Congas	0	0	0	96
Cameroon-Djembe	0	0	0	93
Cameroon-DrumsetBikutsi	0.936	0.427	0.587	103
Cameroon-PercussionSetBikutsi	0	0	0	106

Continued

Cameroon-ShakerBikutsi	1	0.021	0.041	95
China-Bawu	0.889	0.058	0.11	137
China-BeijingOperaGongs	1	0.521	0.685	121
China-BianzhongBells	1	0.286	0.444	105
China-BigErhuPlectrum	1	0.505	0.671	103
China-CeylonGuitar	0	0	0	116
China-ChauGongs	0.348	0.075	0.124	106
China-Dizi	0.75	0.18	0.291	133
China-Dongxiao	0	0	0	117
China-Erhu	0.868	0.4	0.548	115
China-FengGong	1	0.02	0.039	100
China-Gaohu	0.981	0.525	0.684	101
China-GongsTunedMetalMallet	1	0.265	0.42	113
China-GongsTunedSoftMallet	1	0.409	0.58	115
China-GongsTunedWoodmallet	1	0.458	0.629	96
China-Hulusi	1	0.015	0.029	137
China-JinghuOperaViolin	0	0	0	121
China-KouXian	1	0.12	0.214	125
China-Pipa	1	0.008	0.017	119
China-ShanghaiBabyPiano	1	0.008	0.015	130
China-Sheng	0	0	0	133
China-SmallErhuPlectrum	1	0.531	0.694	113
China-WuhanTamTam	1	0.01	0.019	105
China-Xiao	0	0	0	132
China-YangQin	1	0.042	0.081	143
Congo-Bongos	1	0.163	0.28	92
Congo-Sanzas	0	0	0	123
Cuba-Guiro	1	0.017	0.033	121
Cuba-Triangle	1	0.026	0.051	114
Egypt-Fiddle	0.968	0.536	0.69	112
Germany-CrumhornAlto	1	0.062	0.116	130
Germany-CrumhornBass	1	0.113	0.203	115
Germany-CrumhornConsortium	1	0.177	0.301	113
Germany-CrumhornSoprano	1	0.051	0.098	117
Germany-CrumhornTenor	0.667	0.015	0.03	132
Germany-Gemshorn	0	0	0	124
micro avg	**0.946**	**0.128**	**0.225**	7165
macro avg	**0.679**	**0.13**	**0.19**	7165
weighted avg	**0.686**	**0.128**	**0.187**	7165
samples avg	**0.427**	**0.151**	**0.214**	7165

The final machine learning method we used was the XGBoost model, which performed exceptionally well in the single instrument recognition tasks according to Liu et al. (2022). We adapted this model for multi-label classification in the same way as the random forest model and found some interesting results. The micro and macro F1 scores were 0.426 and 0.363, respectively, showing a significant improvement over the random forest model. However, XGBoost exhibited a similar issue with a high precision but very low recall (see Table 4). The XGBoost model achieved a micro precision of 0.829 with a micro recall of only 0.286 and a macro precision of 0.751 with a macro recall of only 0.287. The results were much better than the random forest model; however, it still assumed that most of the audio samples include instruments that were not actually present. This issue is particularly evident for individual instruments such as Brazil-Bongos, Brazil-Pandeiro, China-Bawu, China-FengGong, Cuba-Triangle, and Congo-Sanzas.

Table 4. XGBoost performance.

	precision	recall	f1-score	support
Armenia-Duduk	0.583	0.107	0.181	131
Bali-Gamelan Ensemble	0.967	0.252	0.4	115
Bolivia-Charango	0.933	0.471	0.626	119
Bolivia-Moseno	0.837	0.554	0.667	130
Bolivia-Roncoro Chords	0.528	0.385	0.445	122
Bolivia-Roncoro Solo	0.6	0.023	0.044	132
Brazil-Afuche Cabasa	0.421	0.075	0.128	106
Brazil-Agogo	0.667	0.017	0.034	115
Brazil-Bass Guitar Bossa	0.8	0.481	0.601	108
Brazil-Berimbau	0.25	0.008	0.016	119
Brazil-Bongos	1	0.029	0.056	104
Brazil-BongosCowbell	0	0	0	86
Brazil-Cabasa	0.692	0.178	0.283	101
Brazil-Claves	1	0.144	0.252	111
Brazil-Cuica	0.875	0.074	0.137	94
Brazil-EggShaker	0.5	0.097	0.163	103
Brazil-Guitar	0.923	0.638	0.755	94
Brazil-Pandeiro	0.9	0.089	0.162	101
Brazil-PercussionSet	0	0	0	91
Brazil-RainStick	0.75	0.023	0.045	130
Brazil-Surdo	0.953	0.961	0.957	127
BurkinaFaso-BaraDrum	1	0.067	0.126	89
BurkinaFaso-BassLine	0.857	0.104	0.186	115

Continued

Cameroon-Congas	0.25	0.01	0.02	96
Cameroon-Djembe	1	0.097	0.176	93
Cameroon-DrumsetBikutsi	0.784	0.738	0.76	103
Cameroon-PercussionSetBikutsi	1	0.028	0.055	106
Cameroon-ShakerBikutsi	1	0.095	0.173	95
China-Bawu	0.929	0.19	0.315	137
China-BeijingOperaGongs	0.965	0.917	0.941	121
China-BianzhongBells	0.971	0.324	0.486	105
China-BigErhuPlectrum	0.988	0.786	0.876	103
China-CeylonGuitar	0.24	0.103	0.145	116
China-ChauGongs	0.074	0.019	0.03	106
China-Dizi	0.582	0.429	0.494	133
China-Dongxiao	0.961	0.624	0.756	117
China-Erhu	0.775	0.809	0.791	115
China-FengGong	0.939	0.31	0.466	100
China-Gaohu	0.927	0.752	0.831	101
China-GongsTunedMetalMallet	1	0.558	0.716	113
China-GongsTunedSoftMallet	1	0.626	0.77	115
China-GongsTunedWoodmallet	1	0.667	0.8	96
China-Hulusi	0.667	0.102	0.177	137
China-JinghuOperaViolin	0.944	0.14	0.245	121
China-KouXian	0.96	0.192	0.32	125
China-Pipa	0.714	0.042	0.079	119
China-ShanghaiBabyPiano	0.375	0.046	0.082	130
China-Sheng	0.696	0.12	0.205	133
China-SmallErhuPlectrum	0.99	0.867	0.925	113
China-WuhanTamTam	0.455	0.048	0.086	105
China-Xiao	0	0	0	132
China-YangQin	0.813	0.182	0.297	143
Congo-Bongos	0.926	0.543	0.685	92
Congo-Sanzas	1	0.016	0.032	123
Cuba-Guiro	0.75	0.025	0.048	121
Cuba-Triangle	1	0.061	0.116	114
Egypt-Fiddle	0.908	0.705	0.794	112
Germany-CrumhornAlto	0.971	0.523	0.68	130
Germany-CrumhornBass	0.758	0.409	0.531	115
Germany-CrumhornConsortium	0.976	0.363	0.529	113

Continued

Germany-CrumhornSoprano	0.964	0.462	0.624	117
Germany-CrumhornTenor	0.957	0.333	0.494	132
Germany-Gemshorn	0.081	0.024	0.037	124
micro avg	**0.829**	**0.286**	**0.426**	7165
macro avg	**0.751**	**0.287**	**0.363**	7165
weighted avg	**0.751**	**0.286**	**0.363**	7165
samples avg	**0.743**	**0.319**	**0.422**	7165

Our deep learning baseline model, the bi-LSTM model, was not able to capture distinct features for instrument recognition as it scored 0 across all metrics—F1 score, recall, and precision (see Table 5). However, the bi-LSTM model was expected to be the worst-performing model. It is a common issue with RNN architectures when given high-dimensional inputs such as Mel spectrograms or MFCCs since the long temporal sequences cannot capture the invariance in the frequency domains (Reghunath & Rajan, 2021). However, in our study, we have reduced the input dimension by using the VGGish embeddings, but our dataset was not sufficiently diverse for the model to learn any meaningful information for predictions despite the reduced dimension of 5×128; hence, the model was not able to perform. Despite its previous success in related tasks, the bi-LSTM was the worst-performing model in our study.

Table 5. Bi-LSTM performance.

	precision	recall	f1-score	support
Armenia-Duduk	0	0	0	131
Bali-Gamelan Ensemble	0	0	0	115
Bolivia-Charango	0	0	0	119
Bolivia-Moseno	0	0	0	130
Bolivia-Roncoro Chords	0	0	0	122
Bolivia-Roncoro Solo	0	0	0	132
Brazil-Afuche Cabasa	0	0	0	106
Brazil-Agogo	0	0	0	115
Brazil-Bass Guitar Bossa	0	0	0	108
Brazil-Berimbau	0	0	0	119
Brazil-Bongos	0	0	0	104
Brazil-BongosCowbell	0	0	0	86
Brazil-Cabasa	0	0	0	101
Brazil-Claves	0	0	0	111
Brazil-Cuica	0	0	0	94

Continued

Brazil-EggShaker	0	0	0	103
Brazil-Guitar	0	0	0	94
Brazil-Pandeiro	0	0	0	101
Brazil-PercussionSet	0	0	0	91
Brazil-RainStick	0	0	0	130
Brazil-Surdo	0	0	0	127
BurkinaFaso-BaraDrum	0	0	0	89
BurkinaFaso-BassLine	0	0	0	115
Cameroon-Congas	0	0	0	96
Cameroon-Djembe	0	0	0	93
Cameroon-DrumsetBikutsi	0	0	0	103
Cameroon-PercussionSetBikutsi	0	0	0	106
Cameroon-ShakerBikutsi	0	0	0	95
China-Bawu	0	0	0	137
China-BeijingOperaGongs	0	0	0	121
China-BianzhongBells	0	0	0	105
China-BigErhuPlectrum	0	0	0	103
China-CeylonGuitar	0	0	0	116
China-ChauGongs	0	0	0	106
China-Dizi	0	0	0	133
China-Dongxiao	0	0	0	117
China-Erhu	0	0	0	115
China-FengGong	0	0	0	100
China-Gaohu	0	0	0	101
China-GongsTunedMetalMallet	0	0	0	113
China-GongsTunedSoftMallet	0	0	0	115
China-GongsTunedWoodmallet	0	0	0	96
China-Hulusi	0	0	0	137
China-JinghuOperaViolin	0	0	0	121
China-KouXian	0	0	0	125
China-Pipa	0	0	0	119
China-ShanghaiBabyPiano	0	0	0	130
China-Sheng	0	0	0	133
China-SmallErhuPlectrum	0	0	0	113
China-WuhanTamTam	0	0	0	105
China-Xiao	0	0	0	132
China-YangQin	0	0	0	143

Continued

Congo-Bongos	0	0	0	92
Congo-Sanzas	0	0	0	123
Cuba-Guiro	0	0	0	121
Cuba-Triangle	0	0	0	114
Egypt-Fiddle	0	0	0	112
Germany-CrumhornAlto	0	0	0	130
Germany-CrumhornBass	0	0	0	115
Germany-CrumhornConsortium	0	0	0	113
Germany-CrumhornSoprano	0	0	0	117
Germany-CrumhornTenor	0	0	0	132
Germany-Gemshorn	0	0	0	124
micro avg	0	0	0	7165
macro avg	0	0	0	7165
weighted avg	0	0	0	7165
samples avg	0	0	0	7165

The C-RNN (or C-LSTM) model trained on MFCCs also performed poorly, achieving 0.237 for the micro F1 score and 0.22 for the macro F1 score (see Table 6). Considering that the MFCCs provided relevant information about the input audio, we had expected this model to perform better for instrument detection. However, it also failed to learn to detect some instruments from the training, namely, China-Wuhan TamTam and Brazil-Afuche Cabasa. Even for instruments that the model was able to detect, the precision and recall scores were underwhelming, with most scores for both precision and recall ranging between 0.3 and 0.4. Some instruments, such as Bolivia-Roncoro Chords, had near-zero scores, with just 0.048 for precision and 0.008 for recall.

Table 6. C-RNN (MFCC-trained) performance.

	precision	recall	f1-score	support
Armenia-Duduk	0.224	0.115	0.152	131
Bali-Gamelan Ensemble	0.177	0.122	0.144	115
Bolivia-Charango	0.478	0.37	0.417	119
Bolivia-Moseno	0.232	0.177	0.201	130
Bolivia-Roncoro Chords	0.048	0.008	0.014	122
Bolivia-Roncoro Solo	0.137	0.053	0.077	132
Brazil-Afuche Cabasa	0	0	0	106
Brazil-Agogo	0.067	0.087	0.075	115
Brazil-Bass Guitar Bossa	0.412	0.259	0.318	108

Continued

Brazil-Berimbau	0.333	0.017	0.032	119
Brazil-Bongos	0.151	0.135	0.142	104
Brazil-BongosCowbell	0.084	0.081	0.083	86
Brazil-Cabasa	0.495	0.485	0.49	101
Brazil-Claves	0.024	0.027	0.026	111
Brazil-Cuica	0.102	0.138	0.117	94
Brazil-EggShaker	0.053	0.058	0.056	103
Brazil-Guitar	0.602	0.628	0.615	94
Brazil-Pandeiro	0.562	0.406	0.471	101
Brazil-PercussionSet	0.056	0.077	0.065	91
Brazil-RainStick	0.427	0.515	0.467	130
Brazil-Surdo	0.404	0.449	0.425	127
BurkinaFaso-BaraDrum	0.032	0.022	0.026	89
BurkinaFaso-BassLine	0.359	0.443	0.397	115
Cameroon-Congas	0.114	0.042	0.061	96
Cameroon-Djembe	0.053	0.011	0.018	93
Cameroon-DrumsetBikutsi	0.952	0.961	0.957	103
Cameroon-PercussionSetBikutsi	0.125	0.009	0.018	106
Cameroon-ShakerBikutsi	0.911	0.968	0.939	95
China-Bawu	0.235	0.088	0.128	137
China-BeijingOperaGongs	0.141	0.083	0.104	121
China-BianzhongBells	0.045	0.048	0.046	105
China-BigErhuPlectrum	0.05	0.01	0.016	103
China-CeylonGuitar	0.182	0.034	0.058	116
China-ChauGongs	0.307	0.292	0.3	106
China-Dizi	0.364	0.12	0.181	133
China-Dongxiao	0.102	0.043	0.06	117
China-Erhu	0.467	0.304	0.368	115
China-FengGong	0.039	0.03	0.034	100
China-Gaohu	0.348	0.307	0.326	101
China-GongsTunedMetalMallet	0.588	0.265	0.366	113
China-GongsTunedSoftMallet	0.907	0.852	0.879	115
China-GongsTunedWoodmallet	0.916	0.906	0.911	96
China-Hulusi	0.452	0.139	0.212	137
China-JinghuOperaViolin	0.033	0.008	0.013	121
China-KouXian	0.096	0.056	0.071	125
China-Pipa	0.081	0.025	0.038	119

Continued

China-ShanghaiBabyPiano	0.111	0.038	0.057	130
China-Sheng	0.053	0.008	0.013	133
China-SmallErhuPlectrum	0.965	0.982	0.974	113
China-WuhanTamTam	0	0	0	105
China-Xiao	0.103	0.053	0.07	132
China-YangQin	0.235	0.133	0.17	143
Congo-Bongos	0.259	0.315	0.284	92
Congo-Sanzas	0.221	0.236	0.228	123
Cuba-Guiro	0.036	0.033	0.034	121
Cuba-Triangle	0.064	0.088	0.074	114
Egypt-Fiddle	0.548	0.152	0.238	112
Germany-CrumhornAlto	0.295	0.177	0.221	130
Germany-CrumhornBass	0.302	0.139	0.19	115
Germany-CrumhornConsortium	0.15	0.053	0.078	113
Germany-CrumhornSoprano	0.333	0.068	0.113	117
Germany-CrumhornTenor	0.333	0.144	0.201	132
Germany-Gemshorn	0.034	0.016	0.022	124
micro avg	**0.294**	**0.198**	**0.237**	7165
macro avg	**0.27**	**0.205**	**0.22**	7165
weighted avg	**0.268**	**0.198**	**0.215**	7165
samples avg	**0.302**	**0.209**	**0.226**	7165

Unlike the C-RNN trained with MFCCs, the CRNN trained with Mel spectrograms performed significantly better. This model achieved a micro F1 score of 0.39 and a macro F1 score of 0.361 (see Table 7). Unlike machine learning methods like XGBoost and random forest, its precision and recall values do not suggest overprediction of instrument presence. Despite this advantage, the model still underperformed relative to the results achieved by Reghunath and Rajan (2021), who achieved a micro F1 score of up to 0.65 and a macro F1 score of 0.56 for their C-LSTM (C-RNN) model (trained on Mel spectrograms). The model encountered a similar issue to its counterpart; the CRNN trained with Mel spectrograms was also unable to detect an instrument, namely the China-CeylonGuitar.

Table 7. C-RNN (Mel spectrogram-trained) performance.

	precision	recall	f1-score	support
Armenia-Duduk	0.355	0.168	0.228	131
Bali-Gamelan Ensemble	0.226	0.226	0.226	115
Bolivia-Charango	0.638	0.37	0.468	119

Continued

Bolivia-Moseno	0.346	0.354	0.35	130
Bolivia-Roncoro Chords	0.709	0.459	0.557	122
Bolivia-Roncoro Solo	0.089	0.038	0.053	132
Brazil-Afuche Cabasa	0.133	0.094	0.11	106
Brazil-Agogo	0.107	0.078	0.09	115
Brazil-Bass Guitar Bossa	0.692	0.667	0.679	108
Brazil-Berimbau	0.529	0.076	0.132	119
Brazil-Bongos	0.264	0.269	0.267	104
Brazil-BongosCowbell	0.063	0.07	0.066	86
Brazil-Cabasa	0.43	0.366	0.396	101
Brazil-Claves	0.073	0.036	0.048	111
Brazil-Cuica	0.581	0.457	0.512	94
Brazil-EggShaker	0.289	0.233	0.258	103
Brazil-Guitar	0.715	0.989	0.83	94
Brazil-Pandeiro	0.556	0.347	0.427	101
Brazil-PercussionSet	0.098	0.132	0.113	91
Brazil-RainStick	0.14	0.138	0.139	130
Brazil-Surdo	0.846	0.906	0.875	127
BurkinaFaso-BaraDrum	0.313	0.461	0.373	89
BurkinaFaso-BassLine	0.536	0.774	0.633	115
Cameroon-Congas	0.208	0.052	0.083	96
Cameroon-Djembe	0.167	0.022	0.038	93
Cameroon-DrumsetBikutsi	0.963	1	0.981	103
Cameroon-PercussionSetBikutsi	0.105	0.019	0.032	106
Cameroon-ShakerBikutsi	0.938	0.958	0.948	95
China-Bawu	0.337	0.204	0.255	137
China-BeijingOperaGongs	0.305	0.24	0.269	121
China-BianzhongBells	0.215	0.162	0.185	105
China-BigErhuPlectrum	0.892	0.32	0.471	103
China-CeylonGuitar	0	0	0	116
China-ChauGongs	0.314	0.151	0.204	106
China-Dizi	0.419	0.135	0.205	133
China-Dongxiao	0.452	0.239	0.313	117
China-Erhu	0.815	0.843	0.829	115
China-FengGong	0.593	0.54	0.565	100
China-Gaohu	0.452	0.416	0.433	101
China-GongsTunedMetalMallet	0.4	0.018	0.034	113

Continued

China-GongsTunedSoftMallet	0.867	0.965	0.914	115
China-GongsTunedWoodmallet	0.931	0.99	0.96	96
China-Hulusi	0.294	0.036	0.065	137
China-JinghuOperaViolin	0.55	0.091	0.156	121
China-KouXian	0.469	0.184	0.264	125
China-Pipa	0.346	0.151	0.211	119
China-ShanghaiBabyPiano	0.382	0.1	0.159	130
China-Sheng	0.4	0.135	0.202	133
China-SmallErhuPlectrum	0.926	0.991	0.957	113
China-WuhanTamTam	0.53	0.419	0.468	105
China-Xiao	0.088	0.023	0.036	132
China-YangQin	0.776	0.678	0.724	143
Congo-Bongos	0.473	0.946	0.63	92
Congo-Sanzas	0.411	0.415	0.413	123
Cuba-Guiro	0.124	0.099	0.11	121
Cuba-Triangle	0.07	0.088	0.078	114
Egypt-Fiddle	0.797	0.839	0.817	112
Germany-CrumhornAlto	0.507	0.269	0.352	130
Germany-CrumhornBass	0.717	0.287	0.41	115
Germany-CrumhornConsortium	0.455	0.221	0.298	113
Germany-CrumhornSoprano	0.635	0.342	0.444	117
Germany-CrumhornTenor	0.604	0.22	0.322	132
Germany-Gemshorn	0.073	0.048	0.058	124
micro avg	**0.471**	**0.333**	**0.39**	7165
macro avg	**0.44**	**0.342**	**0.361**	7165
weighted avg	**0.438**	**0.333**	**0.354**	7165
samples avg	**0.49**	**0.348**	**0.38**	7165

The state-of-the-art model by Han et al. (2017) was the best performing of all the models we evaluated, achieving a micro F1 score of 0.55 and a macro F1 score of 0.504. One of its most notable achievements was its ability to detect instruments like the Germany-CrumhornAlto, Germany-CrumhornBass, Germany-CrumhornConsortium, Germany-CrumhornSoprano, and Germany-CrumhornTenor with high precision and recall where other models struggled (see **Table 8**). The model failed to detect some instruments, such as the China-Hulusi and China-JinghuOperaViolin. However, its overall performance was close to results reported by Han et al. (2017), whose model trained on the IRMAS dataset achieved a micro F1 score of 0.602 and a macro F1 score of 0.503. Considering that their

model was built to perform well for instrument recognition in mainstream polyphonic music, our 10% lower performance on the micro F1 score and similar macro F1 score for instrument recognition of cultural instruments was a strong result.

Table 8. Han's CNN performance.

	precision	recall	f1-score	support
Armenia-Duduk	0.667	0.321	0.433	131
Bali-Gamelan Ensemble	0.376	0.635	0.472	115
Bolivia-Charango	0.457	0.622	0.527	119
Bolivia-Moseno	0.316	0.746	0.444	130
Bolivia-Roncoro Chords	0.5	0.074	0.129	122
Bolivia-Roncoro Solo	0.496	0.841	0.624	132
Brazil-Afuche Cabasa	0.119	0.151	0.133	106
Brazil-Agogo	0.1	0.026	0.041	115
Brazil-Bass Guitar Bossa	0.963	0.713	0.819	108
Brazil-Berimbau	0.986	0.597	0.743	119
Brazil-Bongos	0.206	0.067	0.101	104
Brazil-BongosCowbell	0.169	0.151	0.16	86
Brazil-Cabasa	0.72	0.842	0.776	101
Brazil-Claves	0.893	0.982	0.936	111
Brazil-Cuica	0.44	0.543	0.486	94
Brazil-EggShaker	0.789	0.437	0.563	103
Brazil-Guitar	0.895	0.904	0.899	94
Brazil-Pandeiro	0.736	0.772	0.754	101
Brazil-PercussionSet	0.172	0.121	0.142	91
Brazil-RainStick	0.391	0.069	0.118	130
Brazil-Surdo	0.622	0.984	0.762	127
BurkinaFaso-BaraDrum	0.728	0.753	0.74	89
BurkinaFaso-BassLine	0.758	0.791	0.774	115
Cameroon-Congas	0.375	0.031	0.058	96
Cameroon-Djembe	0.038	0.011	0.017	93
Cameroon-DrumsetBikutsi	0.864	0.99	0.923	103
Cameroon-PercussionSetBikutsi	0.6	0.028	0.054	106
Cameroon-ShakerBikutsi	1	0.958	0.978	95
China-Bawu	0.522	0.606	0.561	137
China-BeijingOperaGongs	0.162	0.157	0.16	121
China-BianzhongBells	0.258	0.371	0.305	105

Continued

China-BigErhuPlectrum	0.978	0.437	0.604	103
China-CeylonGuitar	0.385	0.362	0.373	116
China-ChauGongs	0.02	0.009	0.013	106
China-Dizi	0.796	0.677	0.732	133
China-Dongxiao	0.524	0.949	0.675	117
China-Erhu	0.703	0.904	0.791	115
China-FengGong	0.326	0.43	0.371	100
China-Gaohu	0.7	0.832	0.76	101
China-GongsTunedMetalMallet	0.35	0.062	0.105	113
China-GongsTunedSoftMallet	0.972	0.904	0.937	115
China-GongsTunedWoodmallet	0.856	0.865	0.86	96
China-Hulusi	1	0.022	0.043	137
China-JinghuOperaViolin	0.824	0.116	0.203	121
China-KouXian	0.76	0.304	0.434	125
China-Pipa	0.373	0.21	0.269	119
China-ShanghaiBabyPiano	0.614	0.662	0.637	130
China-Sheng	1	0.444	0.615	133
China-SmallErhuPlectrum	0.982	0.973	0.978	113
China-WuhanTamTam	0.138	0.171	0.153	105
China-Xiao	0.033	0.008	0.012	132
China-YangQin	0.916	0.762	0.832	143
Congo-Bongos	0.727	0.783	0.754	92
Congo-Sanzas	0.389	0.415	0.402	123
Cuba-Guiro	0.2	0.091	0.125	121
Cuba-Triangle	0.4	0.07	0.119	114
Egypt-Fiddle	0.62	0.902	0.735	112
Germany-CrumhornAlto	0.842	0.985	0.908	130
Germany-CrumhornBass	0.911	0.887	0.899	115
Germany-CrumhornConsortium	0.84	0.929	0.882	113
Germany-CrumhornSoprano	0.869	0.966	0.915	117
Germany-CrumhornTenor	0.894	0.833	0.863	132
Germany-Gemshorn	0.4	0.081	0.134	124
micro avg	**0.593**	**0.513**	**0.55**	7165
macro avg	**0.582**	**0.513**	**0.504**	7165
weighted avg	**0.587**	**0.513**	**0.504**	7165
samples avg	**0.63**	**0.531**	**0.558**	7165

6. Conclusion

Despite advancements in technology such as applications like Shazam that can identify music within seconds, the trend mainly applies to well-known instruments. Cultural instruments are virtually unrepresented. This paper aims to build on the work of Kailewang and Moieni (2022) by extending multi-instrument detection to cultural instruments, contributing to cultural awareness and the protection and promotion of cultural diversity, which is an obligation under the UNESCO 2005 Convention on the Protection and Promotion of the Diversity of Cultural Expressions, an international treaty ratified by 151 signatory states. This project aims to build on the work of Kailewang and Moieni (2022) by extending multi-instrument detection and applying modern deep learning techniques to preprocess the dataset provided by Cultural Infusion.

We extracted features including Mel spectrograms, MFCCs, Spectral Bandwidth, Spectral Centroid, Spectral Rolloff, Zero Crossing Rate, Root Mean Square energy, and VGGish embeddings for model training. We used random forest as our baseline machine learning model, which performed poorly with our dataset. The XGBoost model, which performed exceptionally well in single instrument detection in a previous paper by Liu et al. (2022), achieved subpar results with our dataset. The bi-LSTM model, our modern deep learning baseline model, was not able to detect any instruments from the test data. The two CRNN models (trained on MFCCs and Mel spectrograms) performed better; however, they struggled to achieve a 0.5 F1 score in either the micro or macro average. The best performing model was the state-of-the-art CNN by Han et al. (2017), which we adapted to detect our cultural instruments. Despite challenges in the dataset, this model matched the macro F1 score of 0.50 achieved by Han et al. (2017) with their conventional dataset, IRMAS, and reached a micro average F1 score of 0.55, only about 10% lower (see **Table 9** and **Table 10**).

Table 9. Macro average result.

Macro avg	Precision	Recall	F1-score
Random Forest	0.679	0.13	0.19
XGBoost	0.751	0.287	0.363
Han	0.582	0.513	0.504
CRNN mfcc	0.27	0.205	0.22
CRNN mel spec	0.44	0.342	0.361

Table 10. Micro average result.

Micro avg	Precision	Recall	F1-score
Random Forest	0.946	0.128	0.225
XGBoost	0.829	0.286	0.426
Han	0.593	0.513	0.55
CRNN mfcc	0.294	0.198	0.237
CRNN mel spec	0.471	0.333	0.39

7. Areas of Improvement

The major challenge we faced for this study was the dataset quality; we compiled the music by combining instruments randomly into one audio file for both the training and testing sets. Hence, the quality of the compositions was not entirely realistic when compared to public datasets that are used for mainstream music recognition or single-instrument recognition, such as IRMAS, OpenMIC-2018, and MedleyDB. In addition, the dataset lacked sufficient variety for training, particularly in terms of the number of samples available for the instruments. Cultural instrument data are scarce, with no publicly available datasets currently offering a broad and diverse set of cultural instruments.

Although we compiled 8000 training samples and 2000 testing samples, the lack of variety per instrument posed challenges. In cases where an instrument had only two recordings, one was used for training while the other was used for testing, causing an issue where the model was not able to detect the unseen data. Another limitation, also noted by Kailewang and Moieni (2022), was that all instruments were present at the same time throughout the audio sample, which does not reflect how instruments typically behave in real music, where different instruments enter and exit at various times.

In the future, the issue of limited instrument variety could be resolved by utilising Generative Adversarial Networks (GANs) such as WaveGAN to generate additional audio samples for each instrument. This experiment was conducted on a subset of the Sound Infusion cultural instrument dataset, which contains approximately 100 unique instruments with various audio samples for each. Our paper explored approximately 63 of these instruments, which leaves room for improvement and scalability.

To strengthen model performance and reduce the risk of overfitting, the data collection process could also be refined by filtering for instruments with at least five available audio samples prior to training.

Acknowledgments

We want to thank Sound Infusion for providing support with their music studio platform containing the instrument samples used to complete the project. We extend our thanks to the CEO of Cultural Infusion, Peter Mousaferiadis, for his support of the project. We would also like to thank Om Kadem, Anjaly Sajeevkumar, Mary Legrand, Aida Hakemi, Mohsen Sadegh Zadeh, and Nicole Lee for their additional support in the process to deploy this algorithm for use on http://www.soundinfusion.io/.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

Chen, R., Akbar, G., & Ajit, N. (2024). *Musical Instrument Recognition in Poly-Phonic*

Audio Through Convolutional Neural Networks and Spectrograms. DigitalNZ. https://digitalnz.org/records/56071138/musical-instrument-recognition-in-polyphonic-audio-through-convolutional-ne

Destatis (2025). *32% of the World's Population Do Not Use the Internet.* https://www.destatis.de/EN/Themes/Countries-Regions/International-Statistics/Data-Topic/Science-Research-Digital/InternetUse.html

Dewi, C., Chen, A. P. S., & Christanto, H. J. (2023). Recognizing Similar Musical Instruments with YOLO Models. *Big Data and Cognitive Computing, 7,* Article 94. https://doi.org/10.3390/bdcc7020094

Han, Y., Kim, J., & Lee, K. (2017). Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 25,* 208-221. https://doi.org/10.1109/taslp.2016.2632307

Hing, D., & Settle, C. (2021). *Detecting and Classifying Musical Instruments with Convolutional Neural Networks.* Stanford University. http://cs230.stanford.edu/projects_winter_2021/reports/70770755.pdf

Humphrey, E. J., Durand, S., & McFee, B. (2018). OpenMIC-2018: An Open Dataset for Multiple Instrument Recognition. *International Society for Music Information Retrieval Conference* (pp. 438-444). ISMIR. https://doi.org/10.5281/zenodo.1492445

Kailewang & Moieni, R. (2022) Multi-Instrument Detection in Culture Music Using Machine Learning Models. *International Journal of Management and Applied Science (IJMAS), 8,* 48-57.

Kaminskas, M., & Ricci, F. (2012). Contextual Music Information Retrieval and Recommendation: State of the Art and Challenges. *Computer Science Review, 6,* 89-119. https://doi.org/10.1016/j.cosrev.2012.04.002

Kundu, R. (2023). *YOLO Algorithm for Object Detection Explained [+Examples].* https://www.v7labs.com/blog/yolo-object-detection

Lei, L. (2022). Multiple Musical Instrument Signal Recognition Based on Convolutional Neural Network. *Scientific Programming, 2022,* Article ID: 5117546. https://doi.org/10.1155/2022/5117546

Li, P., Qian, J., & Wang, T. (2015). *Automatic Instrument Recognition in Poly-Phonic Music Using Convolutional Neural Networks.* arXiv. https://arxiv.org/abs/1511.05520v1

Liu, Y., Yin, Y., Zhu, Q., & Cui, W. (2022). *Musical Instrument Recognition by XGBoost Combining Feature Fusion.* arXiv, 2206.00901. https://arxiv.org/abs/2206.00901v1

Mukhedkar, D. (2020). *Polyphonic Music Instrument Detection on Weakly La-Belled Data Using Sequence Learning Models.* https://www.diva-portal.org/smash/get/diva2:1458608/FULLTEXT02

Reghunath, L. C., & Rajan, R. (2021). Predominant Instrument Recognition in Polyphonic Music Using Convolutional Recurrent Neural Networks. In M. Aramaki, K. Hirata, T. Kitahara, R. Kronland-Martinet, & Ystad, S. (Eds.), *Music in AI Era* (pp. 214-227). Springer.

Vaiedelich, S., & Fritz, C. (2017). Perception of Old Musical Instruments. *Journal of Cultural Heritage, 27,* S2-S7. https://doi.org/10.1016/j.culher.2017.02.014

Wang, A. (2003). *An Industrial Strength Audio Search Algorithm.* ResearchGate. https://www.researchgate.net/publication/220723446_An_Industrial_Strength_Audio_Search_Algorithm

Xu, B., Wang, N., Chen, T., & Li, M. (2015). *Empirical Evaluation of Rectified Activations in Convolutional Network.* arXiv, 1505.00853. https://arxiv.org/abs/1505.00853v2

Zhong, L., Cooper, E., Yamagishi, J., & Minematsu, N. (2023). Exploring Isolated Musical Notes as Pre-Training Data for Predominant Instrument Recognition in Polyphonic Music. *2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)* (pp. 2312-2319). IEEE. https://doi.org/10.1109/apsipaasc58517.2023.10317292