

# **VIRTUAL GUITAR SIMULATOR USING COMPUTER VISION**

**A PROJECT REPORT**

*Submitted By*

**SHASHANK PANDA      195001102**

**TUSHAR NAIR      195001116**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**



**Department of Computer Science and Engineering**

**Sri Sivasubramaniya Nadar College of Engineering**

**(An Autonomous Institution, Affiliated to Anna University)**

**Kalavakkam - 603110**

**May 2023**

# **Sri Sivasubramaniya Nadar College of Engineering**

**(An Autonomous Institution, Affiliated to Anna University)**

## **BONAFIDE CERTIFICATE**

Certified that this project report titled “**VIRTUAL GUITAR SIMULATOR USING COMPUTER VISION**” is the *bonafide* work of “**SHASHANK PANDA (195001102)** and **TUSHAR NAIR (195001116)**” who carried out the project work under my supervision.

Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

**Dr. T. T. MIRNALINEE**  
**HEAD OF THE DEPARTMENT**

Professor,  
Department of CSE,  
SSN College of Engineering,  
Kalavakkam - 603 110

**Dr. A. BEULAH**  
**SUPERVISOR**

Assistant Professor,  
Department of CSE,  
SSN College of Engineering,  
Kalavakkam - 603 110

Place:

Date:

Submitted for the examination held on.....

**Internal Examiner**

**External Examiner**

## ACKNOWLEDGEMENTS

We thank God, the almighty for giving us strength and knowledge to do this project.

We would like to thank and express our deep sense of gratitude to our guide **Dr. A. BEULAH**, Assistant Professor, Department of Computer Science and Engineering, for her valuable advice and suggestions as well as her continued guidance, patience and support that helped us to shape and refine our work.

Our sincere thanks to **Dr. T. T. MIRNALINEE**, Professor and Head of the Department of Computer Science and Engineering, for her words of advice and encouragement and we would like to thank our project Coordinator **Dr. B. BHARATHI**, Associate Professor, Department of Computer Science and Engineering for her valuable suggestions throughout this project.

We express our deep respect to the founder **Dr. SHIV NADAR**, Chairman, SSN Institutions. We also express our appreciation to our Principal **Dr. V. E. ANNAMALAI**, for all the help he has rendered during this course of study.

We would like to extend our sincere thanks to all the teaching and non-teaching staffs of our department who have contributed directly and indirectly during the course of our project work. Finally, we would like to thank our parents and friends for their patience, cooperation and moral support throughout our lives.

**SHASHANK PANDA**

**TUSHAR NAIR**

## **ABSTRACT**

The efforts in the field of virtualisation of the guitar instrument into well modeled software systems has been riddled with practical limitations. This project aims to significantly remove the inaccuracies and drawbacks of existing solutions by accounting for the individual roles that each hand plays in the act of guitar strumming and consolidate them into a single system. The existing solutions in this area either focus on half the problem or do not model the real life mechanics of guitar playing. Use of extra apparatus is involved in some cases. These drawbacks can be addressed by development of a consolidated system which encapsulates all aspects of guitar playing and does not require any apparatus other than a reasonably good computer webcam. Since the project uses computer vision, it does not require any extraneous hardware and the modulation of the system into left and right hand modules tackles the partial solutions discussed prior. The proof of concept of the proposed system is to demonstrate that all the nuances of guitar playing can be captured simplistically using only computer vision and machine learning. Experiments to validate the system involves comparing the chord classification to actual chord intended by the user over a series of attempts, which is formulated into a confusion matrix and metrics are derived from the same.

# TABLE OF CONTENTS

<b>ABSTRACT</b>	<b>iii</b>
<b>LIST OF TABLES</b>	<b>vi</b>
<b>LIST OF FIGURES</b>	<b>vii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 MOTIVATION . . . . .	1
1.2 BACKGROUND . . . . .	2
1.3 PROBLEM DEFINITION . . . . .	3
1.4 ORGANIZATION OF THE REPORT . . . . .	4
<b>2 LITERATURE SURVEY</b>	<b>6</b>
2.1 EXISTING VIRTUAL MUSIC SYSTEMS . . . . .	6
2.2 RESEARCH OBJECTIVES . . . . .	10
<b>3 PROPOSED METHODOLOGY</b>	<b>12</b>
3.1 MEDIAPIPE . . . . .	14
3.1.1 Palm Detection Model . . . . .	15
3.1.2 Hand Landmark Model . . . . .	16
3.1.3 Implementation in MediaPipe . . . . .	18
3.2 LEFT HAND MODULE . . . . .	20
3.2.1 Dataset Creation . . . . .	20
3.2.2 Chord Detection Model Training . . . . .	22

3.2.3	Chord Identification . . . . .	24
3.3	GUITAR SOUND BANK . . . . .	25
3.4	RIGHT HAND MODULE . . . . .	29
3.4.1	Velocity Detection using OpenCV . . . . .	29
3.4.2	Hosting Model on the Cloud . . . . .	30
<b>4</b>	<b>EXPERIMENTAL RESULTS</b>	<b>33</b>
4.1	DATASET DESCRIPTION . . . . .	33
4.2	ECOSYSTEM . . . . .	34
4.3	EXPERIMENTS CONDUCTED . . . . .	36
<b>5</b>	<b>PERFORMANCE ANALYSIS</b>	<b>42</b>
5.1	CONFUSION MATRIX . . . . .	42
5.2	USER INSIGHT ANALYSIS . . . . .	44
<b>6</b>	<b>SOCIAL IMPACT AND SUSTAINABILITY</b>	<b>46</b>
<b>7</b>	<b>CONCLUSIONS AND FUTURE WORK</b>	<b>48</b>
	<b>REFERENCES</b>	<b>50</b>

## LIST OF TABLES

5.1	Model performance . . . . .	42
5.2	Survey results . . . . .	44

## LIST OF FIGURES

2.1	Setup of the drum set . . . . .	7
2.2	Guitar system using red gloves . . . . .	9
2.3	Different phases of the EMG aided guitar project . . . . .	10
3.1	Detailed system architecture . . . . .	13
3.2	Palm detector model architecture . . . . .	15
3.3	Hand landmark model architecture . . . . .	16
3.4	21 Landmark coordinates . . . . .	17
3.5	Mediapipe pipeline . . . . .	18
3.6	Mediapipe Hands . . . . .	19
3.7	Recording and processing of 1200 images for each chord . . . . .	21
3.8	Images for a single chord . . . . .	22
3.9	Convolutional Neural Network . . . . .	23
3.10	Procedure of a 2-D CNN . . . . .	24
3.11	Chord detection using model . . . . .	25
3.12	DAW - Reaper . . . . .	26
3.13	Audio Interface - iRig . . . . .	27
3.14	Guitar - Yamaha F130 . . . . .	27
3.15	Directory containing chord .wav files . . . . .	28
3.16	Placement of hitboxes for sensing the strum . . . . .	29
3.17	Bucket details . . . . .	30
3.18	SSH access to VM from browser window . . . . .	31
4.1	Sample image for each chord . . . . .	34
4.2	Left hand chord detection . . . . .	37
4.3	Landmark Coordinate tracking . . . . .	38
4.4	Playing classified chord sound . . . . .	39
4.5	Virtual Guitar System running locally . . . . .	40
4.6	Virtual Guitar System on the cloud . . . . .	41
5.1	Confusion Matrix . . . . .	42
5.2	Performance metrics from confusion matrix . . . . .	43



## CHAPTER 1

# INTRODUCTION

The classical guitar is a 6-stringed instrument with a hollow wooden body. It has been one of the oldest and most popular forms of music for many centuries. There are more than 10 different types of guitars today, differentiable by the method of playing or appearance. Depending on the scope and position of the guitarist, there are a wide array of accessories that would be necessary to aid in the playing of this instrument. But one commonality that guitarists face is the difficulty of maintaining, storing and transporting such a huge and expensive instrument.

## 1.1 MOTIVATION

There exists a gap in the bridge that joins physical instruments to their software counterparts. Simulation of the natural muscle movements that trigger sounds according to the instrument is not effectively created in software. Pianos, organs and synthesizers can be accurately represented on computer keyboards to an extent but larger, more complicated instruments such as guitars, flutes and violins are yet to be modelled in such a concise way.

A big problem faced by musicians is the unavailability of their instruments at the inception of musical ideas. Guitars in particular aren't carried around too often due to them generally being heavy, space consuming and prone to damage. Not knowing when inspiration will strike, potential loss of Musical Intellectual Property can occur. An accessible and easy to use software system, that doesn't

require any physical apparatus, and encompasses all elements of guitar playing removes this hurdle.

## **1.2 BACKGROUND**

In the market currently there exist a lot of software applications that enable the playing of an instrument by employing some limitations. In case of drums, the individual components would have to be clicked or tapped on to produce the sound. While for guitars, the strumming and chord selection need to be preset manually and then run. This does not give an accurate feel of how the instruments are played in real life. There also exists some amount of learning that needs to be done due to absence of a standard when it comes to developing a simulator as well as the understanding how this mechanism works.

The next breakthrough involved the use of some additional physical apparatus such Electromyogram (EMG) controllers [5], or Musical Instrument Digital Interface (MIDI) recording devices to recreate the feeling of playing a guitar. The EMG device is still a work in progress and an expensive proposition which will not be received well by the general public. Based on a survey conducted by the team, the apparatus are not very well received by instrumentalists and they would rather play the traditional instrument. With the existing guitar simulation technology, strumming velocity can be identified with EMG and chord shape can be recognised individually by Computer Vision, but there are no solutions which provide a consolidated software that identifies chord shape of hand gesture and strumming action simultaneously.

## 1.3 PROBLEM DEFINITION

To create a software system using the concepts of computer vision, which accurately models the real life experience of playing a guitar, accounting for the nuances of muscle movements without the need of any additional sensory apparatus.

- Dataset creation by recording pictures of hand gesture corresponding to chords on the guitar.
- Training and validating CNN model to recognize chords using the created dataset.
- Classification of chords using developed model upon showing gesture.
- Velocity detection of strumming hand (right hand)
- Inferring chord and velocity from human actions to play sound from the database.

The above requirements must be consolidated into an easy to use and smooth-running system that seamlessly gathers real time inputs from a user's video feed by differentiating the left hand from right, and thus prescribing the operations to each respective hand.

Left hand must focus on the formation of the guitar chord using one's fingers, and the right hands motion must trigger the strumming sound of a guitar, once it passes the designated string area.

## **1.4 ORGANIZATION OF THE REPORT**

This report is organized in a chapter format where each chapter addresses a specific aspect of the project work. A brief overview of the content and structure of the report is given below.

In Chapter Two, Literature Survey was done on this problem by scouring through many research papers and blog posts in order to understand what were the various methodologies employed by other researchers and which are the gaps that one could aim to fill.

In Chapter Three, the various modules that were designed in order to create the fully working Virtual Guitar application are illustrated. From the most primitive part, each component has been explained thoroughly to understand how the system works. To understand it in a very broad sense, we require two modules, namely, the Left and Right Hand modules.

In Chapter Four, the Results of experimentation are presented, in which a series of tests and surveys are included . The data on which the experiments were performed on is also laid out.

In Chapter Five, Performance Analysis shows the different metrics of interest pertaining to the application based on the results, like accuracy, precision and F1 score to understand how well the system works with some quantifiable evidence.

In Chapter Six, we understand just how much of an impact this system would have on society and if it has a solid foreseeable future. Developing on the results, we arrive at a conclusion backed by firm evidence and discuss the potential areas to be improved to better run the system which currently lie out of the scope of the developers.

Finally in Chapter Seven, conclusions and scopes for future development that can be made based on the project are laid out.

The end of the thesis, References, lists the resources from where we compared and contrasted various techniques, ideas and methods to finally arrive at a state-of-the-art Virtual Guitar application.

## CHAPTER 2

# LITERATURE SURVEY

## 2.1 EXISTING VIRTUAL MUSIC SYSTEMS

There have been numerous attempts to virtualize musical instruments ranging from guitars to drums to xylophones. Different techniques have been employed to make this work. While some are modelled basing them off of the natural method of playing, others achieve a perfectly working system for the instrument in process of making a new method of playing which involves some amount of additional knowledge specific to that system.

In the system proposed by Bering et. al. [1], they aim to use computer vision and OpenCV libraries to create a virtual drum simulator. To do this, image processing techniques such as thresholding and applying of Gaussian filters were used to extract the contours of the hand from a bounding rectangle. The hands were tracked successfully to strike different areas on the screen to play the corresponding sound. The limitation of this system is that the face gets detected as a hand and users have to keep their head outside the frame of the camera.

The work done by Carl Timothy Tolentino et. al. [4] focuses on providing users with an affordable alternative to purchasing entire drum sets in order to learn drums. They adopt the methodology of computer Vision to do so. Similar to our efforts, only the webcam of a computer is put in use to drive the system. The pipeline constructed in this project is as follows: object detection, event detection, audio synthesis. The use of color-coded drum sticks is adopted, in order to

identify the apparatus to trigger the behavior of a drum on air. Additionally, a yellow slip of paper is stuck to the knee of the player to trigger the kick drum, a piece of equipment that is triggered by the stomping of a pedal in a real drum kit as can be seen in figure 2.1

The author was able to overcome one limitation of this system by adding using color coded blobs on sticks and also adding foot functionalities such as bass drum and hi-hat control. But one downside to this system is that objects of the same/similar color as the apparatus indicators must not be present in the frame, else the result will be undesirable. Blob detection is implemented on the color keypoints, and their movements can be approximated to trigger the intended sound.

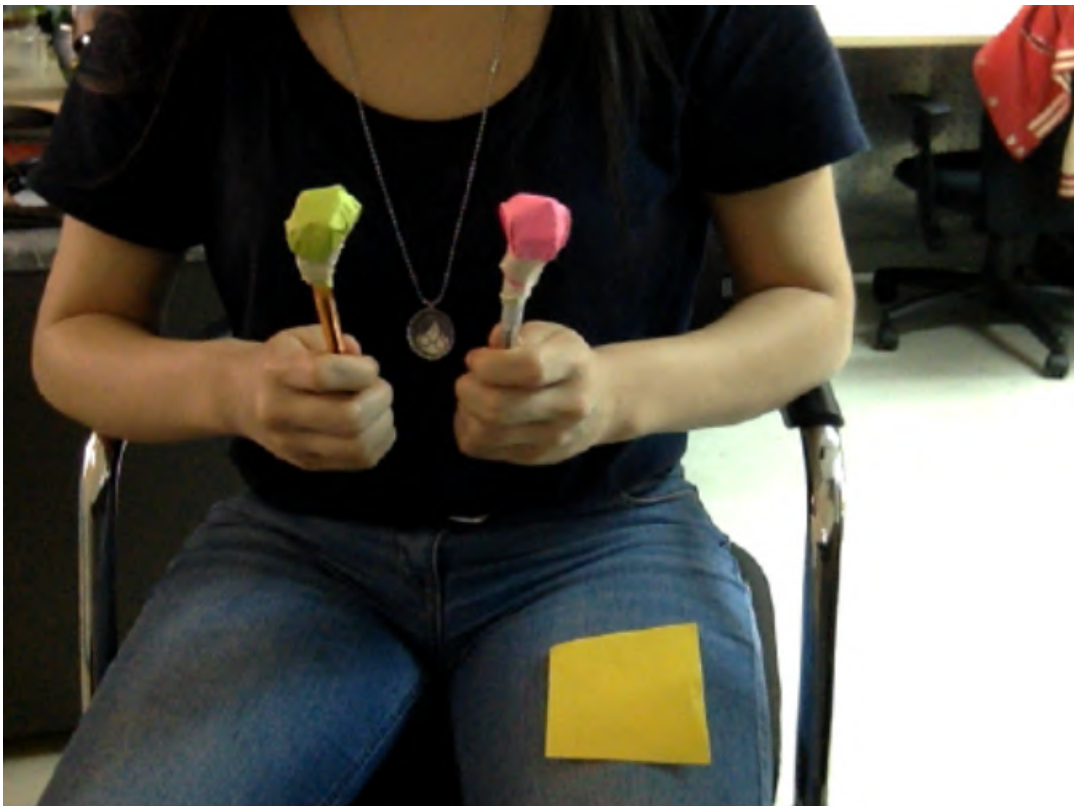


FIGURE 2.1: Setup of the drum set

Like the drums, there have been a few attempts at building models for virtualizing a guitar. The work done by Santiago et. al.[3] is the closest to the system that we aim to build. The user is required to wear a pair of red gloves and is able to pluck at different parts of a line that is drawn on the screen where each part of the line, colored differently, aims to represent the fretboard of a guitar. Every time a point denoting the right hand passes the string, the sound is to be triggered. The notes are generated using the Karplus-Strong Algorithm to simulate the according sound.

A lot gaps were identified in the work done by Santiago et. al, which gives room for improvement. The system was hard wired to be able to play a particular songs that required only 4 notes. Emphasis was laid on the need for gloves or anything red in color to wrap the hands of the user so that the system can recognise the color and performing the required computation. Gesture recognition models were not used and would end up resulting in a poor performance if the color is not identified correctly. The system used MATLAB to play the sounds using the algorithm and this would result in latency issues ranging from 0.5 to 1.5 seconds from the initiation of the strum to when the sound was actually played back. The system is also quite GPU and CPU intensive which further explains why there are issues in latency.

The current state-of-the-art with respect to guitar simulation is the work done by John Edel Tamani et.al. [5] which explains a work in progress that introduces an Electromyogram on the forearm of the player. This introduction of a physical device directly linking to the muscles of the human hand produced more accurate results and felt relatively more natural to play than any other on screen simulator. The EMG model of guitar, being a prototype it had some limitations, being able to detect the different types of strumming but not the chord being help up by the



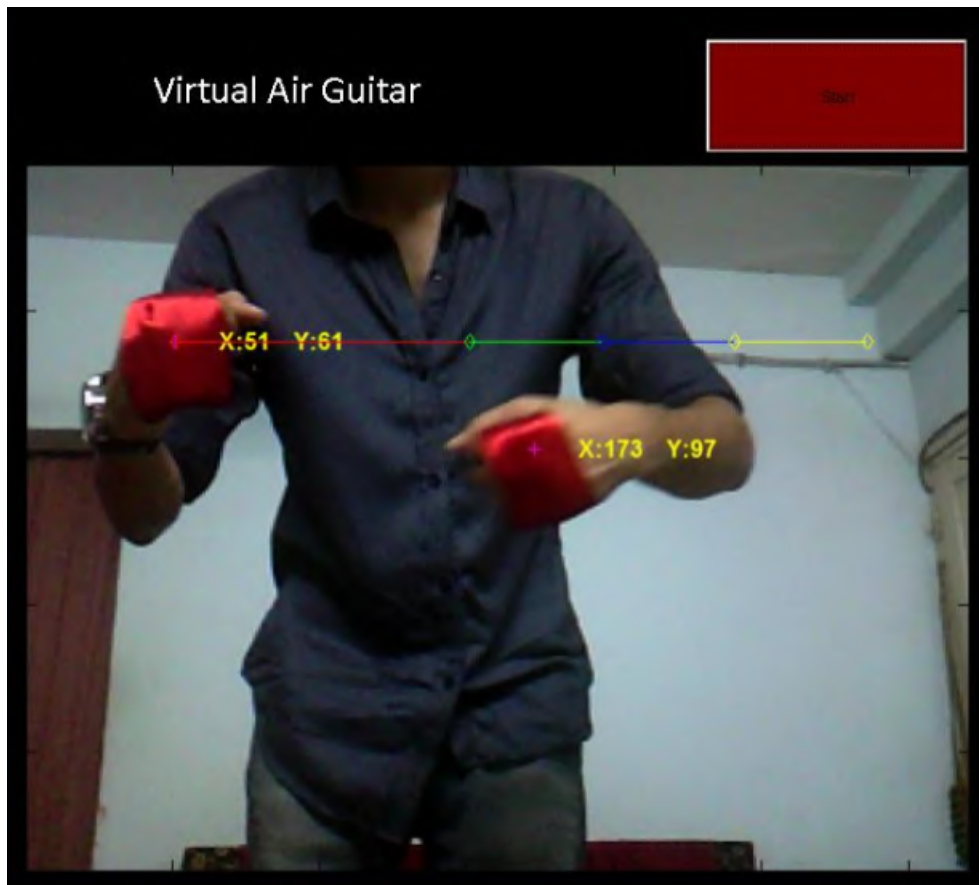


FIGURE 2.2: Guitar system using red gloves

other hand. Even the strumming was predicted with a poor accuracy as the amount of training data collected was too less. Taking more than 30 minutes to collect data from one participant implies the collection of a large dataset is not plausible. The requirement of an expensive equipment like an Electromyogram to try this system out removes the concept of it being available to the general public to try out.

From the figure 2.3 it is observed that this is still in the works and a full fledged air guitar has not been modelled yet. In the current phase of the project, it involved getting participants to try the EMG device and record the upwards and downwards strumming actions to collect the data for training. A k-Nearest Neighbour model was trained on the recorded data to identify with an accuracy what type of strum was performed. The main goal was to be able to take down

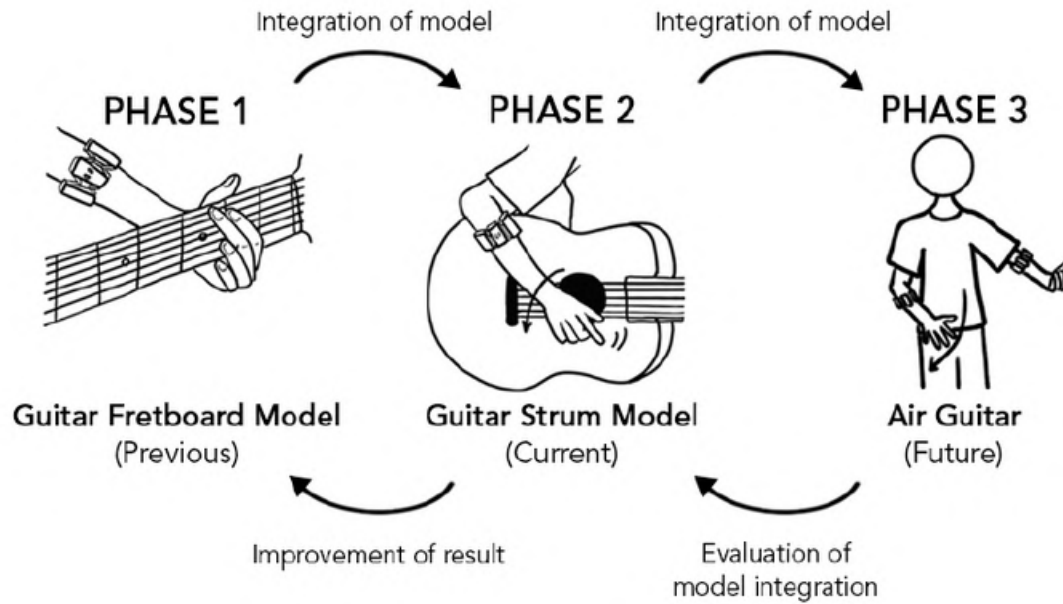


FIGURE 2.3: Different phases of the EMG aided guitar project

musical notes when inspiration strikes while holding the guitar as it is a cumbersome task to switch between a pen, paper and holding the guitar again. The guitar chord fret classifier and classifier of strums would be integrated in the future to produce the fully working air guitar system in future.

## 2.2 RESEARCH OBJECTIVES

When we look at the systems proposed to play the drums [1][4], while they do a good job in playing the right sounds when the hitbox is triggered, it does not model the real life playing of the instrument. This allows for improvement in the area that would eliminate looking up at the screen to hit right inside a specific boundary. A drummer would not be able to pick up the system and play without spending some time to learn the nuances with respect to this system. Thus, when preparing our application we aim to be able to simulate the natural playing of a guitar in the air.

Our system in one fell swoop tackles every single issue faced in [3] by introducing hand tracking methods and not focusing on a particular color. And the creation of the sound bank again eliminates the need to run an algorithm to detect which sound is to be generated and retains the natural sounds a guitar would make. We allow the playing of all the 7 major chords and there is no restriction to which notes can or cannot be played. Only individual string plucking cannot be simulated yet. The latency issues due to requirement of high computing power in order to run MATLAB's background runtime functions are removed by hosting the machine learning prediction model on the cloud and running only a simple python script in the local system. This ensures that any laptop system with a working webcam can run this application.

Challenging the current state-of-the-art [5], the Virtual Guitar system is a deployable, fully functioning application that handles all aspects of the guitar playing from chord detection, identification and playing the respective sound. Eliminating the need of a physical apparatus, it can be marketed to the public and made accessible to anyone that wishes to try it out. Improving on the dataset to better classify chords can be done.

## CHAPTER 3

# PROPOSED METHODOLOGY

To build a Virtual Guitar that is able to accurately simulate the natural playing of the instrument without any lag or latency. Without the use of any external apparatus and just the camera feed along with concepts from the domain of Machine Learning, Cloud Computing and Web development a seamless guitar playing experience is simulated. The system broadly consists of two modules that handle gestures from the left and right hands respectively.

1. **Left Hand :** The position and gesture is captured and determines what chord sound is to be played.
2. **Right Hand :** The movement speed dictates the intensity of the sound that is played on strumming of an actual guitar.

The data is processed remotely using a server hosted on the cloud and the output is presented as the respective guitar sound on the interface.

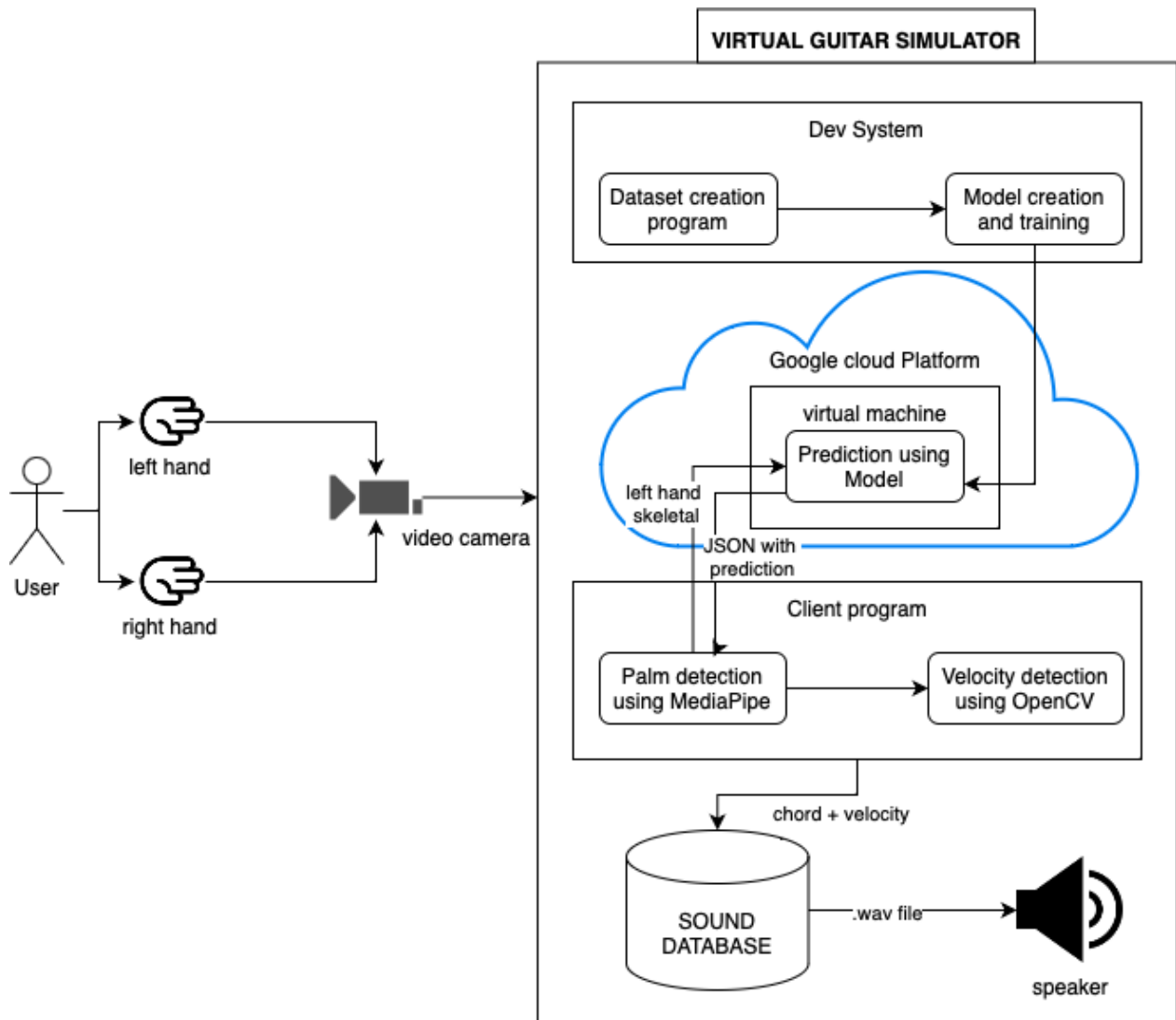


FIGURE 3.1: Detailed system architecture

## 3.1 MEDIAPIPE

MediaPipe is an open-source ML framework offered by Google that aims to deliver fast and reliable solutions using computer vision as the base standard in the lowest level. This framework aims to access sensory data through a static or live audio and video feed. These programs are written in the Python language and can be implemented and customized to suit the particular problem at hand.

In case of hand gesture and position recognition, the MediaPipe Hands solution is the most optimal. This involves using two separate models, a Palm Detection one and a Hand Landmark one. In order for the model to work accurately, it is trained extensively with multiple datasets. These datasets are distinct with respect to the manner and environment from which they were derived to better the model. They are as follows :

1. **In-the-wild dataset** : 6K images of various geographic diversity, lighting conditions and hand appearance that do not account for any complex gestures.
2. **In-house collected gesture dataset** : 10K images of various angles of all physically possible hand gestures but sampled from a set of only 30 people.
3. **Synthetic dataset** : 100k images rendered from a high-quality synthetic hand model where the finger color and thickness can be altered. The images were sampled over different backgrounds, poses and lighting environments.

The inclusion of multiple datasets increases the robustness of the model.

The ML Pipeline of MediaPipe Hands consists of two models, a palm detector and a hand landmark model.

### 3.1.1 Palm Detection Model

A singleshoot detector mode called BlazePalm Detector is used to identify the presence of hands on the screen. Unlike faces, detection of hands is very complex as it has to deal with varied hand sizes and problems with occlusion and self-occlusion. Trying to pinpoint the landmarks directly on the fingers of the hand is difficult due to the large differences that are present. So this model generates the bounding boxes for the entire palm or fist of the hand as an initial step. This ensures less number of anchors and good scene-context awareness even for small objects.

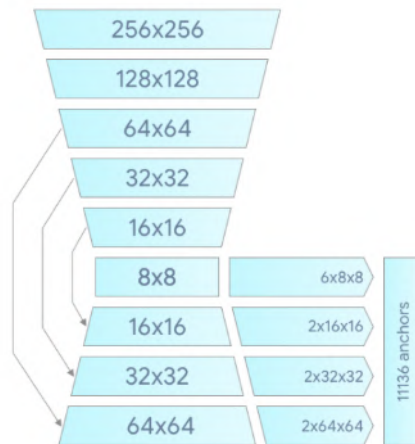


FIGURE 3.2: Palm detector model architecture

This model is run only when the presence of the hand is first detected on the screen. It is not run at every frame of the video unless the said hand was removed from the

frame or a new hand was added into the scene. This saves a lot of computational power by avoiding the running of an unnecessary detection function.

There exists a probability score to check if a hand is present in the frame or not. When this score falls below the acceptable threshold, then it is called a tracking failure and the Palm detection model is triggered to reset the tracking.

### 3.1.2 Hand Landmark Model

In the second stage, we run the hand landmark detection model which will identify and plot 21 distinct landmark coordinate points on the hand in 2.5D. This model is trained on the various hand poses and provides an accurate landmark localization even when the hands are partially visible or occluded via means of regression.

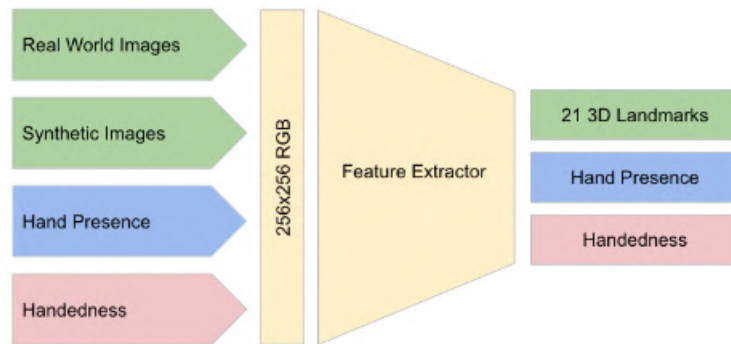


FIGURE 3.3: Hand landmark model architecture

The model has three outputs as shown in figure 3.3:

1. 21 hand landmarks with the x, y and z coordinates showcasing relative depth.
2. A flag that indicates the presence of a hand in the frame.
3. Classification of handedness, whether right or left.



The 21 points are set accurately as learnt from the multiple datasets. This can be seen in figure 3.4 The introduction of relative depth exponentially increases the possibilities of detectable and understandable gestures by the model while evading the issues caused by occlusion. The threshold score or the hand flag that triggers the tracking reset is performed in this model. When two hands are determined in the frame, they also have a feature that defines left or right handedness.

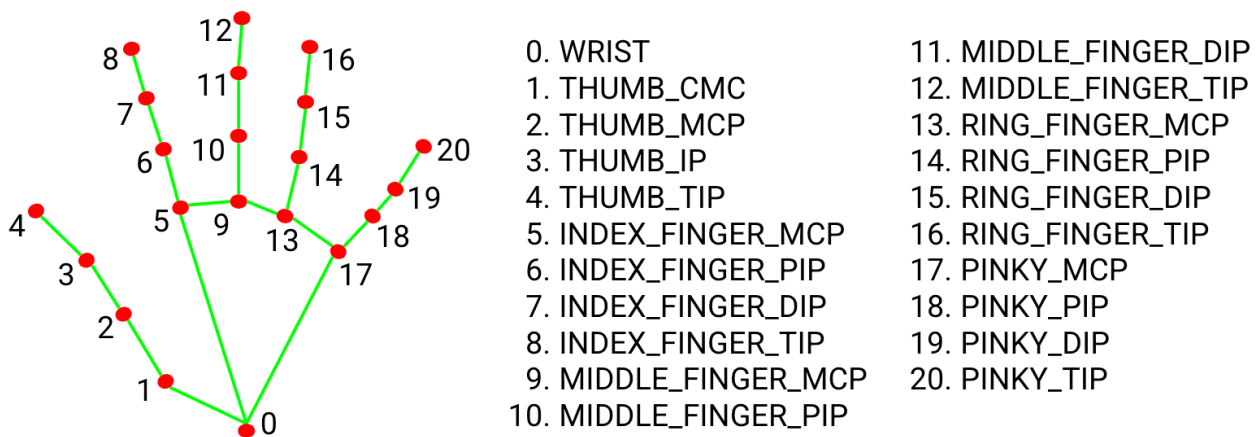


FIGURE 3.4: 21 Landmark coordinates

This model has different versions that can be used based on the hardware availability at the end-user side. GPU intensive versions as well as a more lighter version that would rely on the existing CPU power in a more real time sense while not compromising on the accuracy also exists.

### 3.1.3 Implementation in MediaPipe

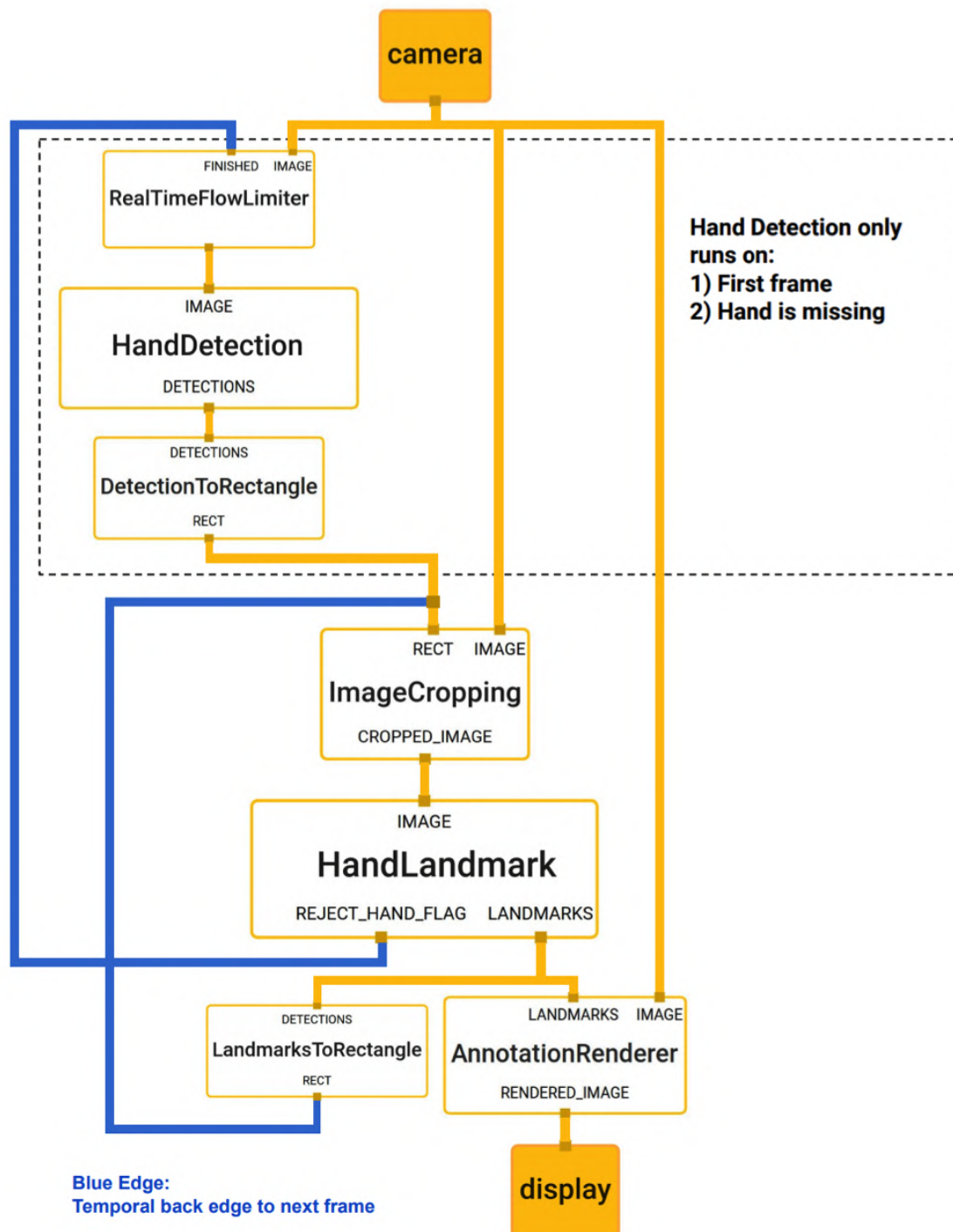


FIGURE 3.5: Mediapipe pipeline

In the most high level perspective, the entire system can be understood as a directed graph with multiple subgraphs. These graphs are made up of the modular components called Calculators. Calculators perform tasks that involve manipulation, cropping, rendering and computations on data.

The Mediapipe graphs consists of two subgraphs in palm detection and hand landmark plotting as observed in figure 3.5 The palm detection is run only as and when needed, saving GPU power. The confidence metric in the hand landmark module will determine whether to run the previous model or not.

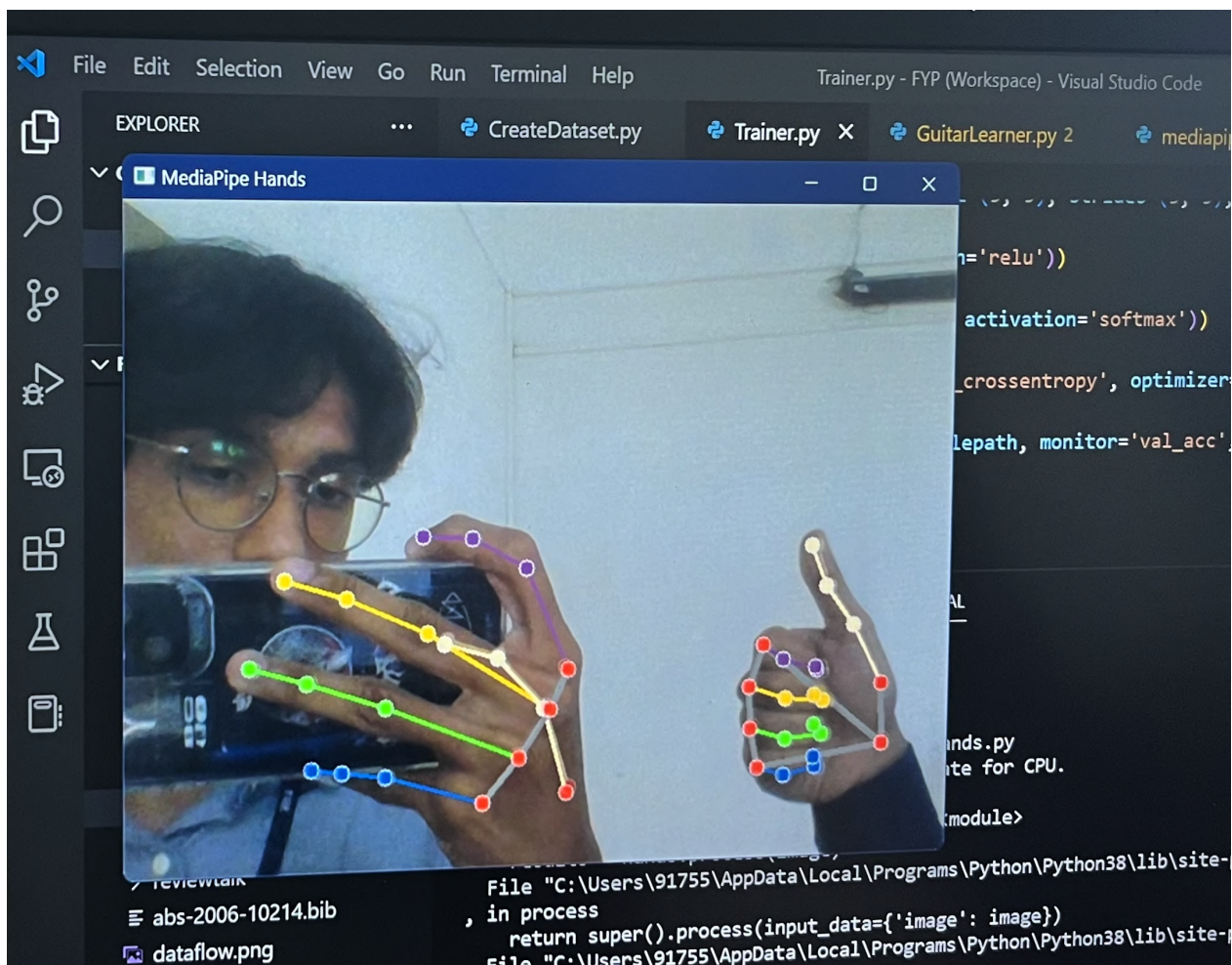


FIGURE 3.6: Mediapipe Hands

The solution finally produces an output as seen in figure 3.6. In this figure one can observe the individual landmark points along with which finger they belong to by the respective colours displayed. The self occlusion and that by an external entity both have been thwarted.

## **3.2 LEFT HAND MODULE**

The left hand module involves creating the dataset of hand gestures that correspond to each of the different chord that can be played on a guitar. The scope of the program limits it to only the major chords. Then a CNN model is trained on this dataset to be able to accurately predict which chord gesture is being held up by the left hand.

### **3.2.1 Dataset Creation**

The dataset must be created and stored in a format that is easy to comprehend for a computer system. This means that a black-and-white colour scheme would be more beneficial, as an RGB image has extra dimensions in the bitmap representation.

Further, contours must be identified and the skeletal frame constructed by MediaPipe must be a bright shade in order to contrast the darker backdrop.

The dataset is recorded by running a program that creates folders on the local system for each of the 7 major chords. In order to do this a bounding box is first

defined around the hand in question and the contour lines are drawn to identify each of the landmark coordinates and their connections. This rectangle with the hand fully inside is then captured from the video feed and processing is done on it. The colors are stripped from the image to make it gray with the lines on the hands in white making it seem like a skeletal representation of the gesture. This can be seen in the figure 3.8

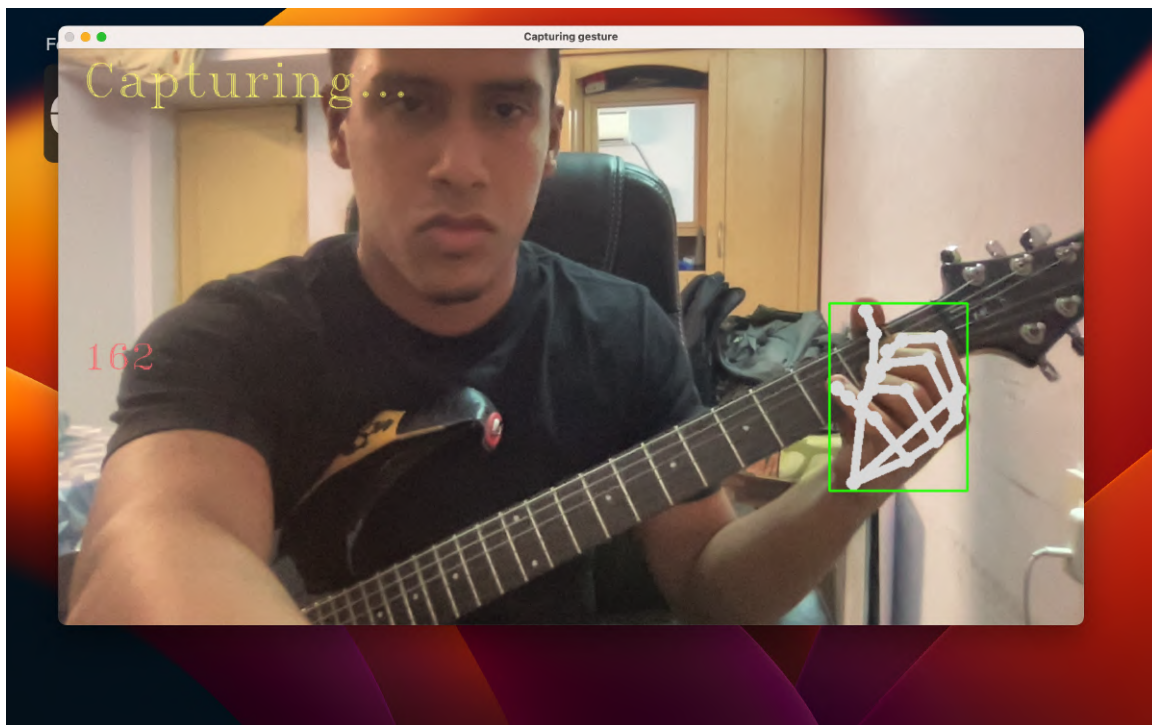


FIGURE 3.7: Recording and processing of 1200 images for each chord



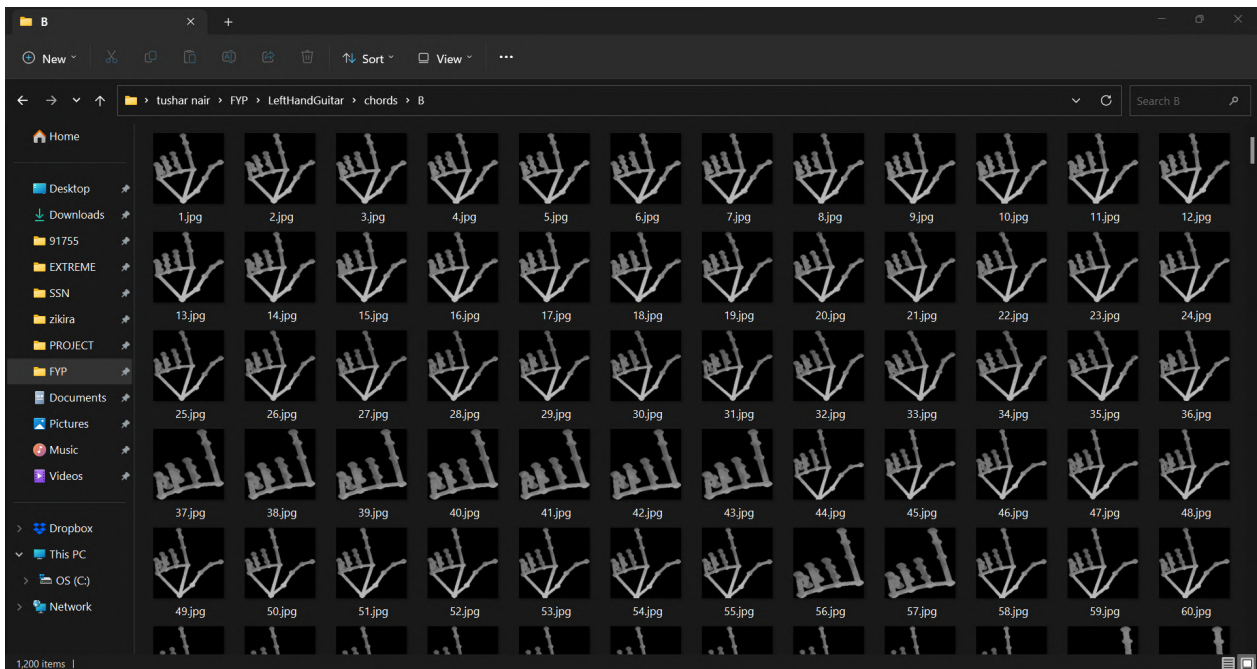


FIGURE 3.8: Images for a single chord

### 3.2.2 Chord Detection Model Training

A CNN model is trained on the 'chords' directory housing more than 8000 images of all the different gestures. Setting a total of 7 classes, the training directory, the batch sizes and the number of epochs to be trained based on the requirement of the system, the model is run on this data.

The Convolutional Neural Network is similar to a traditional Multi-layer BP network. The CNN however, follows a system of locally connected kernels, rather than fully connected networks as in BP networks. This was designed by drawing inspiration from anatomically accurate neurons, where CNN kernels correspond to the receptors in our brain.

A group of connections can share the same weights, which reduces parameters further. CNN follows Downsampling dimension reduction i.e. Pooling. Local

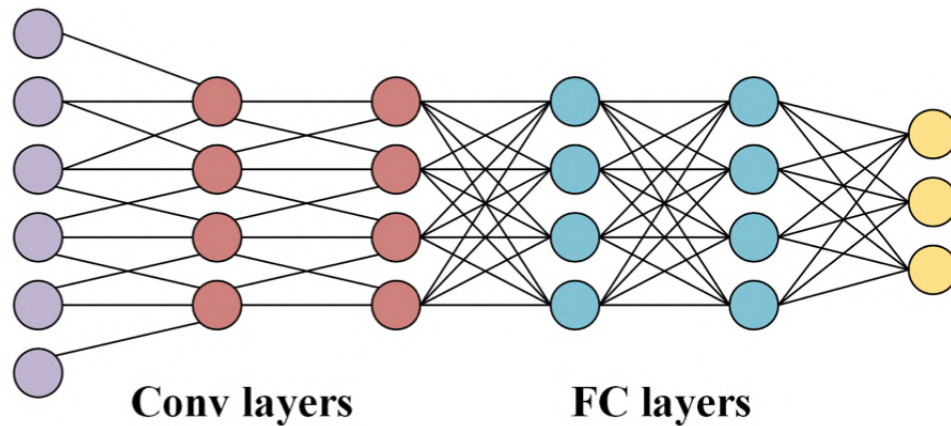


FIGURE 3.9: Convolutional Neural Network

bitmaps are combined by an aggregator function such as 'max', 'avg', etc and the resultant bitmap is dimensionally reduced.

When it comes to using CNNs for a Computer Vision application, the work done by D. Bhatt et. al. [2] explains how the numerous possible architectures for a CNN can be examined and the perfect fit for the object recognition task can be identified.

The CNN is sequential and employs pooling to produce tensors from each 2D convolution layer which is then fed again to the outputs while maintaining the ReLu curve. Finally the multi-dimensional arrays or tensors are flattened to one dimension and then fed to the dense layer where every neuron is connected to the other and classifies which class the image belongs to. The trained model is then stored as a H5 file in the same directory. The model is robust enough to accommodate minor variations in hand gestures and predict the chord value accurately.

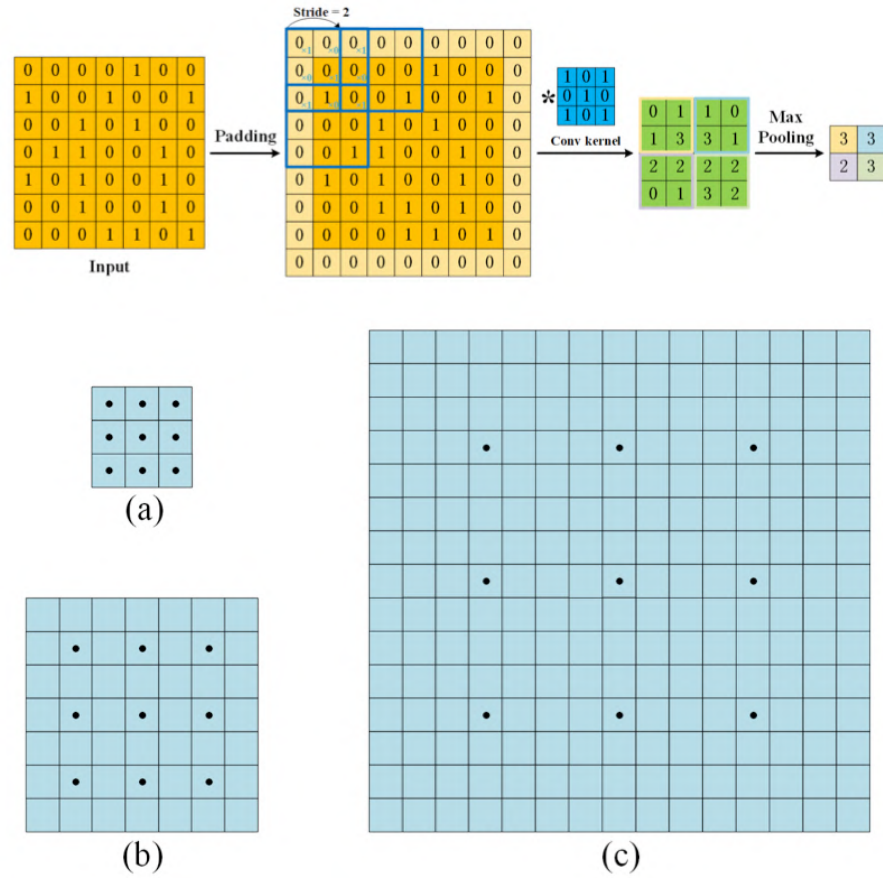


FIGURE 3.10: Procedure of a 2-D CNN

### 3.2.3 Chord Identification

The live feed annotates the left hand with the connectors and hand landmark points and then sends each frame as an image after performing the same processing that was done on the images during dataset creation to the server hosted on the cloud where the trained model performs a prediction which a probability as to which chord class the hand gesture belongs to.



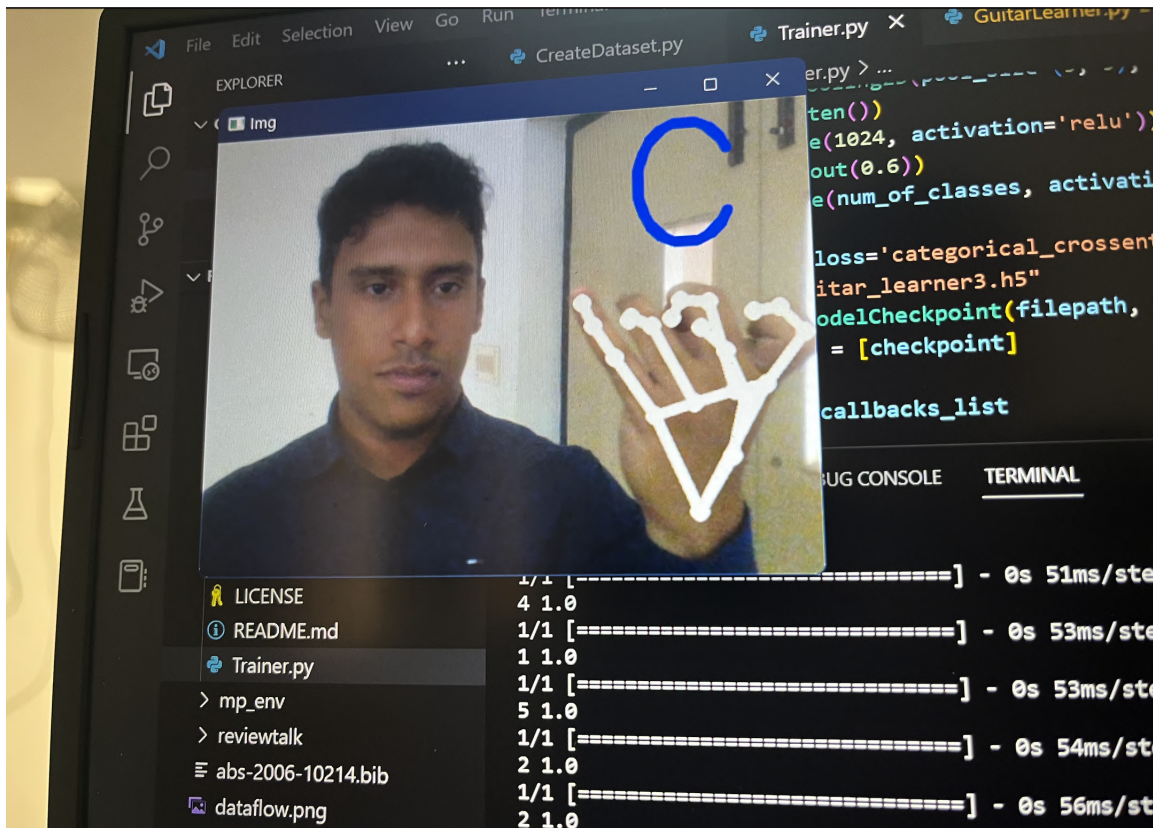


FIGURE 3.11: Chord detection using model

### 3.3 GUITAR SOUND BANK

In the final leg of the system, where we have the classified chord along with the speed of the strum, the corresponding sound needs to be mapped to and played. To maintain the authenticity of the sound, the sounds were not downloaded from an external source but instead recorded on a Guitar with a cherub clip acoustic mic. Reaper is a DAW (Digital Audio Workstation) application that offers recording, editing, mixing and processing functionalities. After setting up the environment for recording with a DAW (Digital Audio Workstation) - Reaper and an Audio interface - iRig that connects the guitar to the Macintosh computer, the sounds were processed and stored in a .wav format in the project directory under 'chord\_sounds'.



FIGURE 3.12: DAW - Reaper

This sound bank has been expanded to accommodate two different levels of strumming based on the velocity. For example, A\_fast.wav and A\_slow.wav. To further improve the sound bank, would be to include the many minor chord sounds that exist as well as individual chord plucking sounds instead of strumming.



FIGURE 3.13: Audio Interface - iRig



FIGURE 3.14: Guitar - Yamaha F130

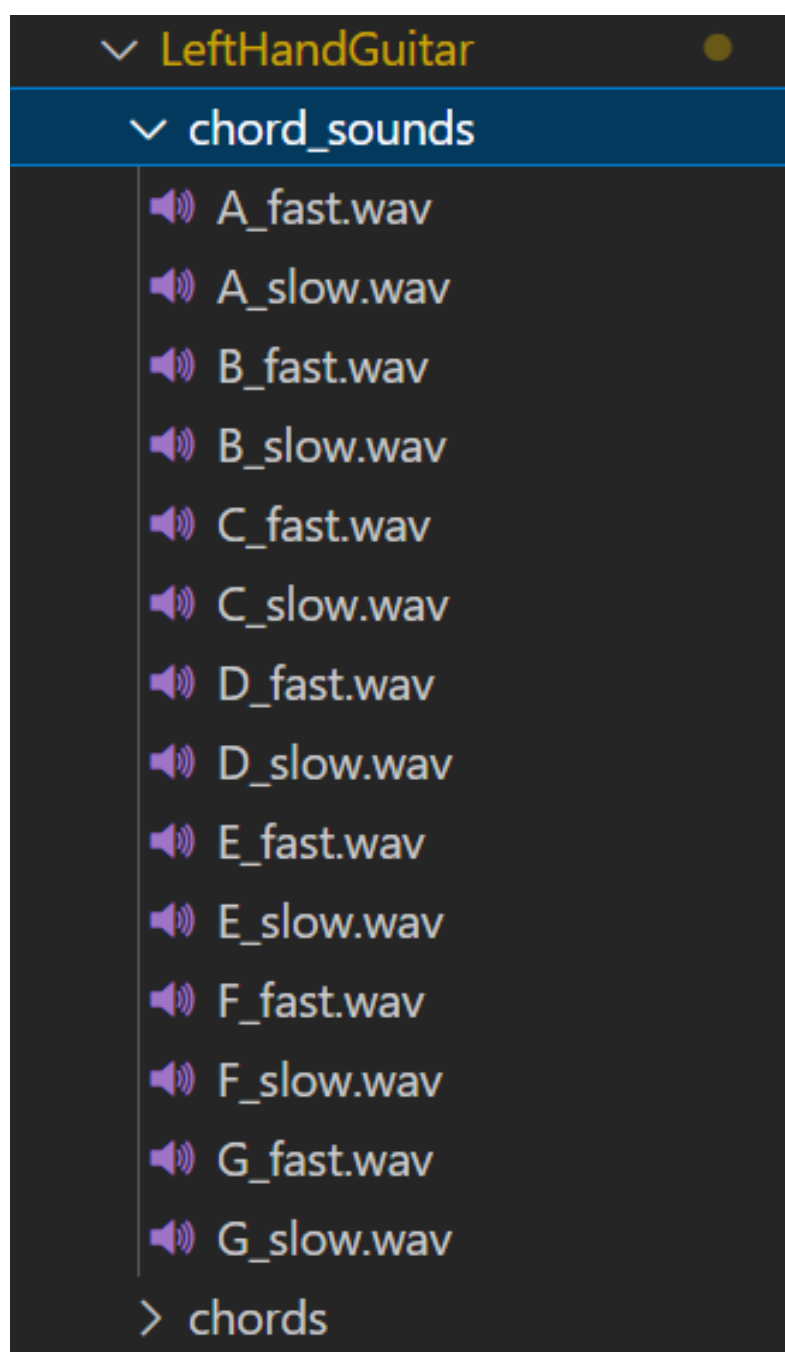


FIGURE 3.15: Directory containing chord .wav files



## 3.4 RIGHT HAND MODULE

### 3.4.1 Velocity Detection using OpenCV

Using the 12th hand-landmark corresponding to the tip of the middle finger, we identify when the finger crosses a pair of horizontal "hit-boxes" close to the halfway point from the bottom of the screen. We get the timestamp of the event when the point crosses the top horizontal line and similarly record the timestamp of the event when it crosses the bottom horizontal line. The time the hand takes to pass in between these two hit-boxes is inversely proportional to the speed of the strum. Using this time value, we can decide to play the strum sound of a higher distortion that gives a buzzing effect or the softer strumming sound as when a guitar is strummed gently.

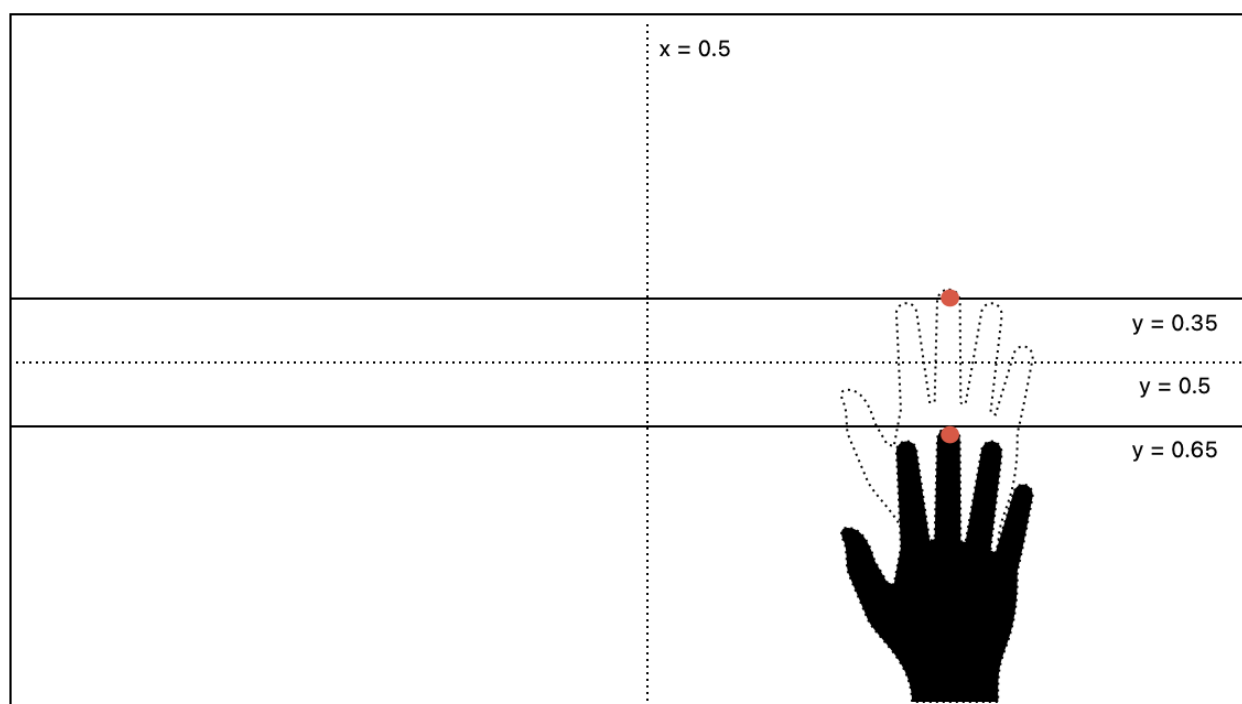


FIGURE 3.16: Placement of hitboxes for sensing the strum

### 3.4.2 Hosting Model on the Cloud

In order to abstract the prediction using the model from the users system, a VM on Google Cloud Platform (GCP) was set up. The specifications of this VM are as follows:

#### Machine configuration :

Machine type: e2-medium

CPU platform: Unknown CPU Platform

Architecture: x86/64

vCPUs to core ratio: -

Custom visible cores: -

Display device: Disabled

GPUs: None

Bucket details

REFRESH

HELP ASSISTANT

LEARN

chordmodelbucket

Location

Storage class

Public access

Protection

us (multiple regions in United States)

Standard

Not public

None

OBJECTS

CONFIGURATION

PERMISSIONS

PROTECTION

LIFECYCLE

OBSERVABILITY

NEW

INVENTORY REPORTS

NEW

Buckets

>

chordmodelbucket

UPLOAD FILES

UPLOAD FOLDER

CREATE FOLDER

TRANSFER DATA

MANAGE HOLDS

DOWNLOAD

DELETE

Filter by name prefix only

Filter

Filter objects and folders

Show deleted data




<input type="checkbox"/>	Name	Size	Type	Created	Storage class	Last modified	Public access	
<input type="checkbox"/>	 <a href="#">guitar_learner.h5</a>	271.5 MB	application/octet-stream	Apr 22, 2023, 4:25:14 PM	Standard	Apr 22, 2023, 4:25:14 PM	Not public	 

FIGURE 3.17: Bucket details

This virtual machine supports 2 concurrent threads simultaneously (this can be upgraded with an upgrade in infrastructure) and has a memory of 4 GB.

We SSH (Secure Shell) into the VM with a pre-configured default key, and access the virtual machine's terminal via a window in our browser. All the VM operations are carried out through this command line interface.

```

CHORD IS: 0
223.178.82.117 - - [23/Apr/2023 10:42:14] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 96ms/step
CHORD IS: 0
223.178.82.117 - - [23/Apr/2023 10:42:16] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 106ms/step
CHORD IS: 0
223.178.82.117 - - [23/Apr/2023 10:42:17] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 108ms/step
CHORD IS: 0
223.178.82.117 - - [23/Apr/2023 10:42:18] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 95ms/step
CHORD IS: 0
223.178.82.117 - - [23/Apr/2023 10:42:19] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 105ms/step
CHORD IS: 0
223.178.82.117 - - [23/Apr/2023 10:42:22] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 107ms/step
CHORD IS: 0
223.178.82.117 - - [23/Apr/2023 10:42:24] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 107ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:36] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 103ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:39] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 106ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:41] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 105ms/step
CHORD IS: 1
223.178.82.117 - - [23/Apr/2023 10:42:44] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 109ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:48] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 96ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:49] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 108ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:50] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 107ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:51] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 106ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:54] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 107ms/step
CHORD IS: 2
223.178.82.117 - - [23/Apr/2023 10:42:56] "POST /api/test HTTP/1.1" 200 -
1/1 [=====] - 0s 108ms/step

```

FIGURE 3.18: SSH access to VM from browser window

The keras model was mounted onto a bucket storage on GCP as shown in the figure 3.17. This was a custom model uploaded as an 'H5' file and works as a regular keras model would. This bucket is referenced by our Virtual machine to run prediction tasks on received skeletal images.

The firewall rule was added to accommodate HTTP POST requests from our client program running on the user's system. The client sends the skeletal image of the left hand's gesture when the strum is triggered, after processing it to match the dataset's expected format.

This image once received by our cloud server, is passed through the model to receive a prediction. This prediction is sent as the response to the HTTP POST and is in JSON format, which is unpacked in the client program. This removes the need for the client's system to be configured with tensorflow or have any of the dependencies installed in the local system and be compatible to run the machine learning model. It also allows us to monitor what chords the clients are playing more often for analytical purposes.



## CHAPTER 4

# EXPERIMENTAL RESULTS

In this section, we present the findings with regards to the specific modules of the whole system and then that of the full working application

### 4.1 DATASET DESCRIPTION

In this system, the dataset was not sourced externally but instead created by us in order to have control on the data that is fed to the machine learning model for training. It also serves to be a right blend of rigid and flexible by incorporating minor changes in the gesture positions of each chord.

This dataset is a collection of more than 8000 images that were captured by the developer. Setting a cap of 1200 images for each of the 7 major chords, this array of images were stored in separate directories. We have incorporated heterogeneity in the images for each chord by recording them in two different ways.

1. While holding an actual guitar in the hand and articulating the fingers in the position that indicates the gesture for a particular chord.
2. Holding up the chord gestures in the most accurate way they are played in the air without the aid of a guitar.

These images were recorded using python and computer vision functions in order to render them as a set of connector points and the lines joining them. The color has

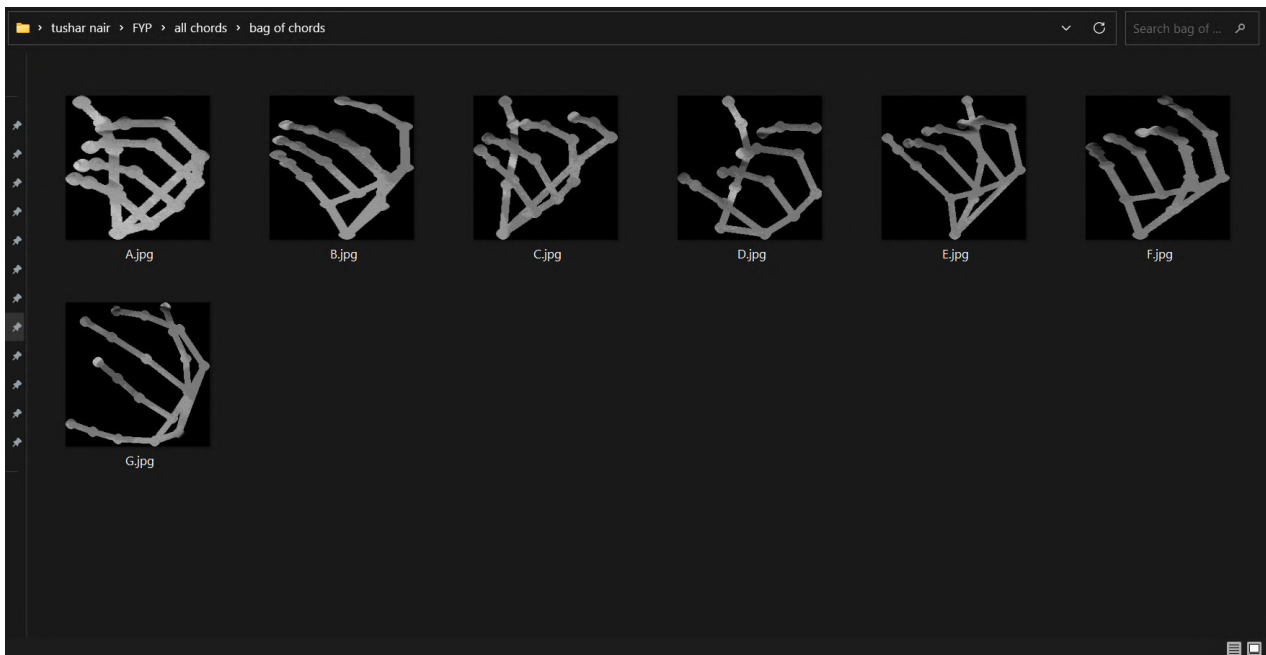


FIGURE 4.1: Sample image for each chord

been stripped down to black and white, maintaining only the useful information to look somewhat like that of a skeletal hand. This can be seen in the figure 4.1 where we see one image from each of the 7 chord classes.

Inconsistencies in recording the dataset were observed when using different lighting and introducing noise in the background. This brings about the idea of further increasing entropy by using different hand types along with a dynamic environment, thereby making the system more robust. A great computational power would be required in order to make the same come to fruition.

## 4.2 ECOSYSTEM

The main goal of the Virtual Guitar system was to eliminate the need of any physical apparatus in it's working and so there is virtually no need of any special

hardware device to use this application. At the very least, it requires any computer or laptop device with a working camera and monitor.

On the flip side, the laptop must have python installed and be able to download and run a simple python script that would have all the function calls defined to run the entire system. The chord detection model is hosted on the cloud using the Google Cloud platform and the same is accessed via API calls and POST requests from the client to the server. Everything gets handled in the same python file.

The last step deals with the containerizing of each module and zipping it all together as a package which when made available on the internet can be downloaded and run by anyone on just about any system without the need for any special dependencies or hardware specifications to be present. All the CPU or GPU intensive work is either already handled or is taken care of by the virtual machine running on the cloud.

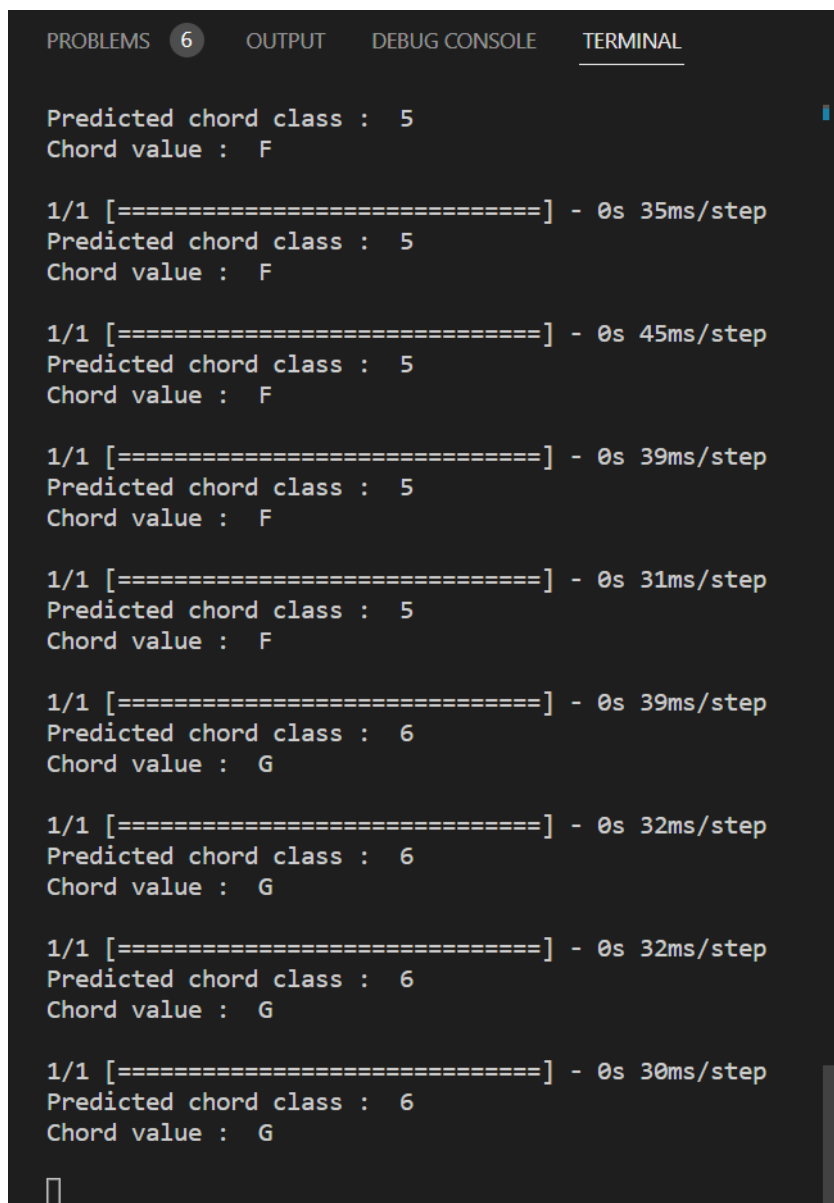
#### **Details of hardware and software used:**

- Camera: **Macbook air M1 webcam**
  - 720p FaceTime HD camera
  - Advanced image signal processor with computational video
- programming language used with version: **python 3.10.8**
- web framework used with version: **Flask 2.2.3**
- library used to detect hand landmarks with version: **Mediapipe 1.16.0**
- Machine learning library used with version: **Tensorflow 3.2.2**

### **4.3 EXPERIMENTS CONDUCTED**

The experiments conducted can be understood by observing them step by step in a modular manner. The whole system has been broken down into units and each one has been tested diligently and finally the whole system too, in order to ensure the seamless working of the application.

The first stage involves the left hand chord prediction by the trained machine learning model. On the left half of the screen the position and gesture of the hand is tracked. On running the trained model, the gesture is recognized and the chord class for the same is classified by the system and further mapped to the respective chord value. The output of this module can be seen in the figure 4.2



```

PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL

Predicted chord class : 5
Chord value : F

1/1 [=====] - 0s 35ms/step
Predicted chord class : 5
Chord value : F

1/1 [=====] - 0s 45ms/step
Predicted chord class : 5
Chord value : F

1/1 [=====] - 0s 39ms/step
Predicted chord class : 5
Chord value : F

1/1 [=====] - 0s 31ms/step
Predicted chord class : 5
Chord value : F

1/1 [=====] - 0s 39ms/step
Predicted chord class : 6
Chord value : G

1/1 [=====] - 0s 32ms/step
Predicted chord class : 6
Chord value : G

1/1 [=====] - 0s 32ms/step
Predicted chord class : 6
Chord value : G

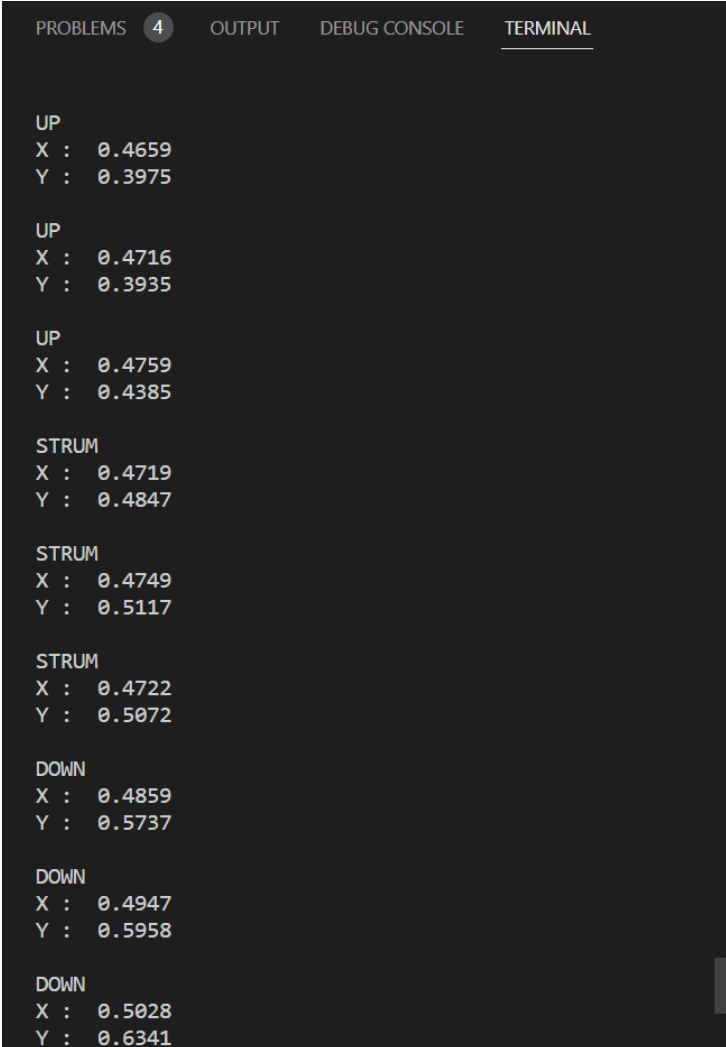
1/1 [=====] - 0s 30ms/step
Predicted chord class : 6
Chord value : G

```

FIGURE 4.2: Left hand chord detection

We can see that the prediction is run accurately at each step, even changing the displayed chord class and value when the left hand gesture changes.

The next stage involves localizing the landmark point 12 as defined by the MediaPipe Hands solution on the right side of the screen. We display the x and y coordinate values of the point in the terminal to track the position of the required landmark point. This is shown in the below figure 4.3.



```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

UP
X : 0.4659
Y : 0.3975

UP
X : 0.4716
Y : 0.3935

UP
X : 0.4759
Y : 0.4385

STRUM
X : 0.4719
Y : 0.4847

STRUM
X : 0.4749
Y : 0.5117

STRUM
X : 0.4722
Y : 0.5072

DOWN
X : 0.4859
Y : 0.5737

DOWN
X : 0.4947
Y : 0.5958

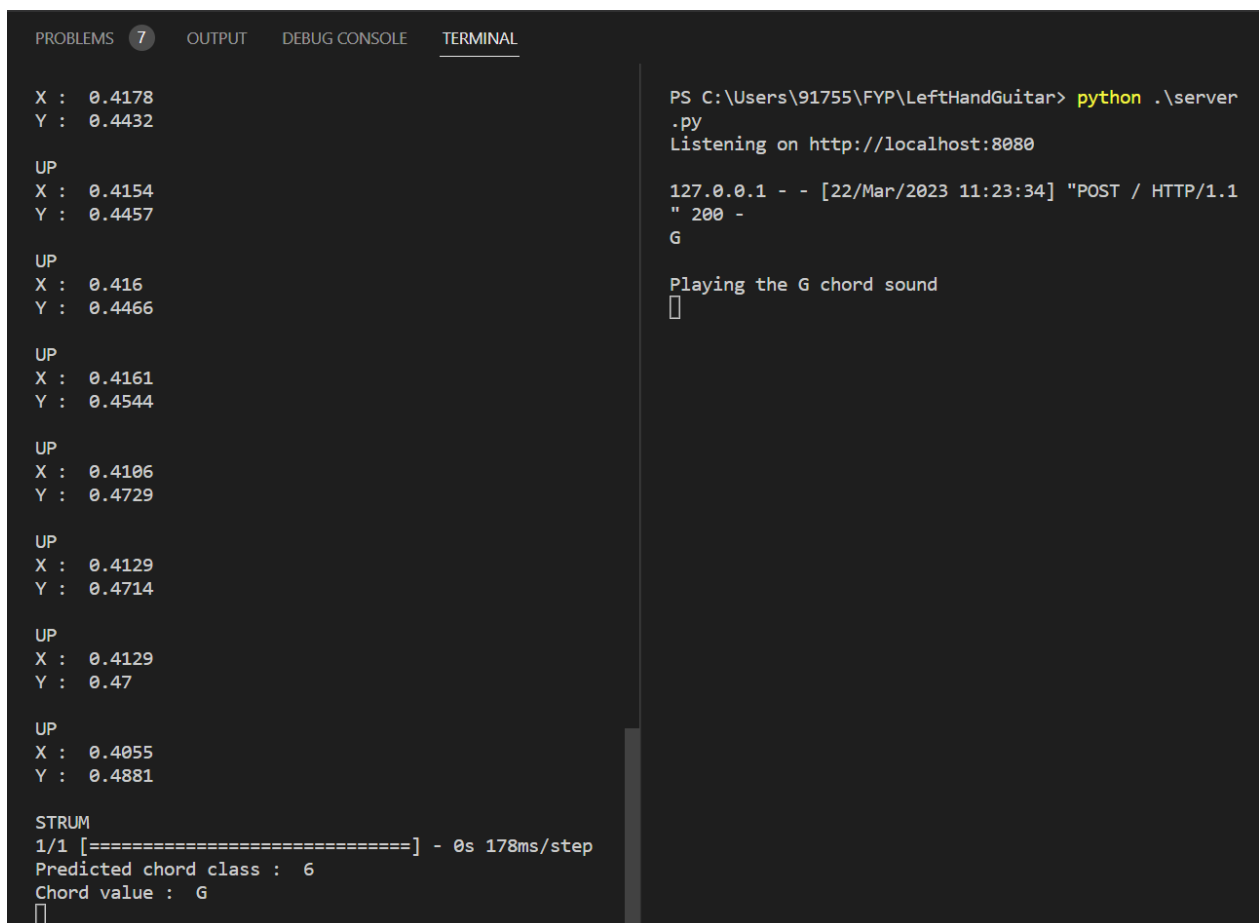
DOWN
X : 0.5028
Y : 0.6341
```

FIGURE 4.3: Landmark Coordinate tracking

On the right half of the screen, as the strumming point moves from the top half of the screen to the bottom half, the program identifies the regions where the landmark is above or below the strum region and also the instant(s) of motion indicating the strum.

Testing the fully working application on the local system involves the left and right side modules running asynchronously.

When the left hand gesture is held up and the right hand strumming point reaches the bottom of the horizontal strum line, the chord value at that instant is predicted by the model and this value is recorded and sent to a local server. The working of both the left and right side asynchronously is observed. When the strumming point is reached, the chord value at that instant is predicted by the model and the time spent in the strum region is flagged as fast or slow. These values are passed to the server. The server then accesses the created sound bank to play the respective sound.



```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL

X : 0.4178
Y : 0.4432

UP
X : 0.4154
Y : 0.4457

UP
X : 0.416
Y : 0.4466

UP
X : 0.4161
Y : 0.4544

UP
X : 0.4106
Y : 0.4729

UP
X : 0.4129
Y : 0.4714

UP
X : 0.4129
Y : 0.47

UP
X : 0.4055
Y : 0.4881

STRUM
1/1 [=====] - 0s 178ms/step
Predicted chord class : 6
Chord value : G

PS C:\Users\91755\FYP\LeftHandGuitar> python .\server.py
Listening on http://localhost:8080

127.0.0.1 - - [22/Mar/2023 11:23:34] "POST / HTTP/1.1" 200 -
G

Playing the G chord sound

```

FIGURE 4.4: Playing classified chord sound

We observe two windows being generated apart from the live camera feed that records video inputs. These windows give a representation of the chord playing hand separately and also the tracked hand landmark points to ensure they are tracked precisely and displaying accurate data. The terminal also shows the server output with the classified chord and the respective sound is generated.

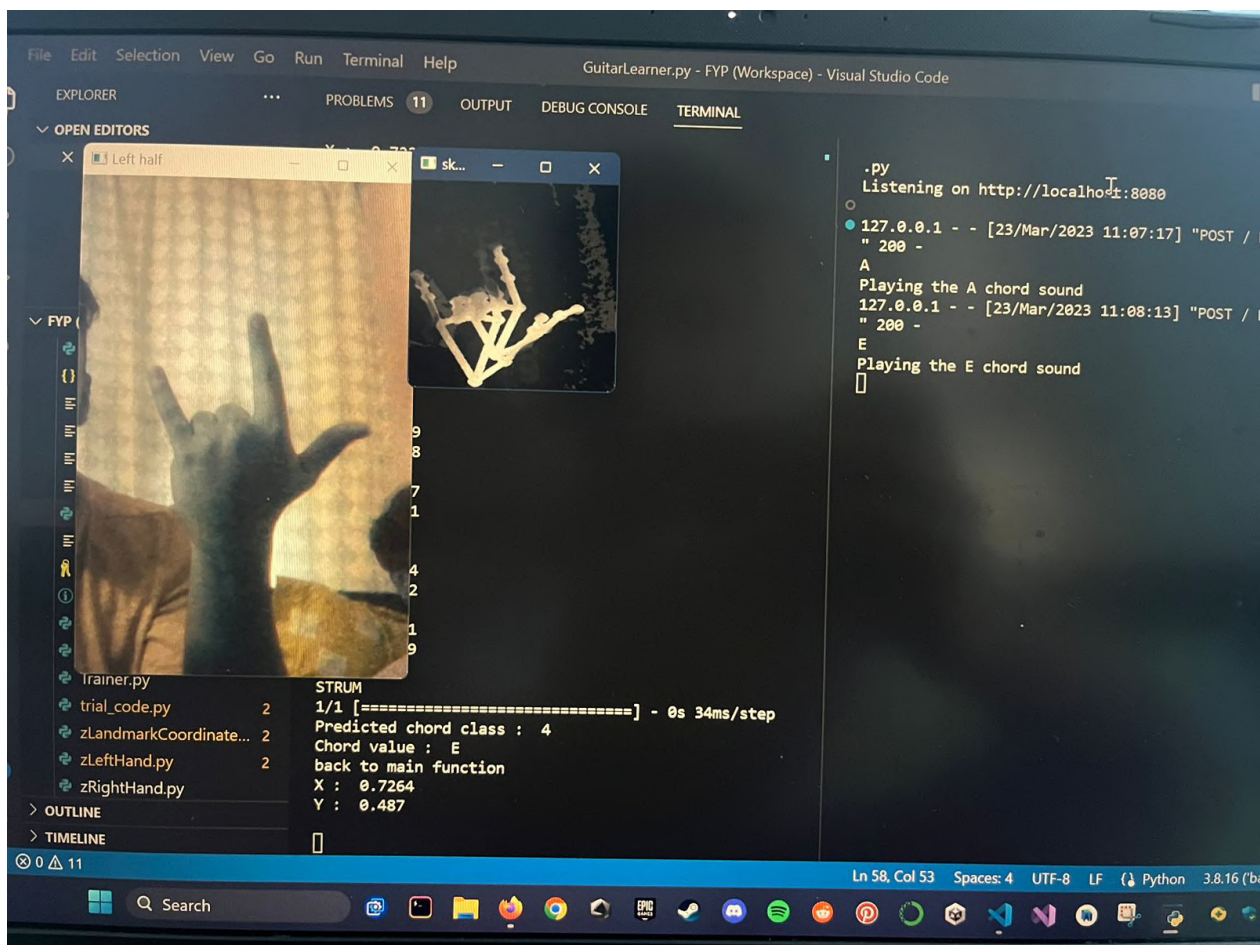


FIGURE 4.5: Virtual Guitar System running locally

The problem with running everything locally would necessitate the end-user to download and have all the dependencies to run the machine learning model. There were also issues with latency as the application took time to respond between queries to the ML model. This is impractical and the final, fully working application overcomes these challenges.





## CHAPTER 5

# PERFORMANCE ANALYSIS

Model	Accuracy	Val Accuracy
CNN Model	0.99	0.98

TABLE 5.1: Model performance

The machine learning model was trained over 40 epochs on the created dataset to produce satisfactory results with accuracy metrics greater than 95%. It finished with a CNN Error percentage of 1.07%.

## 5.1 CONFUSION MATRIX

		Predicted Values						
		A	B	C	D	E	F	G
Actual Values	A	7	-	-	3	-	-	-
	B	3	2	-	-	-	5	-
	C	3	-	5	-	-	2	-
	D	4	3	-	3	-	-	-
	E	3	-	2	-	1	4	-
	F	1	-	-	-	-	9	-
	G	-	-	1	-	-	4	5

FIGURE 5.1: Confusion Matrix

The above performance analysis was performed by continually checking the chord that plays when a particular chord is shown to the camera. Each chord is tested 10 times and the total number of successful prediction is tallied out of 10.

The ground truth in the above experiment is the knowledge of the guitar player conducting the experiment of which chord is to be held while noting the observations.

Class	n (truth)	n (classified)	Accuracy	Precision	Recall	F1-score
A	21	10	75.71%	0.7	0.33	0.45
B	5	10	84.29%	0.2	0.40	0.27
C	8	10	88.57%	0.5	0.63	0.56
D	6	10	85.71%	0.3	0.5	0.38
E	1	10	87.14%	0.1	1.0	0.18
F	24	10	77.14%	0.9	0.38	0.53
G	5	10	92.86%	0.5	1.0	0.67

FIGURE 5.2: Performance metrics from confusion matrix

Although the model has a great accuracy, the results observed when running the model are not satisfactory due to the external environment being dynamic in nature. The chord gestures being held up, while being correct, are not being tracked accurately by the camera and here a discrepancy with end results.

## 5.2 USER INSIGHT ANALYSIS

Prompt	Strongly Agree	Agree	Disagree	Strongly Disagree
I would use this app for regular practice	0%	25%	62.5%	12.5%
It is convenient to use in its current state	25%	37.5%	37.5%	0%
It effectively replaces the need for a guitar	0%	0%	0%	100%
I have come across similar guitar virtualization tools in the market	50%	25%	25%	0%
I think this application has market potential	87.5%	0%	0%	12.5%
I would recommend this app to a friend	87.5%	12.5%	0%	0
<b>Average</b>	Agree % =	58.33%	Disagree % =	41.67%

TABLE 5.2: Survey results

A survey was conducted on a uniform pool of 8 guitarists ranging from skill levels of beginner to professional. They were asked a series of 6 yes/no questions and their insights were recorded and tabulated. The overall reception of this project was satisfactory, and scopes of improvement were shared with us.

## CHAPTER 6

# **SOCIAL IMPACT AND SUSTAINABILITY**

The Virtual Guitar system is not simply a shot in the dark but instead a well devised, highly directed shot towards an extremely rapidly growing market of people on the internet with or even without any great interest in the instrument. Guitarists of all levels of playing expertise can and will have a hand at trying out this revolutionary software system that could potentially make someone's whole career while also holding true to the converse.

At the beginner level, it eliminates the need for someone new to the instrument to pay up such a large amount when they aren't sure if they'll seriously pick up the instrument or not. It gives them a chance to get familiar with the nuances of playing the instrument. For guitarists that have experience in using the instrument for a while, they can use the system to play any musical inspiration that comes to them without ever having the need to carry around the expensive, delicate and large physical guitar.

The music community is embracing any kind of new technology in the space by viewing it as something that is created by the product of advancement in technology and science, demarcating the new generation of music while also respecting the ways that have been well established in composing tunes for a long time. While nothing looks like it would entirely replace the use of the original physical instrument, it in turn eggs on the researchers to keep pushing and finding better solutions to the same problems.

With the right software updates, features to record and mix the audio recorded on the system can be added which would appeal to not just instrumentalists but also music producers. This adds a whole new dimension to the application. It also serves as a platform to learn the various chord gestures and the accompanying sounds for that chord. This opens up new avenues in the market yet again by turning out to be an educational tool if enough guides are created for an amateur to watch, learn and practice.

There is no dearth to the potential end users of the system as well as possible feature updates that could be introduced to the same, thereby, making it an application that could firmly hold its ground in the market as well as combat the battle of time.

## CHAPTER 7

# CONCLUSIONS AND FUTURE WORK

In this study, the proof of concept was established, that there is an appreciable scope in the process of virtualization of the guitar instrument. A responsive and real time system was developed upon an architecture encompassing the Mediapipe framework, OpenCV library and Google Cloud platform.

The chord detection module yielded below average results, and there is work to do in this area. By improving the model for prediction of chords, either by adding heterogeneity to the training dataset or adding variations in the lighting, hand position and hand type, we would improve the accuracy of the sounds emitted by our system to the real-life guitar playing experience. The latency issues that persisted were eliminated after hosting the system on the cloud and the performance in this aspect was far better than what was anticipated.

The core components of the end-user application have been developed to a large extent, but the containerization and deployment of a ready-to-use end user application either as a native desktop application or a web application is yet to be carried out. Additionally, a comprehensible interface for accessing the same can be prepared and marketed.

To handle large volumes of traffic that is generated by the POST operation to our cloud server, our virtual machine solution must be expanded to accommodate growing user count by dynamically scaling-up. This can be achieved by using a virtual load balancer or a virtual machine scale set, both of which are available on most major cloud platform.



Considering the scope of the project, with some financial aid, the virtual machine on the cloud can be upgraded and several better options for a machine learning classification model can be trained and tested.

## REFERENCES

1. Bering, Shannon & Famador, Sandra. (2016). Virtual Drum Simulator Using Computer Vision. 370-375. 10.12792/icisip2016.066.
2. D. Bhatt et al., “CNN Variants for Computer Vision: History, Architecture, Application, Challenges and Future Scope,” *Electronics*, vol. 10, no. 20, p. 2470, Oct. 2021, doi: 10.3390/electronics10202470.  
mi
3. Santiago JR, Samuel SJ, Sawn R. Modified Virtual Air Guitar: A Concept Realized using Image Processing Techniques. *EAI Endorsed Trans Context Aware Syst App* [Internet]. 2015 Mar. 12 [cited 2023 Apr. 27];2(3):e2. Available from: <https://publications.eai.eu/index.php/casa/article/view/2029>
4. Tolentino, Carl Timothy & Uy, Agatha & Naval, Prospero. (2019). Air Drums: Playing Drums Using Computer Vision. 1-6. 10.1109/ISMAC.2019.8836175.
5. Tamani, John & Christian, Jan & Cruz, Blaise & Cruzada, Joshua & Valenzuela, Jolene & Chan, Kevin Gray & Deja, Jordan Aiko. (2018). Building Guitar Strum Models for an Interactive Air Guitar Prototype.
6. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C. L., & Grundmann, M. (2020). Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214.
7. Z. Li, F. Liu, W. Yang, S. Peng and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999-7019, Dec. 2022, doi: 10.1109/TNNLS.2021.3084827.