Abstract

Don't Touch The Art!
Gesture Controlled Musical Instrument

Manya Sachdev

2023

Hand gestures are a natural and intuitive way to interact with machines. This project blends hardware and software components to create a multi-sensory, engaging experience for the viewer. The final product is an interactive musical instrument consisting of six sound rods placed in a circular formation. In the centre, is a hammer-like attachment that rotates and strikes the sound rods. The software component applies a sequential machine learning model to detect and recognize hand gestures passed to it via a video feed. Libraries used include OpenCV and CVZone. The HandTrackingModule from CVZone marks skeleton-based landmarks which helps increase precision in classification. The hardware circuit is built using an Arduino UNO and A4988 motor driver. The motor used is a NEMA 17 stepper motor. The classified gesture defines the movement pattern in the stepper motor which rotates the hammer attachment and strikes the sound rods, adding a real-time visual and auditory feedback to the viewer.

Don't Touch The Art!

Gesture Controlled Musical Instrument

A Capstone Project
Presented to the Faculty of Computer Science
of
Ashoka University
in partial fulfillment of the requirements for the Degree of
Postgraduate Diploma in Advanced Studies and Research

by
Manya Sachdev

Advisor: Debayan Gupta

December, 2023

Contents

List of Figures

List of Tables

Acknowledgements

I am deeply grateful to the entire staff at Makerspace for being so patient with my never ending list of questions. I am very thankful to my advisor, Professor Debayan Gupta, who always met my ideas with positivity and was extremely prompt with any and all communication. Lastly, I'm very thankful for my brother and father whom I went to for all construction related doubts. Their respective engineering and architecture degrees have really paid off for me.

Chapter 1

Introduction

India Art Fair opened doors to their first digital art exhibit in February, 2023. Art installations in this section played on various computer science principles such as Virtual Reality, Augmented Reality, Computer Vision and Image Projection. An installation that particularly drew me was a piece called ”Log Kya Kahenge” by the artist, Mira Felicia Malhotra. [1]. ”Log Kya Kahenge” used virtual reality to add a meaningful layer to a canvas painting through the lens of an iPad. This was the only viewer-interactive installation in the section. It was also the most crowded. The installation allowed me to see firsthand, the impact of *interactivity* in viewership experience.

Interactivity in art can play on multiple senses: Visual, Auditory, Tactile, Gustatory, Olfactory, Vestibular and Proprioceptive. As I waited for my turn to hold the iPad and view the complete painting, I found myself wanting to make an interactive art installation of my own. With this project, I wanted to push my boundaries on how many senses I could call upon in one single art piece. However, the one essence of traditional art that I wanted to retain was *Don't touch the art.*

1.0.1 Background

An early example of involving viewers in the creation phase of art is Myron Krueger's Videoplace [2]. Krueger set out to design an interactive viewership experience, releasing its first iteration in 1975 without the use of a computer. Over the next decade, he went on to build his own computing system to achieve his vision of artificial reality. Using real time image processing techniques, he combined live video feeds of a viewer's shadow with graphic elements on a projection screen. Today, Videoplace houses over 25 different interaction patterns for users to choose from and create art with.

This sort of a scenario, where humans are directly influencing the art produced, is extremely prevalent today. A current day example of such an artist is TeamLab, an international art collective. The "artist" behind the exhibits is made up of a diverse team of programmers, architects, designers, engineers, mathematicians and CG animators. One of their most notable pieces is called Rain Room, an enclosed area with a constant stream of water being dispersed from the ceiling. This stream of water however, never touches any person actually in the room. Various sensors and cameras are placed intelligently around the enclosure. They detect human presence and movement. They calibrate the start and stop of the water stream around the individual [3], giving viewers an immersive and experimentative experience. The viewer has the ability to make their experience completely unique each time.

Gesture recognition is a pivotal part of the larger Human-Computer Interaction (HCI) realm. It gives users an opportunity to communicate with systems in an intuitive and humanely expressive way. Hand gestures are particularly universal. Every culture, language and community uses hand gestures in communication. These could be in conjunction with spoken words; for instance, waving while greeting or giving a thumbs up when wishing good luck. It could also be entirely separate, forming an all encompassing communication tool on its own; for instance, American

Sign Language or musical orchestra conducting. Whatever the use case, hand gestures have immense potential in communicating information to another entity.

1.0.2 Project Overview

The project combines multiple computer science principles in one reactive product. Titled "Don't touch the art", the installation allows viewers to *play* a musical instrument and record a composition on it.

The physical structure comprises of 6 sounds rods, suspended horizontally in the air with the help of a cotton thread/elastic string. These rods are spaced equidistant from a hammer-like attachment placed in the centre of the entire setup. The hammer is attached to the head of a stepper motor which is mounted on the supporting wall of the structure.

A machine learning model has been trained to identify 6 different types of hand gestures. Each gesture corresponds to a specified degree rotation in the stepper motor. The gestures are read via the laptop's webcam and classified in real-time, resulting in immediate movement in the motor. At the end of the interaction period, the entire composition is played back.

1.0.3 Objectives

Capture a sequence of user inputs via a camera and reciprocate with a multi-sensory response in real-time.

Specifics

1. Seamless integration of hardware and software.
2. Write a machine learning model for accurate hand gesture readings.

3. Make the viewer an active participant; leave them with the desire to have multiple rounds of interaction.

Chapter 2

Literature Review

Hand gesture recognition can be carried out using multiple different techniques. An in-depth review and exploration was given in [4]. The paper differentiates between static hand gestures and dynamic hand movements in hand gesture recognition. Static gestures focus on the stable shape of the the hand eg. a thumbs up. On the other hand, dynamic gestures focus on the movement pattern of the hand eg. waving. For either type, the paper outlines the approaches for gesture recognition - 1. Glove based approach 2. Computer Vision Approach. Under the CV approach, it mentions multiple options for recognition techniques. A color based recognition can be done using a glove marker. This takes into account pixel values and classifies them into skin or not skin. Another technique is the skeleton based recognition. This approach adds geometric markers on top of the gesture image to improve detection of complex features.

Gesture control over motors using the OpenCV library was explored in [5]. Hand movements were captured using an external, high-definition camera and commands for motor movement were sent to a Raspberry Pi board which was interfaced with a robotic arm (controlled via servo motors). Computer vison was chosen as the HCI method of interaction due its low cost and simplicity in hardware requirements.

In an article that explores gesture controlled drones [6], image segregation from live video feeds was discussed. Over a series of experiments, it was discovered that variabilities in scene setting greatly influence the accuracy of correct classification. Bright spaces with low background noise make for good backdrops for the camera frame. A distance of 3ft was particularly identified as improving the accuracy score.

Hand gestures can be replacements for other hardware interaction tools such as remotes, joy sticks and mice. This paper [7] discusses the different funcitonal roles that getsures can play.

It identifies three such roles:

1. Semiotic - gestures that communicate information.

2. Ergotic - gestures that manipulate objects in the real world.

3. Epistemic - gestures that learn from their environment.

Chapter 3

Methodology

”Don’t touch the art!” uses a laptop’s default webcam to detect hand gestures from a live video feed. It captures the gesture, passes it through an image classification model and records the recognized class. It sends the recognized class as an integer input to the Arudio microcontroller via a serial connection formed in the Arduino IDE. Once the microcontroller receives the signal, it initializes movement in the stepper motor according to a pre-defined mapping between specific integer values and number of steps. The viewer can input any number of gestures in this manner and get a real-time feedback on the sound produced. Once the desired number of gestures have been passed, the entire sequence of movements is played back, allowing the user to listen to their complete composition.

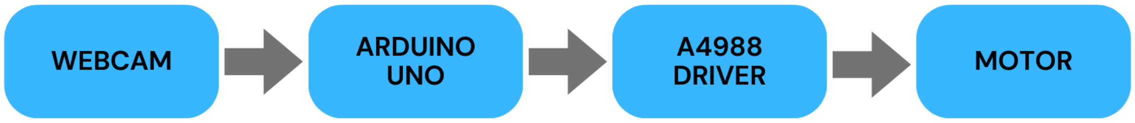

Figure 3.1: Block Diagram

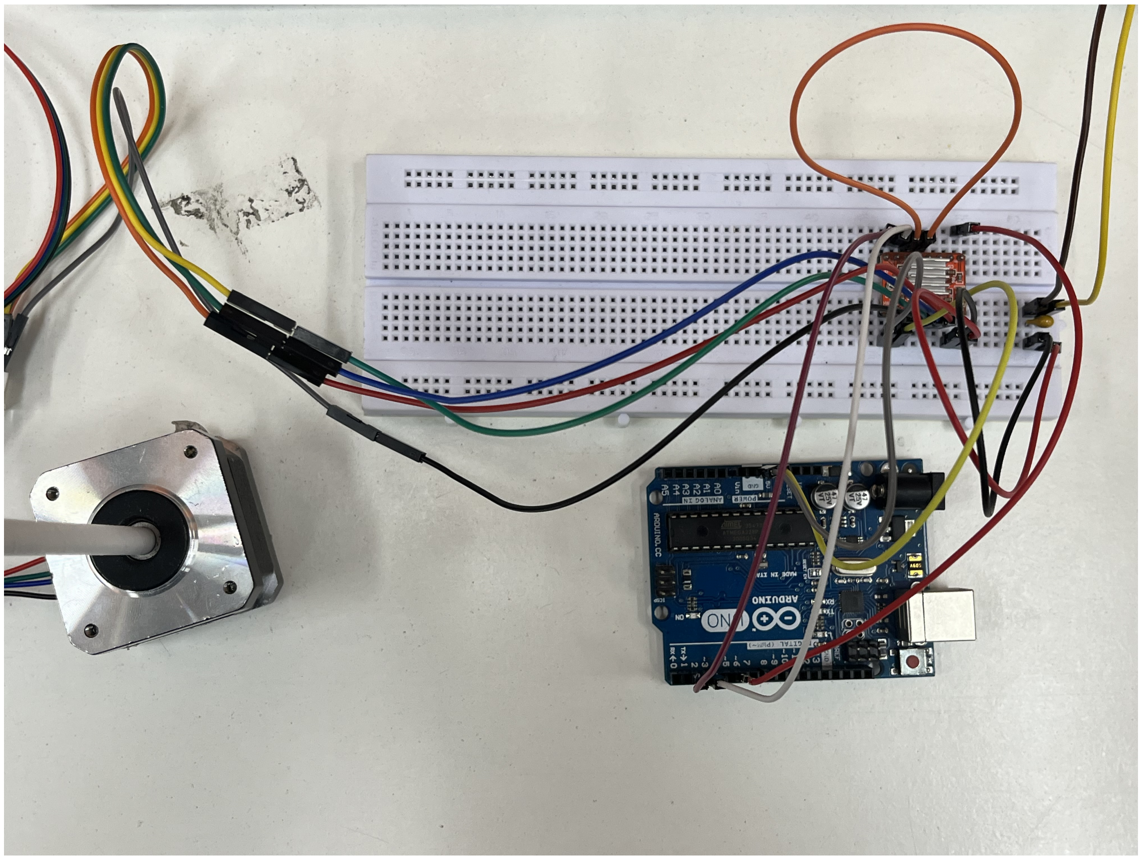

Figure 3.2: Circuit

3.0.1 Hardware Setup

Arduino Uno Microcontroller

Widely available electronics platform which contains a microcontroller and several pins for input and output of signals. It provides an open source platform on which the user can run software and control hardware. Using the arduino IDE, one can write and upload code to the board for later execution.

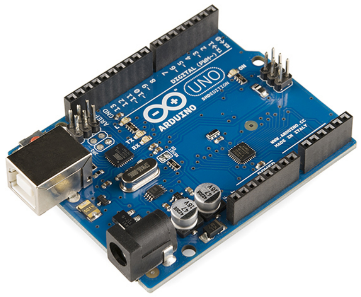

Figure 3.3: Arduino Uno

A4988 Stepper Motor Driver

This is the interface device between the arduino and the stepper motor. It receives electrical signals from the arduino microcontroller and converts them into precise movements in the stepper motor. The driver helps regulate the current flowing through the coils of the motor in order to prevent overheating and ensure reliable and repeatable performance. The A4988 motor driver also supports micro stepping, making it a great candidate for use in robotics applications where precise movement it required.

NEMA 17 Stepper Motor

A widely used stepper motor. The 17 corresponds to the size of the motor faceplate (1.7 inches). The motor uses sets of coils, energized in sequence to rotate in discrete

Figure 3.4: A4988 Driver

steps. Compared to traditional DC motors, stepper motors provide much more precise control over angular positioning. The stepper motor is usually paired with a motor driver such as the A4988 used in this project to achieve accurate and consistent movements in electronics projects.

Figure 3.5: Stepper Motor

Decoupling Capacitor

A decoupling capacitor provides temporary energy storage for electronics systems. This reduces voltage fluctuations in the circuit as the integrated circuits switch operations. As the current demands change between operations, the capacitor acts as a buffer and smooths out the current delivery to ensure reliable and accurate operation of the electronic components used in a circuit.

A4988	Connection
DIR	Pin 2 (Arduino)
STP	Pin 3 (Arduino)
SLP	RST
RST	SLP
EN	Pin 6 (Arduino)
GND	GND (Arduino)
VDD	Pin 5 (Arduino)
1A, 1B, 2A, 2B	Stepper Motor
GND	Decoupling Capacitor
VMOT	Decoupling Capacitor

Table 3.1: A4988 Pinout

Power Supply

The Arduino board is powered using a laptop. Power transmission is facilitated with the a help of USB cable that is plugged into the laptop's COM5 port.

The stepper motor has a higher power requirement. It is powered using an external power source: a KA3005D Digital Control DC Power Supply. This supply has the capacity to go upto 30V and 5A. For the purpose of this project, the voltage has been set to 24V and the amperage has been capped at 2A.

Figure 3.6: External Power Supply

3.0.2 Software Setup

Visual Studio Code was used as the main code editor. A script for data collection was run and a total of 2625 images - approximately 350 images per class - were collected using the computer's built-in webcam. The organised data was uploaded to Google Drive which was mounted to a Colab notebook containing the script for the machine learning model. The trained model was uploaded back to VS Code where a test script was run, checking for real-time gesture recognition. The recognised gesture was sent over a serial connection to the Arduino board, which sent signals to move the stepper motor accordingly.

Machine Learning Model

A Convolutional Neural Network was designed using the Tensorflow and Keras libraries in Python. Once the model was initialized as a sequential model, layers were added linearly.

The first layer, a two dimensional convolutional layer, is the initial layer that performs low-level feature extraction. It specifies the dimensions of the input expected-256x256 pixels and 3 color channels. The second convolutional layer is initialized with the same values in terms of number of filters, kernel size and activation function. This layer is meant to capture additonal features. The third and final convolutional layer is initialized with twice the number of filters as before. It is responsible for high-level feature abstraction.

Each convolutional layer is followed immediately by a MaxPooling layer. This reduces spational dimensions while retaining important features.

The Flatten layer flattens the output passed to it from the convolutional layers into a one dimensional array. This is followed by two Dense layers. The first, with 256 neurons, is a fully connected layer that learns the complex patterns in the flattened 1D features. The second dense layer, or output layer, is initialized with number of neurons

= 7 to represent the 7 image classes. The activation is defined as Softmax, a function that is used in multi-image classification. Softmax takes a vector of unnormalized scores and converts them into probabilities which represent the likelihood of each class being the correct one. The class with the highest probability is declared as the predicted class.

Model summary:

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896
 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)
 conv2d_1 (Conv2D)           (None, 125, 125, 32)      9248
 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 32)        0
 g2D)
 conv2d_2 (Conv2D)           (None, 60, 60, 64)        18496
 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 64)        0
 g2D)
 flatten (Flatten)           (None, 57600)             0
 dense (Dense)               (None, 256)               14745856
 dense_1 (Dense)             (None, 7)                 1799
=================================================================
Total params: 14776295 (56.37 MB)
Trainable params: 14776295 (56.37 MB)
Non-trainable params: 0 (0.00 Byte)
```

The full code to the model can be found here.

Serial Connection

A serial connection was established between the computer and the Arduino using a USB cable. Both devices agree on a baud rate of 9600 to transfer data. This allowed for real-time interaction between the two systems which is a key component of this project.

Stepper Motor Control

Motor movement was defined in the Arduino IDE using the AccelStepper library. Step and Direction pins were initialized and a maximum speed parameter was set.

Seven movement patterns were defined. Each movement pattern was mapped to a particular integer input. The integer input was expected to come from the laptop via the serial connection formed. The following loop, continuously checks for any incoming data. If data is received, it calls the controlSteps() function to facilitate the desired motor movement.

```
void loop()
{
  if(Serial.available()>0){
    receivedData = Serial.readStringUntil('\n');
    controlSteps(receivedData);
  }
}
```

3.0.3 Installation Design

Multiple iterations of the design were produced and rough prototypes were made. They were measured in terms of: Resource feasibility, quality of sound produced and visual aesthetics.

Int	Gesture	Degree
0	Index pointing	360
1	Peace Sign	180
2	Three fingers	120
3	Four fingers	90
4	Fist	45
5	Thumbs up	Directional Switch for following movements
6	Index finger + thumb	360, slow speed

Table 3.2: Hand Gestures and Corresponding Motor Movement.

The installation comprises of:

1. 6 hollow steel sound rods of varying lengths

2. Acrylic base - 180x400mm

3. Two hexagonal support structures - height - 76.2mm

4. Hammer attachment

5. Spring

6. Wooden hammer head

A thick cotton string runs through each sound rod. The string is fixed to the hexagonal structures, suspending the rod in the air.

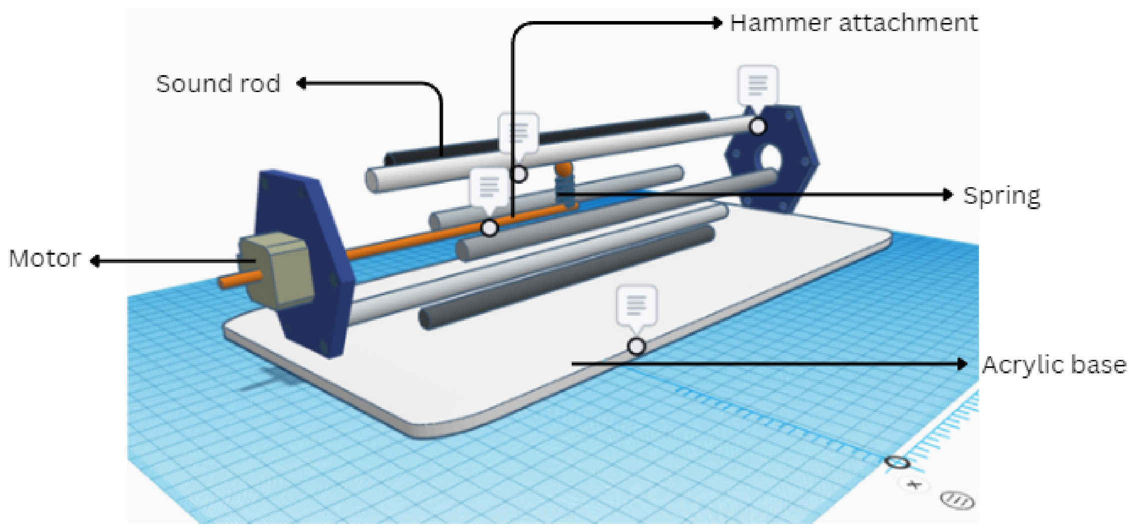

Figure 3.7: Final Design

Chapter 4

Data Collection

Images for the dataset were collected using a python script. The script employed the OpenCV and cvzone libraries.

```
cap = cv2.VideoCapture(0)

detector = HandDetector(maxHands=1)
```

The above code snippet initializes a video capture object. The id for the camera in this setup was 0, indicating the laptop's default webcam. The HandDetector module was imported from cvzone.HandTrackingModule and the number of hands to be detected was set as 1.

A while-loop was set up to initialize a continued stream of frame captures. Hands were automatically annotated with a bounding box and hand landmarks with the help of the HandDetection module. Image saving was triggered by the press of the *s* key on the keyboard. Each save was accompanied by a counter value displayed in the terminal to help indicate the number of images captured for a class upto that point.

4.0.1 Image Processing for Data Collection

Once an image was captured, it was cropped to focus on just the hand area, as defined by the bounding box. Next, the image was resized to 300x300 pixels to ensure uniformity in values. Lastly, the cropped and resized image was placed in the centre of a white background. This was done to ensure that all images in the dataset have the same height and width.

```
x, y, w, h = hand['bbox']
imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
imgCrop = img[y-offset:y+h+offset, x-offset:x+w+offset]
imgCropShape = imgCrop.shape
aspectRatio = h/w

if aspectRatio>1:
    k = imgSize/h
    wCal = math.ceil(k*w)
    if not imgCrop.size:
        print("Image is empty.")
    else:
        imgResize = cv2.resize(imgCrop, (wCal, imgSize))
    imgResizeShape = imgResize.shape
    wGap = math.ceil((imgSize-wCal)/2)
    imgWhite[:, wGap:wCal+wGap] = imgResize
else:
    k = imgSize/w
    hCal = math.ceil(k*h)
    imgResize = cv2.resize(imgCrop,(imgSize, hCal))
    imgResizeShape = imgResize.shape
    hGap = math.ceil((imgSize-hCal)/2)
    imgWhite[hGap:hCal+hGap, :] = imgResize
```

4.0.2 Image Preprocessing for Training

```
data = tf.keras.utils.image_dataset_from_directory(
    ....
    image_size=(256, 256),
    validation_split=0.2,
    ....
)
data = data.map(lambda x, y: (x / 255.0, y))
```

Images were inputted in a 256x256 pixel format. They were rescaled to fit a much smaller range of values - [0,1] compared to [0,255]. This made convergence faster and the training process more stable.

Labels

```
data_iterator = data.as_numpy_iterator()
batch = data_iterator.next()
image, labels = batch
```

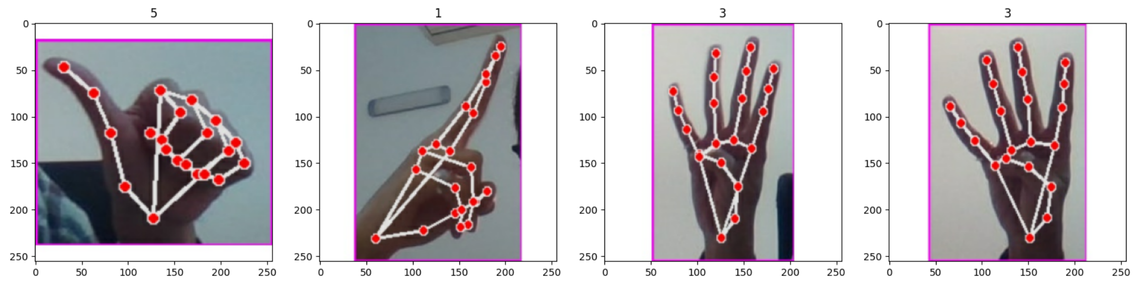

Figure 4.1: Examples of Scaled and Labelled Images

4.0.3 Evaluation Metrics

We will be testing the model on three metrics - Accuracy, Precision and Recall.

Categorical Accuracy

Categorical accuracy is commonly used for multi-class classification. It returns the proportion of correctly classified samples against the total number of samples.

Precision

Precision returns the ratio of correctly classified samples to the total number of classified samples for a particular class.
precision = true_positives / (true_positives + false_positives).

Recall

Recall measures the model's sensitivity. It returns the ratio of correctly classified samples to the the number of correctly classified and incorrectly misclassified samples for a particular class.
recall = true_positives / (true_positives + false_positives)

Chapter 5

Results

Model Evaluation Metrics

Prescision score:0.9902597665786743: Classes were identified correctly 99.03% of the time.

Recall score: 0.9902597665786743: Recall, or sensitivity was at 99.03% as well, implying that the model effectively captures the majority of the positive instances in the dataset.

Accuracy score: 0.9902597665786743: Accuracy shows the overall correctness of the model. An accuracy of about 99.03% was seen which implies that the model performed extremely well in all cases.

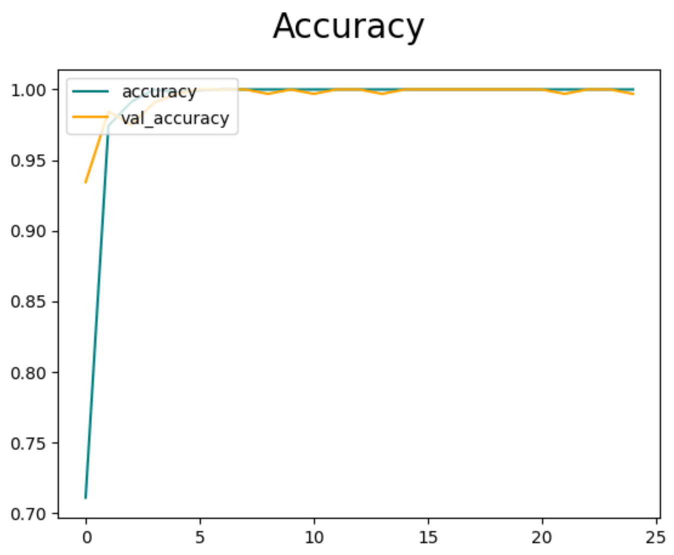

Figure 5.1: Accuracy

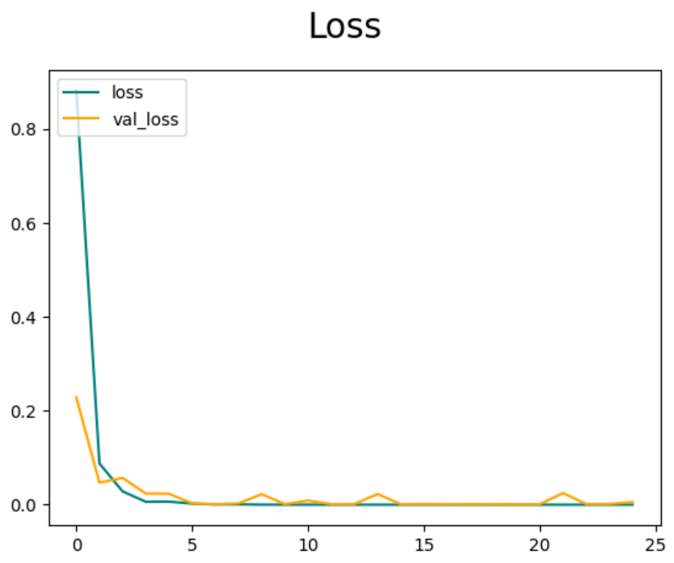

Figure 5.2: Loss

Chapter 6

Discussion

6.0.1 Challenges Encountered

Model Overfitting

In the first few iterations of the machine learning model, over-fitting the training dataset was a big problem faced. The model performed excellently on the training and validation set. It even performed well on new images passed to it.

However, the scope of the project demanded hand gestures to be classified from a live video feed. I had to really experiment to find out which hand gestures actually worked and what placement the hand needed to be in context of the entire frame. Lighting was also an issue. Images, as compared to frames from a live video stream, were much clearer. As such, the model performed better in brightly lit areas as compared to darker ones.

Hardware Setup

This was my first project involving hardware. At times, I struggled with finding the *right* equipment. In terms of the microcontroller, I started with an ADIY FLY board but soon switched to an arduino which significantly helped accelerate the building of my bare circuit. For the motor, I started out with a DC motor but shifted to a stepper

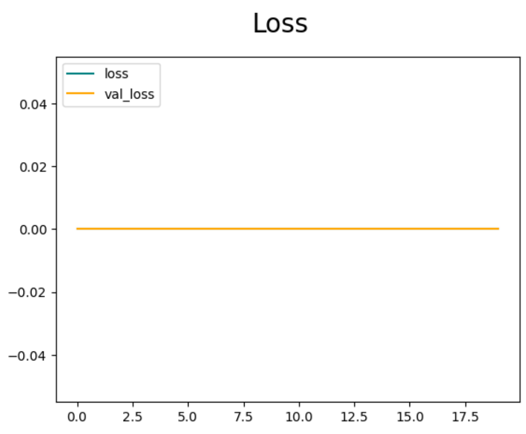

Figure 6.1: Evaluation metrics from an early iteration of the model

motor midway to have precise control over speed, direction and degree rotation.

The learning curve was steep, and looking back, the motor code has room for added complexity.

Installation Design

Choosing the right design for the installation was a convoluted process. In most designs, there was a conflict between visual appeal and auditory response.

The first iteration was aesthetically appealing. The lines were clean and it looked like a complete, stand-alone product. The open nature of the structure made for a visually stimulating experience since there was no obstruction in the line of sight of the viewer. The issue here lied with the audiotory response. The sound rods were to be fixed on a base with the help of a spring. As a result, the sound produced on striking the rod was muted and unpleasant. It had a negative effect on the overall interaction experience.

The second iteration solved for the auditory issue. Instead of having the rods fixed on a base, they were hung using threads in a wind-chime like set up. However, I wanted the product to rest on flat surface, not be suspended in the air like a wind chime. To allow this, the outer structure had be built as a bird cage to allow for both

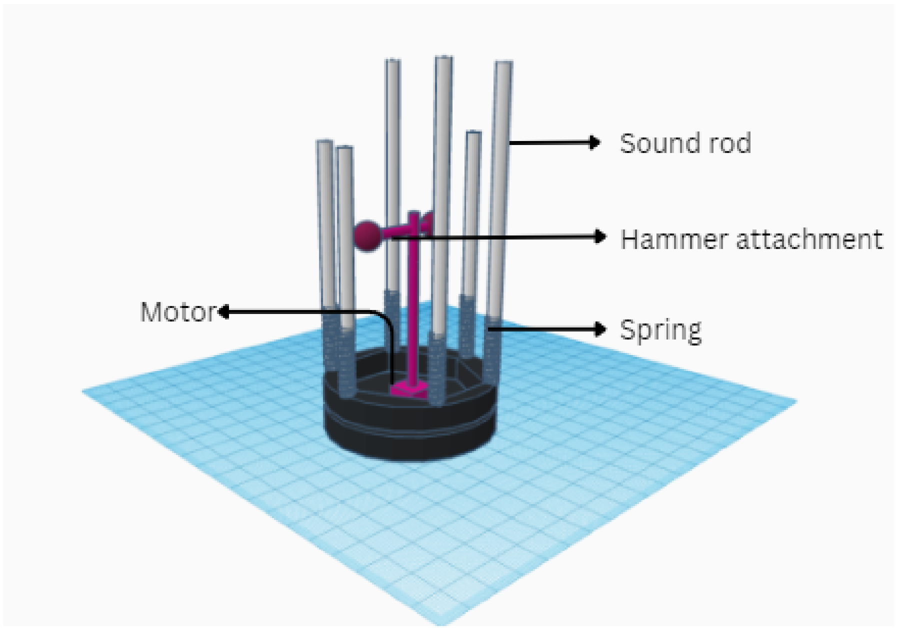

Figure 6.2: First Iteration

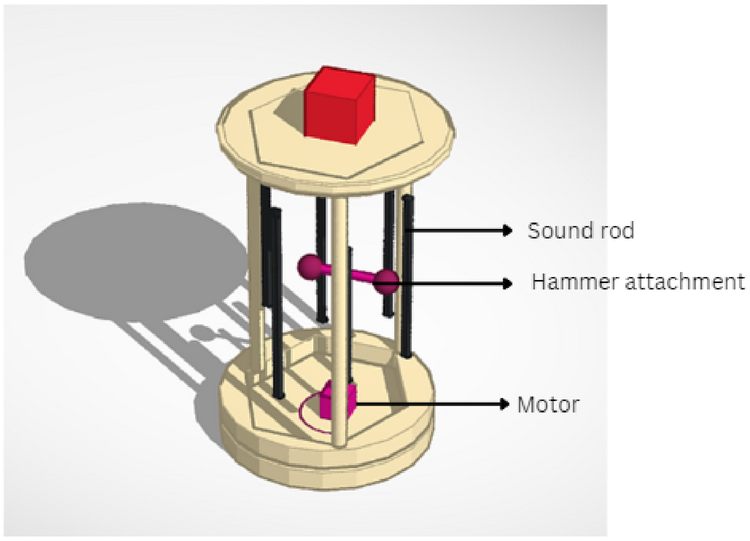

Figure 6.3: Second Iteration

rod suspension and structural support. The installation apeared cluttered. The line of view was obstructed and the visual experience was not as stimulating.

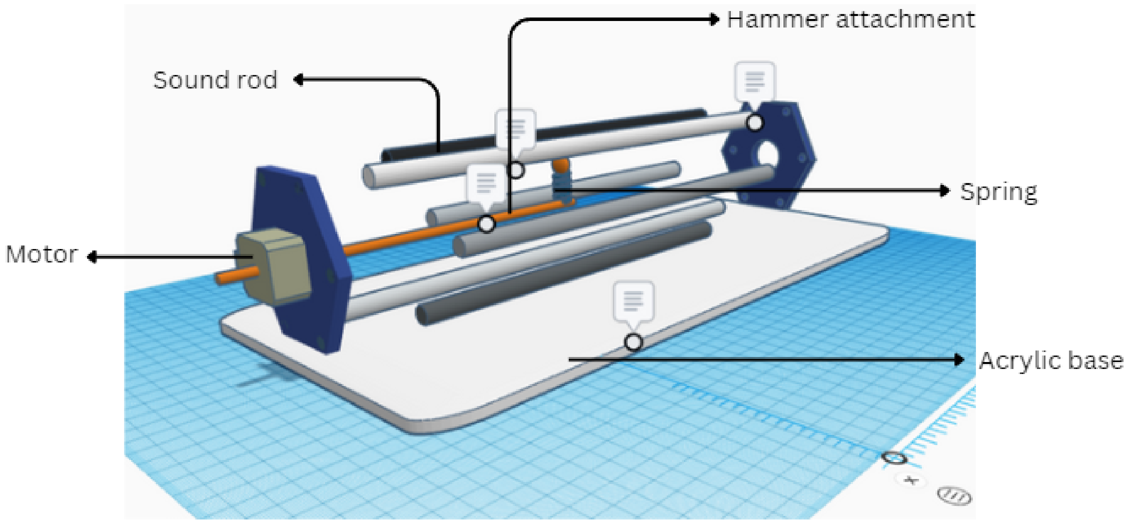

Figure 6.4: Final Iteration

The final iteration solves both issues to a certain degree. The horizontal construction allows the side panels to act as support, negating the need for additional beams. Further, by threading the hollow sound rods using a thread, we minimise the sound muting that was experienced in the first sketch, resulting in a pleasant auditory experience.

Chapter 7

Conclusion

The objective of the project was to build a gesture controlled musical instrument with the help of a stepper motor and supporting devices - an Arduino UNO microcontroller and an A4988 motor driver. The motor displays proper movement patterns according to the specified hand gestures fed to it. Seven image classes were successfully classified in the model. Each class of images i.e. each hand gesture, was mapped to an integer value that was passed to the microcontroller's IDE over a serial connection. The data was received and the code was flashed to move the motor a specified number of steps.

The *Esc* key marks the end of the interaction sequence. At this point, the entire sequence of gestures passed as inputs is played together, allowing the user to listen to their composition.

7.0.1 Enhancements

The model can be expanded to recognize more gestures. It can take two hand inputs at one time, one to control the degree of rotation in the motor, and the other to control the speed. It could also be modified to recogize hand movements in place of static gestures.

For the installation, we could add a second motor and be exposed to a new range of movement patterns. For instance, the second motor could be programmed to touch and effectively mute a particular sound rod. The different kinds of sounds that could be produced in such a setup would be very interesting.

Lastly, a lighting component could be added to the installation with the help of wLEDS. These could either be hard coded depending on specific movement patterns in the motor. Alternatively, we could introduce a microphone attachment and send sound signals to the LEDs and have an immersive, audio reactive lighting experience.

Bibliography

[1] Rakhi Bose. Mumbai artist uses ipad to reveal dystopic reality of ’normal’ families. In *Feynman and computation*. Outlook Weekender, 2023.

[2] Videoplace 1975.

[3] MoMA. Rainroom.

[4] Oudah Munir. Hand gesture recognition based on computer vision: A review of techniques. MDPI, 2020.

[5] Jedidiah Paterson. Gesture-controlled robotic arm utilizing opencv. IEEE, 2021.

[6] Kathiravan Natarajan. Hand gesture controlled drones: An open source library. IEEE, 2018.

[7] Lenman Sören. Computer vision based hand gesture interfaces for human-computer interaction. 2002.