Project Report on

*Square Game using Machine learning*

Masters of Computer Applications

Mini Project

MCA Semester I

**Submitted By**

Harshada Naik

Roll No. 1333

**Institute of Computer Science**

**Mumbai Educational Trust**

# University of MumbaiAcknowledgements

I would like to thank **Dr. Abhijit Banubakode**, Principal and **Prof. *Flavia Gonsalves***, project guide, who were a constant source of help and played an important role in the successful execution of this project.

**CERTIFICATE OF APPROVAL**

This is to certify that the project report titled

*Square Game using Machine Learning*

Is a bonafide record of the work done by

*Harshada Naik*

*Roll No. 1333*

MCA Semester I

Under the guidance of

Prof. *Flavia Gonsalves*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Dr. Abhijit Banubakode Prof. *Flavia Gonsalves*

Principal Project Guide

MET-ICS MET-ICS

**CERTIFICATE OF APPROVAL**

This is to certify that the project report titled

*Square Game using Machine Learning*

Is a bonafide record of the work done by

*Harshada Naik*

*Roll No. 1333*

This project is approved as Mini Project for

MCA Semester I

University of Mumbai

Examiner Examiner

Sign : \_\_\_\_\_\_\_\_\_ Sign : \_\_\_\_\_\_\_\_\_

Name: \_\_\_\_\_\_\_\_\_ Name: \_\_\_\_\_\_\_\_\_

Date : \_\_\_\_\_\_\_\_\_ Date : \_\_\_\_\_\_\_\_\_

**Index**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Contents** | **Page No.** |
| **1** | **Scope of Project** |  |
| **2** | **Project Description & Limitations** |  |
| **3** | **UML Diagrams** |  |
| **4** | **Screen Shots** |  |
| **5** | **Conclusion** |  |
| **6** | **Biblography** |  |

**Scope of Project**

This is a web application game which can be played on both the platform such as android and web as well. The objective of this square game is to train and apply machine learning model based on image classification and sound classification.

This game has been developed using javascript language, HTML, css and machine learning model with the help of Teachable machine. This game also includes object oriented concepts. A machine learning project starts with obtaining data, processing, model training/evaluation and ends in model deployment.

This is the square game in which the object moving to the left, right, up and down direction over voice controls and also by showing the images by enabling camera is moving to those specified directions by using machine learning concepts.

The game mainly consists of two square components. we just need to take our one big square to the another small sqaure so that when one sqaure’s head get attached to the head of another square, The biggest square get increase in size and grows and as the number of square attaches to each other, the length of the biggest square increases which makes the game more interesting. This all the action happens over a voice control as well as the image classification which is a machine learning concepts.

There are four steps for preparing a machine learning model:

1. Preprocessing input data
2. Training the deep learning model
3. Storing the trained deep learning model
4. Deployment of the model

**Project Description & Limitations**

A Project Description is a document that outlines the details of a specific project in a structured format covering all stages of the project and the processes involved in it.

**Hardware Requirements**

* RAM : 2 GB (Minimum)
* Processor : 1 GHZ or higher
* Hard disk : 32 GB or more
* System: intel core i3
* Camera
* Microphone

**Software Requirements**

* Languages: javascript, HTML, CSS
* Machine Learning model – Teachable Machine
* Operating system – windows 10
* Camera permission
* Microphone permission

**System Coding conventions:**

1. **Javascript**

JavaScript was initially created to “make web pages alive”.

The programs in this language are called scripts. They can be written right in a web page’s HTML and run automatically as the page loads.

Scripts are provided and executed as plain text. They don’t need special preparation or compilation to run.

In this aspect, JavaScript is very different from another language called Java.

There are ways to interact with the camera/microphone and other devices, but they require a user’s explicit permission. So a JavaScript-enabled page may not sneakily enable a web-camera, observe the surroundings and send the information to the NSA.

There are at least *three* great things about JavaScript:

* Full integration with HTML/CSS.
* Simple things are done simply.
* Supported by all major browsers and enabled by default.

JavaScript is the only browser technology that combines these three things. That’s what makes JavaScript unique. That’s why it’s the most widespread tool for creating browser interfaces.

That said, JavaScript can be used to create servers, mobile applications, etc.

1. **Teachable machine**

Teachable Machine is a web-based tool that makes creating machine learning models fast, easy, and accessible to everyone.

How do you can use it?

1. **Gather**

Gather and group your examples into classes, or categories, that you want the computer to learn.

1. **Train**

Train your model, then instantly test it out to see whether it can correctly classify new examples.

1. **Export**

Export your model for your projects: sites, apps, and more. You can download your model or host it online.

Teachable Machine is flexible – use files or capture examples live. It’s respectful of the way you work. You can even choose to use it entirely on-device, without any webcam or microphone data leaving your computer.

**Images:** Teach a model to classify images using files or your webcam.

**Sounds:** Teach a model to classify audio by recording short sound samples.

**Poses:** Teach a model to classify body positions using files or striking poses in your webcam.

The models you make with Teachable Machine are real TensorFlow.js models that work anywhere javascript runs, so they play nice with tools like Glitch, P5.js, Node.js & more.

Plus, export to different formats to use your models elsewhere, like Coral, Arduino & more.

**Project Description:**

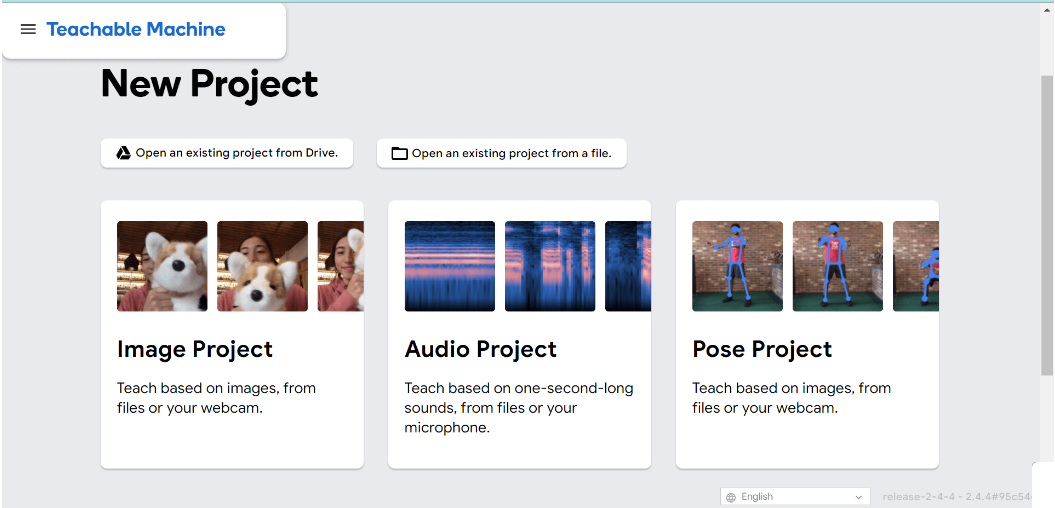
1. **Build Machine Learning model using Teachable Machine.**

A teachable machine is a web-based tool that quickly and easily creates models. It is also flexible. It can be used to teach a model how to identify images and pose through images or live webcam. It is free and best for students. Teachable Machine creates a **Tensorflow** model, which can be integrated with any website app, Android application, or other platforms. There is no need to create an account. It was so easy.

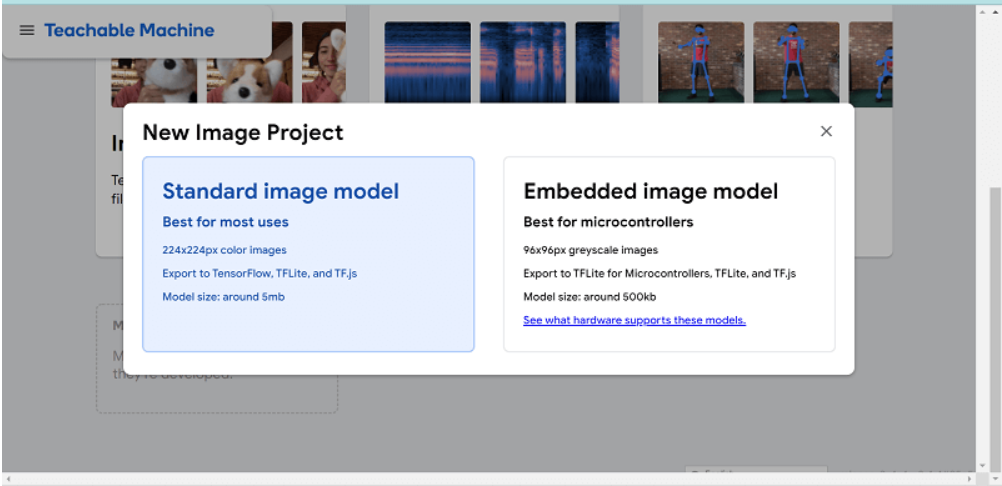
**Let's Build a Model for image Model.**

**Step 1:** Go to Teachable Machine:  <https://teachablemachine.withgoogle.com/train>.

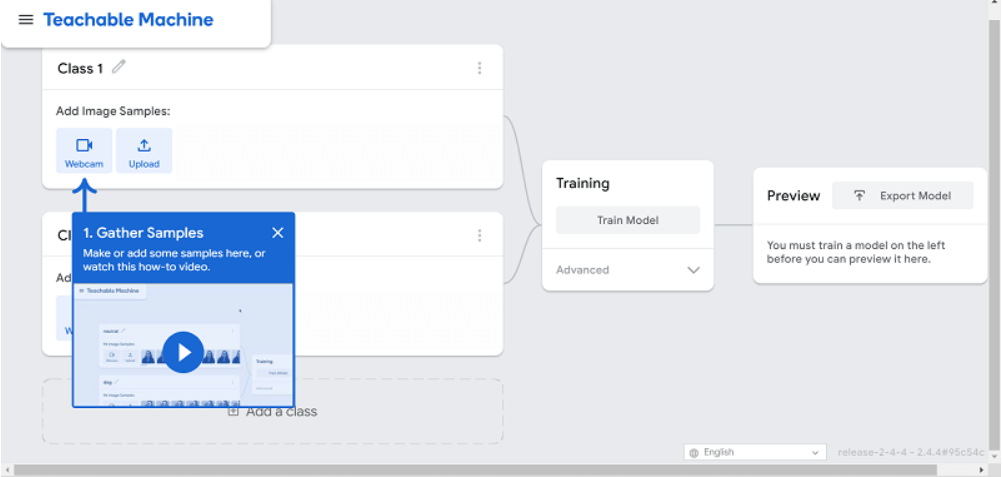
We will be directed to the below-shown screen that consists of three options - Image, Audio, and Pose.



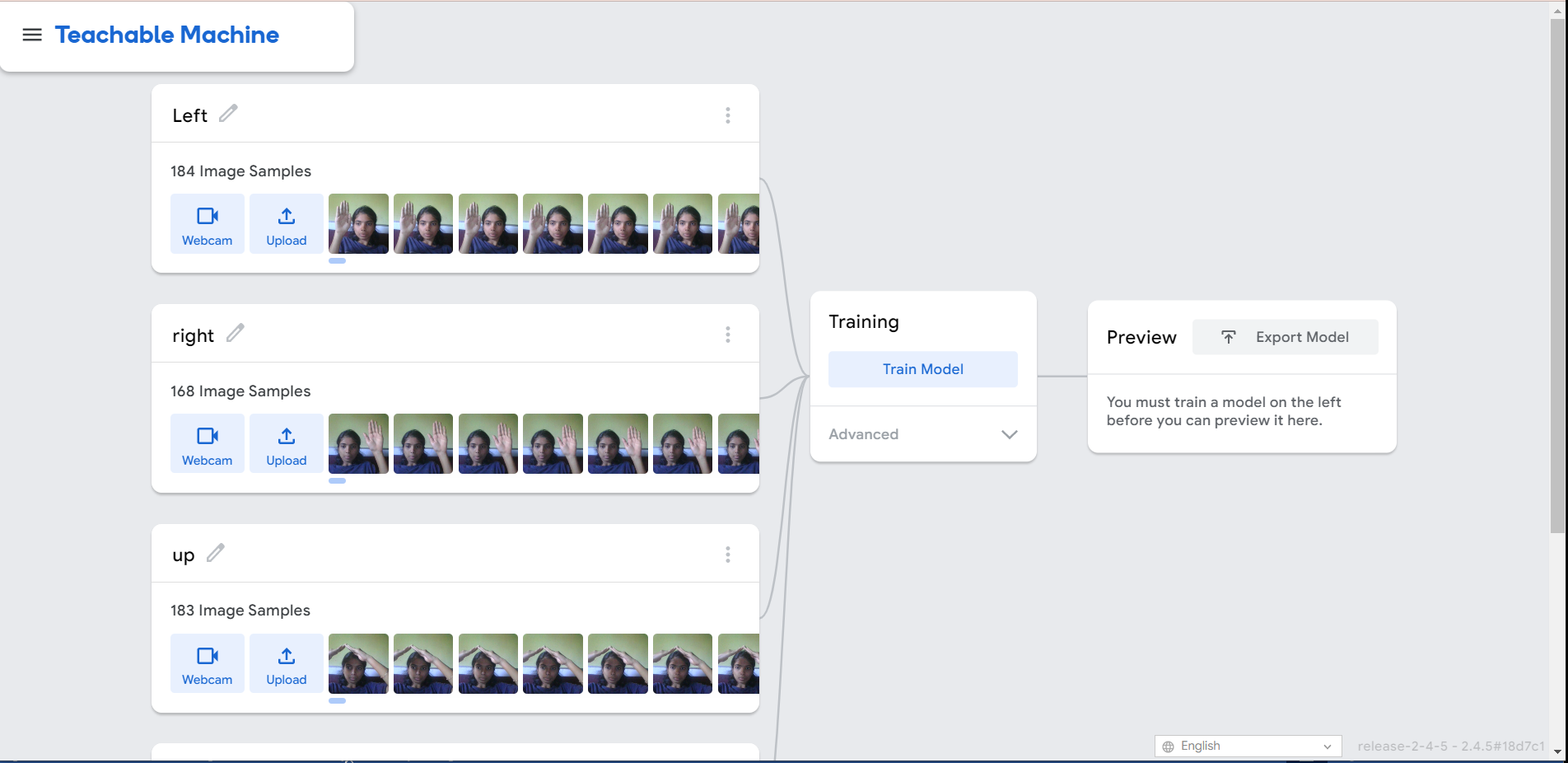
**Step 2:** Choose an image project. We will see two options again: standard or embedded. We aren't making this for micro-controllers, so we recommend choosing a standard. If the users are interested, then select Embedded Image Model. Even if they choose Embedded, the process will remain the same. It is only the model that will differ.



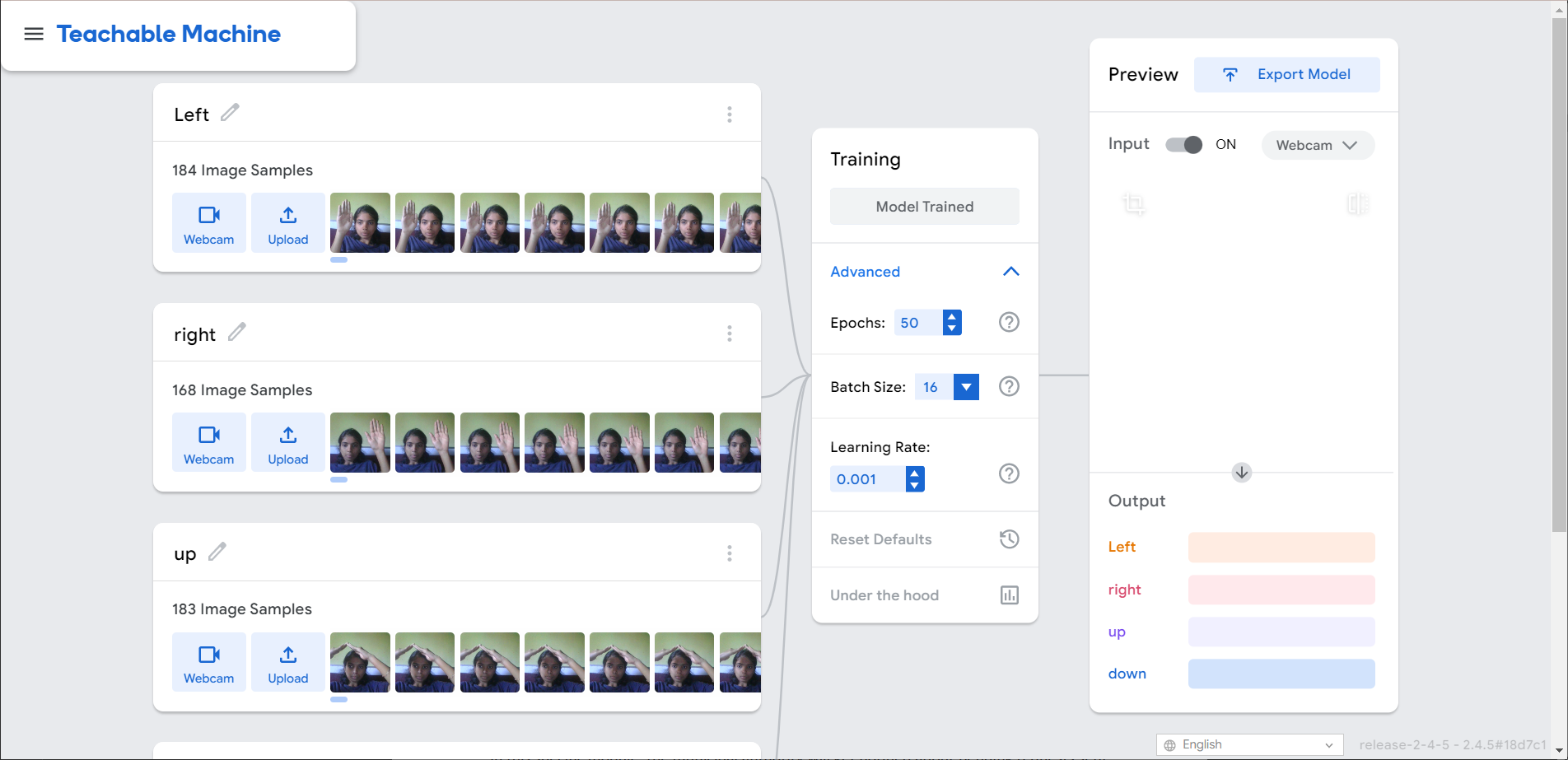
Clicking on **Standard Image Project** will take us to the screen below. Here we can add classes to the model. we have two choices: **upload images from the databank** or **use the live camera to capture images**.



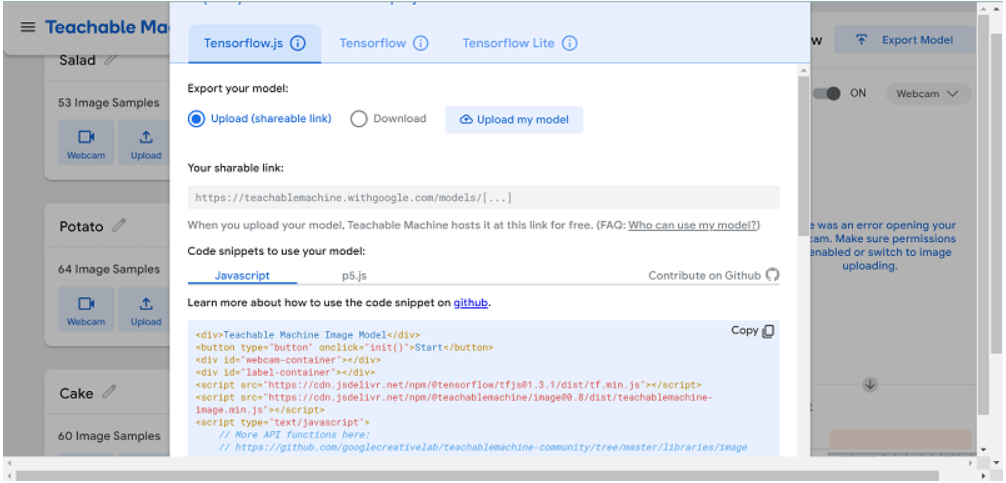
**Step 3:** Now create classes and upload the images. We will only create four classes: **Left, Right**, **Up**, Down. We have replaced class1 with left, class2 with right, class3 with up and class4 is now down. The user can make classes as many as they need in their particular project.



Click on **Train Model** after we have uploaded the images. There are three options available: **Batch Size, Epochs** and **Learning rate**. These options are not something we have ever heard of, so we don't be alarmed if they're new to us. It's important that we play with the models and determine which values give the best accuracy to make it more efficient. A model is useless if it's not accurate. We can adjust their values to find the best model. Here we will use default values.



**Step 4:** After the model has been trained, it is time to export it.



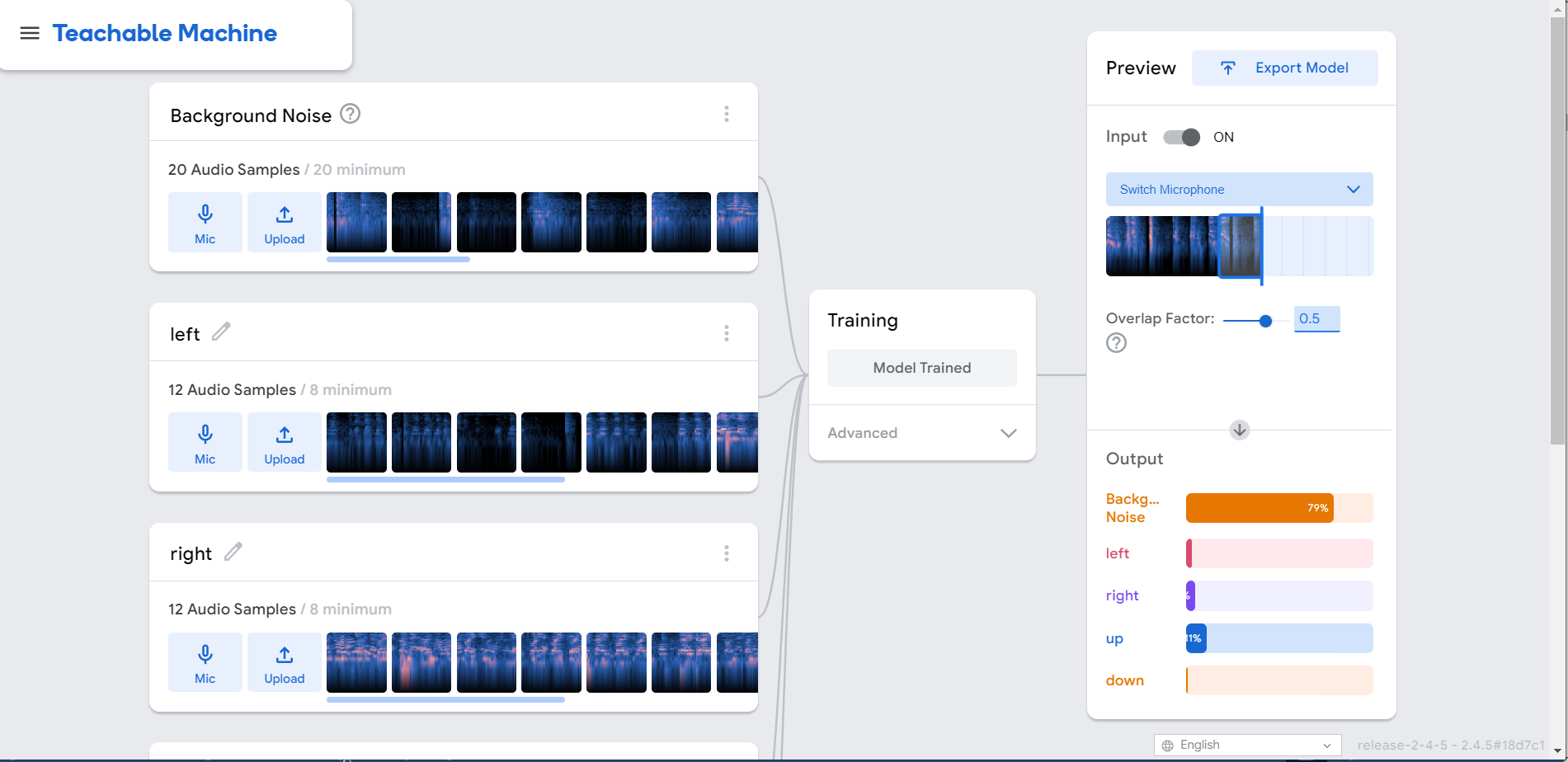
We will see several options when we click Export Model. The code snippets can help integrate the model into our application. **Tensorflow.js** models are compatible with all JavaScript libraries and frameworks. Some frameworks only support a specific type of model. We will check to see if our library or framework supports this model.

The download of the model can take some time. This is how we create a machine-learning model.

We can also create models for **audio** and **pose**, similar to the image project.

**Let's Build a Model for Audio Model.**

An audio project will create a model capable of detecting sound. We created five classes: Background Noise, left, right, up and down. In the preview section, after training the model, we tested the model's efficiency using noise. In the Output of the preview, we can see more background noise. We need to increase the number of samples to improve the model's learning.



1. **Create Html file**

This file contains some basic HTML tags like div, h1, title, etc. also we’ve used bootstrap (CDN is already included) along with that we have used some libraries with script tag.

**Libraries:**

1. **cdnjs.cloudflare.com** is an ultra-fast, reliable, globally available content delivery network for open-source libraries. Cloudflare works with the maintainers of the cdnjs project and distributes the latest versions as they are released.
2. **Ml5. js** aims to make machine learning more approachable to a broad audience of artists, designers, creative coders and students. The library provides access to machine learning algorithms and models in the browser, building on top of TensorFlow. It builds on Tensorflow.js and provides friendly access to machine learning algorithms and models in the browser.
3. **p5.sound** extends p5 with Web Audio functionality including audio input, playback, analysis and synthesis.

**Code:**

<!DOCTYPE html>

<html lang="en">

<head>

<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.5.0/p5.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/1.5.0/addons/p5.sound.min.js"></script>

<script src="https://unpkg.com/ml5@0.4.1/dist/ml5.min.js"></script>

<script src="path/to/p5.sound.js"></script>

<link rel="stylesheet" type="text/css" href="style.css">

<meta charset="utf-8" />

</head>

<body>

<main>

</main>

<script src="sketch.js"></script>

<script src="square.js"></script>

</body>

</body>

</html>

1. **Creating a javascript file – Square.js**

* So, here first will create a square class. So that I will create a square object in my code and it has a constructor function.
* Here we are going to call it a body is an array and with index 0 it creats a vector. createVector object has an X and Y. and will create this vector at 0, 0. And the direction of the X and Y is also set to 0 to move it horizonatally and vertically only.
* Update() function is use to update the body when the square gets collide with each other. It will create a copy of the object and will get added into the head of another object.
* Show() function is use to display raectangle with a black fill color and increased in size as it get collided and also it will move to the specified direction with 1 pixel.
* setDir(x, y) function is use to set the direction, so whatever comes into the function sets those things.
* eat(pos) function is to increase the length of the black object if its get collided to the another object which is of red color.
* Inside the grow() function, here we are expanding a array by pushing a blank vector at the end of the array.

**Code:**

class Square {

constructor() {

this.len = 0;

this.body = [];

this.body[0] = createVector(floor(w/2), floor(h/2));

this.xdir = 0;

this.ydir = 0;

}

setDir(x,y) {

this.xdir = x;

this.ydir = y;

}

update() {

let head = this.body[this.body.length-1].copy();

this.body.shift();

head.x += this.xdir;

head.y += this.ydir;

this.body.push(head);

//this.body[0].x += this.xdir;

//this.body[0].y += this.ydir;

}

grow() {

let head = this.body[this.body.length-1].copy();

this.len++;

this.body.push(head);

}

endGame() {

let x = this.body[this.body.length-1].x;

let y = this.body[this.body.length-1].y;

if (x > w-1 || x < 0 || y > h-1 || y < 0) {

return true;

}

for (let i=0; i < this.body.length-1; i++) {

let part = this.body[i];

if (part.x == x && part.y == y) {

return true;

}

}

return false;

}

eat(pos) {

let x = this.body[this.body.length-1].x;

let y = this.body[this.body.length-1].y;

if (x == pos.x && y == pos.y) {

this.grow();

return true;

}

return false;

}

show() {

for (let i = 0; i < this.body.length; i++) {

fill(0);

noStroke();

rect(this.body[i].x, this.body[i].y, 1, 1);

}

}

}

1. **Creating a javascript file – sketch.js**

* All work will be done by the canvas, so firstly we need to access the canvas in our JS file with a 640 by 520 canvas with createCanvas() mathod.
* Function controlSquare() is used to control the square to the left, right and up, down direction as per the label shows there on screen which gets printed by machine learning model which we have trained by enabling a camera.
* Inside the draw function, Scale by rez i. e. resolution increases the size of the square that means it will scaling it up as we increases the size. Here will also called the update and show function which is specified in another js file.
* framerate will slow down speed or increases the speed of the object so as to control the speed of the object.
* So with inside the FoodLocation() we gonna create a another object which is of red color, and if the two object get collided it will also get the new food location.
* endGame()function is use when head of the object get collided to any of the sides the game will get over and screen turns out to the red color.

**Implementing machine learning model for Image Classification:**

1. First, we have to train the model for image classification as I’ve mentioned above stepwise.
2. After training the model everything happens right there locally in the browser for that you have to be connected to the internet to access the teachable machine website. Model is training in the browser.
3. Then save the model by clicking to the Export model button.
4. Here I’ve use it with Tensorflow.js and a library called ml5 JS which is compatible model for ml5 and Tensorflow JS is the same model.
5. Now, click on upload my model. After that, you can a see a url which we going to use in our project to apply that model.
6. You can also test the model by previewing itself in the browser.
7. What this going to do is, it will capture the video from the camera and display it on canvas and will add bits of code to load the model, give the video to the model and get the result back, and display that result back to the canvas.
8. Code Implementation:

* First will have single variable for video, we connect to the capture device – the webcam in setup() function.
* In draw() function, we draw the video on the canvas
* Now step 1 is to load the model to the project, for that we are going to use preload() function which will load any important assests, images, data files, models, before the program starts in Setup.
* We are also going to create another variable classifier and in preload() function, set the classifier equal to ML5.imageClassifier, and then URL of the model with extension of .JSON. We will copy that URL from the training page, and paste it right there in the ML5 imageClassifier function
* Now Step 2 is to classifying the model, for that will create classifyVideo() function. My variable that stores the model now is called classifier. So the function in ML5 to classify an image is called “Classify”. And then the first argument to the function is the image that I want to classify, which is the video and gotResults which will going to report and event back to get a callback.
* Step 3 is to get the cclassification which can be done by writing a function called gotResults which we have written in previous step. It can give an error message or it can receive some results. So if there will be error, it going to display the error otherwise it will display the label by classifying the images as left, right, up and down and also controlling the object.
* Step 4 is to draw the label using textsize, text, fill function which will get displayed on the canvas.

**Code:**

let video;

//let flipVideo;

let label = 'waiting...';

let classifier;

//step 1: Load the model

function preload() {

classifier = ml5.imageClassifier('https://teachablemachine.withgoogle.com/models/AXYeq3Rsx/model.json');

}

let square;

let rez = 20;

let food;

let w;

let h;

function setup() {

createCanvas(640, 520);

video = createCapture(VIDEO);

video.hide();

//flipVideo = ml5.flipImage(video);

//step 2: start classifying

classifyVideo();

w = floor(width / rez);

h = floor(height / rez);

frameRate(5);

square = new Square();

foodLocation();

}

function classifyVideo() {

//flipVideo = ml5.flipImage(video);

classifier.classify(video, gotResults);

}

function foodLocation() {

let x = floor(random(w));

let y = floor(random(h));

food = createVector(x, y);

}

function controlSquare() {

if(label === 'left') {

square.setDir(-1, 0);

} else if (label === 'right') {

square.setDir(1, 0);

} else if (label === 'down') {

square.setDir(0, 1);

} else if(label === 'up') {

square.setDir(0, -1);

}

}

function draw() {

background(220);

image(video, 0, 0);

textSize(32);

fill(255);

text(label, 10, 50);

scale(rez);

if (square.eat(food)) {

foodLocation();

}

square.update();

square.show();

if (square.endGame()) {

print('End Game');

background(255, 0, 0);

noLoop();

}

noStroke();

fill(255, 0, 0);

rect(food.x, food.y, 1, 1);

}

//step 3. get the classification

function gotResults(error, results) {

if(error) {

console.error(error);

return;

}

label = results[0].label;

controlSquare();

classifyVideo();

}

**Implementing machine learning model for Sound Classification:**

1. The steps for creating the model will be the same. We just have to select sound instead of images while training the model for sound classification.
2. When will preview the model in the browser, it will classify the sound as per our requirements. And in project it will print as a label on the canvas and move to the specified direction.
3. Code Implementation:

* Now step 1 is to load the model to the project, for that we are going to use preload() function which will load any important assests, images, data files, models, before the program starts in Setup.
* We are also going to create another variable classifier and in preload() function, set the classifier equal to ML5.soundClassifier, and then URL of the model with extension of .JSON. We will copy that URL from the training page, and paste it right there in the ML5 soundClassifier function.
* Now Step 2 is to classifying the model, for that will create classifyAudio() function. My variable that stores the model now is called classifier. So the function in ML5 to classify an audio is called “Classify”. And then the first argument to the function is the sound that I want to classify, which get printed on the canvas as a label and gotResults which will going to report and event back to get a callback.
* Step 3 is to get the cclassification which can be done by writing a function called gotResults which we have written in previous step. It can give an error message or it can receive some results. So if there will be error, it going to display the error otherwise it will display the label by classifying the images as left, right, up and down and also controlling the object.
* Step 4 is to draw the label using textsize, text, fill function which will get displayed on the canvas.

**Code:**

let video;

let label = 'waiting...';

let classifier;

//step 1: Load the model

function preload() {

classifier = ml5.soundClassifier('https://teachablemachine.withgoogle.com/models/rymbs3AEM/model.json');

}

let square;

let rez = 20;

let food;

let w;

let h;

function setup() {

createCanvas(640,520);

//step 2: start classifying

classifyAudio();

w = floor(width / rez);

h = floor(height / rez);

frameRate(5);

square = new Square();

foodLocation();

}

function classifyAudio() {

classifier.classify(gotResults);

}

function foodLocation() {

let x = floor(random(w));

let y = floor(random(h));

food = createVector(x, y);

}

function controlSquare() {

if(label === 'left') {

square.setDir(-1, 0);

} else if (label === 'right') {

square.setDir(1, 0);

} else if (label === 'down') {

square.setDir(0, 1);

} else if(label === 'up') {

square.setDir(0, -1);

}

}

function draw() {

background(220);

//textAlign(CENTER, CENTER)

textSize(32);

fill(255);

text(label, 10, 50);

scale(rez);

if (square.eat(food)) {

foodLocation();

}

square.update();

square.show();

if (square.endGame()) {

print('End Game');

background(255, 0, 0);

noLoop();

}

noStroke();

fill(255, 0, 0);

rect(food.x, food.y, 1, 1);

}

//step 3. get the classification

function gotResults(error, results) {

if(error) {

console.error(error);

return;

}

label = results[0].label;

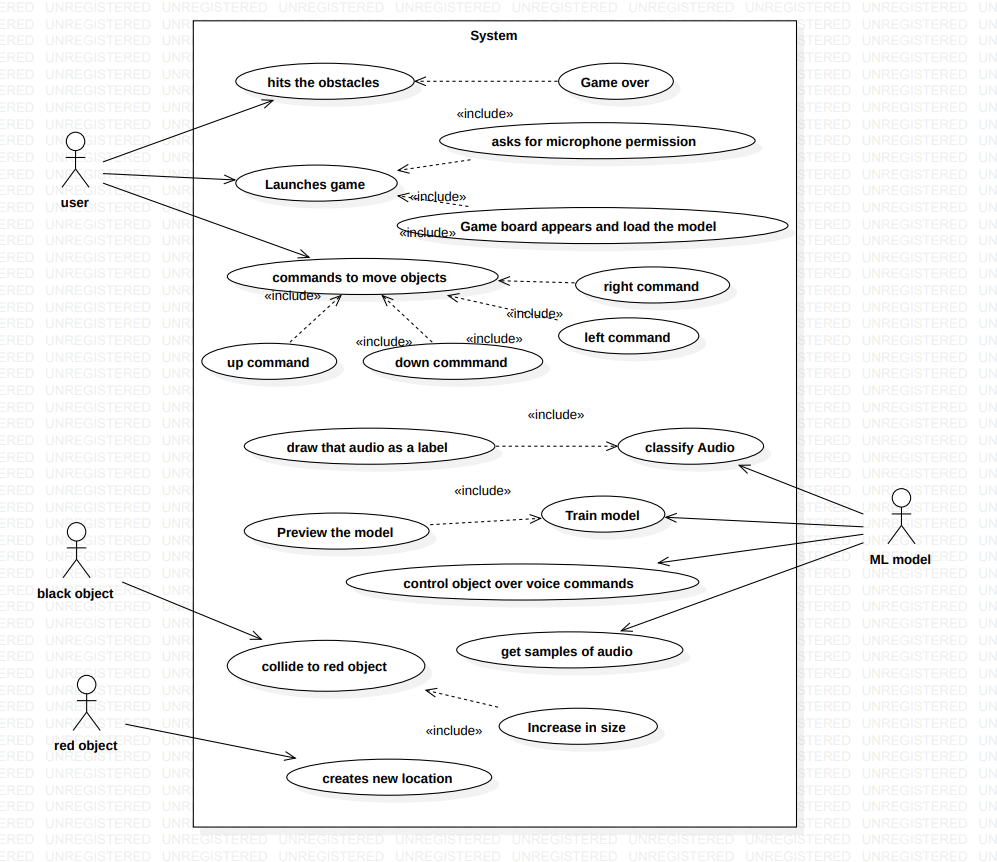
controlSquare();

classifyAudio();

}

**UML Diagrams**

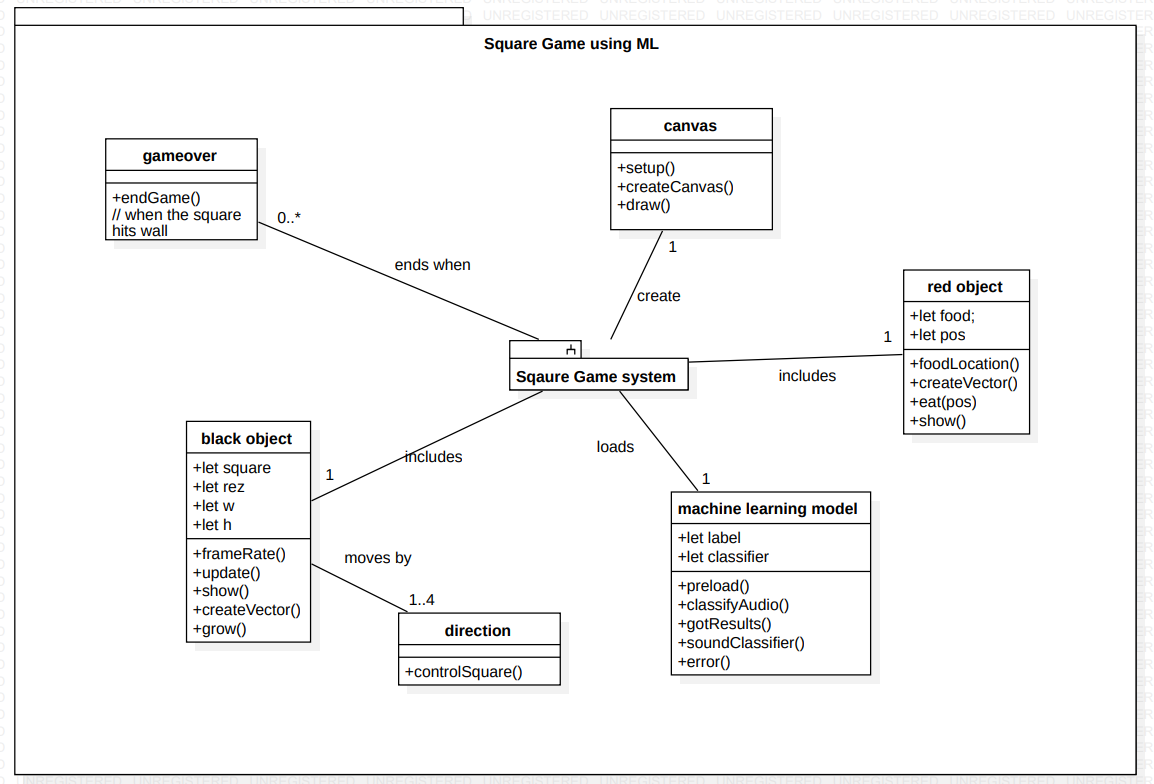
Use case Diagram for sound Classification



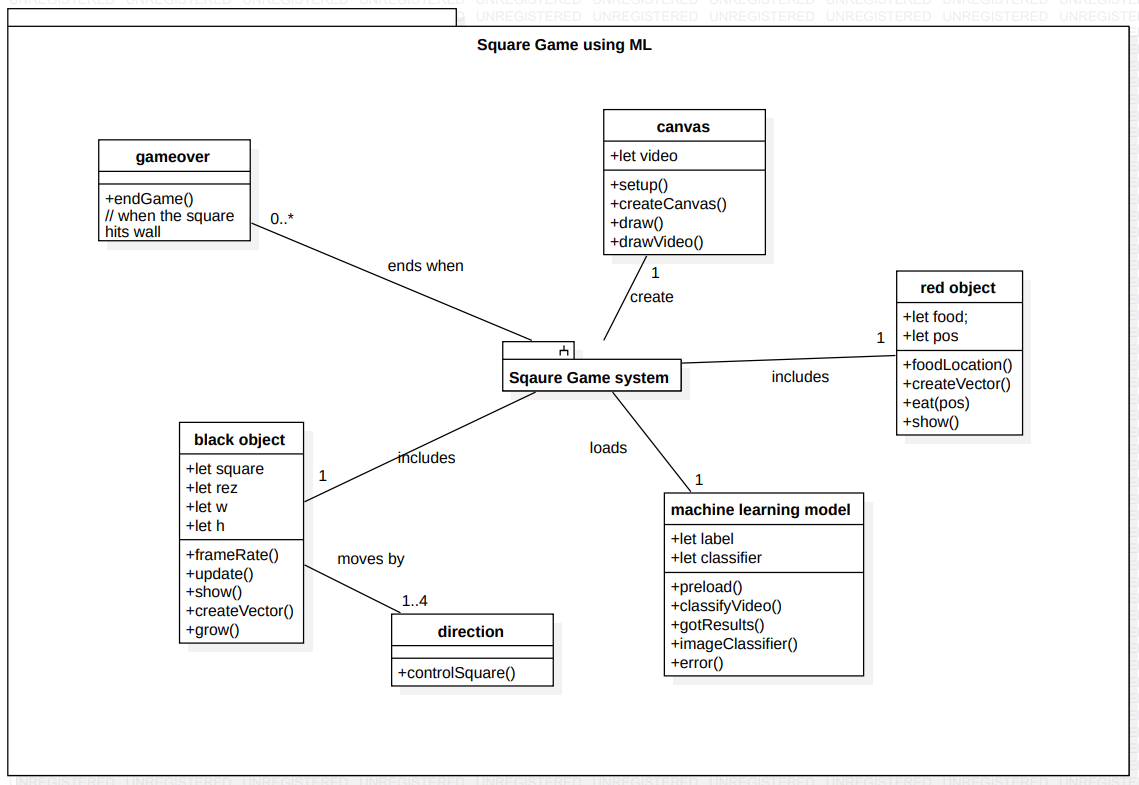
Use case Diagram for image Classification



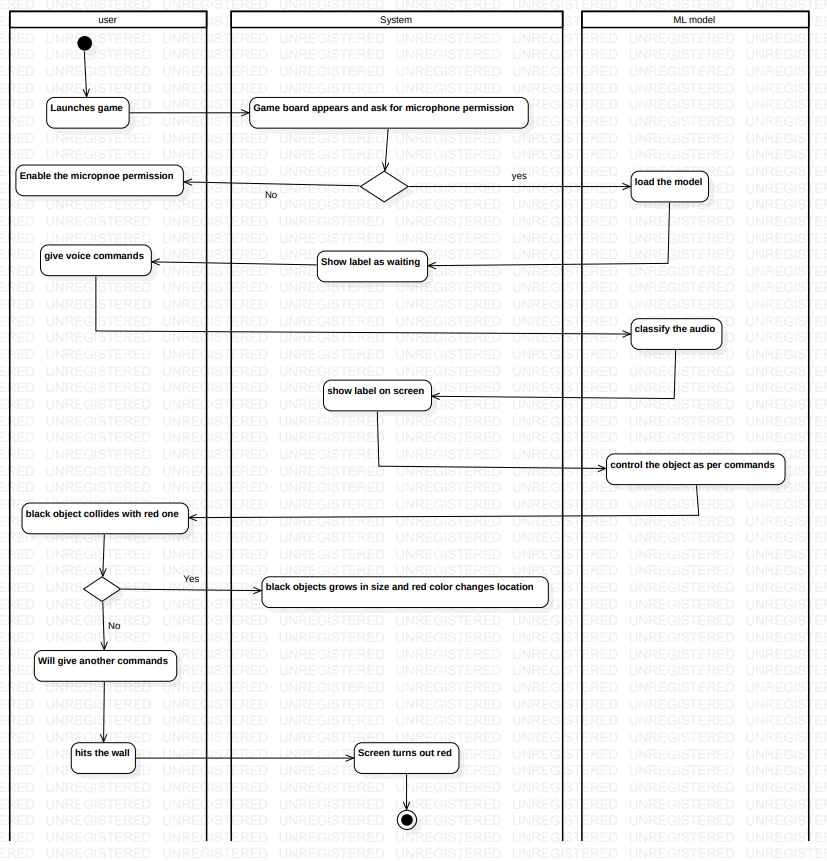
Class Diagram for sound Classification



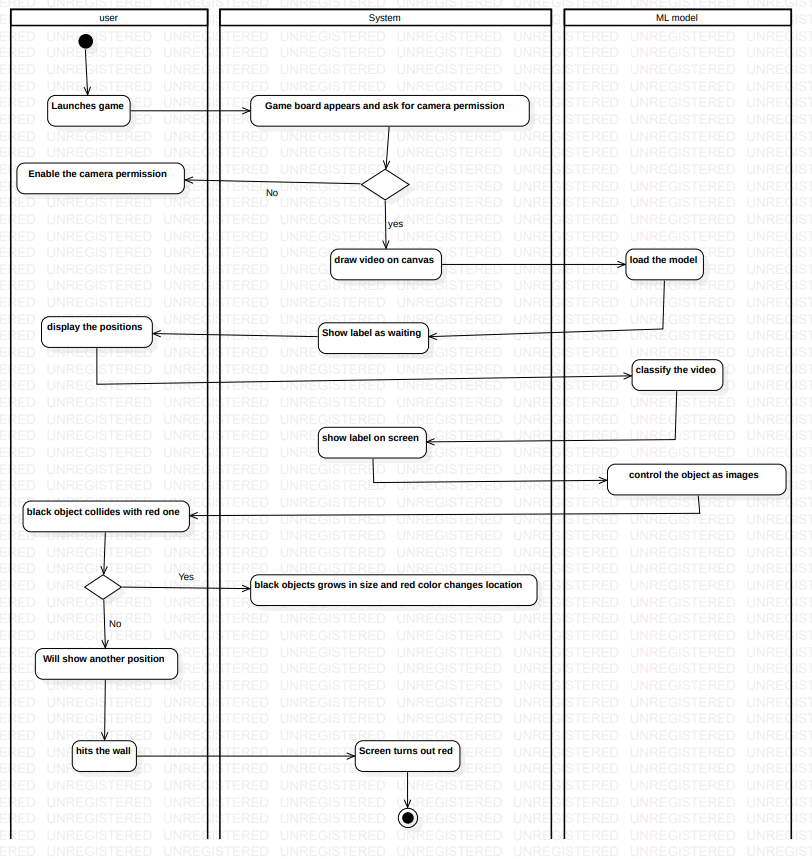
Class Diagram for image Classification



Activity Diagram for sound Classification



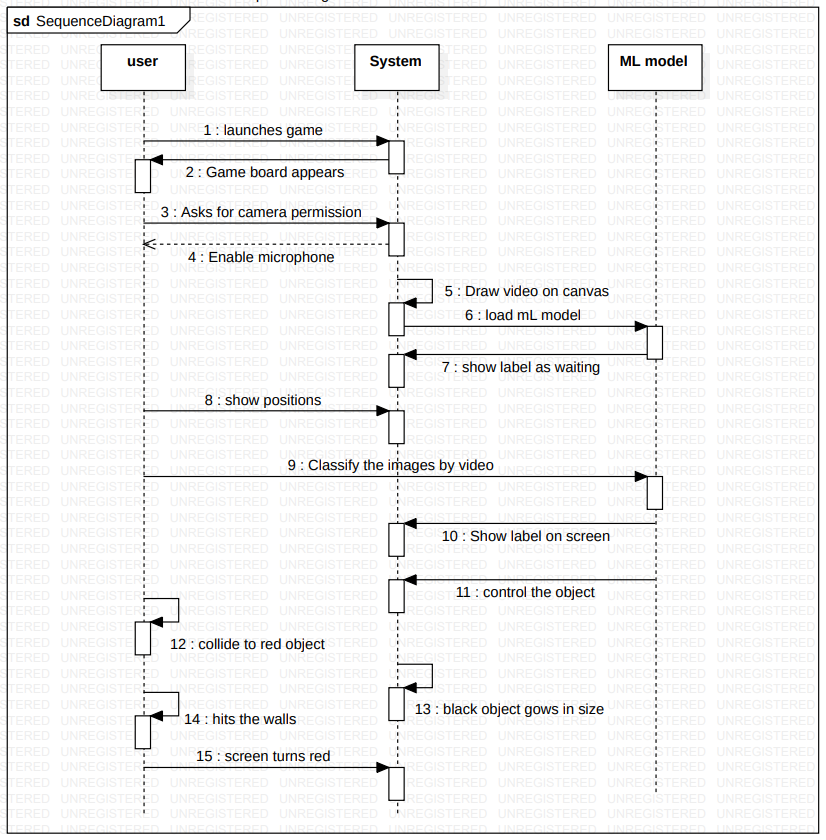
Activity Diagram for image Classification



Sequence Diagram for sound Classification

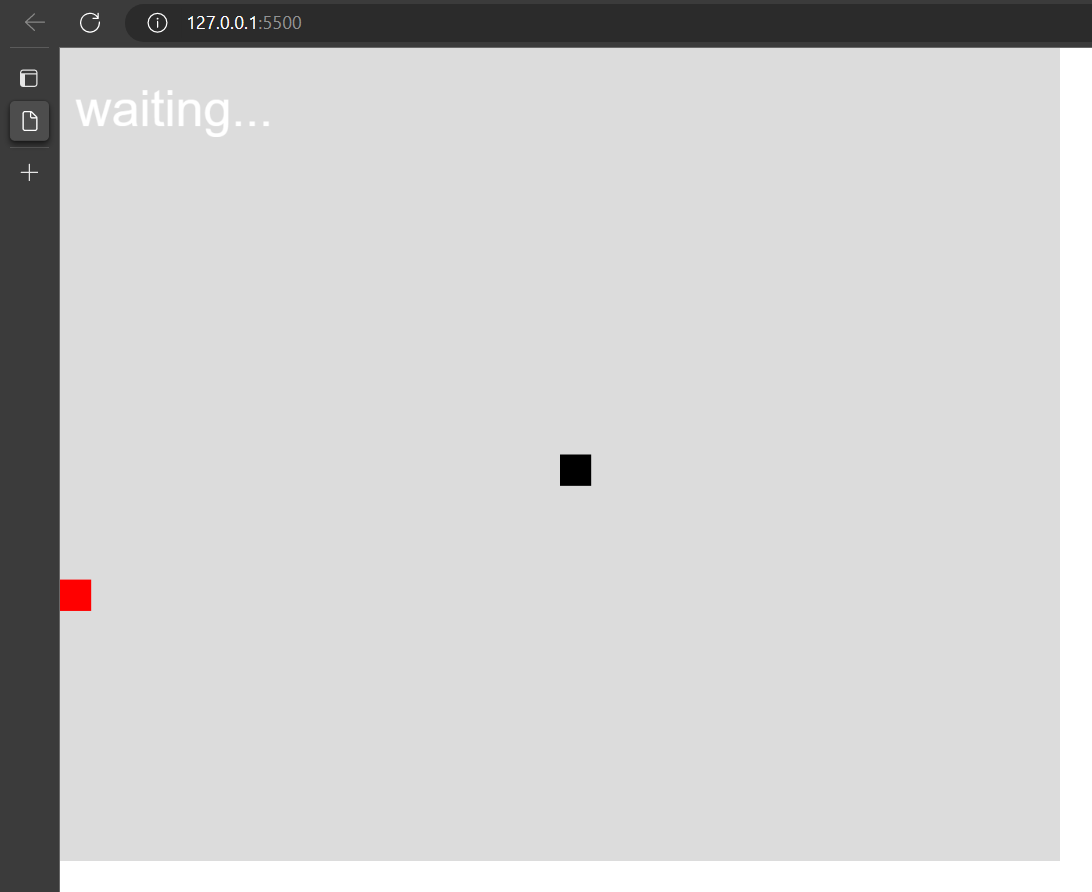


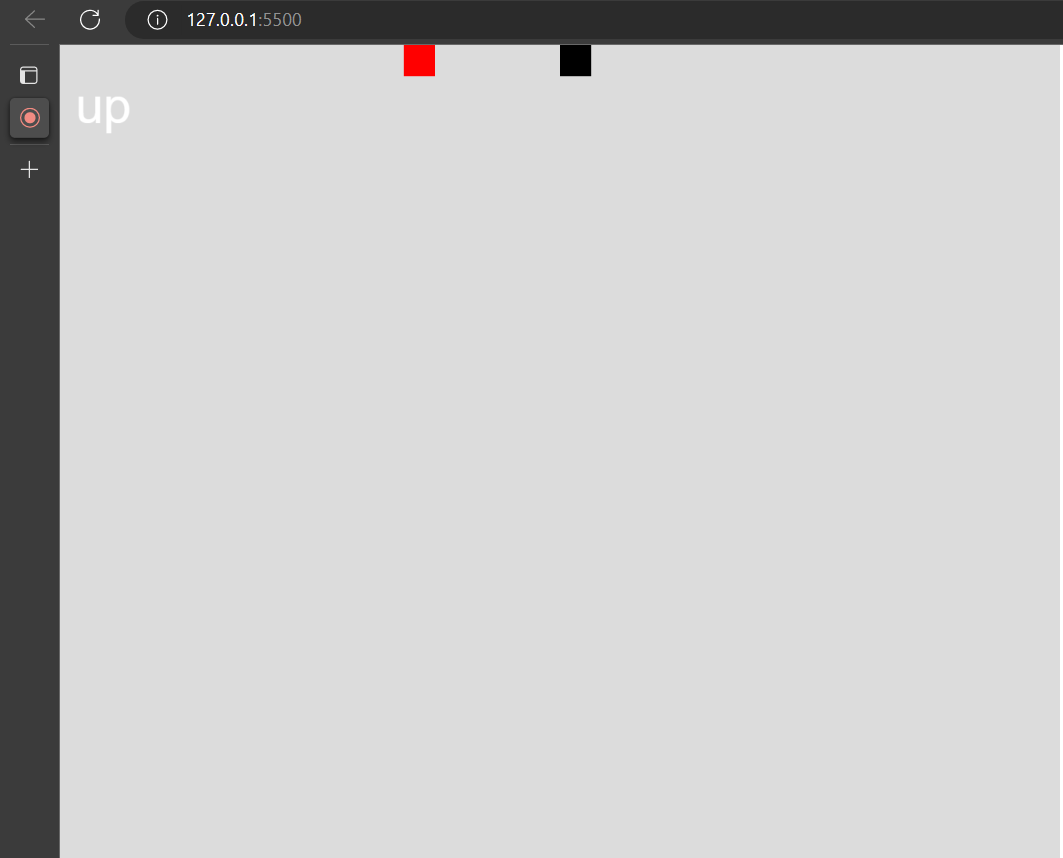
Sequence Diagram for image Classification

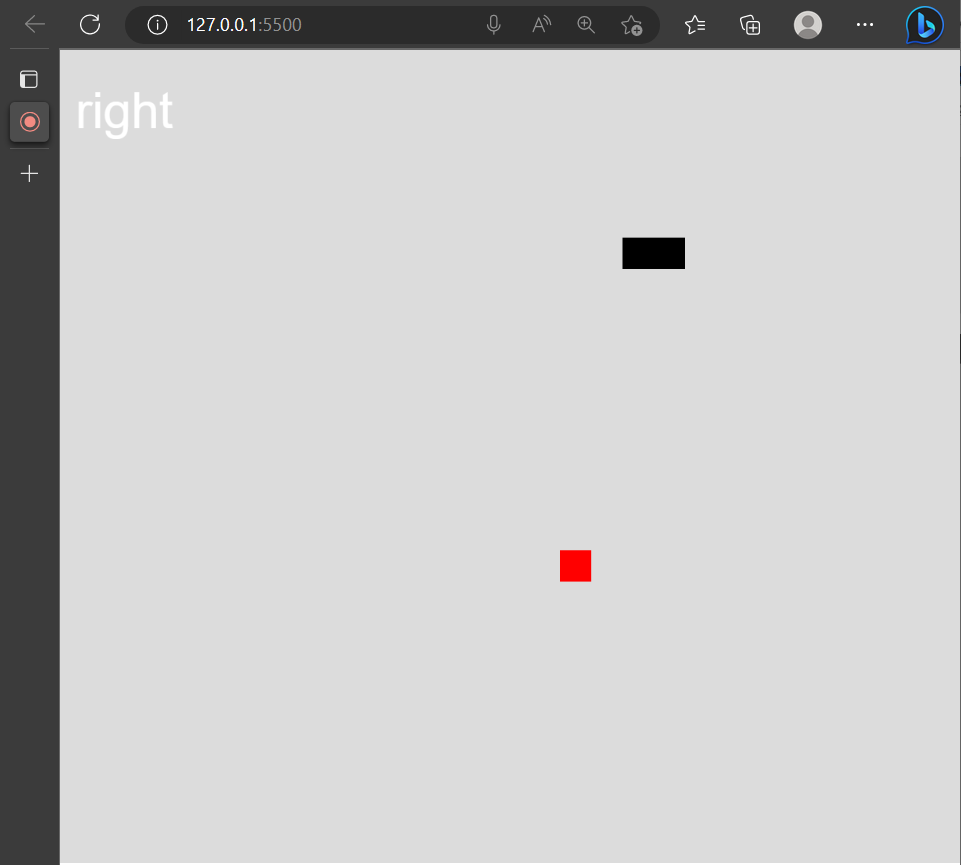


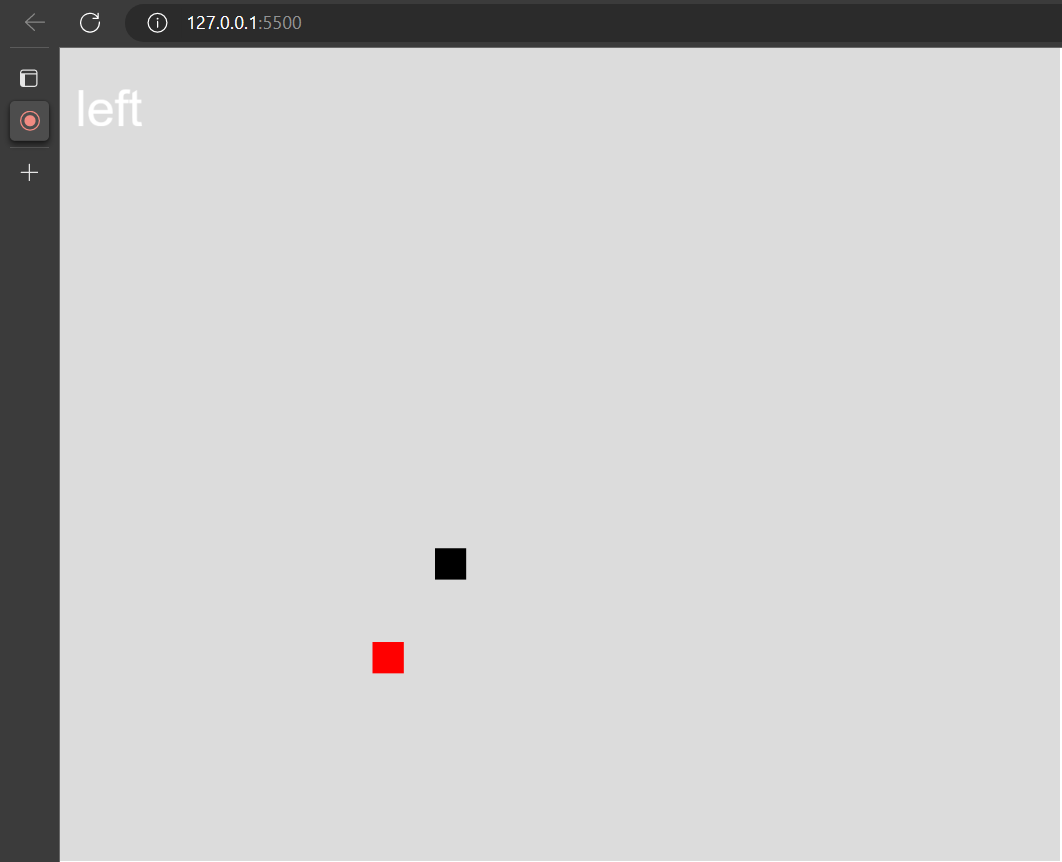
**Screenshots**

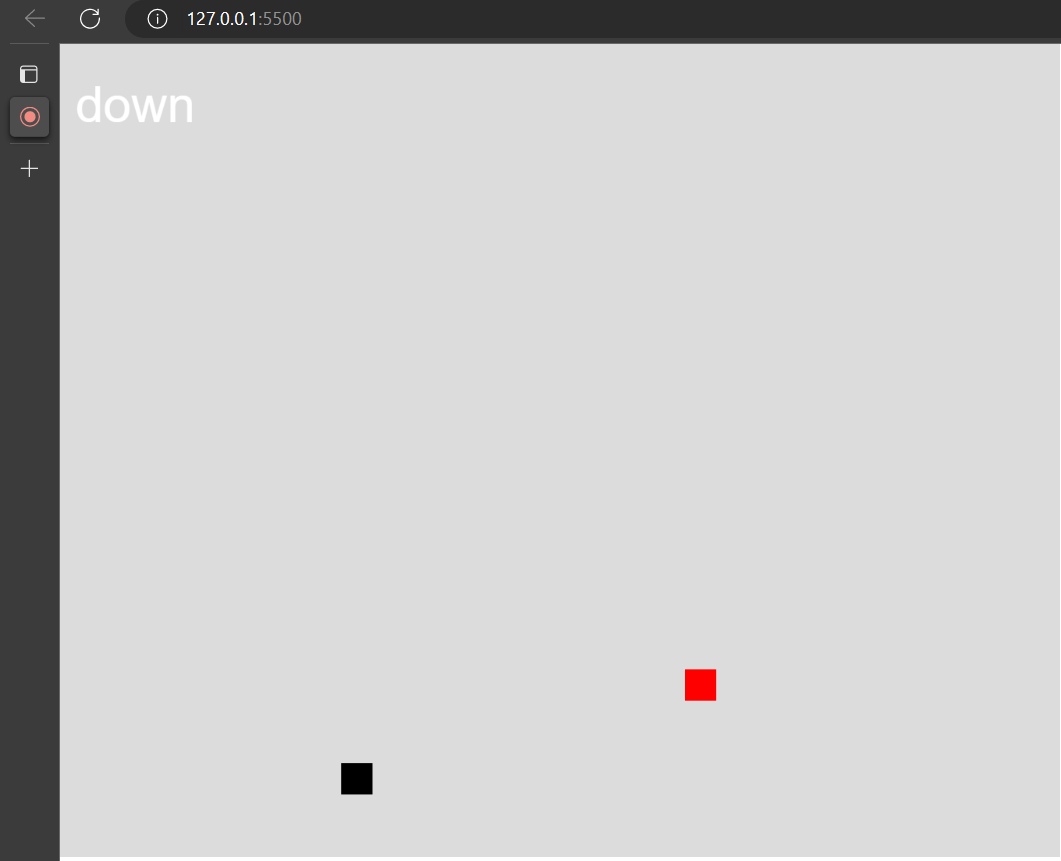
**For Sound Classification:**



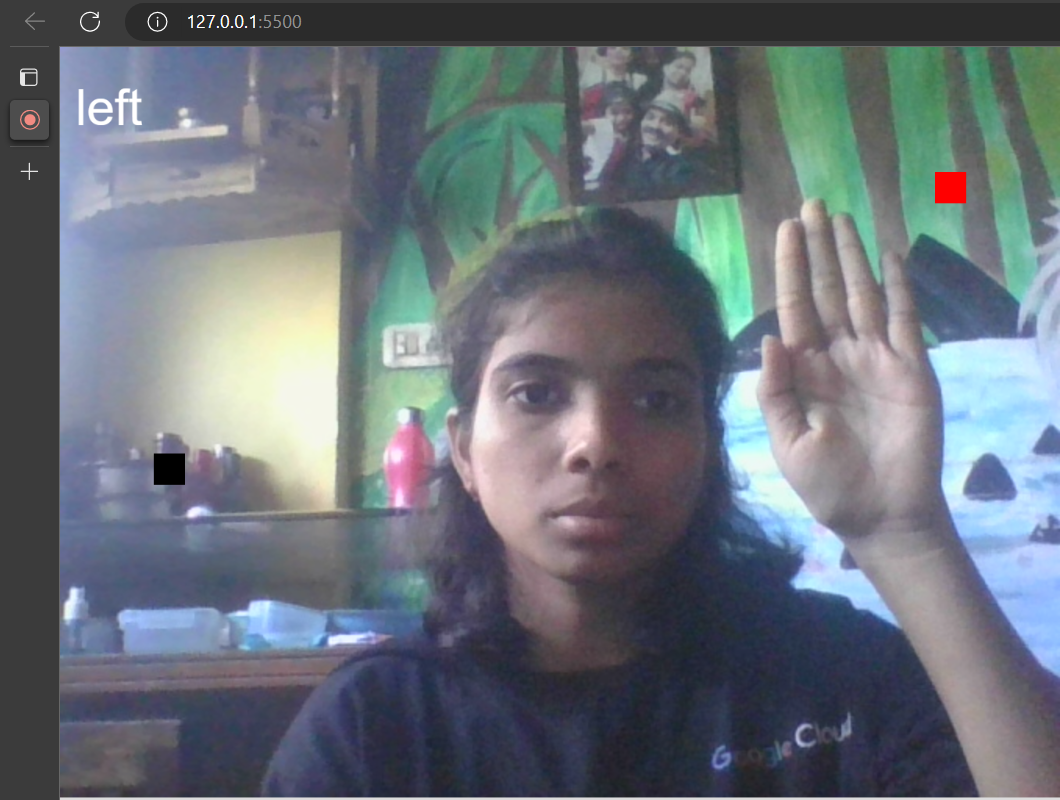


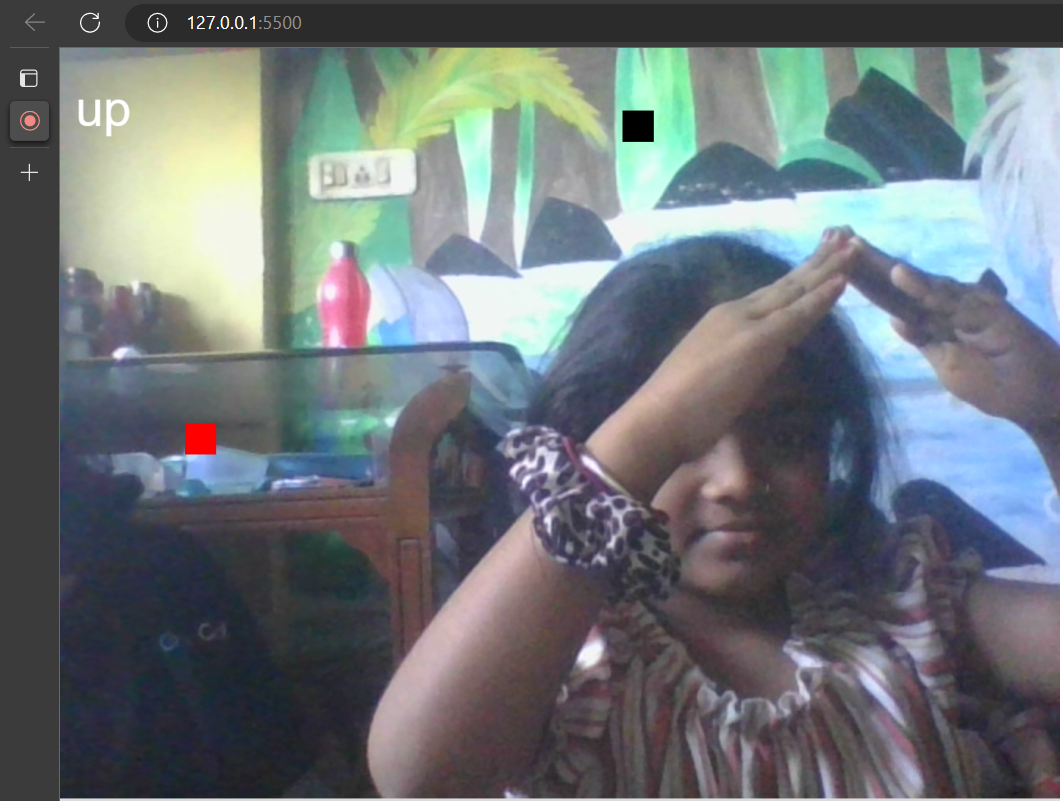


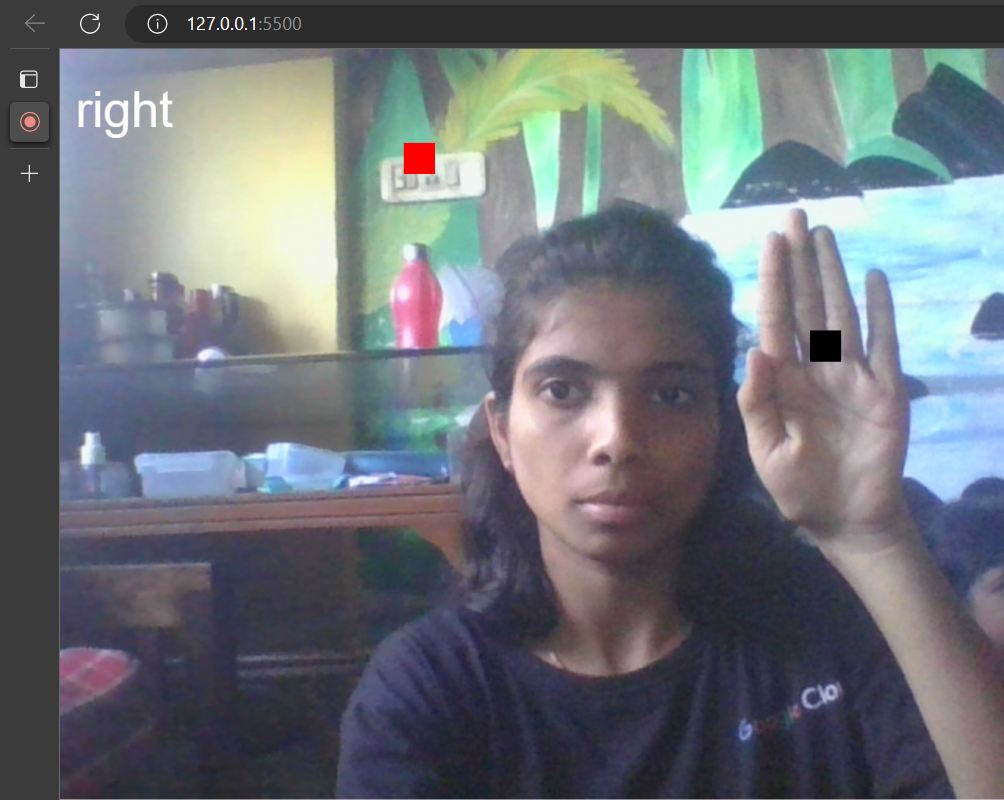




**For image Classification:**







**Conclusion**

Machine learning is a powerful tool for making predictions from data. However, it is important to remember that machine learning is only as good as the data that is used to train the algorithms. In order to make accurate predictions, it is important to use high-quality data that is representative of the real-world data that the algorithm will be used on.

Machine learning is quickly growing field in computer science. It has applications in nearly every other field of study and is already being implemented commercially because machine learning can solve problems too difficult or time consuming for humans to solve. To describe machine learning in general terms, a variety models are used to learn patterns in data and make accurate predictions based on the patterns it observes.

So far in this project, we have focused on how to prepare and use ML algorithms in Go. This included the preparation of data, Setting Up the Development Environment, and the use of data to build models. We also looked at how to integrate an existing ML model into an application, Using Pretrained Models. Deploying Machine Learning Applications. To conclude, we have take a look at the different stages in a typical project, and how to manage process of developing and deploying a successful ML system.

**Bibliography**

* <https://teachablemachine.withgoogle.com/>
* <https://cdnjs.com/libraries/p5.js>
* https://p5js.org/libraries/