# Exploring deep learning for music genre classification on a laptop

Muneeb I. Chaudhary[*]

*Save the Sitar, www. savethesitar. org*

Mubeen I. Chaudhary[†]

*Save the Sitar, www. savethesitar. org*

January 6, 2023

## Contents

**Abstract**

The rise of AI-based music apps has increased the importance of music genre classification. Deep learning models have been shown to be remarkably effective. However, these models incur a significant cost in terms of memory and computational power. This may not be feasible in mobile devices or in home computers. Instead of using computationally intensive deep learning models, this research explores deep learning models that are trainable on resource-constrained devices. An augmented version of the GTZAN data set is used. Apart from deep artificial neural networks (ANNs) and convolutional neural networks (CNNs), a simple transfer learning model is also proposed. The transfer learning model shows a signficantly higher test accuracy as compared to the other models.

[*]Electronic address: `savethesitar@gmail.com`; Corresponding author

[†]Electronic address: `mubeen.i.chaudhary@gmail.com`

## 1 Introduction

This project was imotivated by our long-term interest in building AI-based music apps to help stem the tremendous decline of interest in south-Aisan classical music in Pakistan. While deep learning has been shown to be tremendously useful in music information retrieval (MIR), it is notoriously resource-hungary. Given the difficulty of deploying such hardware at multiple physical locations transferring data to locations with computers armed with powerful GPUs is an attractive possibility. However, there are multiple well-known problems with such an approach. The heavy environmental cost that such computations and data transfers incur is of significant concern. The issues of privacy and latency may also make this option unattractive in certain situations.

Apart from these valid concerns regarding the resource-hungriness of deep learning models, we feel there are two other important reasons for focussing on resource-light models. We feel that machine learning may be going through something akin to the personal computer revolution of the 1980s, when computers became ubiquitously used. Machine learning libraries like keras and librosa have already gotten to a point that even high school students can start using them. There are already websites like `https://machinelearningforkids.co.uk` that offers pre-trained models to children to start experimenting with machine learning. We hope that a useful tool like machine learning will soon be accessible to children through Scratch-like [1] block-based environments, where instead of just using pre-trained models, they can design their own models. Such a change will require computations to be carried out locally on laptops or mobile devices. The second issue is linked to a problem that we faced while working in Pakistan, a developing country. Some of the models that we tried to build were simply too heavy for our laptop. During our efforts to locate powerful computers, we were shocked to learn that most of the universities in Lahore did not have access to computational power adequate for running deep learning algorithms. The free version of Google Colab also got overwhelmed by some of the models we wanted to try. Given that over three-fourth of the world's population lives in the developing world, we feel that is imperative to also conduct research on models that have universal accessibility.

This work was done as a step towards helping us build machine-learning powered apps for Pakistan's classical music. The problem of waning interest in traditional music is not unique to Pakistan but cuts across cultures. Given this broad scope, we would like more young people to start thinking of ways that they can use computer science to cause meaningful change. This report is written with this idea in mind.

We begin by discussing the prerequisites that any new entrant into this are will require. After a brief literature survey, we will discuss the data set that we used to train our models. We will then discuss the different deep ANNs and CNNs we used to classify genres. We achieve a test accuracy of 70%. We then propose a simple transfer learning model based on YAMNet[8]. This model results in a test-accuary of 84%. We will conclude with, what we learned from this project and discuss possible directions for future explorations.

## 2 Prerequisites

We found edX and Coursera to be extremely helpful in developing the necessary programming and theoretical background. The first course we would recommend is a Python programming course on edX Introduction to Computer Science and Programming Using Python. The second one is the machine learning specialization on Coursera Machine Learning Specialization. While these two sources are strictly not required to begin working in this field, they certainly provided us with a lot of confidence and useful knowledge. Since we did not know any signal processing or any features of music, we turned to the YouTube course and the accompanying GitHub repository of the Sound of AI [2], [3]. Once we had learned the basics from there, we got help from multiple online resources and web pages.

## 3 Literature survey

History of music and AI go all the way back to 1957 when Lejaren Hiller and Leonard Isaacson [4], [5] used rule-based AI to generate the first computer-generated composition. Modern machine learning techniques and much more powerful computers have dramatically transformed the landscape since that time.

For music genre classification the state of the art classification accuracy is above 90% for the GTZAN dataset [9], [10], [11], [12]. This accuracy has been reached by designing sophisticated deep learning architectures including CNNs, RNNs and LSTMs [13] and using multiple audio and video features. These features include industry-standards such as mel frequency cepstal coefficients (MFCCs), spectral roll-off and various kinds of spectra including mel spectrograms. On the other hand student research projects [14], [15], [16], [17], [18], [19], [20] have mostly managed to achieve an accuracy between 50%-80%.

For some of the reasons mentioned above there has been lightweight deep learning models have been of interest [7]. Many transfer learning models have also been used to lighten the mahine learning model [6].

## 4 Our data set

The GTZAN genre data set was presented by G. Tzanetakis and P. Cook [21] . It comprises of 100 files of each of the 10 musical genres. Each file is 30 seconds long. While many problems have been identified about this dataset [22], this is still an industry standard and has frequently been used in MIR-related work [9, 10, 11, 12]. At the very basic level we found a couple of files to be corrupted. We replaced those files in our data set by downloading an appropriate audio file from the internet. We also augmented this data set with an online GitHub repository on qawwalis [23]. This augmentation was important to us because we are particularly interested in applying machine learning techniques to classical music of Pakistan. This data set consists of 71 qawwalis by 23 different artists. The length of each clip is 1-minute. We randomly cut a 30-second segment from each qawwali to get 71 30-second clips. We then downloaded 15 qawwalis from the internet and augmented this data set by another two 30 second clips from each of the qawwalis (and discarded one clip because we only needed 29 clips). Hence in the end we had 100 qawwali clips, each of 30 seconds length. Hence in the end we had 11 genres with 100 files each of length 30 seconds. Our data set and code is available at the Github repository.

## 5 Model based on MFCCs

MFCCs are one of the most used features in audio signal processing. These are useful in defining the timbre of music. We began working with a fully-connected feedforward artificial neural network(ANN). We also tried some simple convolutional neural networks (CNNs) to compare them to the feedforward models. We ran our models on the 11 genres with varying file time lengths. We sliced our files into 10 seconds, 3 seconds and 1 second files. As an example when we sliced our files into 3 seconds, we generated 10 files from each original file. This allowed our data set to increase 10-fold. As we ran our models, various questions arose. These were

1. Should we average the MFCCs over frames? Initially in order to reduce the size of the network we averaged the MFCCs over the entire duration of the file. So for example if we are working with a 3 second audio file, at a sampling rate of 22 kHz, we should get $22050 \times 3 = 66150$ samples. Since the MFCCS are based on the short-time Fourier Transform (STFT), the typical hop length is 512 samples. This means that the number of audio frames we have is $\lceil 66150 \div 512 \rceil = 130$. We computed our $n$ MFCCs ($n$ was typically 30) for each of these frames, and then averaged over the 130 frames. So our data for each 3 second clip was 30 dimensional. Our accuracy varied between 60% to 63%. Our expectation was that this averaging would lead to a significant loss in information. Hence if we were to remove this averaging procedure, our results should improve. But the problem arose that the complexity of data exploded. So previously for a 3 second audio file we had 30 MFCCs, now for the same file, our data had become 30 x 130 = 3,900 dimensional. We used a feedforward ANN with 512 x 256 x 64 x 16 x 11 neurons so that we could compare it with our earlier averaged result. The resulting accuracy was extremely low (less than 15%). We even tried working with a more complex ANN but found that the computational power at

our disposal made working with such large ANNs a time consuming process. What it clearly demonstrated was that if we wanted to work with MFCCs without any averaging, feedforward ANNs would not be very successful. We then moved on to try CNNs on this data set. This will be discussed below in point 7

2. What is the effect of reducing the data? As stated earlier, while working with shorter duration files (10, 3 or 1 second files), we decided to randomly pick only one short file from the 30-second long file. This meant that for each genre, instead of 100 30-second files, we got 100 3-second files (instead of a 1000 files). This led to a reduction in accuracy. So for example for a 3-second file, the identification accuracy dropped by about 4%.

3. Should we increasing data size by chopping it up? Most researchers working on GTZAN data set increased the size of their data by slicing each 30 second file into shorter files (3 seconds or 5 seconds) [9, 10, 15]. When we ran our ANN on the data we did not find any such advantage.

| length (s) | total files | accuracy |
|---|---|---|
| 30 | 1,100 | 63.9% |
| 3 | 11,000 | 63.1 % |
| 1 | 33,000 | 59.5% |

   This seems to run contrary to the established norm in the field regarding data that more is better. However, we should note that by the time we got to 1 second files, the accuracy goes down a bit. That is to be expected on such short time scales. Based on our findings it seems perfectly fine to run the neural network on 30-second files.

4. What should be the complexity of the network? For frame-averaged MFCCs we found that the 512 x 256 x 64 x 16 fully connected ANN seemed to work well. When we experimented with a small network 32 x 32 x 16 we found the accuracy going down by about 4%. The table below shows data for working with 11,000 files each of 3 second length.

| Network | accuracy |
|---|---|
| 512x256x64x16 | 63.1% |
| 32x32x16 | 59.7 % |

5. What is the ideal number of MFCCs? We also experimented with increasing and decreasing the number of MFCCs and using the first and second derivatives of the MFCCs. We found out that the first and second derivatives do not contribute as much as increasing the number of MFCCs does. From speech processing websites we learnt that roughly 13 MFCCs should be sufficient to capture the sound data. We found that by increasing the number of MFCCs the to about 30, the training cost did not go up significantly but for every 10 MFCCs added we get an improvement of about 5%. The result of using 30 MFCCs seems to be superior to the result of using 13 MFCCs along with their first and second derivatives. Increasing the number of MFCCs significantly beyond 30 did not lead to a significant improvement in results.

6. How much data do we need?

7. Do ANNs and CNNs behave differently to change of parameters? It was not clear to us that each of the parameters studies with feedforward ANNs would also apply to CNNs without any change. We experimented with running different file lengths (primarily 30 seconds and 3 seconds). We found out that between that as with ANNs, CNNs do not suffer significant performance change because of this change of length. We also reduced the data to a 100 files of varying length to see the impact on CNNs. CNNs were also adversely effected by reducing the input to a 100 files only and lost accuracy on a similar scale to the ANN network. Our conclusion was that CNNs and ANNs behave similarly to change in data.

The results for a 2D-CNN with 64 x 32 x 32 x 16 x 64 (dense layer) network (along with an 11-neuron softmax)

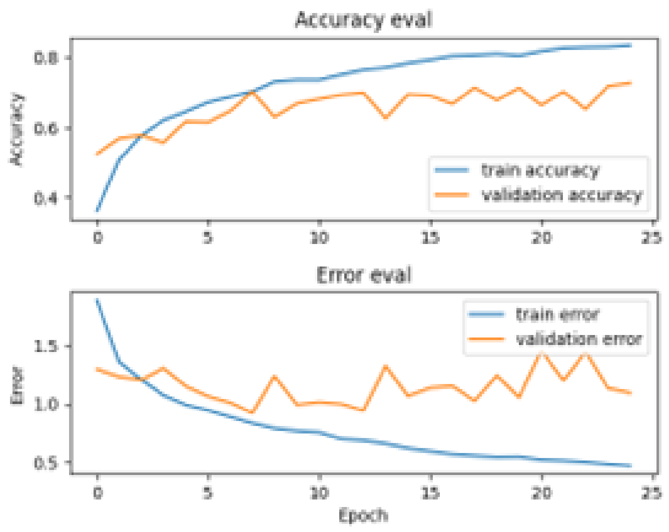


Figure 1: Accuracy versus epochs for a CNN using MFCCs without averaging over frames

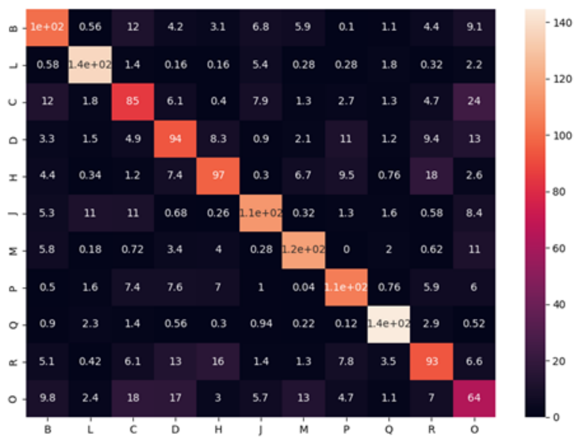


Figure 2: Confusion matrix for a CNN using MFCCS without averaging over frames

Interestingly, the model had the most trouble with the genres of country, hip-hop, reggae, and rock music. Indeed, various authors have reported issues with rock and other genres [9, 15]. The experimentation in the rest of this report has been conducted on 3 second files.

## 6 Model based on spectrogram

The mel spectrogram is another widely used feature. This spectrogram, by emphasizing lower frequencies, manages to mimic the qualities of a human ear better than a linear spectrogram. The lure of deep learning has been the idea that no human feature engineering is required. So we calculated the mel spectrogram of our audio data and fed it into a CNN. We varied the number of convolutional layers, filters, strides etc. but found the results to be fairly stable. The CNN based on the spectrogram did not result in a significant performance performance improvement over 30 MFCCs. The accuracy stayed close to 68%-70%. Our results are shown in figures 3 and 4.

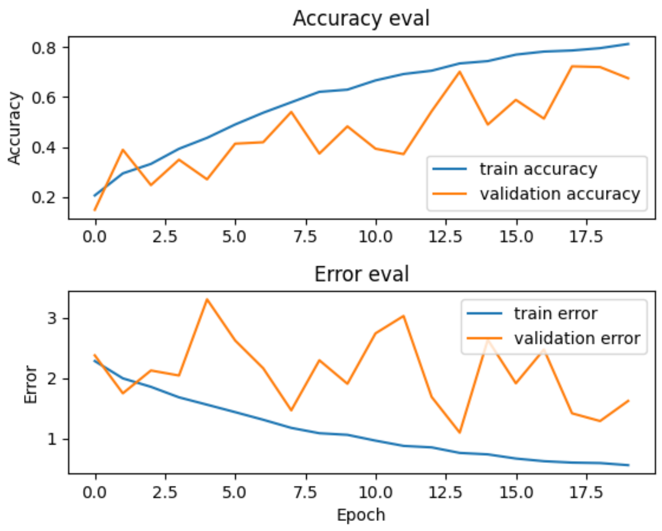


Figure 3: Accuracy versus epochs for a CNN using Mel spectrogram

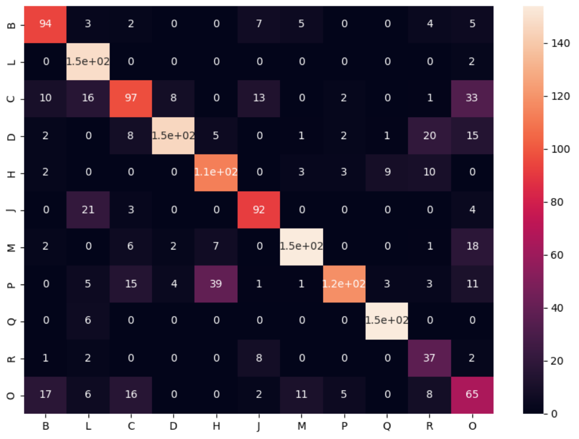


Figure 4: Confusion matrix for a CNN using a Mel spectrogram

This results was enlightening because it showed that feature engineering seemed to provide useful information that was comparable to the machine learnt features in CNNs.

## 7 Model based on raw audio data

We were curious as to how if we directly fed raw audio data into a neural network, what kind of performance will we get. We tried to run a deep neural network on raw audio data. The input dimension being 66150 for a 3 second file was computationally extremely difficult for our laptops to handle. We then tried our CNN model. Even then the laptop simply hung up. Instead we tried using Google Colabs. The free GPU available there also could not deal with this network. So we were stuck. On the suggestion of Dr. F. Sheikh, we started looking at a pretrained model called YAMNet. YAMNet is a pre-trained deep neural network that is freely available on the TensorFlow website. It can predict 521 different audio classes. The thought was that we could use the embedding of this pretrained model to feed into a simple neural network that would classify the music genres that we were interested in. The philosophy of transfer learning is that YAMNet has identified many of the high level features necessary for sound detection that we can directly use for genre classification. We tried two experiments with YAMNet. In one we put in a 512 neuron fully connected layer before the 11 neuron classification layer. In the other experiment we removed the fully connected 512 neuron layer. The results clearly showed that the 512 layer led to a worse result. Also that without our 512 neuron layer, YAMNet based model quickly converged to its best accuracy within an epoch or two. This leads us to conclude that YAMNet by itself is a highly trained model, and adding another full-connected layer in front of it, is probably not the best thing to do, since it has the potential of mixing its output, leading to deteriorated outcomes.

As discussed earlier, we decided to use transfer learning from YAMNet and use the output of the last but final layer and feed it as input into our neural network. The output of this layer is 1024 * 2 * length in seconds of the sample file (actually the last factor is not 2 but 1/0.48, but for 3 second files this approximation is o.k. ). We added a 512 neuron layer (with relu activation) before coupling it to the final 11 output layer with softmax activation. The motivation for this experiment was that since the output layer of YAMNet was so large, we must use some kind of a downsizing middle layer to get reasonable results. However, when we removed this layer and directly coupled it to the 11 output layer with softmax activation, the accuracy improved. Our results are shown in figures 5 and 6.

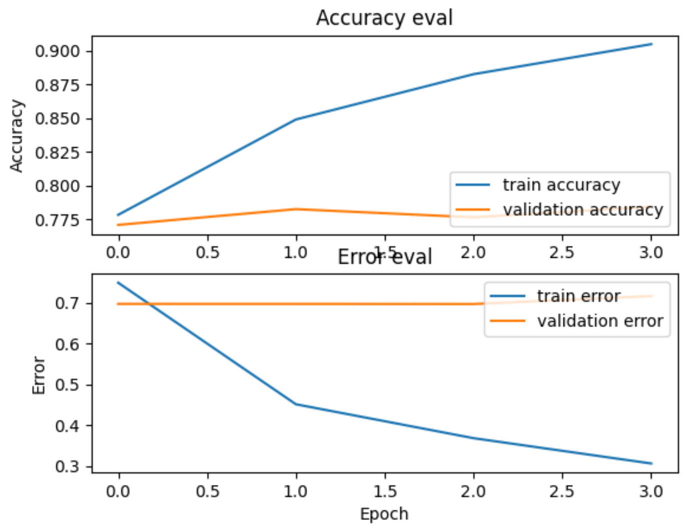
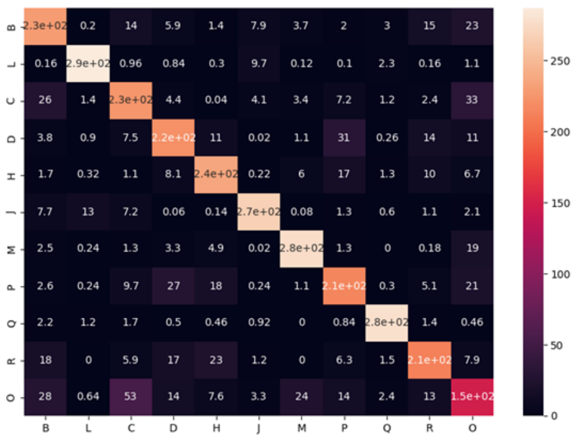


Figure 5: Accuracy versus epochs for a YAMNet-based transfer learning model using raw audio



|  | B | L | C | D | H | J | M | P | Q | R | O |
|---|---|---|---|---|---|---|---|---|---|---|---|
| B | 2.3e+02 | 0.2 | 14 | 5.9 | 1.4 | 7.9 | 3.7 | 2 | 3 | 15 | 23 |
| L | 0.16 | 2.9e+02 | 0.96 | 0.84 | 0.3 | 9.7 | 0.12 | 0.1 | 2.3 | 0.16 | 1.1 |
| C | 26 | 1.4 | 2.3e+02 | 4.4 | 0.04 | 4.1 | 3.4 | 7.2 | 1.2 | 2.4 | 33 |
| D | 3.8 | 0.9 | 7.5 | 2.2e+02 | 11 | 0.02 | 1.1 | 31 | 0.26 | 14 | 11 |
| H | 1.7 | 0.32 | 1.1 | 8.1 | 2.4e+02 | 0.22 | 6 | 17 | 1.3 | 10 | 6.7 |
| J | 7.7 | 13 | 7.2 | 0.06 | 0.14 | 2.7e+02 | 0.08 | 1.3 | 0.6 | 1.1 | 2.1 |
| M | 2.5 | 0.24 | 1.3 | 3.3 | 4.9 | 0.02 | 2.8e+02 | 1.3 | 0 | 0.18 | 19 |
| P | 2.6 | 0.24 | 9.7 | 27 | 18 | 0.24 | 1.1 | 2.1e+02 | 0.3 | 5.1 | 21 |
| Q | 2.2 | 1.2 | 1.7 | 0.5 | 0.46 | 0.92 | 0 | 0.84 | 2.8e+02 | 1.4 | 0.46 |
| R | 18 | 0 | 5.9 | 17 | 23 | 1.2 | 0 | 6.3 | 1.5 | 2.1e+02 | 7.9 |
| O | 28 | 0.64 | 53 | 14 | 7.6 | 3.3 | 24 | 14 | 2.4 | 13 | 1.5e+02 |

Figure 6: Confusion matrix for YAMNet-based transfer learning model on raw audio

## 8 Results and lessons learned

The results reported in this section are the summary of what we found out during this project. Since this report is written with a focus on amateurs and high school students, we feel that we should explicitly state these lessons, so that other people do not repeat the mistakes we made. Retrospectively all these mistakes look silly, but while we were downloading, cutting our audio files, writing code, and tweaking our hyperparameters these did not.

1. The best performance we got was from a neural network with 4 layers with an accuracy of about 60%. This was fairly robust in the sense that by increasing the number of neurons or layers we could not get a better test accuracy. In some sense we feel that this was pointing out the limitations of using feed-forward neural networks.

2. The file length from 3 second to 30 second seemed to have no significant impact on the quality of the prediction. At a 1 second level, there was a slight degradation in accuracy.

3. We found out that the YAMNet model gave an average prediction accuracy of nearly 80%.

4. Adding additional layers to YAMNet seemed to deteriorate its performance. Hence it seems that YAMNet has already done all the necessary feature extraction and only a softmax output layer is required to run it.

5. It was also interesting that when we even chose a 3 second clip from a 30 second file, and used only a 100 files per genre as our entire data set, the results were similar to when we split up the 30 second file into 10 files and used those files to train our network. Retrospectively it seemed obvious since YAMNet was a pretrained model. The 512 layer is not particularly useful. So the only training that has to be done is for the 11 neuron layer. Even 60% or 70% of the 100-file data should be enough to train the neural network.

6. The first mistake we made was in our data set. Early on we were splitting data into training, validation and testing we sliced our files into 3 second files, and then split the training, validation, and testing data. This contaminated our validation and test set. It was later on that we realized that we had violated this cardinal rule. This we found out during our discussions with each other. Later, when we were reading research papers we found a pertinent comment in the fundamental GTZAN paper that "Frames from the same audio file are never split between training and testing data in order to avoid false higher accuracy due to the similarity of feature vectors from the same file". [21].

7. Our programming teachers and courses had always advised us to comment our code well. For class projects, we had found this requirement to be a waste of time at best. This project that has gone on for a few months has clearly demonstrated to us that good commenting is critical for being able to do anything useful with code.

8. We also found that symbolic constants are critical to be able to tune hyperparameters.

## 9 Conclusions and future work

There are multiple ideas that need to be worked on in the future.

1. Clearly generation of relevant data sets is crucial to this effort moving forward. GTZAN, Qawwal Rang and our own data set have issues that may put to question many of the finds in this report. We feel that just like NMIST and other data sets that have become the industry standard, music classification data sets also need to be standardized.

2. The accuracy and the robustness of any implemented model should get close to 90%. This may require us to couple two or more networks, use majority voting or use some pre-trained network.

3. We want to explore the various methods of lightweight neural networks such as pruning or quantization.

## 10 Acknowledgments

We would like to thank Dr. Fahim Sheikh for his guidance, support, and time. We learned a lot through personal communication with him and from his original work Qawwal Rang: An audio data set for genre recognition of Qawwali and Github Qawwal Rang.

# Appendices

## A Making your own data set

For our work on instrument classification and genre classification we had to build or augment existing data sets. In both cases we needed to download music from the internet.

In order to make your own data set, three steps are required.

1. Downloading
2. Converting to an appropriate format
3. Cutting to the appropriate length

The code that we used to accomplish these tasks can be found in THIS DIRECTORY. It is important to note that we needed to install ffmpeg to be able to convert mp3 files (that pytube) that Python downloads to wav files that librosa easily manipulates.

## References

[1] M. Resnick et. al., “Scratch: Programming for all”, *Communications of the ACM*, vol. 52, pp 60–67. 2009

[2] V. Velardo, “The sound of AI," YouTube, [Video files]. Available:`https://www.youtube.com/@ValerioVelardoTheSoundofAI/playlists`. [Accessed: July 28, 2022].

[3] V. Velardo, `https://github.com/musikalkemist`

[4] L. Hiller and L. Isaacson, “Musical composition with a high-speed digital computer”, *Journal of Audio Engineering Society*, vol. 6, pp154-60. 1958

[5] L. Hiller and L. Isaacson, *Illiac Suite. Score*, Theodore Presser Co, New York, USA, 1957

[6] X. Zhang, J. Zhou, W. Sun, and S.K. Jha, “A Lightweight CNN Based on Transfer Learning for COVID-19 Diagnosis”, *Computers, Materials and Continua*, vol 72, pp1123-1137. 2022

[7] S. Parmanand et. al., “A lightweight deep learning model for automatic segmentation and analysis of ophthalmic images”, *Scientific Reports*, vol. 12, pp 8508-8525, 2022. https://doi.org/10.1038/s41598-022-12486-w

[8] “Transfer learning with YAMNet for environmental sound classification”. [online] `https://www.tensorflow.org/tutorials/audio/transfer_learning_audio`

[9] A. Elbir and N. Aydin,“ Music genre classification and music recommendation by using deep learning”, *Electron. Lett.*, vol. 56, pp. 627-629, 2020, https://doi.org/10.1049/el.2019.4202

[10] J. Liu, C. Wang and L. Zha ,“A Middle-Level Learning Feature Interaction Method with Deep Learning for Multi-Feature Music Genre Classification’, *Electronics*, vol. 10(18), pp 2206-2223, 2021. doi: 10.3390/electronics10182206

[11] C. Liu, L. Feng, G. Liu, H. Wang, and S. Liu, “Bottom-up Broadcast Neural Network for Music Genre Classification”, 2019. [online] `https://arxiv.org/abs/1901.08928`

[12] W. W. Y. Ng, W. Zeng and T. Wang, "Multi-Level Local Feature Coding Fusion for Music Genre Recognition," , *IEEE Access*, vol. 8, pp. 152713-152727, August 2020.

[13] C. P. Tang, K. L. Chui, Y. K. Yu, Z. Zeng and K. H. Wong, "Music Genre classification using a hierarchical Long Short Term Memory (LSTM) model", International Workshop on Pattern Recognition IWPR 2018 ,University of Jinan, Jinan, China, 2018. [online] `http://www.cse.cuhk.edu.hk/~khwong/c2018_IWPR2018_LSTM_music_classification.pdf`

[14] R. Adragna and Y. H. Sun, "Music Genre Classification", MIE 234 Project Report, Univ. of Toronto, Toronto, Canada, 2018. [Online] `https://www.eecg.utoronto.ca/~jayar/mie324/musicgenre.pdf`

[15] D. A. Huang, A. A. Serafini and E. J. Pugh, "Music Genre Classification", CS 229 Project Report, Stanford, California, U.S.A. 2018. [online] `https://github.com/derekahuang/Music-Classification`

[16] J. Mendes, "Learning Techniques for Music Genre Classification adn Build a Music Recommendation System", M. S. thesis, School of Computing, National College of Ireland. 2020. [online] `https://norma.ncirl.ie/4455/1/jonathanmendes.pdf`

[17] T. Lahovnik and V. Podgorelec, "Music genre classification based on spectrograms of the sound recording using an ensemble of CNNs," *Proceedings of Student Computing Research Symposium (SCORES'2̂2)*. Ljubljana, Slovenia Oct. 2022, pp 9-12, doi: : https://doi.org/10.51939/scores22

[18] A. Pun and K. Nazirkhanova, "Music Genre Classification with Mel Spectrograms and CNN", CS 229 Project Report, Stanford, California, U.S.A. 2021. [online] `https://cs229.stanford.edu/proj2021spr/report2/81973885.pdf`

[19] J. Tulisalmi-Eskola, "Automatic Music Genre Classification – Supervised Learning Approach", Dept. of Information Tech., M. Eng. thesis, Metropolia University of Applied Sciences, Helsinki, Finland, 2022. [Online] `https://www.theseus.fi/bitstream/handle/10024/651811/Tulisalmi-Eskola_Johanna.pdf`

[20] J. Yang, "Music Genre Classification with Neural Networks: An Examination of Several Impactful Variables", B.S.thesis, Dept. Comp. Sci., Trinity Univ., Texas, U.S.A, 2018. [Online]. Available: `https://digitalcommons.trinity.edu/compsci_honors/44/`

[21] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol 10, pp. 293-302, July 2002

[22] B. L. Sturm, "An analysis of the GTZAN music genre dataset," *MIRUM '12: Proceedings of the second International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies*, pp 7-12. November 2012

[23] F. Sheikh, "Qawwal Rang", `https://medium.com/@fahim.sheikh/qawwalrang-a4fc0d2b2b59` (accessed July 3, 2022)