2018YL\_LD7029BVZ31

MSc Computer Science & Digital Technologies Project

DISSERTATION 2019/20

Raga Classification with Artificial Neural Network

*Develop Browser based prediction model of*

*Indian Carnatic Ragas*

*using Artificial Neural Network*

*&*

*Javascript*

16033168

Sripriya Barrow

DECLARATION

I declare the following:

That the material contained in this dissertation is the end result of my own work and that due acknowledgement has been given in the bibliography and references to all sources be they printed, electronic or personal.

The maximum word count of this Dissertation is *10000 which has been adhered to.*

That unless this dissertation has been confirmed as confidential, I agree to an entire electronic copy or sections of the dissertation to being placed on Blackboard, if deemed appropriate, to allow future students the opportunity to see examples of past dissertations. I understand that if displayed on Blackboard it would be made available for no longer than five years and that, students would be able to print off copies or download. The authorship would remain anonymous.

I agree to my dissertation being submitted to a plagiarism detection service, where it will be stored in a database and compared against work submitted from this or any other School or from other institutions using the service. In the event of the service detecting a high degree of similarity between content within the service this will be reported back to my supervisor and second marker, who may decide to undertake further investigation that may ultimately lead to disciplinary actions, should instances of plagiarism be detected.

I have read the UNN/CEIS Policy Statement on Ethics in Research and Consultancy and I confirm that ethical issues have been considered, evaluated and appropriately addressed in this research. Ethics Form submitted online for approval.

Print Name: Sripriya Barrow. DATE: 10/Oct/2019

ABSTRACT

Deep Learning and Neural Net are cutting edge technologies of today and Javascript is gaining popularity in the world of Artificial Intelligence though very recently, more so in the past 2 years. Deep learning has taken a mighty leap in the past 5 years in many key areas such as Medical, Space, Research and automation driving.

Convolutional Neural Networks (CNN) are the buzz word in image classification. They form a part of the deep learning model. They have multiple hidden layers through which the image passes through various filters, and through cycles of this repeated process, the CNN detects features and aspects that aid in successful classification of images. We have many pre-trained models on images of physical objects that are used by enthusiasts and researchers for various projects. Audio image processing using CNN or Long short-term memory network (LSTM) is still very new and there are not too many mature models available and no recognised dataset at all for audio images, though there are some dataset on actual audio recordings of sounds.

Keras is a python based high-level neural network API which has multi-backend. Keras is largely used in AI and makes the model creation very simple. TensorflowJS on the other hand, is new and only been in use since 2016.

This dissertation project will be a study of integrating end-to-end, from audio recording, image processing to model creation and prediction for Raga classification using Keras and Google’s TensorflowJS API. I will be using VGG (Visual Geometric Group) and MobileNet convolutional neural net (CNN) models which are acknowledged as the best for image classification and test the prediction accuracy for the dataset on both VGG and Mobilenet. I will also analyse how the change in some hyper-parameters affects our model’s predictive power. I will do this by converting music .wav files to sound image forms and dataset. I will finally present our findings by using this prediction model to browser-based application that predicts an audio image.

Table of Contents

[INTRODUCTION 7](#_Toc21555370)

[History 7](#_Toc21555371)

[Common AI technologies 7](#_Toc21555372)

[What is Deep Learning? 8](#_Toc21555373)

[A brief on Raga 8](#_Toc21555374)

[The Project Goal 9](#_Toc21555375)

[Focal Point 10](#_Toc21555376)

[Web Audio Recorder 10](#_Toc21555377)

[The audio Context 11](#_Toc21555378)

[Audio Stream visualization 12](#_Toc21555379)

[Audio Recorder - Web UI 13](#_Toc21555380)

[The View Spectrogram 13](#_Toc21555381)

[Web App Code Directory Tree 14](#_Toc21555382)

[Aspects of Sound Wave 15](#_Toc21555383)

[Sound Envelope 16](#_Toc21555384)

[Fast Fourier Transform 17](#_Toc21555385)

[Filter Bank 18](#_Toc21555386)

[Mel-frequency Cepstral Co-efficients (MFCCs) 19](#_Toc21555387)

[Spectrogram 20](#_Toc21555388)

[Chroma 21](#_Toc21555389)

[Creating Model in Keras 21](#_Toc21555390)

[Why Keras? 21](#_Toc21555391)

[Convolutional Neural Network 21](#_Toc21555392)

[Data and organisation 23](#_Toc21555393)

[Method 1: 23](#_Toc21555394)

[Method 1 - Datasets 23](#_Toc21555395)

[Validation accuracy 23](#_Toc21555396)

[Method 2: 23](#_Toc21555397)

[Method 2 – Datasets 24](#_Toc21555398)

[Directory Structure 24](#_Toc21555399)

[Method 1 – Training with Keras 24](#_Toc21555400)

[Method 2 – Training with Keras 29](#_Toc21555401)

[Setting up Google Colab 31](#_Toc21555402)

[Testing and Prediction in Keras 34](#_Toc21555403)

[Testing & Prediction with TensorflowJS 36](#_Toc21555404)

[Keras to TensorflowJS 36](#_Toc21555405)

[Prediction with TensorflowJS 38](#_Toc21555406)

[Fine Tuning the model 43](#_Toc21555407)

[VGG16 43](#_Toc21555408)

[MobileNet 45](#_Toc21555409)

[The Project Findings 49](#_Toc21555410)

[Observation: 49](#_Toc21555411)

[Problems with Method1 of training with Keras: 49](#_Toc21555412)

[Evaluation, Analysis and Approach 50](#_Toc21555413)

[Data Augmentation 50](#_Toc21555414)

[Optimizers 51](#_Toc21555415)

[SGD – Stochastic gradient descent 51](#_Toc21555416)

[Activation functions 51](#_Toc21555417)

[Loss Functions 52](#_Toc21555418)

[Limitations/ Trip Ups 52](#_Toc21555419)

[Test Results 52](#_Toc21555420)

[Test Inference 52](#_Toc21555421)

[Test Sampling Results 54](#_Toc21555422)

[Conclusion 55](#_Toc21555423)

[The selection of the Raga Classes 55](#_Toc21555424)

[Key features of the 3 chosen Raga Classes 56](#_Toc21555425)

[Observations: 56](#_Toc21555426)

[Models 57](#_Toc21555427)

[VGG16 57](#_Toc21555428)

[MobileNet 57](#_Toc21555429)

[INFERENCE 57](#_Toc21555430)

[Future improvements 57](#_Toc21555431)

[Works Cited 59](#_Toc21555432)

[Other References (Playlists) 62](#_Toc21555433)

[Appendix 63](#_Toc21555434)

[Appendix 1 - Source Code 63](#_Toc21555435)

[Appendix 2 - Flow Diagram 63](#_Toc21555436)

[Appendix 2 – Use Case 64](#_Toc21555437)

[Appendix 4 – TroubleShooting 65](#_Toc21555438)

[Troubleshooting References 65](#_Toc21555439)

[Appendix 5 – Pre-trained Model Summary 67](#_Toc21555440)

[Appendix 6 – Project Plan & Gantt Chart 72](#_Toc21555441)

# 

# INTRODUCTION

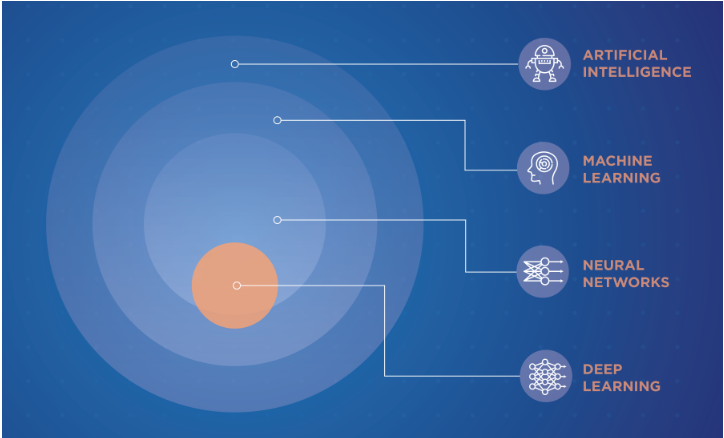
## History

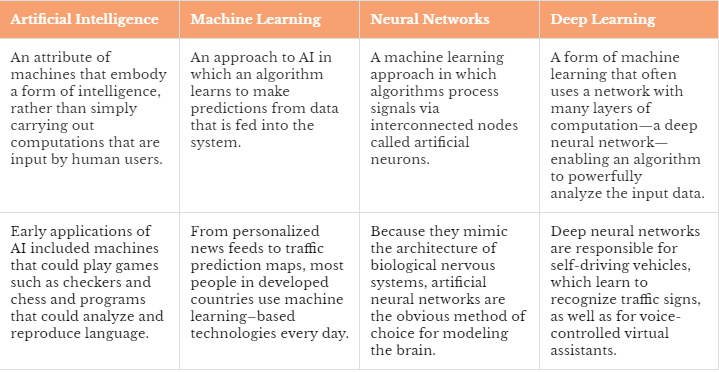
Artificial Intelligence was coined back in the mid-1950s. A mathematician named John McCarthy, also known as the father of AI, used the term to describe machines doing tasks that people call intelligent. In 1956, he and Marvin Minsky organized the Dartmouth Summer Research Project on AI. A few years later the AI Project later to be called AI Lab was created.

In recent years, AI has become a household name and refers to machines which replicate human behaviour and cognition. With evolving technology, amongst, anxiety, fears and excitement, humans have achieved incredible heights with AI, machine learning and deep learning.

## Common AI technologies

The following image and table clearly explain the difference between the terms Artificial Intelligence, Machine Learning, Artificial Neural Networks and Deep Learning which are most often interchangeably used but are very different on the context of use (JEF AKST, 2019).





## What is Deep Learning?

(Marr, 2018)

Typically, in AI, machines do tasks that require human intelligence. AI becomes Machine Learning when machines can learn through experience and display cognitive skills without human interference. While deep learning is part of Machine learning where artificial neural networks (ANN) use algorithms that reflect the patterns of the functions of the human brain. This learning is brought by large recorded datasets called training datasets. This enables the algorithm to perform a task in repetition to learn by practise to get better performance, in the similar way, humans learn.

Deep learning gets its name as the neural network has many layers between the input and the output later and therefore “Deep”.

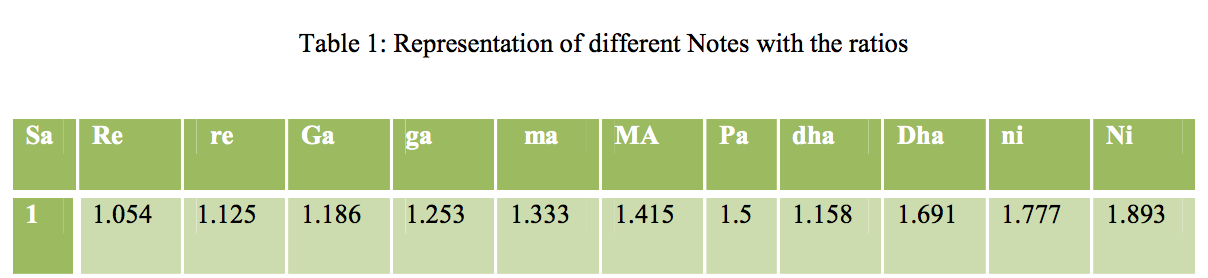
Some examples of deep learning which are commonly used are phone assistants such as Alexa and Siri, driverless cars, facial recognition etc.

## A brief on Raga

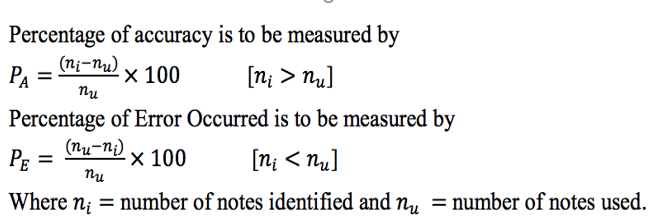
Raga means colour. In Indian music, it is a set of notes that forms a melodic scale and provides a hue of melody and expression. Indian Classical music has 2 variations – the Carnatic played in the South and Hindustani played in the North. Each Raga has an ascending and a descending scale. The Raga is complete (Sampurna) if it has all 7 notes of the octave on the ascending and the descending scale. A Raga can have 5 sets of notes (Audav) or 6 set of notes (Shadav). A Raga is not just the notes in a certain order but also involves the usage of each of the notes and the linking of the notes (shrutis) which are used to improvise the character of the Raga.

Ragas thus have a personality and relay deep human emotions. With all the complexity a Raga demonstrates, it is a good candidate for this dissertation and research to understand and learn how the neural networks will understand its patterns and how accurately it would classify based on the image classification methodology.

As this dissertation is part of web engineering, we will achieve this through Web technologies and demonstrate the same. (Chakraborty & De, 2012)



Mathematically, the formula to calculate the accuracy of each note is as follows. This is not relevant to this project, but it aids in understanding the notes within the scale of the Raga.



# The Project Goal

Raga is a pattern of arrangement of notes (scale) and usage of each note within the scale. Musical pattern recognition is a very complex and challenging task, especially visualization by human brain as it is a mixture of audio (sound waves) and expression (speech). Though we can enjoy music, identifying them without training seems to be difficult especially true with Ragas. Therefore, many mathematicians and scientists have taken the help of mathematical expression to help solve musical analysis and recognition.

With the advancement of AI and Neural network, especially, CNN (Convolutional neural network) and RNN (Recurrent Neural network), complex classification/prediction models have become much simpler than requiring higher mathematical skills. This project will endeavour to use these AI tools, mainly google AI tools and APIs, to train and predict using image classification technology. We have many examples where image recognition and classification have been very effective using Artificial Neural Networks. Music also can be viewed as an image, the spectrogram, a visual representation of the frequencies with respect to time, of its waveform (audio form) to.

With music which has a very undecipherable image pattern to human visualization and recognition, this project will be a journey to test if the image classification of the neural network can successfully identify the Raga patterns using the training dataset with the same algorithm and filters as applied to identifying material objects in the world.

This project will also make a good groundwork for future research towards understanding the relation between complex musical sound and neural network

## Focal Point

The focal point for this project is to demonstrate the following as a browser-based Application:

I) Produce a Javascript Sound Recorder using Web Audio API and HTML5.

II) Create data models using Python Keras for use in the Javascript prediction model.

III) Write a prediction model using Web API TensorflowJS to train and classify music images using neural net.

# Web Audio Recorder

The sound recorder is developed using Web Audio API. Web Audio API is a Javascript API that simplifies processing and synthesizing audio. This is a very powerful API that allows us easy access to high level capabilities of high-end audio systems such as, mixing, filtering all within the web browser.

This app is developed with NodeJS as the web server.

To save recordings we use a third-party API called BinaryClient (orefalo, 2012) (ericz, 2013) (Playing PCM stream from Web Audio API on Node.js, 2019) (noamc, 2014) . The following code sits in the client and streams the audio to the server:

A screenshot of a cell phone

Description automatically generated

The following code in the server writes the recording to wav format.

A screenshot of a cell phone

Description automatically generated

## The audio Context

(Web Audio API, 2019) (gabrielpoca/browser-pcm-stream, 2015) (Geisendörfer, 2009)

MDN web docs define AudioContext as an interface that represents audio-processing graph built from audio modules linked together represented as AudioNode. AudioContext controls creation of the nodes and execution of audio modulation and synthesis.

AudioContext is the (gabrielpoca/browser-pcm-stream, 2015) starting point of audio synthesis in the Web API and everything happens within this context.

A screenshot of a cell phone

Description automatically generated

createMediaStreamSource() method of the AudioContext Interface reads the stream from the mediaRecorder (default is microphone) allows creation of the MediaStreamAudioSourceNode.

AnalyserNode is another interface that represents a node which is able to provide real-time analysis of frequency and time domain information. This node allows us to process the audio data and create visualizations.

A screenshot of a cell phone

Description automatically generated

The scriptProcessorNode is the interface for audio-processing that links two buffers, the input and the output. The buffer size should be a power of 2 between 256 and 16384. Smaller the number for the buffer size, the lower the latency.

## Audio Stream visualization

AudioContext.createAnalyser() method creates the AnalyserNode for an audio. The Analyser Node captures audio data from MediaStreamSource using Fast Fourier Transform(FFT) at a certain frequency domain based on the fftSize specified (Default is 2048).

The 3 key variables of the AnalyserNode are – the fftSize, frequencyBinCount which is half of FFT and Uint9Array(frequencyBinCount) which determines the datapoints that will be collected for the given FFT.

The datapoints collected as dataArray is then passed as an argument to getByteTimeDomainData() method which is the snapshot of the audio data for a given point of time.

This then can be visualised in different forms by plotting it onto HTML5 canvas. The following code snippet draws the audio signal into a wave format.

A screenshot of a cell phone

Description automatically generated

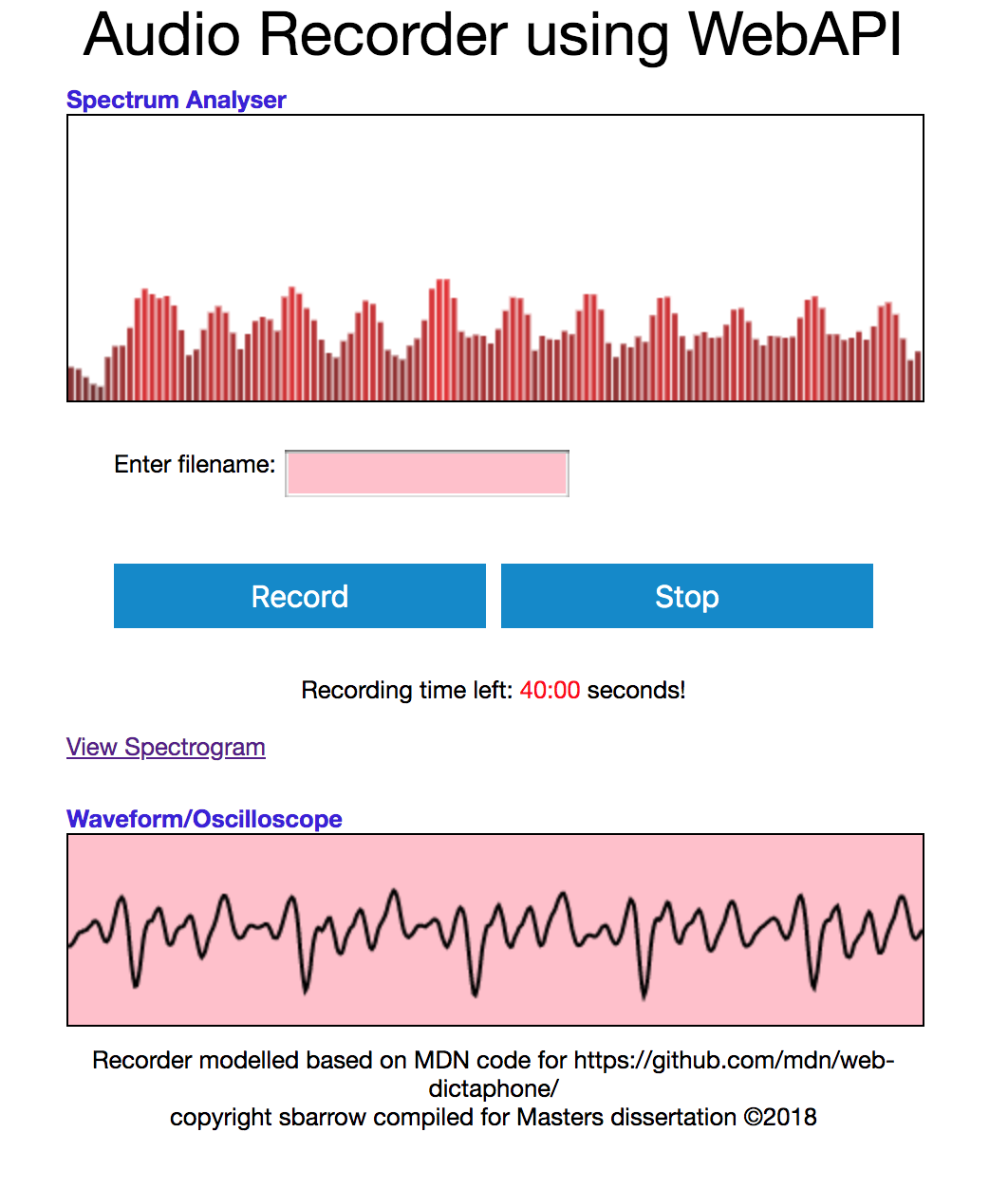
## Audio Recorder - Web UI

The sound recorder Web UI has a visual aspect of the spectrum Analyser and the Waveform.

To view the Spectrogram of the Audio, user can click on the “View Spectrogram” link.

To record Enter the filename of the output and press on Record. All recordings are 40 second clips save in wav format.

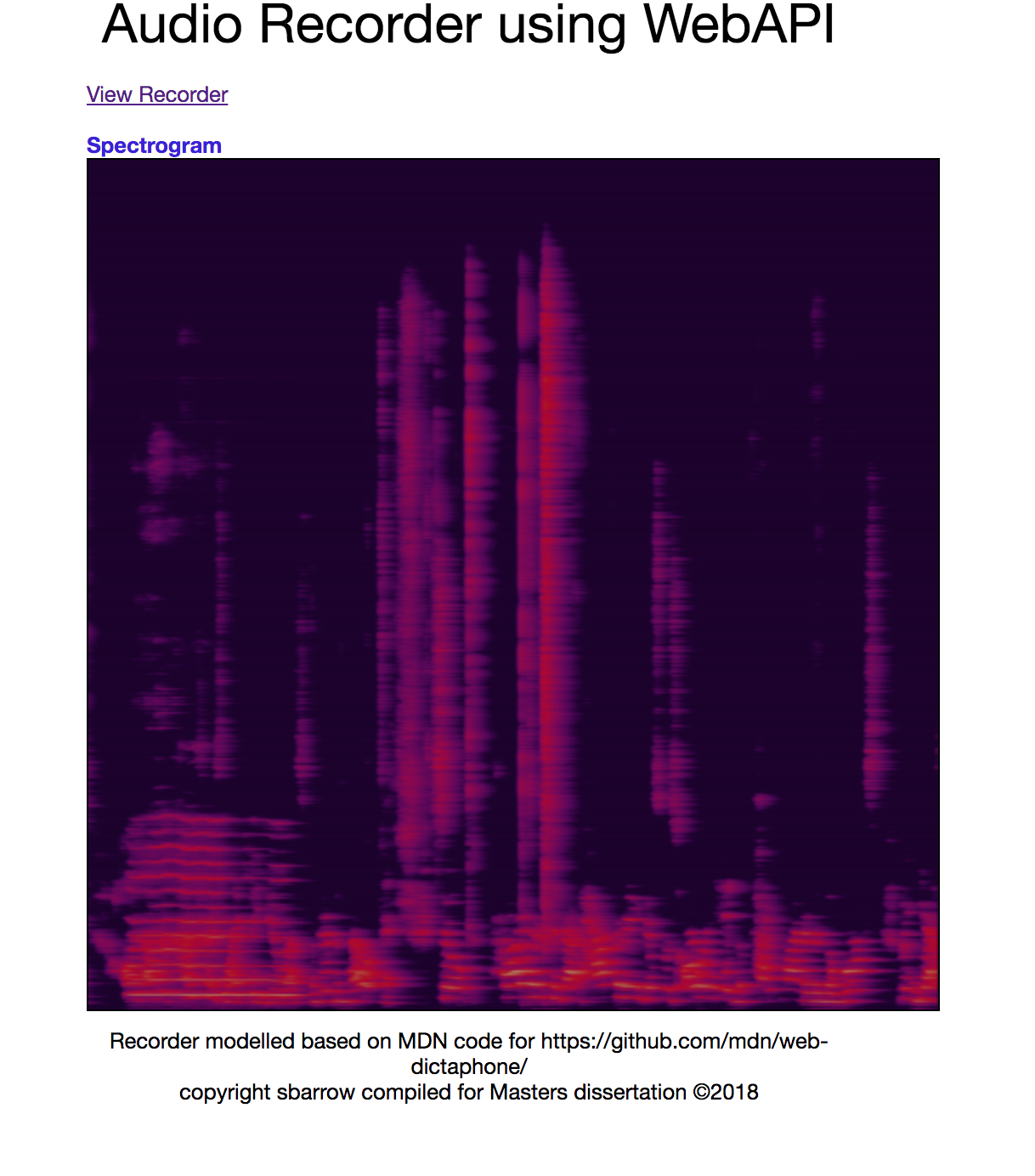
While recording the timer sets to show time remaining. Stop button, stops recording and resets audio.



### The View Spectrogram

On Click of “View Spectrogram” displays the Spectrogram window for the live audio.

“View Recorder” takes back to the Main app page.



A screen shot of a computer

Description automatically generated

### Web App Code Directory Tree

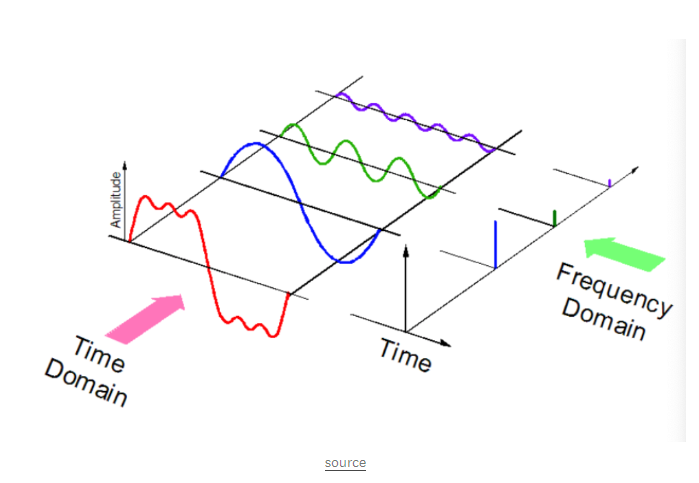
This is the structure of our Node JS App.



## Aspects of Sound Wave

(Fayek, 2019)

Sound is an audio signal with parameters such as Amplitude, Frequency, Time Domain etc. The wave form is typically a sum of all frequencies or smaller waveforms and can be broken down into multiple wave components. (Pandey, 2018)



Since we will be using the Spectrograms as basis of our project to work with image classification using CNN, we will see the several forms of images that can be relevant in training our Neural network and that could be extracted from the audio signal.

This section is based on the following writeups:

<http://haythankfayek.com/2016/04/21/speech-processing_for-machine_learning.html>

<https://www.teachmeaudio.com/recording/sound-reproduction/sound-envelopes/>

### Sound Envelope

Sound Envelope is nothing but the varying level of the sound wave with time, also known as the ADSR (attack, decay, sustain and release) of the audio signal.

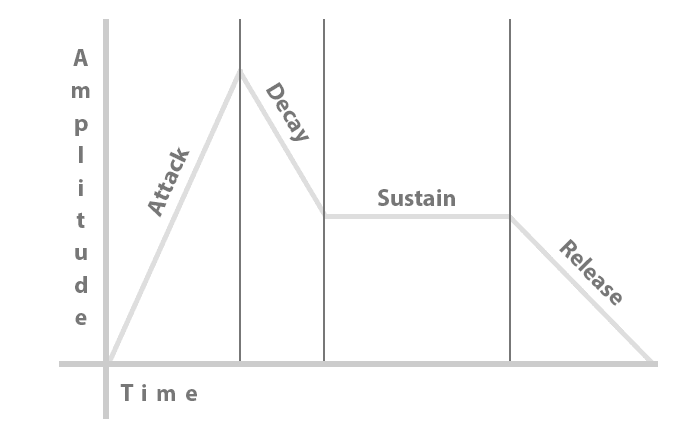
The ATTACK is the amount of time required for the onset of the sound. This is where the amplitude reaches the maximum.

DECAY is the progressive reduction of the amplitude as the sound drops. This follows as the attack reaches its peak.

SUSTAIN is the period where sound still holds true before fading out.

RELEASE is the final fade in amplitude.

In terms of amplitude the sound envelope can be visualised as below ADSR Graph:



We would be using this sound envelope to clean our audio. To achieve this, we will calculate the envelope of the audio signal. From this envelope, we will detect to see if the magnitude is fallen too low. If the signal is too low, it has died down and not important and therefore, we can remove those dead spots.

A screenshot of a cell phone

Description automatically generated

Where y is the audio signal, rate is the rate of the audio signal and threshold is the tested value which is 0.0005. So, across the signal if the mean is greater than the threshold value, then the signal is relevant to us.

A screenshot of a cell phone

Description automatically generated

### Fast Fourier Transform

(Steven W. Smith, 2019), (Thompson, 2017)

To decompose the audio signal into individual frequencies we apply the Fast Fourier transform function.

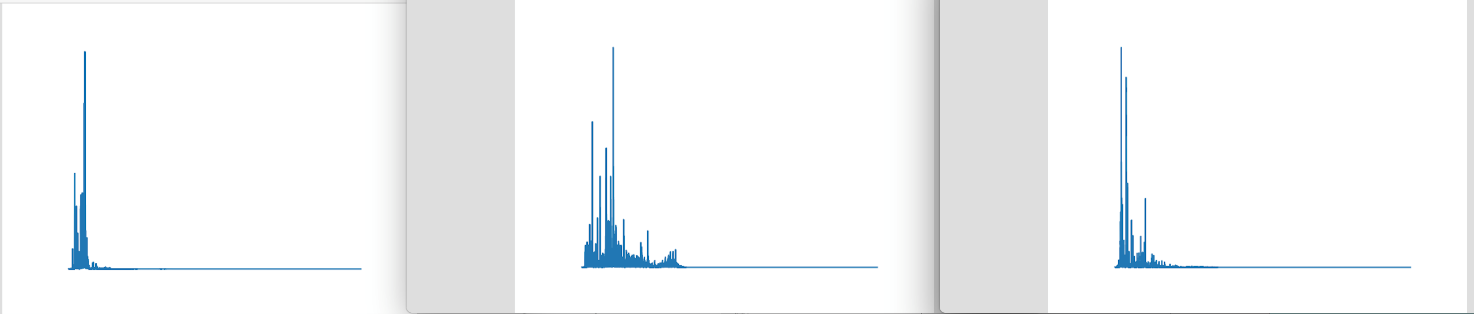
The FFT iterates through each frequency within a frequency spectrum and determines the amplitude of the frequency at a single point in time. It is mathematically denoted as

A picture containing object

Description automatically generated

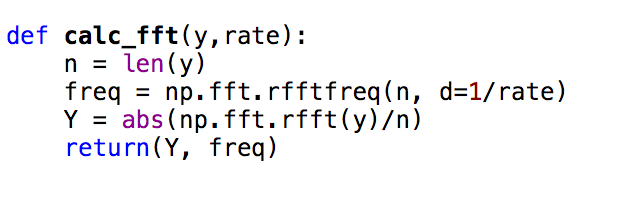
But the mathematical formulation is out of this project scope as I will be using python tools such as numpy or librosa to extract these variables.

Some of the realtime visualisation of the samples I am using are as below in the order of the classes – Kalyani, Kamboji and Mohanam.



The following function using numpy derives the FFT for the signal processed which is then plotted as the above figures. Each FFT image is a 3 second representation of the audio signal.

Librosa is a python library which is integrated with scikit-learn packages and very useful for complex and scientific calculations.



A screenshot of a cell phone

Description automatically generated

### Filter Bank

Digital Filter banks are of 2 types, namely, Analysis and Synthesis. Analysis is when we want to analyse the components of the spectrum. Synthesis is to transform the input. In Analysis spectrum we have one input and many outputs, unlike synthesis where it is many inputs to one output. In our case, we are analysing the spectrum and get the desired frequencies and remove very low frequencies.

Filter bank is the process of applying triangular filters usually 40 filters on a Mel-scale to the power spectrum to get the frequency bands. Mel-scale produces the same effect as the perception of human auditory senses by avoiding low frequencies and processing only the high frequencies. This again filters out all low frequencies. The representation of the filters is as below:

A screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated

The filters we have set at 26 and the nfft = 1103 calculated as rate/40.

The filter bank for the 3 classes – Kalyani, Kamboji and Mohanam can be visualised as below:

A close up of a logo

Description automatically generated

### Mel-frequency Cepstral Co-efficients (MFCCs)

(Nair, 2018)

The filter bank as calculated before cannot be used in machine learning algorithms as they are highly correlated. To decorrelate the filter banks, we apply Discrete Cosine Transform (DCT) to yield compressed form (DataGenetics, 2019). The key parameters are nfilt and nfft as in the filter bank with numcep = 13 which is half of nfilt. The mfcc is calculated as below:

A screenshot of a cell phone

Description automatically generated

The MFCCs representation of the classes, Kalyani, kamboji and mohanam are as below:

A close up of a logo

Description automatically generated

### Spectrogram

Spectrograms are visual representation of the audio file. I had used **regular spectrogram** as explained under section “Creating neural net with Keras”. But later reading through various other audio implementations including spotify, realised mel-spectrogram is a better representation of the audio files.

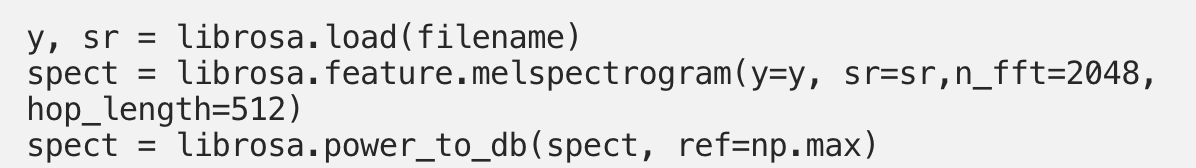
A regular spectrogram is squared magnitude of the STFT (short term Fourier transform) of the signal. The mel-spectrogram is logarithimic and makes it more understandable to the human visual cortex.

The visuals of a regular spectrogram for a 3 second signal is as below:

A picture containing monitor, electronics

Description automatically generated

With librosa, it is very easy to convert the same audio to a **mel-spectrogram** with the 3 lines of code as below:



y = signal

sr = sampling rate = 44100

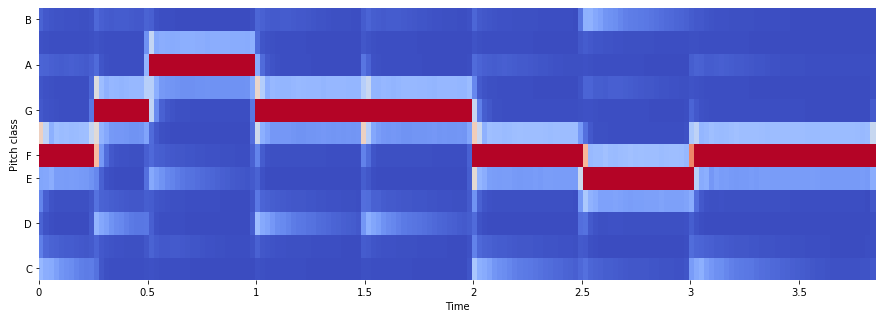
n\_fft = 2048 = window length, which is about 10ms, the shortest time it takes a human to pick the sound. If we move through the audio signal, it by default will move by 10ms.

hop\_length = 512

### Chroma

(Chroma (Unknown))

Constant-Q Transform is similar to mel-scale that it is a logarithmic frequency axis. This has clear pitch indication which makes it suitable for genre classification. This can be visualised as below:



This might be a candidate for another project but is worth mentioning here.

# Creating Model in Keras

(Xu, 2019)

(Dwivedi, 2018)

(Michalak, 2016)

(Building powerful image classification models using very little data, 2019)

## Why Keras?

Keras is an established high-level neural networks API and is well integrated with backend Tensorflow, CNTK and Theano.

Since this project goal is about producing prediction model using Javascript API, and not creating datasets, Keras simplifies the creation of dataset thus allowing the focus from being distracted.

Keras also seamlessly runs on CPU and GPU and allows dataset to be converted to the model as required by TensorflowJS.

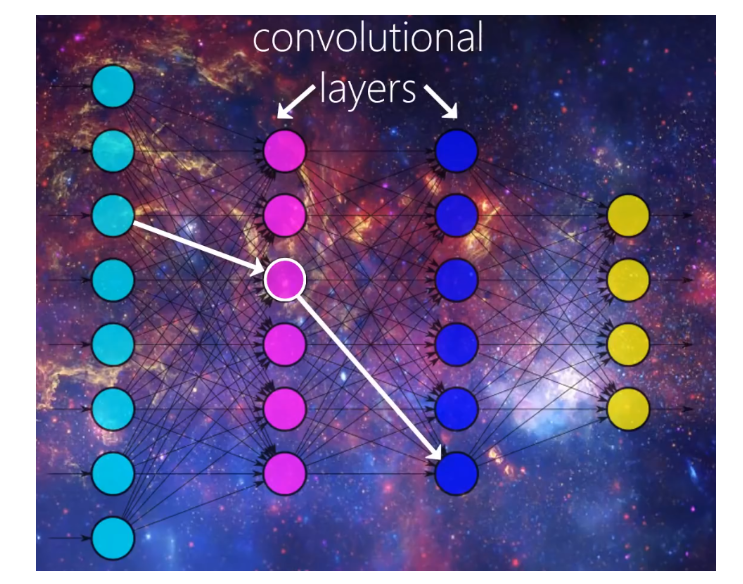
Keras simplifies coding as it enables to produce efficient model that is required by Tensorflow for prediction.

## Convolutional Neural Network

(Deshpande, 2016)

(Saha, 2018) , (Singh, 2018)

(Siddhu, 2019), (Brownlee, 2019)

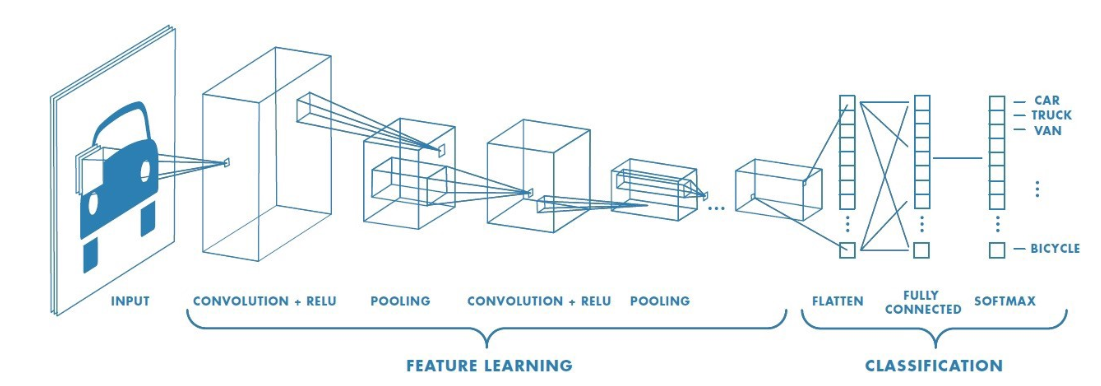


With the rapid advancement of Artificial Intelligence, there has been many breakthroughs in the bridging of gap between humans and machines. Researchers, bloggers, amateurs to enthusiasts have been playing with various aspects of AI and deep learning.

There are many universities who have been working primarily on Computer Vision, where machines are trained to detect images or images of objects.

How do I know the difference between a convolutional neural network a.k.a convnet and other multi-perceptron networks? Convnets have hidden layers that contain convolutional layers which have filters that perform certain kind of filter operation to detect a feature. Detecting a feature is like pattern matching.

Eventhough, CNNs are mainly used in image detection and classification, it can also be used for other classification problems. CNN can have other non-convolutional layers as well. But the main layers are convolutional and when an input is passed through these layers they match for a certain patterns using filters, for example, a filter might identify only circles, another can detect squires, another can detect edges and so on.



As the image passes from layer to layer over and over again, with time, the model starts learning features and starts associating the image with a relative label. It not only trains by detection, but it also corrects its errors on validation of these images, by adjusting the weights and bias to get accuracy on detection these images.

## Data and organisation

### Method 1:

1. Split the recordings to 4s clips using wavepad, a third party tool.
2. Convert all the clips (.wav files) to spectrograms (.png files) using the below (in Mac)

|  |
| --- |
| for f in \*.wav; do sox "$f" -n spectrogram -t "$f" -c "$f" -o "${f%%.wav}.png"; done |

1. Create a folder my\_nbs to hold the Jupyter notebook scripts/programs (\*.ipynb)
2. Under my\_nbs, create the images directory (raag\_images).
3. Within raag\_images, we require 3 folders – train, check (to validate) and test.
4. Within Train and Check, we create folders to hold our classes.
5. The classes are initially RaagM (Raag Mohanam) and RaagS (Raag Sankarabharanam.

#### Method 1 - Datasets

**Training set**:

Class 1 - RaagM – 288 images

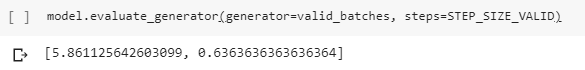
Class 2 - RaagS – 288 images

**Validation set**:

Class 1 - RaagM – 13 images

Class 2 - RaagS – 13 images

#### Validation accuracy



The evaluate generator generates the mean for validation loss and accuracy. The validation loss of 5.86 is high but, I have attributed it to the small dataset.

### Method 2:

Other researchers who have been working on speech recognition and audio classification of Western genres have inferred mel-spectrogram as a better candidate than a Spectrogram for audio classification. For this reason, I will follow method 2 to create mel-spectrograms and try a few more Ragas.

To achieve this, I shall,

1. Record and order signals for training, validation and test data.
2. Clean audio using python envelope
3. Split clean audio files into 3s audio signals.
4. Save all signals into mel-spectrogram

#### Method 2 – Datasets

**Training set**:

Class 1 - Kalyani – 624

Class 2 - Kamboji – 608

Class 3 – Mohanam - 608

**Validation set**:

Class 1 - Kalyani – 624

Class 2 - Kamboji – 520

Class 3 – Mohanam - 437

**Test set**:

Class 1 - Kalyani – 267

Class 2 - Kamboji – 80

Class 3 – Mohanam – 118

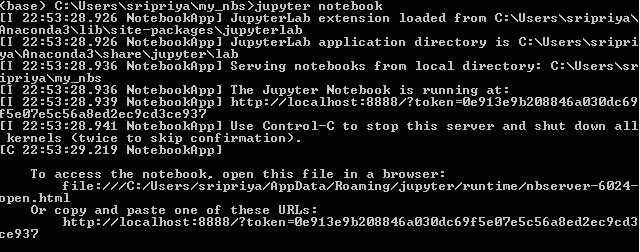
## Directory Structure

### Method 1 – Training with Keras

1. Open Anaconda prompt and from within **my\_nbs** directory, and invoke “jupyter notebook”.



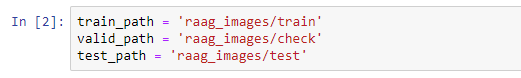
1. Once Jupyter notebook starts up, we see the connection link as below:



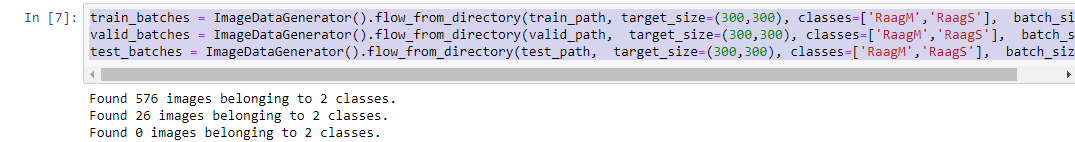
1. Declare variables, importing packages and functions



1. Set the path for the images



1. Normalize images for the neural net.





Batch\_size – the number of images to be processed in one batch

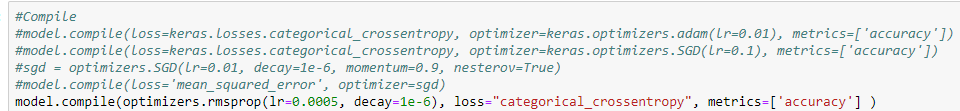
Target\_size – normalizes image size

Classes – Types of Raag samples to be classified.

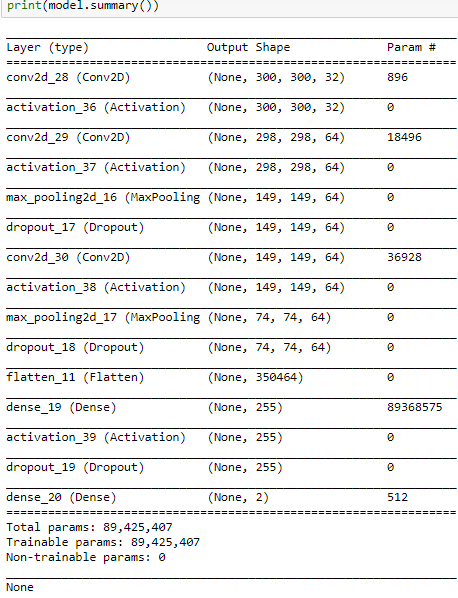
1. Define the neural net training model



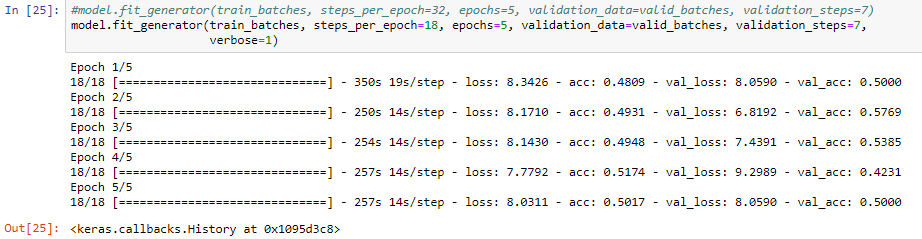
1. Compile the model in Keras



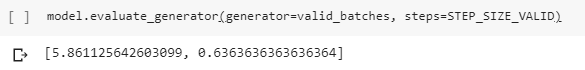
1. View the model summary



1. Train the neural net



1. Evaluate model to provide the average accuracy across epochs.



### Method 2 – Training with Keras

The Mel-spectrogram are inferred by many researchers as a better image representation of audio sounds as they make sound more human readable.

Mel-scale is mathematically non-linear representation of the frequency scale. Librosa allows us to apply this principle using short time fourier transform.

A close up of a logo

Description automatically generated

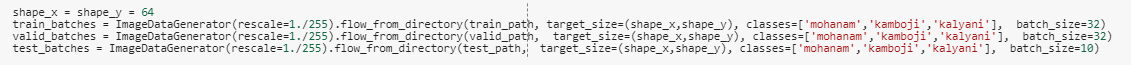
1. Initiate seed Numpy, Python and Tensorflow. This will ensure, the program behaves as expected for each run.



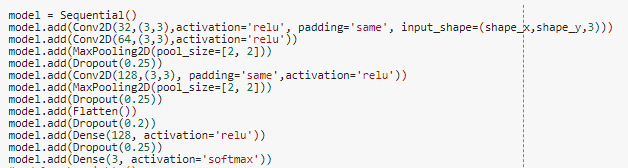
1. Setup image path



1. Normalize images as required for the model

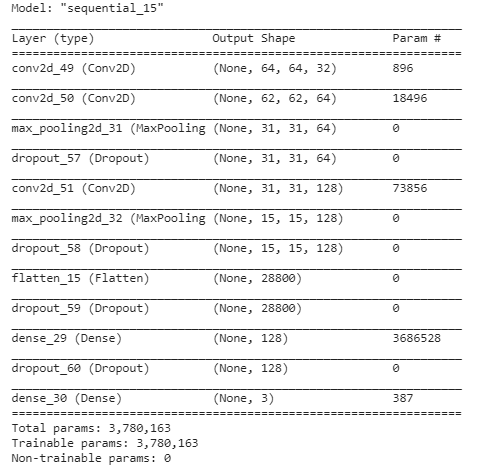


1. Define the model

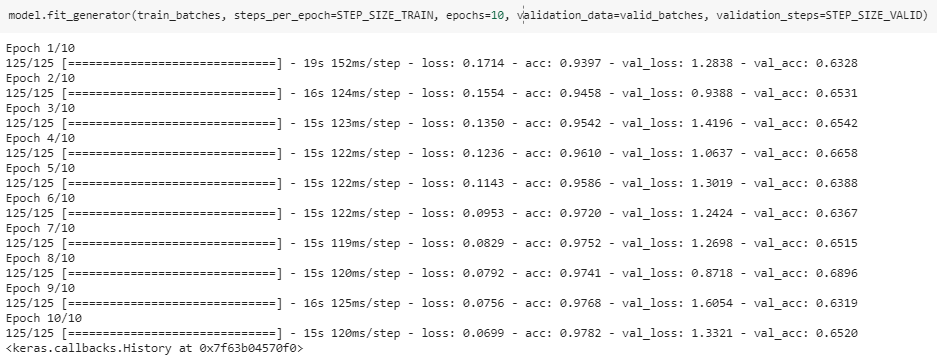


1. Compile the model and view summary.

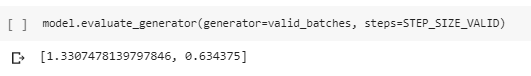




1. Fit model using Fit\_generator.



1. Evaluate model.



### Setting up Google Colab

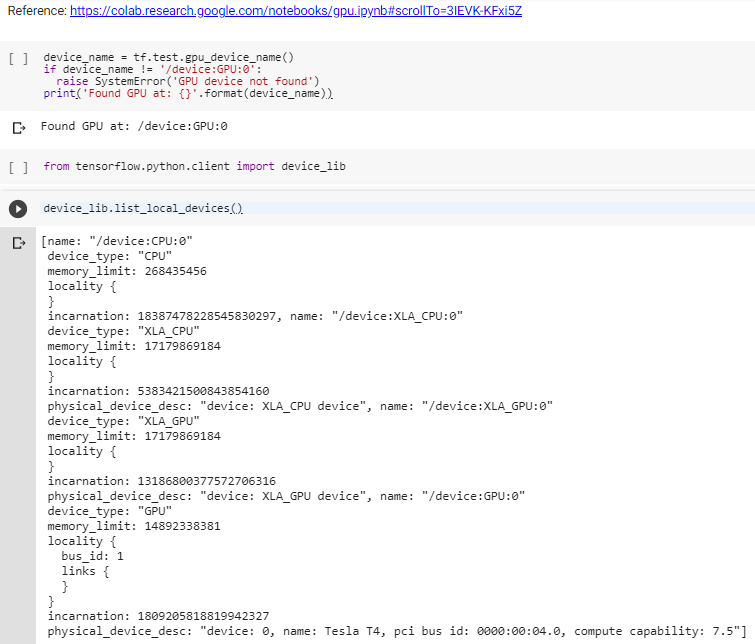
As my model in my local drive takes 3 hours to complete, the GPU is a preferred option. Since, Tensorflow requires CUDA to run on GPU and CUDA is only supported by Nvidia card, I found setting up GPU is a hard call. The workaround will be to use Google Colab which is free for students and researchers and provided the ability to run Neural net on Jupyter lab on GPU.

To set this environment, there are a few measures we need to take.

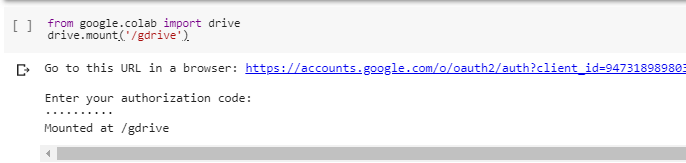
1. Login to Google account.
2. Go to Google Colab on Google Chrome.
3. Copy all project directory into google drive. (colab works from google drive or git by default)
4. To open Jupyter Notebook, Open create new notebook on colab.
5. Click on the down arrow and select “connect to hosted runtime”.



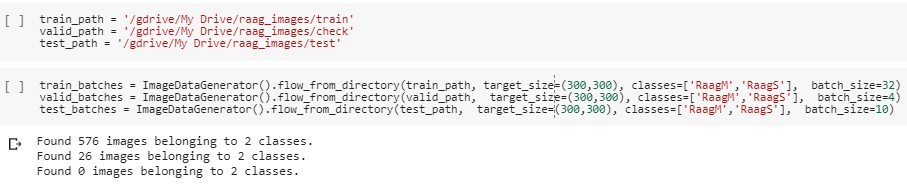
1. From Runtime menu, select “change runtime type”. Change “Hardware accelerator” to GPU.
2. To test the environment is connected to GPU, we can use the following scripts available in Colab:



1. We have to mount the google drive for the program to be able to access the images and follow instructions:



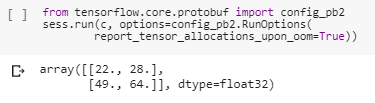
1. Mounting the drive successfully, will result in the batches reading the images:



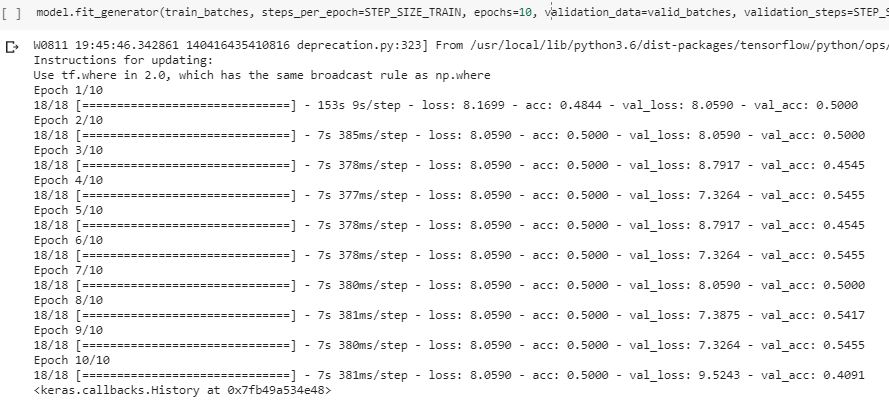
1. We also run the following to allow GPU to grow on demand (error handling detailed in troubleshooting section):



1. The following fixes out of memory issue:



1. The training took a fraction of the time in GPU compared to the CPU.

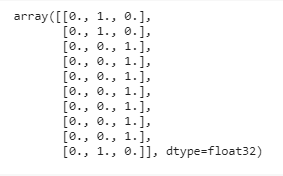


### Testing and Prediction in Keras

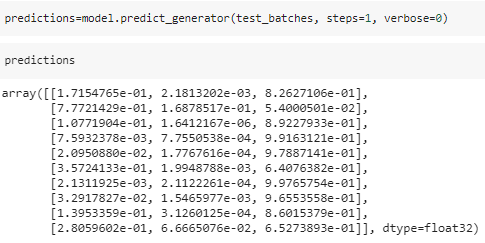
To Test and predict our test batch using the above model, we follow the below procedure.

1. Assign test batch to labels and images

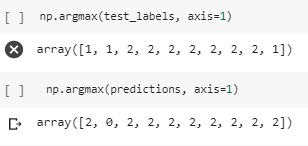




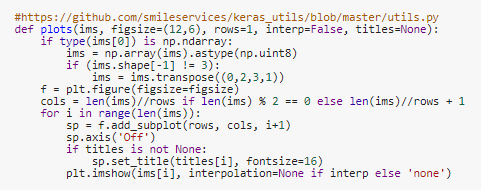
1. Using predict.generator, run the model to predict and classify the images.



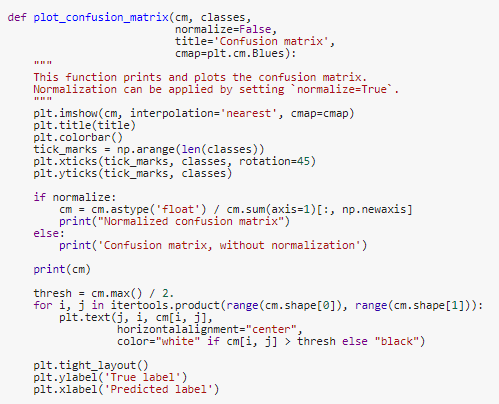
1. View the actual test labels vs prediction results



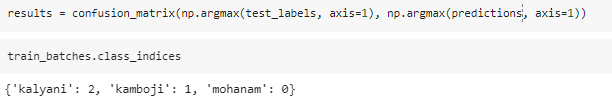
1. The following function from sklearn allows us to compute the confusion matrix for the model results.



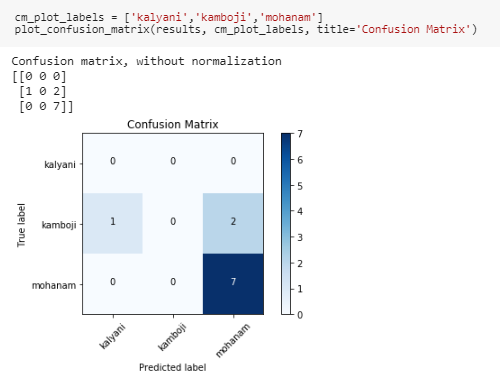
1. The following function from sklearn allows us to view the confusion matrix graphically.



1. Call the function plots to get Confusion matrix. The class indices show the keys and index of the classes.



1. The following visualises the result of the above function.



1. The above graph represents
   1. that no sample was taken (as this is randomised and therefore no prediction for Raga Kalyani.
   2. Out of 2 samples for Raga Kamboji, 1 was predicted as Kalyani and 2 as mohanam.
   3. Raga Mohanam has 100% accuracy in prediction.

To further test this, I have also developed an UI as part of the NodeJS Recorder app as below.

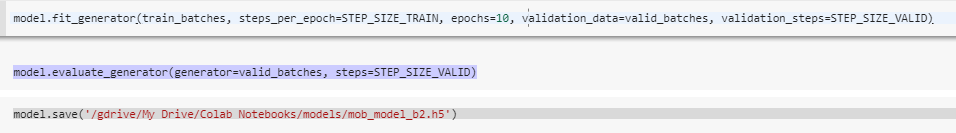
### Testing & Prediction with TensorflowJS

#### Keras to TensorflowJS

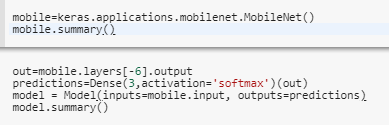
Tensorflow,js is a javascript library. This is open source and was developed by Google for training and deploying ML models in browser based apps especially NodeJS.

There is also ML5.js which is a high level Javascript API built over TensorflowJS.

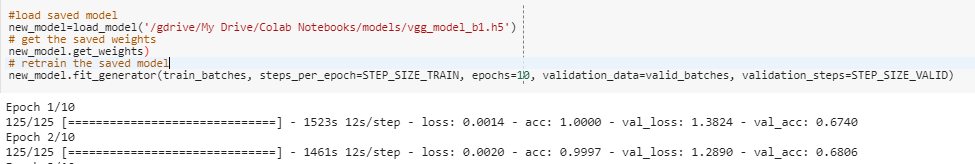
Before taking a keras trained model to TensorflowJS we have to convert it to tensorflow format. To do this, after we fit the model in Keras, we can save the model using model.save() function.



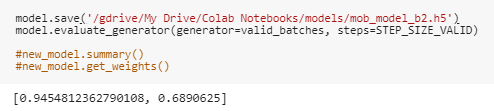
I will evaluate MobileNet in the browser as it returns the best validation accuracy.



We can also retrain by loading the model. To load the saved model,



The above code loads a saved model. Displays weights of the saved model. And retrains it. Retraining starts from the previously saved state.



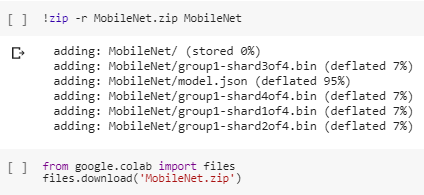
Install tensorflowjs in python environment. I have installed within the Jupyter Notebook in Google Colab. (tfjs, 2019) (Model conversion  |  TensorFlow.js, 2019)



Run the following code to convert the keras model to tensorflowjs model.



Zip and download the directory.



#### Prediction with TensorflowJS

1. This is the UI for prediction app using TensorflowJS. This is driven by NodeJS server from localhost:3000

A screenshot of a cell phone

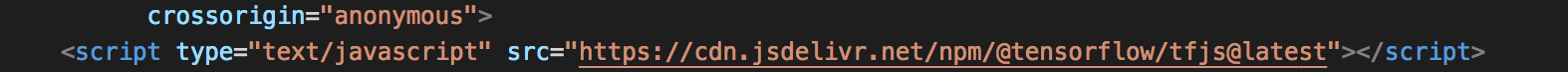
Description automatically generated

1. The below code allows user to browse and select the file using the FileReader(). Selected file is set to dataURL.

A screen shot of a computer

Description automatically generated

1. Tensorflow is loaded using the script tag.



1. Sample code to test our Web UI. The second line should confirm that the tf model is loaded successfully.

A close up of a clock

Description automatically generated

1. The console log shows the messages logged to the console and the tf object.

A screenshot of a social media post

Description automatically generated

1. The progress bar indicates the model loading.

A screenshot of a cell phone

Description automatically generated

1. The image selected by the user is displayed as below.

A screenshot of a cell phone

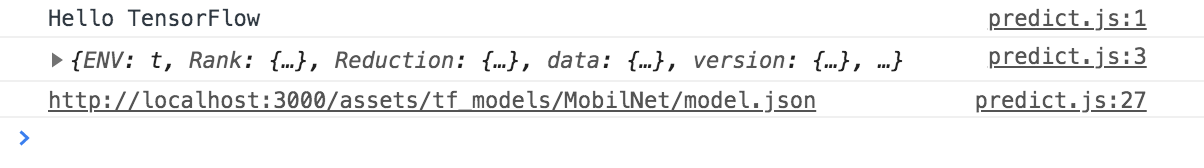
Description automatically generated

1. The model can be selected by user as below:

A screenshot of a cell phone

Description automatically generated

1. The model is loaded using await which returns a promise to be returned sometime in the future. This is wrapped within an async function and runs asynchronously.



1. The selected model after load completion.

A screenshot of a cell phone

Description automatically generated

1. When the predict button is called, this calls the predict function of the tensorflow.

A close up of a sign

Description automatically generated

1. We retrieve the first prediction for display.

A screenshot of a cell phone

Description automatically generated

## Fine Tuning the model

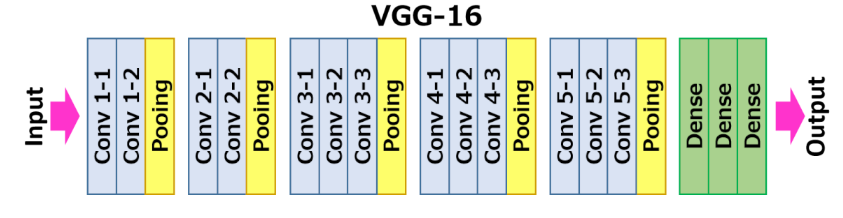
Fine tuning is a process of using weights from pre-trained models to better the performance.

I will use VGG16 and MobilNet to demonstrate the prediction to evaluate and conclude this project.

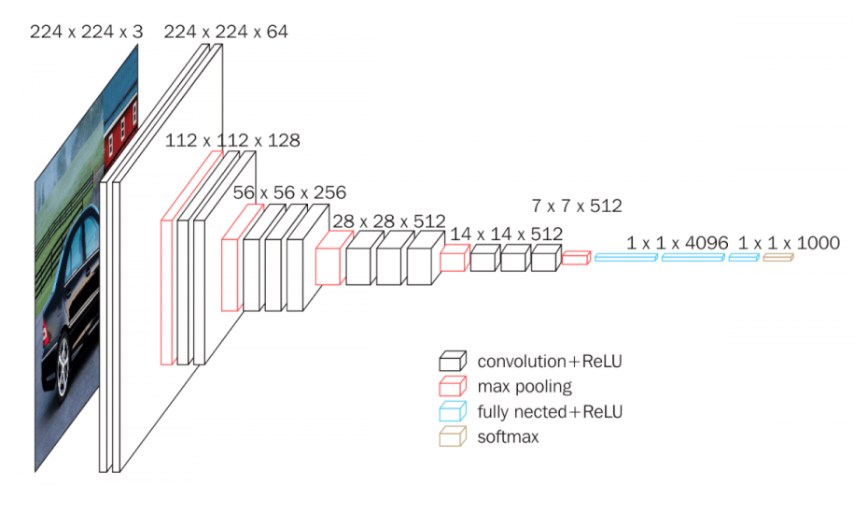
### VGG16

VGG16 is a very huge model and therefore might not be a good candidate for our browser-based prediction model. (Siddhu, 2019)

VGG16 won the best dataset award and it has over 14m images.



This model has images of size 224 x 224 and its model is represented by the below figure (Neurohive, 2018):



1. Download VGG16 model. To ensure, the output and input shape to fit my training model, we need to *set include\_top =False* and input\_tensor as the shape of my tensor model. The default size for VGG16 is (224,224,3).



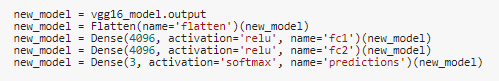
1. The size is 588 MB



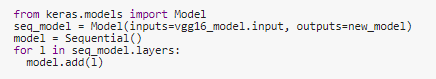
1. To re-use the weights of the pre-trained model, set trainable parameter to False.



1. Define the Dense layers and the output shape

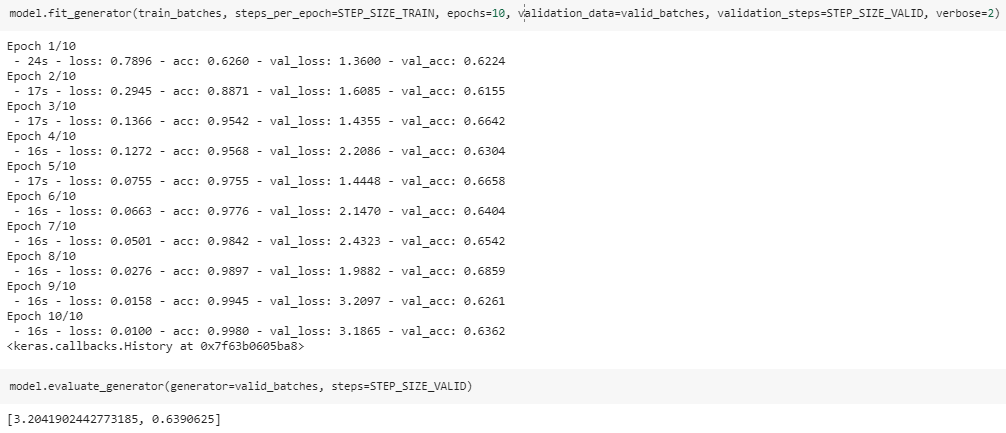


1. Add layers to current model. The current model will have the input shape same as the training batch and the output as defined above.

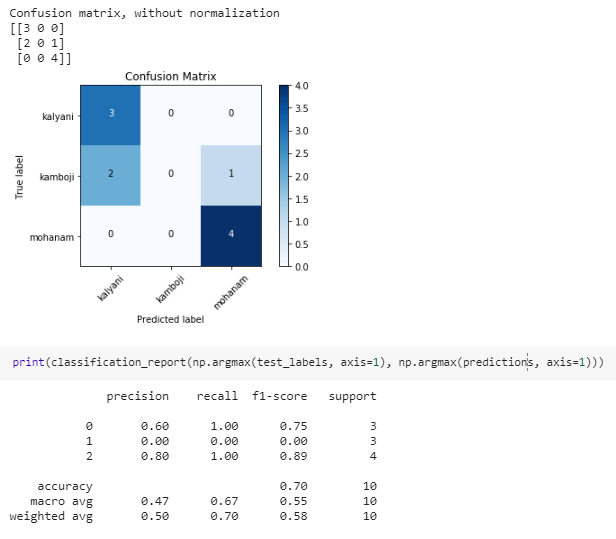


1. Compile and Fit generate and evaluate.



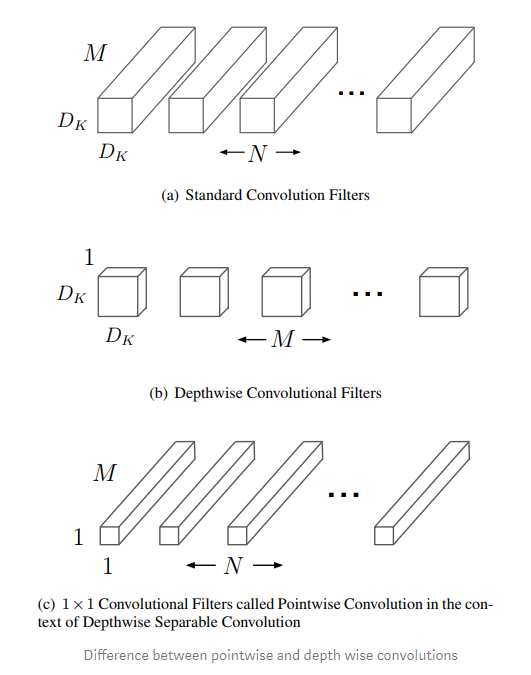


1. View Confusion matrix and Classification Report. Raga Kalyani and Raga Mohanam have 100% accuracy. Raga Kamboji is overfitting.

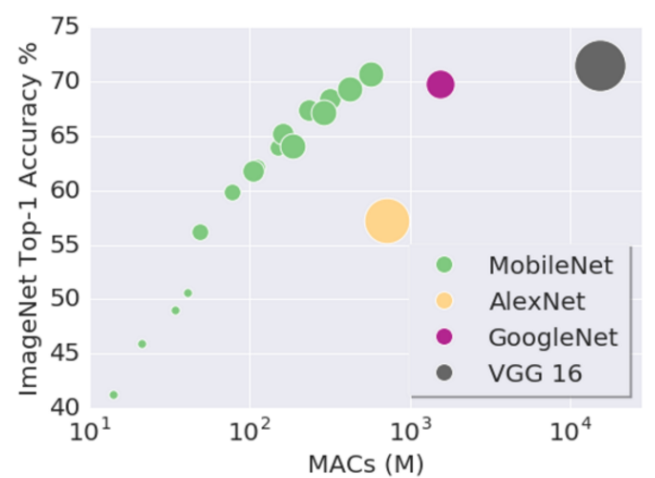


### MobileNet

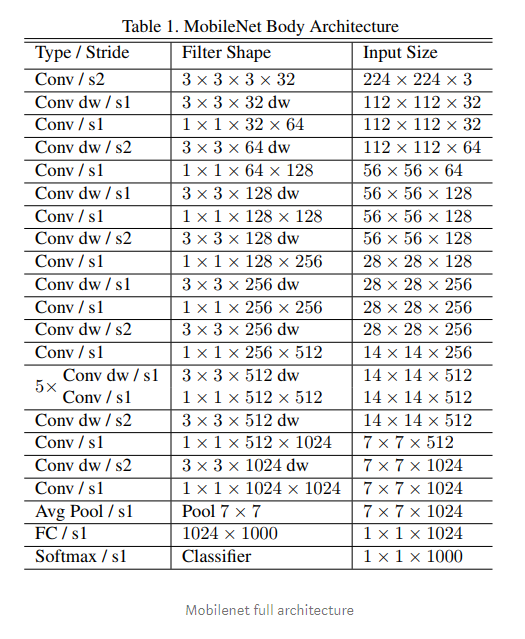
The second model is MobileNet which is a lightweight architecture and designed for mobile and browser-based solutions. It uses depth-wise separable convolutions. This means it performs single convolution on each colour channel instead of flattening and combining all 3 channels which is how VGG16 convolves. (Culfaz, 2018)



MobileNet is high-performing low maintenance model. The following figures show MobileNet performance over other Imagenet Models:



The MobileNet model architecture is as below:



1. Preprocessing Image Data for Mobilenet. Image data has to be processed in a certain format for MobileNet and therefore, note the preprocessing\_function called below:



1. Download MobileNet



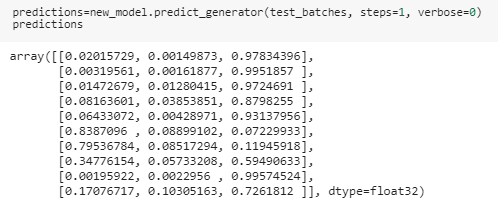
1. Change the output layer to fit in the current Model for Raga classification



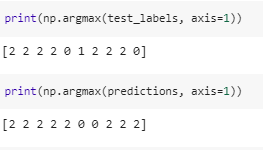
1. Compile and fit.



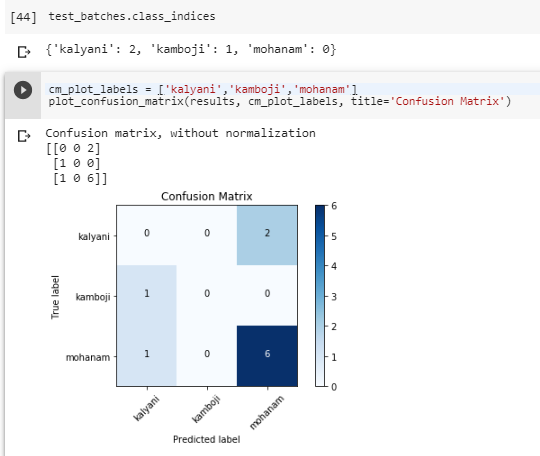
1. Prediction



1. Test vs Prediction labels



1. Confusion Matrix



# The Project Findings

## Observation:

1. The dataset is going to be unique as there is no pre-trained model for Raga classification, though there is pre-trained dataset for genre classification of but for Western music.
2. The scope of this project is audio classification through images and therefore, we will base our dataset on image-based models using CNN to predict.
3. Due to the unique nature of the dataset, the quantity of dataset available is too low for the CNN model.

### Problems with Method1 of training with Keras:

When we use visual processing using CNN with regular images, the results have been exceptionally good. With audio processing, though, the spectrograms which are 2 dimensional images of audio, the results have not been very convincing.

The reasons for this are:

1. Sounds are transparent in nature. The visual images and sounds do not add up in the same way. Though the human eyes can identify visual images to objects, it is much harder to comprehend an audio image in a similar manner. So, we say, visual images of objects are opaque while sound is transparent.
2. A pixel in a visual image can be assumed to belong to one object, while on a Spectrogram, the discrete sound events do not separate into layers.
3. The frequency in a spectrogram is a sum of its sine waves and therefore do not belong to a single sound.
4. CNNs use image filters which are 2 dimensional and the weight is shared between x and y axis. The assumption behind this is that wherever it may lie in the image, the features carry same meaning. For example, a car is expected to be the same whether moved left, right, up or down in the image.
5. Whereas in a Spectrogram, the two axes represent 2 different things – the time and frequency. Moving the frequency up or down will change the nature of the sound.
6. With harmonics, the frequencies move together in a certain pattern or relationship which complicates feature analysis of the sound using a Spectrogram.

## Evaluation, Analysis and Approach

The most widely used and accepted methodology for Audio classification is using mel-spectrogram.

Mel spectrogram “is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.” (Mansar, 2018)

Mel Spectrogram transforms raw audio input to 2-dimensional feature map with dimensions as time and frequency and the values representing amplitude.

Mel Spectrogram is proven to result in better predictions than 1D raw audio.

Taking the above scientific theory, we will follow Method 2 in implementing dataset for this project Model.

The constraint that will remain for this project is the size of the dataset as it is very limited.

### Data Augmentation

Data Augmentation is a useful method especially when you want to skew data to provide the neural net with different perspective of the data. This works very well with visual images as they are opaque and changing their perspective does not change the actual image. But with sounds which are considered to be transparent, this technique has not proven to be very useful from various testing results as below.

We have tried some of the following data augmentation techniques and concluded that it might be more effective with more epochs. But with 100 epochs it is not very effective on sound images.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| height\_shift\_range=0.5,  horizontal\_flip=True,  rotation\_range=90,  brightness\_range=[0.2,1.0],  zoom\_range=[0.5,1.0] | | | | | | | |  |
| optimizers.rmsprop(lr=0.0005, decay=1e-6) | |  |  |  |  |  |  |  |
| optimizers.Adam(lr=0.0001) |  |  |  |  |  |  |  |  |
| SGD(lr=1e-4, momentum=0.9) |  |  |  |  |  |  |  |  |

### Optimizers

Optimizers play a vital role in how the model performs. They are generally paired with parameters (Keras Support doc), and loss function (Keras Losses).

There are many optimizers available in Keras, and we will understand optimizers within the scope of this project.

#### SGD – Stochastic gradient descent

Key parameters include learning\_rate and momentum. It works very well with shallow networks.

Adam = RMSprop + Momentum

This is the most used optimizer as has low memory requirements and works with very little tuning of hyperparameters.

### Activation functions

Activation functions are like gates within the Neural net to pass or fail the result of the given inputs. Pass takes the step to the next layer to make further decistions. Fail stops the node from going any further in the network. (Keras Activations)

There are a few activation functions available but for this project, we will limit the use to RELU, Softmax and Sigmoid.

1. RELU – The rectified linear activation outputs a positive number as it is and zero otherwise. This is the default activation.
2. Softmax usually used in the output layer, uses the following function, where j is the output of one node.

softmax = [j/sum\_of\_exps for j in exps]

1. Sigmoid, also known as the Logistic activation is used to predict the probability of an output. The result always tends to be 0 or 1.

### Loss Functions

Loss function estimates the error of the current state to the actual state. This is repeated often during the neural network through forward and back propagation as part of learning and training through the model.

As this project is about classification of images into Ragas, we will be using **categorical\_crossentropy** to classify the images into different classes.

### Limitations/ Trip Ups

1. TensorflowJS, earlier known as deeplearning.js was only available in 2017 and therefore the concept of AI in Javascript is new. All earlier models were Python based and still Keras in Python is mostly used for training as this can be resource intensive on a browser.
2. The limitation to this project is resource. Though there are not a number of samples either as sound signals or as the image data of sound that exists. And the samples had to be taken from public domain, mainly youtube, to avoid any copyright infringement.
3. My hardware were both intel and the process was CPU intensive and 1 cycle of data fit took over hours to complete in CPU. I needed Nvidia to run the CUDA API that supports AI in GPU and both my Macbook and PC had only Intel default GPU and therefore, not fit for purpose unless it was upgraded.
4. I later found after I visited a google AI conference, that Google Colab comes with built-in GPU ready to use. With Google Colab, there was still issues with Resource management etc. where there were very little articles available and therefore required a lot of search on fixes.

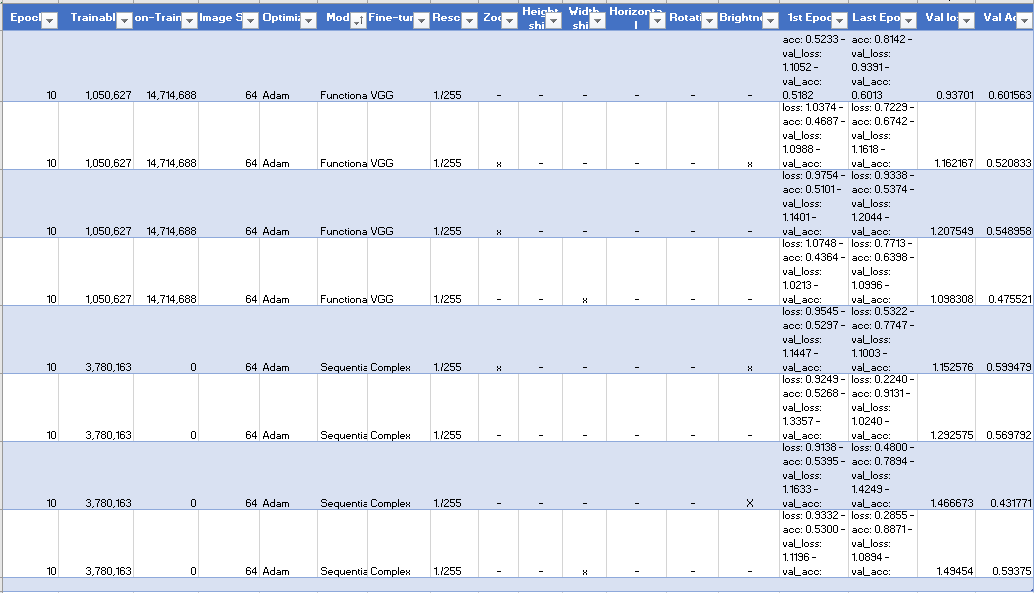
# Test Results

Due to the time it takes to train these models, I have only performed sample testing taking a few test scenarios as published in the next page.

### Test Inference

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Pre-trained? | Epochs | Validation Accuracy | Comments |
| Adam | No | 10 | 59% | This was a simple convolutional Model created from scratch. |
| VGG16 | Yes | 10 | 60% | Each cycle took upto 40 mins. |
| MobileNet | Yes | 10 | 69% | Each cycle took up to 30 mins. |

### Test Sampling Results



# Conclusion

The initial plan was to write this project in Angular JS and use TensorflowJS for model creation, training and prediction. The complexity of this module required server-side scripting and therefore Angular JS did not seem to be the right candidate as it will only fit a part of the whole plan which is to manage data.

Node JS seem to have integrated with most APIs required including Web Audio GL for recording, to processing pre-trained models in the browser for prediction.

I had to resort to Python for processing audio to desired format due to the simplicity of the code. Though audio processing was not the goal of the project, it benefitted from data cleaning and producing Mel-Spectrogram for better results.

There is much to improve on this project, as Neural Net in Javascript is a very new subject and Google is still encouraging more people to use TensorflowJS even for Modelling. TensorflowJS is in its baby stage and a lot more improvement can be expected in the coming years to make it robust and capable as Python APIs, which has huge number of users and in a more mature stage.

### The selection of the Raga Classes

(Hitxp)

The following table provides the similarities and differences between the Raga Classes I have selected for the project.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | **Mohanam** | | **Kamboji** | | **Kalyani** | |
| **Indian Notation** | **Western Equivalent** | **Keyboard key** | **Asc** | **Desc** | **Asc** | **Desc** | **Asc** | **Desc** |
| s | C | First key in an octave | s | s | s | s | s | s |
| r1 | C# (C Sharp) | Second key in an octave |  |  |  |  |  |  |
| r2 / g1 | D | Third key in an octave | r2 | r2 | r2 | r2 | r2 | r2 |
| r3 / g2 | D# (D Sharp) | Fourth key in an octave |  |  |  |  |  |  |
| g3 | E | Fifth key in an octave | g3 | g3 | g3 | g3 | g3 | g3 |
| m1 | F | Sixth key in an octave |  |  | m1 | m1 |  |  |
| m2 | F# (F Sharp) | Seventh key in an octave |  |  |  |  | m2 | m2 |
| p | G | Eighth key in an octave | p | p | p | p | p | p |
| d1 | G# (G Sharp) | Ninth key in an octave |  |  |  |  |  |  |
| d2 / n1 | A | Tenth key in an octave | d2 | d2 | d2 | d2 |  | d2 |
| d3 / n2 | A# (A Sharp) | Eleventh key in an octave |  |  |  | n2 |  |  |
| n3 | B | Twelveth key in an octave |  |  |  |  | n3 | n3 |

### Key features of the 3 chosen Raga Classes

0 - Mohanam

|  |
| --- |
| **Arohanam/Ascending:   S R2 G3 P D2 S** |
| **Avarohanam/Descending:   S D2 P G3 R2 S** |

This is a five-scale raga.

1 - Kamboji

|  |
| --- |
| **Arohanam/Ascending:   S R2 G3 M1 P D2 S** |
| **Avarohanam/Descending:   S N2 D2 P M1 G3 R2 S** |

This is an unequal scale raga where the number of ascending and descending notes differ.

2 – Kalyani

|  |
| --- |
| **Arohanam/Ascending:   S R2 G3 M2 P D2 N3 S** |
| **Avarohanam/Descending:   S N3 D2 P M2 G3 R2 S** |

This is a complete Raga where all 7 notes are in order.

**Justification**: The selection of ragas was decided by resource availability and to choose varying Raga to evaluate through the neural net.

#### Observations:

1. Clearly stresses the importance of having a varied and large dataset for the neural net to uniquely pick the differences. For this project I was unable to gather enough data as I had limited resources available (youtube) to collect data from.
2. There is also the signature of the raga how each note is played in relation to the other, which the neural net will be able to pick as the features of each Raga which cannot be visualized.

## Models

### VGG16

Award winning dataset and complex model which is very successful in image classification.

Drawbacks: Too huge and therefore very slow to process. Just the weights alone are very large.

Each cycle for the current dataset took over 40 mins.

Inference: Has performed relatively well with very small dataset for training on the unusual images which are nowhere similar to the image dataset it has been trained on.

I have been able to achieve a 60% accuracy.

### MobileNet

Lightweight model and dataset suitable for browser-based applications.

Drawbacks: Smaller dataset. Requires an extra level of image pre-processing as it expects a specific data format.

I have been able to achieve 69 % accuracy.

Inference: This surpassed the performance of VGG16 though a considerably smaller dataset. The reason could be how the model filters the images by individual pixes unlike VGG16 which does in blocks.

## INFERENCE

The following were inferred as outcome of this project:

1. ADAM was the best optimizer of the chosen amongst Adam, SGD and RMSProp.
2. MobileNet model was the best performing with 69% validation results
3. For some reason, test predictions were not favourable to Raga Kamboji. The reason could be that Kamboji has similarities with other 2 ragas selected and therefore, most images except a few had more unique features.
4. This project completed all its objectives, but there are many areas for improvement, which couldn’t be performed within the scope of the project due to limited time and resources.

## Future improvements

1. Though CNN has been known to be the ideal model for image classification, it had proven to be very resource intensive. For the audio images, it is claimed that LSTM perform better, but as this is more resource intensive than CNN, this might be a candidate to work in future to better the current state of the project.
2. This can also be taken further for PHD research purposes.
3. This project was a simple endeavour but a huge learning experience with very limited resource. With technology advancing each day, there will be better scopes and possibilities in terms of resources and new capabilities.
4. A good size dataset would be atleast a minimum of 5 to 10 hours of recording of each Raga, which would produce phenomenal results, based on the current results.
5. The 3s sampling was a good start. But for a complex algorithm such as raga classification, it might help to have longer clips to provide the model a better view of notes in relation to other notes surrounding it.

# Works Cited

(Unknown), S. (n.d.). *1.3.1. The NumPy array object — Scipy lecture notes*. Retrieved from Scipy-lectures.org: https://scipy-lectures.org/intro/numpy/array\_object.html

Brownlee, J. (2019, Jul 5). *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*. Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/

*Building powerful image classification models using very little data*. (2019). Retrieved from Blog.keras.io: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Chakraborty, S., & De, D. (2012, Dec). Pattern Classification of Indian Classical Ragas based on Object Oriented Concepts. Kolkata, West Bengal, India. Retrieved from Pdfs.semanticscholar.org: https://pdfs.semanticscholar.org/77bf/e8b5491037a357cc2d012fc312e373a26370.pdf

Chroma (Unknown). (n.d.). *chroma.* Retrieved from Musicinformationretrieval.com: https://musicinformationretrieval.com/chroma.html

Culfaz, F. (2018, Nov 6). *Transfer Learning using Mobilenet and Keras.* Retrieved from Medium: https://towardsdatascience.com/transfer-learning-using-mobilenet-and-keras-c75daf7ff299

DataGenetics. (2019). *Discrete Cosine Transformations*. Retrieved from Datagenetics.com: http://datagenetics.com/blog/november32012/index.html

Deshpande, A. (2016, Jul 29). *A Beginner's Guide To Understanding Convolutional Neural Networks Part 2*. Retrieved from Adeshpande3.github.io: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

Dwivedi, P. (2018, Dec 13). *Using CNNs and RNNs for Music Genre Recognition.* Retrieved from Medium: https://towardsdatascience.com/using-cnns-and-rnns-for-music-genre-recognition-2435fb2ed6af

ericz. (2013). *Package - binaryjs*. Retrieved from Npm.taobao.org: https://npm.taobao.org/package/binaryjs

Fayek, H. (2019). *Speech Processing for Machine Learning: Filter banks, Mel-Frequncy Cepstral Coefficients(MFCCs) and What's In-Between.* Retrieved from Haytham Fayek: https://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

*gabrielpoca/browser-pcm-stream*. (2015, Oct 25). Retrieved from GitHub: https://github.com/gabrielpoca/browser-pcm-stream/blob/master/tpl/index.jade

Geisendörfer, F. (2009, Sep 28). *Streaming file uploads with node.js » Debuggable - Node.js Consulting*. Retrieved from Debuggable.com: http://www.debuggable.com/posts/streaming-file-uploads-with-node-js%3A4ac094b2-b6c8-4a7f-bd07-28accbdd56cb

Heaton, J. (Director). (2019). *2019, Installing TensorFlow, Keras, & Python 3.7 in Mac* [Motion Picture].

Hitxp. (n.d.). *Western Notations Guide to Convert Classical Indian Carnatic Notations - Western Notations | HitXP by Gurudev.* Retrieved from Hitxp.com: http://www.hitxp.com/keyboard-music-notes/western-notations-guide-convert-classical-indian-carnatic/

JEF AKST. (2019, May). *A Primer: Artificial Intelligence Versus Neural Networks.* Retrieved from The Scientist Magazine®: https://www.the-scientist.com/magazine-issue/artificial-intelligence-versus-neural-networks-65802

Keras Activations. (n.d.). *Usage of activations.* Retrieved from keras.io: https://keras.io/activations/

Keras Losses. (n.d.). *Losses - Keras Documentation.* Retrieved from Keras.io: https://keras.io/losses/

Keras Support doc. (n.d.). *Optimizers - Keras Documentation.* Retrieved from Keras.io: https://keras.io/optimizers/

Mansar, Y. (2018, Apr 23). *Audio Classification : A Convolutional Neural Network Approach.* Retrieved from Medium: https://medium.com/@CVxTz/audio-classification-a-convolutional-neural-network-approach-b0a4fce8f6c

many. (2019, Sep 29). *Sample for report\_tensor\_allocations\_upon\_oom and RunOptions . Issue #17076 . tensorflow/tensorflow.* Retrieved from GitHub: https://github.com/tensorflow/issues/17076

Marr, B. (2018, Oct 1). *What Is Deep Learning AI? A Simple Guide With 8 Practical Examples.* Retrieved from Forbes.com: https://www.forbes.com/sites/bernardmarr/2018/10/01/what-is-deep-learning-ai-a-simple-guide-with-8-practical-examples/

Michalak, P. K. (2016, Oct 26). *Music Genre Recognition*. Retrieved from DeepSound: http://deepsound.io/music\_genre\_recognition.html

*Model conversion  |  TensorFlow.js*. (2019). Retrieved from TensorFlow: https://www.tensorflow.org/js/guide/conversion

*Model conversion  |  TensorFlow.js*. (2019). Retrieved from TensorFlow: http://tensorflow.org/js/guide/conversion

Nair, P. (2018, Jul 24). *The dummy’s guide to MFCC*. Retrieved from Medium: https://medium.com/prathena/the-dummys-guide-to-mfcc-aceab2450fd

Neurohive. (2018, Nov 20). *VGG16 - Convolutional Network for Classification and Detection.* Retrieved from Neurohive.io: https://neurohive.io/en/popular-networks/vgg16/

noamc. (2014, Aug 31). *noamtcohen/AudioStreamer*. Retrieved from GitHub: https://github.com/noamtcohen/AudioStreamer/blob/master/public/js/site.js

orefalo. (2012, Sep 5). *binaryjs/binaryjs*. Retrieved from GitHub: https://github.com/binaryjs/binaryjs/blob/master/examples/fileupload/server.js

Pandey, P. (2018, Dec 13). *Music Genre Classification with Python.* Retrieved from Medium: https://towardsdatascience.com/music-genre-classification-with-python-c714d032f0d8

*Playing PCM stream from Web Audio API on Node.js*. (2019). Retrieved from Solved: https://code.i-harness.com/en/q/13e8b78

Saha, S. (2018, Dec 15). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way.* Retrieved from Medium: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Siddhu, R. (2019, Mar 22). *Audio Classification Using CNN — An Experiment.* Retrieved from Medium: https://medium.com/x8-the-ai-community/audio-classification-using-cnn-coding-example-f9cbd272269e

Singh, J. (2018, Dec 20). *Audio and Image Features used for CNN.* Retrieved from Medium: https://medium.com/datadriveninvestor/audio-and-image-features-used-for-cnn-4f307defcc2f

Steven W. Smith, P. (2019). *How the FFT works*. Retrieved from Dspguide.com: https://www.dspguide.com/ch12/2.htm

Tara N. Sainath, O. V. (n.d.). CONVOLUTIONAL, LONG SHORT-TERM MEMORY, FULLY CONNECTED DEEP NEURAL NETWORKS. New York, USA.

tfjs. (2019). *tfjs-converter.* Retrieved from tfjs-converter: https://huningxin.github.io/tfjs-converter/

Thompson, J. (2017, Apr 17). *Audio Signals in Python*. Retrieved from Inspiration Information: http://myinspirationinformation.com/uncategorized/audio-signals-in-python/

*Web Audio API*. (2019). Retrieved from MDN Web Docs: http://developer.mozilla.org/en-US/docs/Web/API/Web\_Audio\_API

Xu, A. Y. (2019, Feb 27). *Urban Sound Classification using Convolutional Neural Networks with Keras: Theory and Implementation.* Retrieved from Medium: https://medium.com/gradientcrescent/urban-sound-classifications-using-convolutional-neural-networks-with-keras-theory-and-486e92785df4

# Other References (Playlists)

Heaton, J. (2019). *Learn Deep Learning with Python, Keras and TensorFlow with Applications of Deep Neural Networks*. [podcast] 2019 Version of Applications of Deep Neural Network. Available at: https://www.youtube.com/watch?v=LCXEuJSwd1E&list=PLjy4p-07OYzulelvJ5KVaT2pDlxivl\_BN [Accessed 8 Oct. 2019].

Ng, A. (2016). *Lecture 1.1 — Introduction What Is Machine Learning — [ Machine Learning | Andrew Ng ]*. [podcast] Machine Learning — Andrew Ng, Stanford University [FULL COURSE]. Available at: https://www.youtube.com/watch?v=PPLop4L2eGk&list=PLLssT5z\_DsK-h9vYZkQkYNWcItqhlRJLN&index=2&t=23s [Accessed 8 Oct. 2019].

Karpathy, A. (2016). *CS231n Winter 2016: Lecture1: Introduction and Historical Context*. [podcast] CS231n Winter 2016. Available at: https://www.youtube.com/playlist?list=PLkt2uSq6rBVctENoVBg1TpCC7OQi31AlC [Accessed 8 Oct. 2019].

# Appendix

## Appendix 1 - Source Code

All source code files are available in my github repository: <http://github.com/sribarrow/audiorecording>

Required environment: NodeJS, Python 3.6

I had installed Miniconda to create multiple environment sandboxes.

For the NodeJS app, npm install will install all dependencies from package.json file.

For Python, various dependencies have to be manually installed.

Installation procedures for Python can be from the following source:

(Heaton, 2019)

## Appendix 2 - Flow Diagram

A close up of a map

Description automatically generated

## Appendix 2 – Use Case

A picture containing text, map

Description automatically generated

## Appendix 4 – TroubleShooting

|  |  |
| --- | --- |
|  | **Error:** [**NET::ERR\_CERT\_AUTHORITY\_INVALID**](https://stackoverflow.com/questions/47700269/google-chrome-localhost-neterr-cert-authority-invalid)  If while recording, the above error in Google Chrome occurs, action the solution below.  Solution:  1. Go to <chrome://flags/#allow-insecure-localhost>  2. Enable the below option |
|  | *Error " Sequential.fromConfig called without an array of configs*  *Solution – This occurs when keras model is trained and saved in gpu environment.* |
|  | *Error loadModel from url doesn't work in Node*  *Deprecated. loadLayersmodel function to be used.* |
|  | *Fetch argument has invalid type*  *Same as 3.* |

### Troubleshooting References

GitHub. (2019). *Sample for report\_tensor\_allocations\_upon\_oom and RunOptions · Issue #17076 · tensorflow/tensorflow*. [online] Available at: https://github.com/tensorflow/tensorflow/issues/17076 [Accessed 8 Oct. 2019].

GitHub. (2019). *Resource exhausted: OOM when allocating tensor | Problem to compute on GPU · Issue #461 · rstudio/keras*. [online] Available at: https://github.com/rstudio/keras/issues/461 [Accessed 8 Oct. 2019].

shape, U., Beker, D. and Mathew, J. (2019). *Understanding the ResourceExhaustedError: OOM when allocating tensor with shape*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/46066850/understanding-the-resourceexhaustederror-oom-when-allocating-tensor-with-shape [Accessed 8 Oct. 2019].

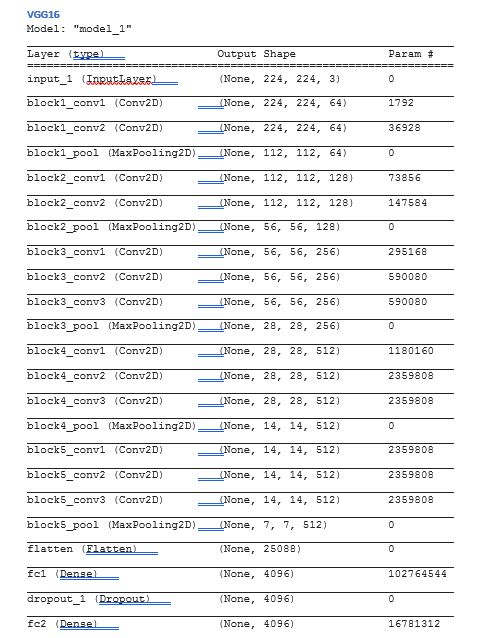
&rdquo;, E., Aryan, S., Aryan, S. and Singh, P. (2019). *Error " Sequential.fromConfig called without an array of configs "*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/52566747/error-sequential-fromconfig-called-without-an-array-of-configs [Accessed 8 Oct. 2019].

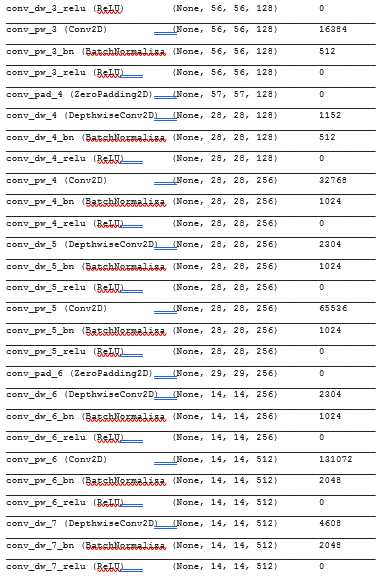
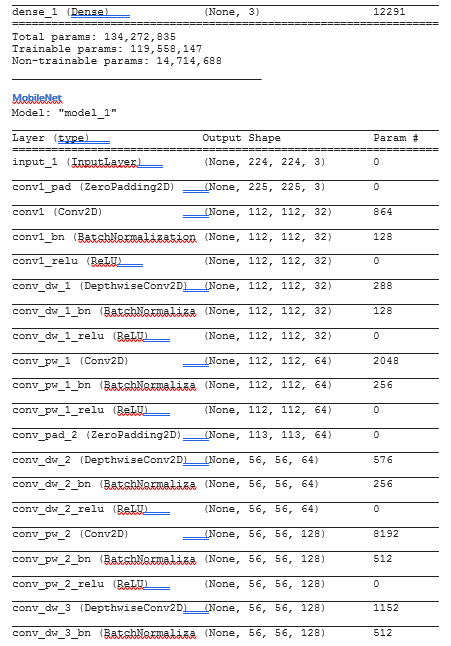
type, T., Iona, H., Iona, H. and Iona, H. (2019). *Tensorflow neural network TypeError: Fetch argument has invalid type*. [online] Data Science Stack Exchange. Available at: https://datascience.stackexchange.com/questions/14429/tensorflow-neural-network-typeerror-fetch-argument-has-invalid-type?rq=1 [Accessed 8 Oct. 2019].

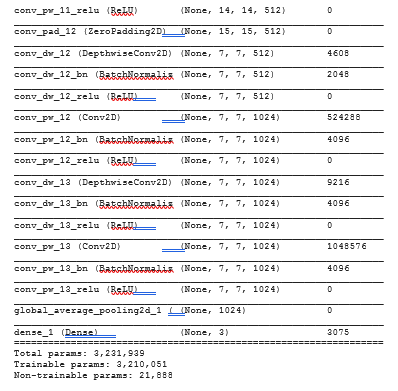
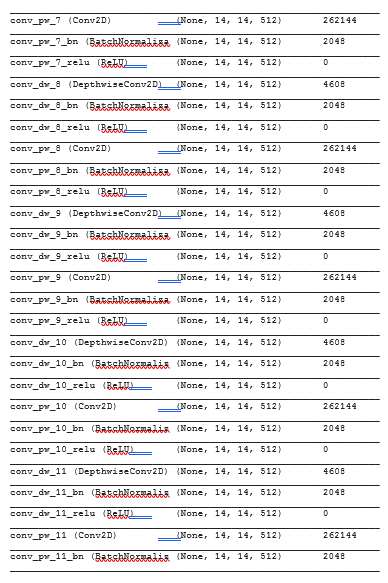
GitHub. (2019). *loadModel from url doesn't work in Node · Issue #410 · tensorflow/tfjs*. [online] Available at: https://github.com/tensorflow/tfjs/issues/410 [Accessed 8 Oct. 2019].

function&rdquo;?, W. (2019). *Why am I getting the error "tf.fromPixels is not a function"?*. [online] Stack Overflow. Available at: https://stackoverflow.com/questions/55402551/why-am-i-getting-the-error-tf-frompixels-is-not-a-function [Accessed 8 Oct. 2019].

## Appendix 5 – Pre-trained Model Summary







## Appendix 6 – Project Plan & Gantt Chart

