Musical Instrument Recognition in Audio Signals using a VGG-style CNN Architecture

Rayani Venkat Sai Rithvik

***Abstract*— In this project, we explore the application of machine learning and deep learning models in identifying musical instruments in audio signals. There are several directions to take the results of the project forward. One such use is employing the models in primary musical education. Recognition of the instrumental in a musical piece is a fundamental step in music theory. The analysis of musical instruments in songs can also be used to study pop culture and predict future trends. It can also be used in accurate close captioning for people who are hard of hearing, thereby increasing media accessibility. The log-mel spectrograms are used to model the human ability to distinguish between different sounds by their timbre. Extraction of these coefficients from the audio signals is essential for training and testing the dataset on the models in ways similar to how the human auditory system works. A simpler version of the VGG-Net architecture is used to train the dataset, and the performance is eventually compared to Support Vector Machines(SVMs). The accuracy of the Convolutional Neural Network(CNN) model plateaus at approximately 65-66%. The results are analysed, and the future scope of the project is discussed.**

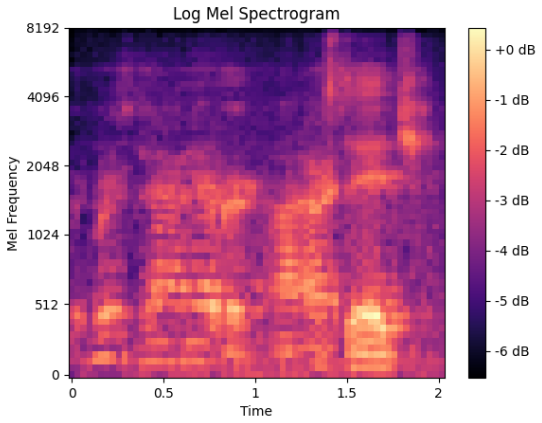

Fig. 1: Log-Mel spectrogram of an audio signal featuring a human singing voice

I. Introduction

Recognition of musical instruments in audio signals through machine learning and deep learning can be a stepping stone for several applications. It is one of the most basic steps in Music Information Retrieval(MIR). Machine learning models can be used as a study aid in primary musical education. It is also possible for aspiring musicians and music labels to study pop culture trends and cater to the audience's demands. Streaming platforms can use the data as a factor for designing their recommender systems. The results can also be directly used in closed captioning, which increases ease of access, especially in current times, as we are more inclusive and empathetic to the differently-abled. Better-quality subtitles can also improve the user experience, as several people rely on closed captions for complete enjoyment and understanding of content. We can automate several tasks in the domain of MIR to perform them quickly in real-time using machine learning.

Humans differentiate between musical instruments by timbre, even when the same note is being played at the same amplitude. Timbre is defined as the quality of the sound that depends on the physical characteristics of the source's material. There is a central frequency and several harmonics associated with a sound, and all these frequencies form the timbre. The timbre of an audio signal can be represented by the log-Mel spectrograms and Mel-Frequency Cepstral Coefficients(MFCCs). The log-Mel spectrograms are used widely in audio processing as they closely model the human auditory system. Thus, they are used as inputs.

II. Literature Review

Slim Essid et al. [2] have used the Mel-frequency Cepstral Coefficients(MFCCs) and a few temporal characteristics as inputs and Gaussian Mixture Models(GMMs) and Support Vector Machines(SVMs) as models, where SVM with an RBF kernel gave the best result. A. Erornen and A. Klapuri [3] have used feature vectors with spectral and temporal characteristics of the audio signals as inputs on the kNN algorithm to achieve an accuracy of 80% for specific instruments.

Peter Li et al. [4] used the raw audio and MFCCs on a basic convolutional neural network(CNN) to achieve an accuracy of 82.74%. The CNN was trained using stochastic gradient descent with a batch size of sixteen. Jongpil Lee et al. [5] used the raw waveform as the input to a 1-D VGG-style CNN, which is termed SampleCNN. On the MagnaTagATune(MTAT) dataset consisting of music samples and the other two datasets consisting of speech, this model achieved an accuracy of 84%.

P. K. Shreevathsa et al. [6] compared an artificial neural network(ANN) and a CNN, which are trained on fourteen music instrument classes, each with around 4000 samples and found the CNN to be more accurate. The CNN model used in this project follows an approach similar to that of the authors of this paper. Blaszke M and Kostek B. M [7] use sets of individual convolutional neural networks (CNNs) per tested instrument. The model is used to classify four instruments. In [8], the authors have taken a part of the IRMAS dataset with six instruments and performed most

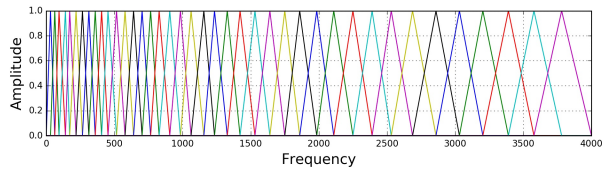

Fig. 2: Mel filter banks

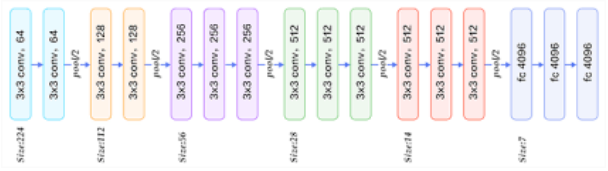

Fig. 3: VGG-16 Architecture

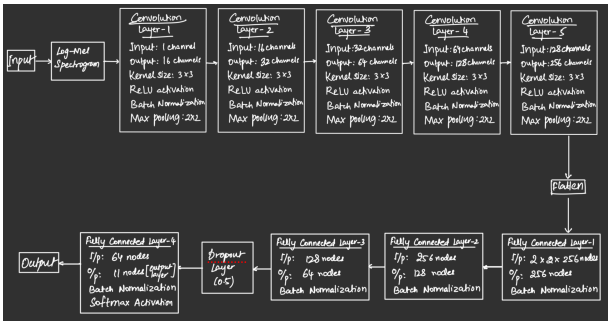

Fig. 4: The model used, which is based on VGG-Net

of the classic machine learning algorithms such as SVM, kNN, decision trees, logistic regression, random forests, etc. The highest accuracy was reported by SVM(0.76), and it is interesting to see if the results will hold for the entire training dataset with eleven instruments.

III. Methodology

A. Dataset Description

The IRMAS dataset for instrument recognition in musical audio signals is being used for this project. The dataset was first used for the evaluation in the following article: Bosch, J. J., Janer, J., Fuhrmann, F., Herrera, P. "A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals", in Proc. ISMIR (pp. 559-564), 2012. In the interest of time and memory, we shall consider only a part of the dataset for this project, as it is large enough for splitting into training and testing data. The dataset consists of files with eleven classes: cello (cel), clarinet (cla), flute (flu), acoustic guitar (gac), electric guitar (gel), organ (org), piano (pia), saxophone (sax), trumpet (tru), violin (vio), and human singing voice (voi). It consists of 6705 audio files in 16 bit stereo wav format sampled at 44.1kHz with a single predominant instrument. They are excerpts of 3 seconds from more than 2000 distinct recordings. The number of files per instrument are: cel(388), cla(505), flu(451), gac(637), gel(760), org(682), pia(721), sax(626), tru(577), vio(580), voi(778).

B. Proposed Workflow

We first need to preprocess the raw audio files and extract their features. We convert all the audio signals into log-Mel spectrograms and use the spectrograms for further training and testing. It is to be noted that Mel-Frequency Cepstral Coefficients can also be obtained using these log-Mel spectrograms. The MFCCs are widely used in audio processing applications such as speech recognition, music processing and sound classification. For our CNN architecture, we use log-Mel spectrograms, while we also compute MFCCs for the sake of establishing a baseline result according to [8]. The following steps describe how the features are extracted.

1) Sampling and Fast Fourier Transform(FFT): The audio signal is divided into short, overlapping windows. We assume that the characteristics of the signal remain relatively constant over short time intervals and sample the signal accordingly. The Fourier Transform is calculated for each window/sample. The signal in the time domain is converted to the frequency domain, and each sample's frequency content is stored.

2) Mel filterbank: The spectrum obtained from the Fourier transform is passed through the Mel filters, which imitate the perception of sound. It is created by overlapping filters spaced according to the Mel scale, with the distance as it is heard to the human ear. Then, the logarithm of the spectrum is taken to mimic the way intensity is sensed by the human auditory system. The frequency in Hertz can be converted to the Mel scale using the following formula:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

3) Spectrogram: The spectrogram is generated by decomposing the magnitude of the signal into its components corresponding to the frequencies on the Mel scale. The spectrogram is called log-Mel spectrogram as we have taken the logarithm of the spectrum after passing it through the Mel filters. To get MFCCs, we decorrelate the coefficients and apply Discrete Cosine Transform(DCT).

The extracted features, along with the labels, are stored and used as an input to a CNN architecture that is derived from the VGG-Net. The architecture of the simplified version of the VGG-Net used in the project is described next.

C. Model Architecture

Karen Simonyan and Andrew Zisserman developed the Visual Geometry Group 16-layer model(VGG-16) wrote the paper titled "Very Deep Convolutional Networks for Large-Scale Image Recognition"[9]. In this model, the sixteen layers use 3x3 convolutional filters, out of which thirteen are convolutional layers. Their stride is one, which means the filter is moved pixel by pixel. ReLU activation functions are used at the end of each layer except the last one, where Softmax is used, which outputs the probabilities of multiple classes. The highest one is chosen. The first thirteen layers also have Max-Pooling layers between every two layers, which have stride as two, skipping a pixel while

Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 16, 64, 64]	160
ReLU-2	[-1, 16, 64, 64]	0
BatchNorm2d-3	[-1, 16, 64, 64]	32
MaxPool2d-4	[-1, 16, 32, 32]	0
Conv2d-5	[-1, 32, 32, 32]	4,640
ReLU-6	[-1, 32, 32, 32]	0
BatchNorm2d-7	[-1, 32, 32, 32]	64
MaxPool2d-8	[-1, 32, 16, 16]	0
Conv2d-9	[-1, 64, 16, 16]	18,496
ReLU-10	[-1, 64, 16, 16]	0
BatchNorm2d-11	[-1, 64, 16, 16]	128
MaxPool2d-12	[-1, 64, 8, 8]	0
Conv2d-13	[-1, 128, 8, 8]	73,856
ReLU-14	[-1, 128, 8, 8]	0
BatchNorm2d-15	[-1, 128, 8, 8]	256
MaxPool2d-16	[-1, 128, 4, 4]	0
Conv2d-17	[-1, 256, 4, 4]	295,168
ReLU-18	[-1, 256, 4, 4]	0
BatchNorm2d-19	[-1, 256, 4, 4]	512
MaxPool2d-20	[-1, 256, 2, 2]	0
Flatten-21	[-1, 1024]	0
Linear-22	[-1, 256]	262,400
BatchNorm1d-23	[-1, 256]	512
Linear-24	[-1, 128]	32,896
BatchNorm1d-25	[-1, 128]	256
Linear-26	[-1, 64]	8,256
BatchNorm1d-27	[-1, 64]	128
Dropout-28	[-1, 64]	0
Linear-29	[-1, 11]	715
Softmax-30	[-1, 11]	0

Fig. 5: Summary of the model architecture

moving(downsampling) and reducing spatial complexity. The three layers at the end are fully connected neural networks, where each neuron is connected to all the activation functions of the previous layer. The last fully connected network has the same number of outputs as the number of classes.

In interest of time, memory and speed, a smaller but similar CNN architecture is used. Five convolutional layers are used with Rectified Linear Unit (ReLU) activation functions and batch normalization(BN) with batch-size equalling number of output nodes to stabilize the training process. The ReLU function outputs the input directly if it is positive and zero otherwise, which helps remove the vanishing gradient problem. The output from the convolutional layers is flattened and is fed to four fully connected layers, with BN applied to each layer. After the last layer, the dropout regularization technique, where some weights are ignored with a probability of 50%. This reduces dependencies among the neurons and reduces the chances of overfitting. Lastly, the Softmax function is used to output the probabilities, where the highest probability corresponds to the predicted class.

D. Training Paradigm

This is a supervised learning model, as we know the labels for the audio files beforehand and are expecting the model to output the correct label. During data preprocessing, the log-mel spectrograms were standardized by subtracting the mean of the data and dividing by the standard deviation of the data to ensure zero mean and unit variance. This ensured no particular feature of a class dominated the training process, and the model converged faster. The standardization is applied independently for each dimension across the entire dataset.

$$X_{\text{normalized}} = \frac{X - \mu_X}{\sigma_X}$$

Some white noise was also introduced to the dataset to ensure the model is accurate even in a different environment, and does not overfit the specific clean audio samples. We are splitting the dataset into training, validation, and testing sets into the ratio of 70-10-20 for the model to be trained on enough samples. Thus, we have 4693 training samples, 663 validation samples, and 1349 testing samples. The model is trained for 30 epochs with a batch size of 64, where the weights of the highest validation accuracy(which means the model performs best for unseen data) are saved to avoid overfitting the training set. The Adam optimizer is used to control the learning rate during the training process for faster convergence. The Categorical Cross Entropy loss function is used as the objective function to minimize. This function is used for multi-class classification where the label for each input is unique.

$$\text{Categorical Cross-Entropy Loss} = -\sum_{i=1}^{C} y_i \cdot \log(p_i)$$

y_i is the indicator function for the actual class (1 if the sample belongs to class i, else 0). p_i is the predicted probability of the sample belonging to class i. C is the total number of classes.

IV. Experimental Results

The following metrics are used to analyze the results of the model:

- Accuracy: Ratio of correct predictions and total number of predictions
- Precision: Ratio of true positives and total positives predicted
- Recall/Sensitivity: Ratio of true positives to the actual number of positives
- F1-Score: Harmonic mean of precision and recall.
- Confusion Matrix: Visual representation of correct and wrong predictions.

After training for 30 epochs, the results as per the above metrics are as follows:

Class	Precision	Recall	F1-score
cla	0.638	0.619	0.628
flu	0.652	0.577	0.612
gac	0.662	0.781	0.717
gel	0.535	0.603	0.567
org	0.704	0.754	0.728
pia	0.805	0.723	0.762
sax	0.492	0.496	0.494
tru	0.664	0.688	0.675
vio	0.529	0.470	0.498
voi	0.818	0.832	0.825

TABLE I: Precision, Recall, and F1-score per Class

There are three types of averaging used for the evaluation of the model's performance:

- Micro-average: Calculated by considering the total number of true positives, true negatives, false positives and

Metric	Value
Accuracy	0.6597
Precision (Weighted)	0.6610
Recall (Weighted)	0.6597
F1-Score (Weighted)	0.6587
Precision (Micro)	0.6597
Recall (Micro)	0.6597
F1-Score (Micro)	0.6597
Precision (Macro)	0.6525
Recall (Macro)	0.6525
F1-Score (Macro)	0.6466

TABLE II: Performance Metrics

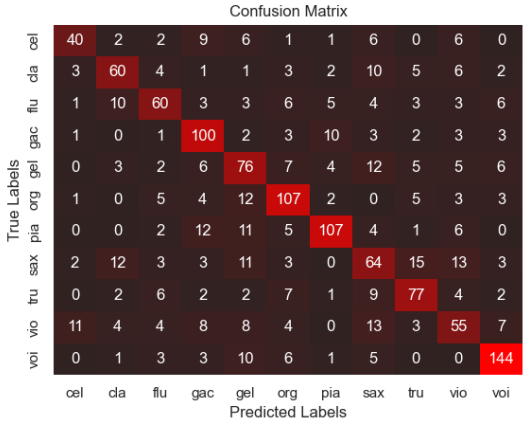

Fig. 6: Confusion Matrix

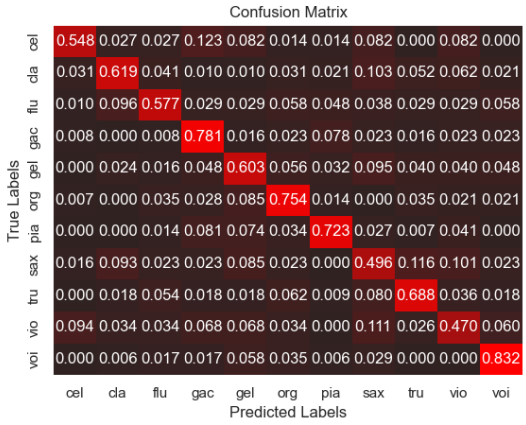

Fig. 7: Normalized Confusion Matrix

false negatives for all classes. Equal importance is given to each sample regardless of the class it belongs to.

- Macro-average: Calculated by taking the average of the respective metric independently calculated for each class. Each class has equal importance regardless of the number of samples in the class.
- Weighted-average: Calculated with the weights of the average being the size of the class, with higher weight for larger classes.

While the dataset is not overwhelmingly imbalanced, there are differences in the sizes of the class. Hence, for completeness, the metrics are given for all the methods of averaging.

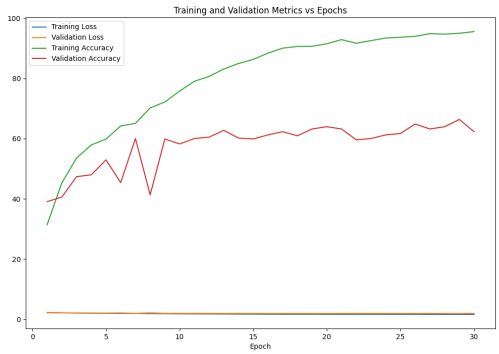

Fig. 8: Training and Validation Accuracy

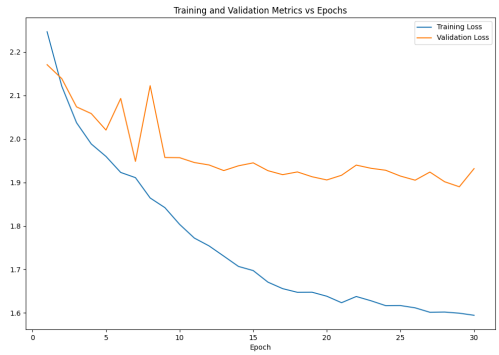

Fig. 9: Training and Validation Loss

The accuracy that the model could achieve is 65.97%. From the confusion matrix we can see that the model is able to classify human voice, acoustic guitar, organ, and piano mostly well. Violin, saxophone and cello are not being classified well. We also see that the model is mostly confusing violin, saxophone and cello, which can be attributed to the fact that they sound similar to each other(especially violin and cello). While the dataset is not terribly imbalanced, the fact the number of samples of cello is the least among all the instruments could also have influenced the confusion. Saxophone is mostly misclassified as trumpet, while flute is mostly misclassified as organ. The electric guitar is mostly misclassified as saxophone. Amongst the higher precision instruments, the model seems confused between piano and acoustic guitar.

V. Conclusion/Discussion

A. Other Observations and Approaches

Different dropout rates, number of layers, and activation functions like Leaky ReLU, Parametric ReLU were used while training the model on the dataset. However, it is to be noted that the accuracy of the model is approximately 64 to 66% in all the cases. With an increase in the number of layers, the peak accuracy is achieved in fewer epochs. For example, the model described in the report achieves 65.97%

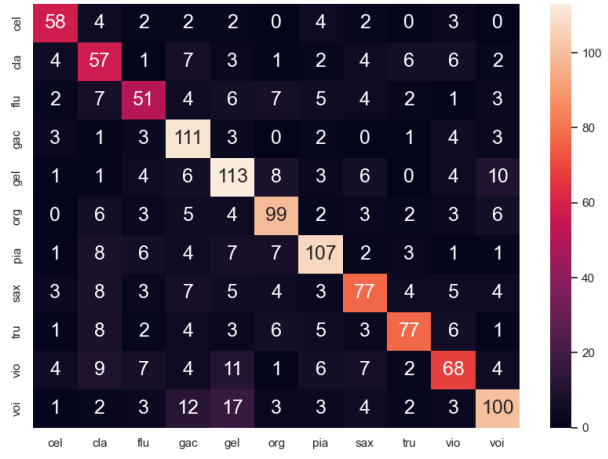

Fig. 10: Confusion Matrix with SVM

accuracy in about 30 epochs. Training it for more epochs results in a slight decrease in the accuracy, which can be attributed to overfitting, i.e., the model learns too much from the training data and cannot correctly predict the labels of unseen training data. The saturation at roughly 65% indicates the model has learnt all it could from the dataset's features.

In the Literature Review, it is mentioned that [8] takes six instrument classes from the IRMAS dataset and achieves 76% accuracy with Support Vector Machine(SVM) algorithm, with Radial Basis Function(RBF) kernel, parameters C = 10 and gamma = 0.1, and twenty MFCCs and other spectral characteristics such as RMS, spectral centroid, spectral bandwidth, rolloff, zero crossing rate as features. When applied for all the eleven instruments with twenty MFCCs as the features, the SVM algorithm gives an accuracy ranging from 65 to 70% based on the train-test split. Regardless, we can say that the model used in this project lags behind the SVM algorithm by a small margin.

B. Future Scope

While the accuracy of the model is reasonable, it is definitely not the best that can be achieved. However, we need to address the saturation of the accuracy at 65%, potentially due to the model being unable to learn further from the features. We can experiment with other features in addition to the log-Mel spectrograms, such as sonograms[10]. We can also perform data augmentation in several ways, such as resampling, pitch shifting, time warping etc., to increase the size of the dataset and make it more balanced. We can also experiment with architectures like Recurrent Neural Networks(RNNs), which can help learn from temporal characteristics. We can also try other regularization techniques like L1 and L2 regularization, and other concepts such as fusion of multiple models.

It is also to be noted that the testing data of the IRMAS dataset has not been worked with in this project. The testing dataset consists of 2874 audio files in 16-bit stereo WAV format sampled at 44.1kHz, with the length varying from 5 to 20 seconds. These files have one to three instruments playing in them, i.e., each input can correspond to multiple labels. Using the Softmax function at the output can help identify multiple instruments; however, it will be interesting to see if training on data with predominantly a single instrument playing, i.e., a single label, can result in a reasonable accuracy in predictions with multiple labels.

VI. References/Resources

1) Bosch, J. J., Janer, J., Fuhrmann, F., and Herrera, P. "A Comparison of Sound Segregation Techniques for Predominant Instrument Recognition in Musical Audio Signals", in Proc. ISMIR (pp. 559-564), 2012
2) Essid, Slim, Gaël Richard, and Bertrand David. "Musical instrument recognition by pairwise classification strategies." IEEE Transactions on Audio, Speech, and Language Processing 14.4 (2006): 1401-1412.
3) Eronen, Antti, and Anssi Klapuri. "Musical instrument recognition using cepstral coefficients and temporal features." 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100). Vol. 2. IEEE, 2000.
4) Li, Peter, Jiyuan Qian, and Tian Wang. "Automatic instrument recognition in polyphonic music using convolutional neural networks." arXiv preprint arXiv:1511.05520 (2015).
5) Lee, Jongpil, et al. "Raw waveform-based audio classification using sample-level CNN architectures." arXiv preprint arXiv:1712.00866 (2017).
6) P. K. Shreevathsa, M. Harshith, A. R. M. and Ashwini, "Music Instrument Recognition using Machine Learning Algorithms," 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM), Dubai, United Arab Emirates, 2020, pp. 161-166
7) Blaszke M, Kostek B. Musical Instrument Identification Using Deep Learning Approach. Sensors. 2022; 22(8):3033.
8) K. Racharla, V. Kumar, C. B. Jayant, A. Khairkar and P. Harish, "Predominant Musical Instrument Classification based on Spectral Features," 2020 7th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2020, pp. 617-622, doi: 10.1109/SPIN48934.2020.9071125.
9) Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In International Conference on Learning Representations, 2015.
10) Dhanalakshmi, P & S., Prabavathy & Rathikarani, V.. (2020). SVM Based Musical Instrument Sound Classification using MFCC and Sonogram. 10.13140/RG.2.2.28731.44322.