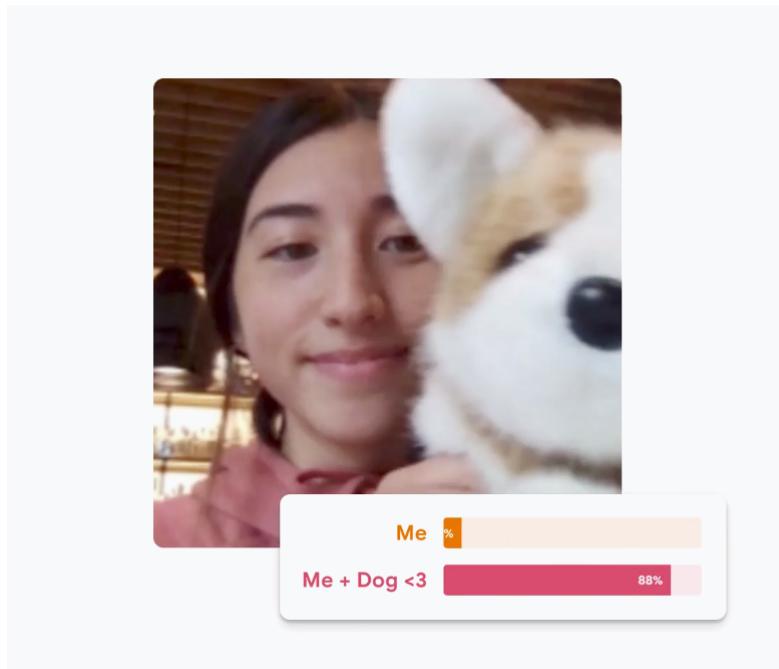


基于Teachable Machine的贪吃蛇小游戏

项目说明

背景

本项目使用了 Google 的 Teachable Machine 项目，该项目可以直接在网页上使用自己电脑配置的摄像机训练自己的机器学习模型，并且可以在网页上直接使用该模型进行预测。利用这个项目，我们可以自己训练模型并且直接上传到 Google 的服务器使用 Google 的 tensorflow.js 库直接在自己的网页中使用。



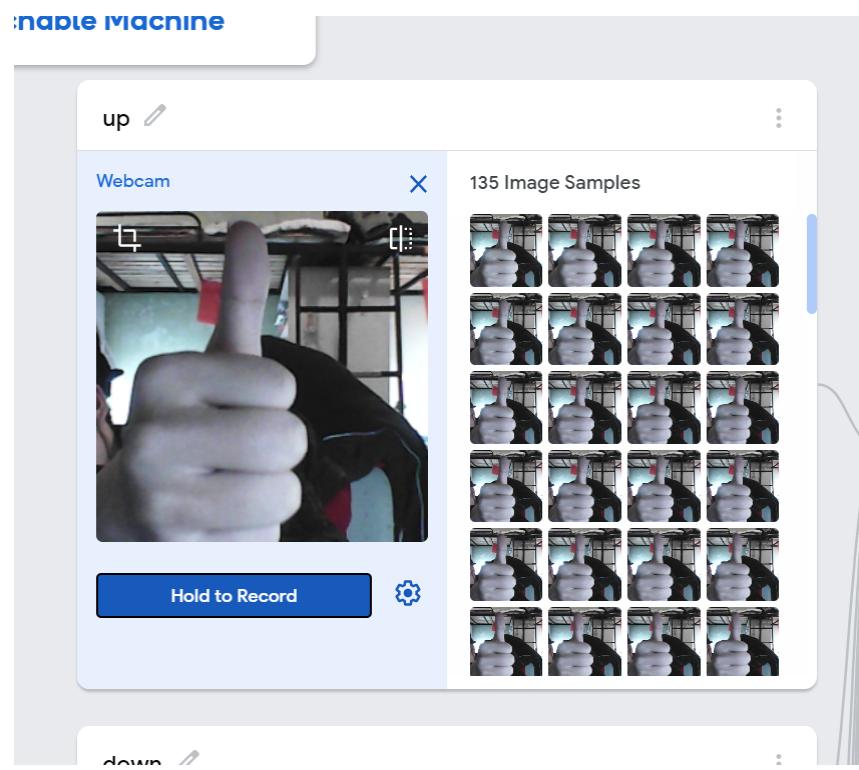
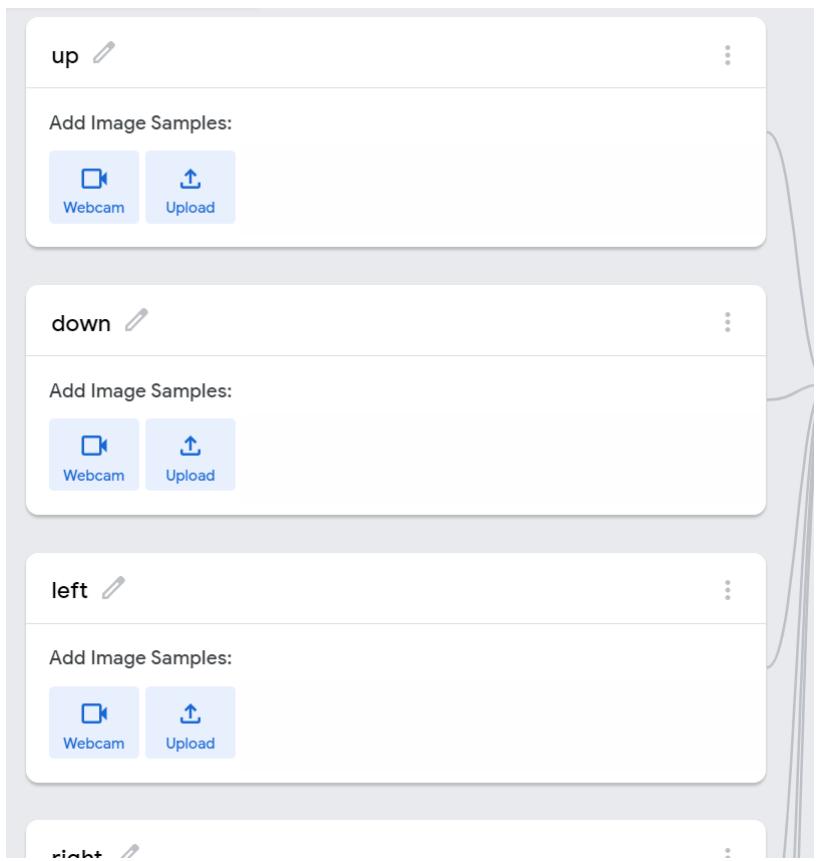
项目简介

基本思路是将键盘控制方向的贪吃蛇(监听键盘 wsad)改为模型预测的方向预测值以控制蛇的走向，其余的工作就是贪吃蛇游戏逻辑的实现部分。

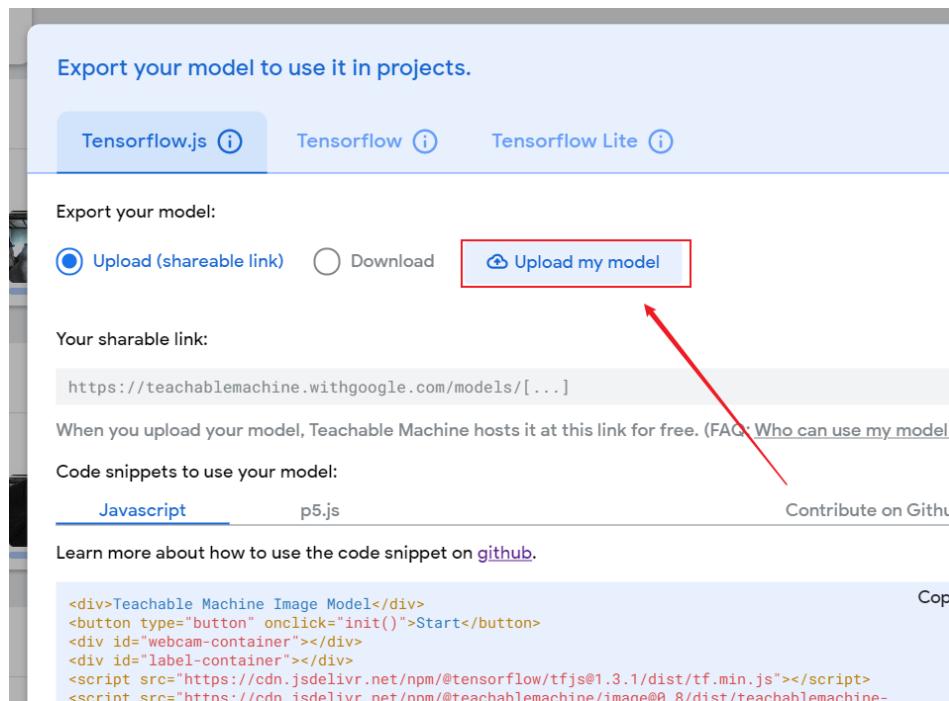
项目实现

模型训练

通过 Teachable Machine 网站进行模型的训练，我们先创建 5 个 class : up , down , left , right , no 分别代表向下，向上，向左，向右和无方向五个状态，训练的数据通过摄像头采集，我使用自己的手指手势所指方向的图片来训练。

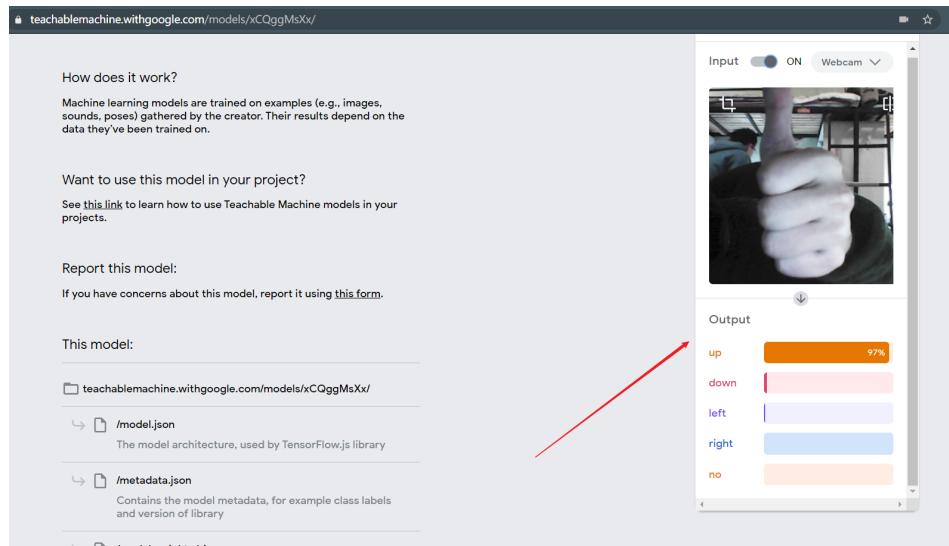


训练好的模型可以上传到 Google 的服务器，可以得到一个 URL 例如我的模型为 <https://teachablemachine.withgoogle.com/models/xCQggMsXx/>，并且可以直接使用下面的 Demo 源代码。



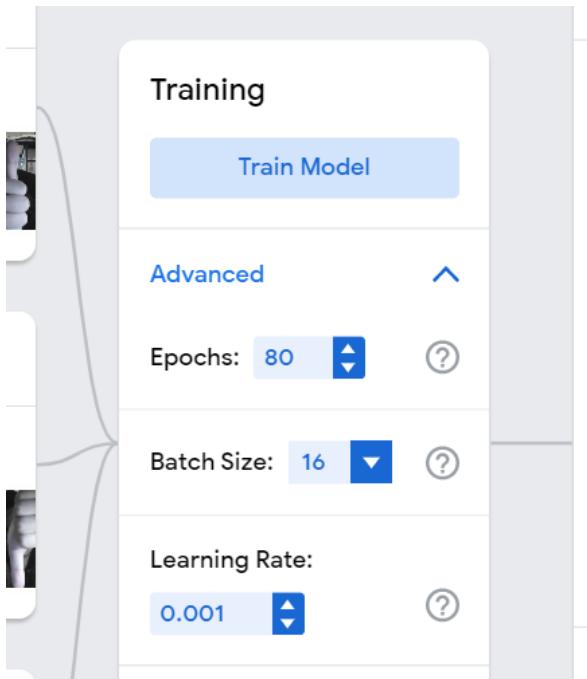
模型的评估与修改

将获取的 URL 在浏览器打开，对我们训练的模型的准确度进行评估，通过观察其预测的准确度与稳定性，并且通过对评估而对其进行修改。



修改模型主要从以下几个方面考虑：

- 每个 Class 的图片的数量
- 每个 Class 中每张图片之间的区别程度(明暗, 手势幅度)
- 训练的参数(Epochs, Batch Size, Learning Rate)

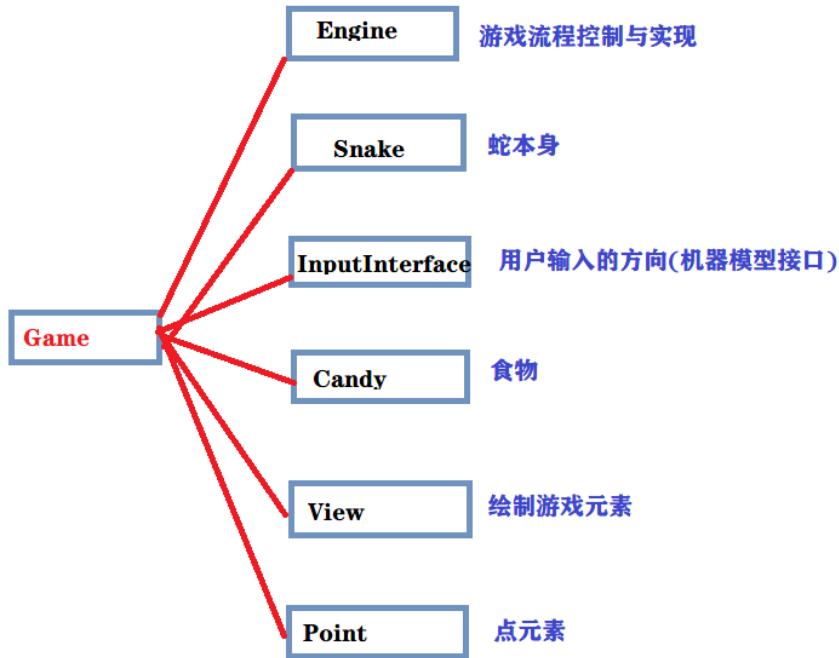


最后通过若干次修改，得出“最优”的模型，`upload`之后在我们的项目中使用。我最后得出的的“最优”模型做出了以下修改与调整：每个 `Class` 中的图片大约为 250 张，在这些图片中有环境灯光明暗，手势摆动幅度等微小差别，训练的参数为 `Epochs = 80; Batch Size = 16; Learning Rate = 0.001`。于是，就得到了一个可以通过摄像头识别手指指向的机器学习 [模型](#)。

小游戏逻辑的实现

该部分主要是 `JS` 语言游戏逻辑实现部分(见 `~/js/snake.js`)，因为本项目的重点是机器学习模型的使用，所以对于游戏逻辑实现部分不过多赘述：

游戏使用了面向对象的知识，定义了以下几个 `Object`：`Engine`，`Snake`，`Candy`，`View`，`Point`，`InputInterface` 主要的 `Object`，其中 `InputInterface` 对象是使用我的模型的接口。



模型在游戏中的使用

先使用 Teachable Machine 给出的实例 `model.js`，利用其给出的 API，得出预测出值最高的方向，然后通过一个全局变量 `dir` (储存当前模型预测的方向)，传给 `snake.js` 用于改变蛇的运动方向。

- 加载模型

```

// 模型URL
const URL = "https://teachablemachine.withgoogle.com/models/xCQggMsXx/";
let model, webcam, maxPredictions;
// 初始化模型
init();
// 识别的方向 - 全局变量
var dir;
async function init() {
    const modelURL = URL + "model.json";
    const metadataURL = URL + "metadata.json";

    // 加载模型
    model = await tmImage.load(modelURL, metadataURL);
    maxPredictions = model.getTotalClasses();

    const flip = true;
    webcam = new tmImage.Webcam(200, 200, flip); // width, height, flip
    await webcam.setup(); // request access to the webcam
    await webcam.play();
    window.requestAnimationFrame(loop);
}

```

```

var webcamele = document.getElementById("webcam-container");
// 当模型加载完成以后，加载网页摄像头，并清空等待提示
webcamele.innerHTML="";
webcamele.setAttribute("style","");
webcamele.appendChild(webcam.canvas);
}

```

- 通过模型预测方向

```

// 预测
async function predict() {
    // 获取预测
    const prediction = await model.predict(webcam.canvas);

    // 通过预测值绘制进度条
    var up = document.getElementById("up");
    var down = document.getElementById("down");
    var left = document.getElementById("left");
    var right = document.getElementById("right");
    var no = document.getElementById("no");
    up.setAttribute("style", "width: " +
    (parseFloat(prediction[0].probability.toFixed(2)) * 100).toString() +
    "%");
    down.setAttribute("style", "width: " +
    (parseFloat(prediction[1].probability.toFixed(2)) * 100).toString() +
    "%");
    left.setAttribute("style", "width: " +
    (parseFloat(prediction[2].probability.toFixed(2)) * 100).toString() +
    "%");
    right.setAttribute("style", "width: " +
    (parseFloat(prediction[3].probability.toFixed(2)) * 100).toString() +
    "%");
    no.setAttribute("style", "width: " +
    (parseFloat(prediction[4].probability.toFixed(2)) * 100).toString() +
    "%");

    // 预测方向
    if(prediction[0].probability.toFixed(2) > 0.80) dir = 0;
    else if(prediction[1].probability.toFixed(2) > 0.80) dir = 1;
    else if(prediction[2].probability.toFixed(2) > 0.80) dir = 2;
    else if(prediction[3].probability.toFixed(2) > 0.80) dir = 3;
}

```

- 在 `snake.js` 中使用预测出的方向

```

/**
 * INPUTINTERFACE OBJECT
 *
 * 方向对象

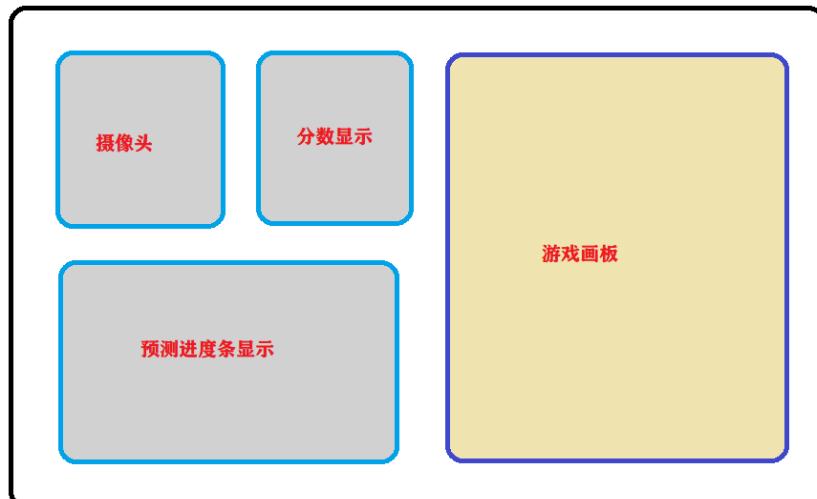
```

```
/*
function InputInterface() {
    var lastDirection = null;
    // 监听方向变化
    this.lastDirection = function () {
        switch (dir) {
            // 左
            case 2:
                lastDirection = -2;
                break;
            // 上
            case 0:
                lastDirection = 1;
                break;
            // 右
            case 3:
                lastDirection = 2;
                break;
            // 下
            case 1:
                lastDirection = -1;
                break;
        }
        return lastDirection;
    };
}
```

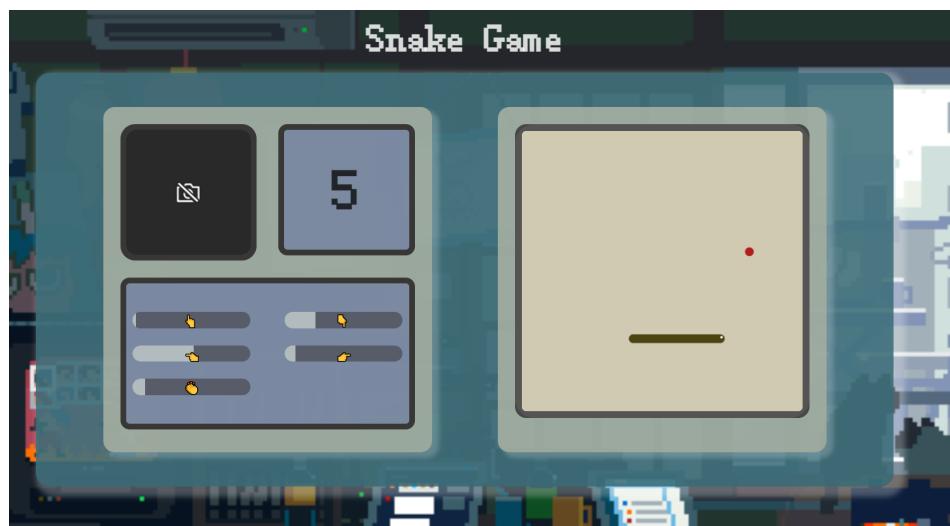
项目前端的完善

使用 [Bootstrap 4](#) 前端框架完善小游戏页面，页面中显示电脑摄像头捕捉画面，游戏画板，分数显示，方向预测显示(进度条表示各方向的可能性大小)

- 页面设计



- UI 实现



项目运行

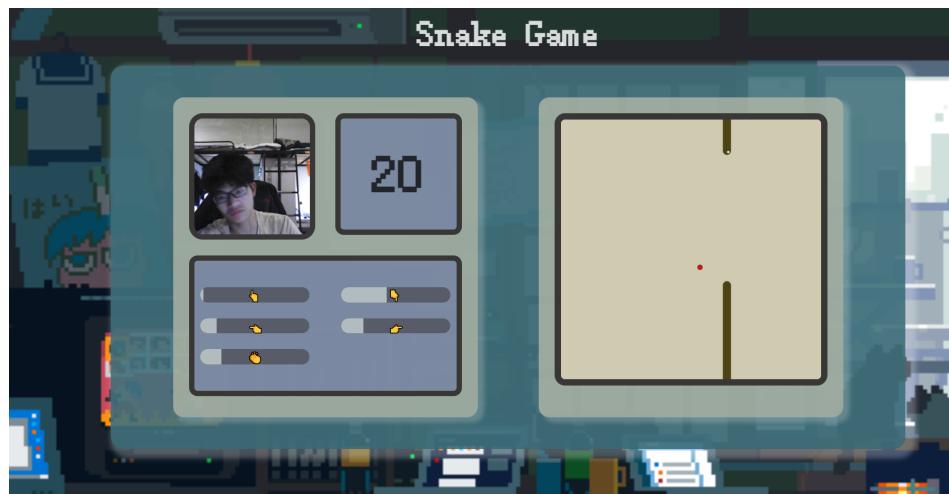
运行说明

浏览器打开项目中的 `inde.html` (我的浏览器及版本号为 Chrome 89.0.4389.82)

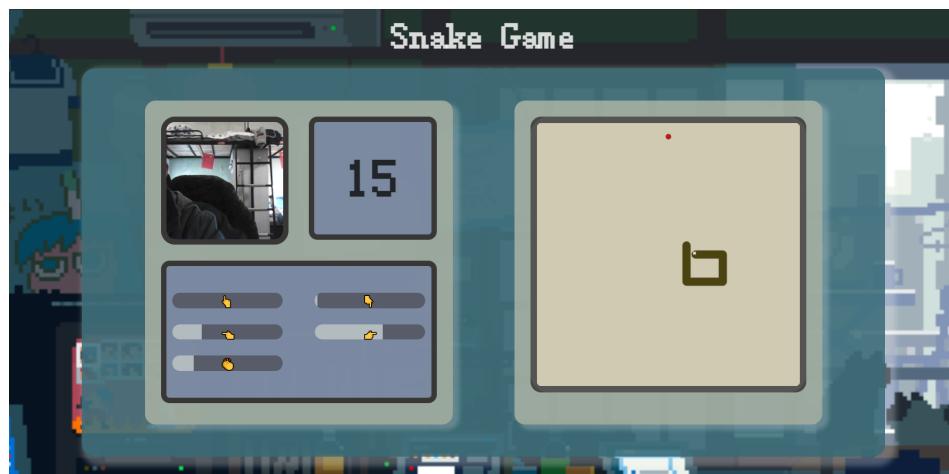
打开后，等待模型加载。



模型加载完成后，显示电脑摄像头捕捉画面，可用手势操控贪吃蛇运动方向。



如果游戏结束，则分数清零，自动开始下一局游戏。



运行结果截图

上下左右四个方向的截图：

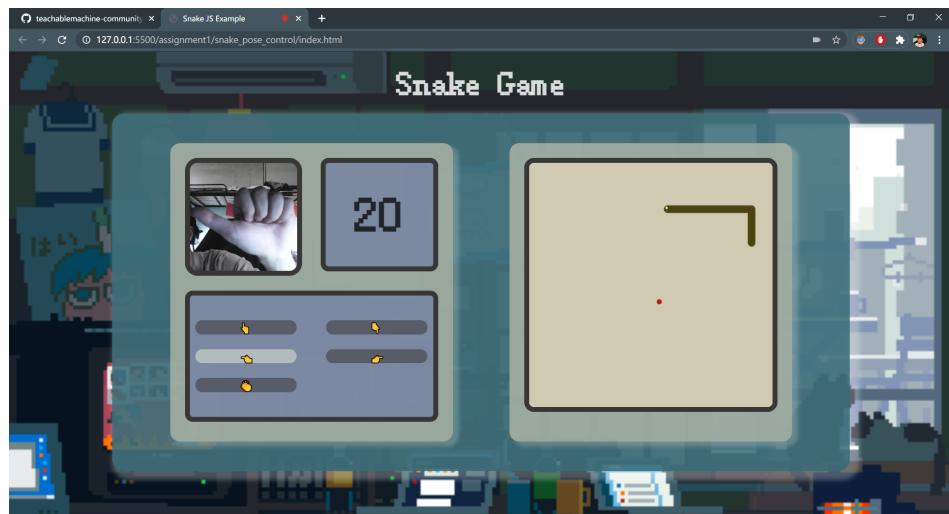
- 上↑



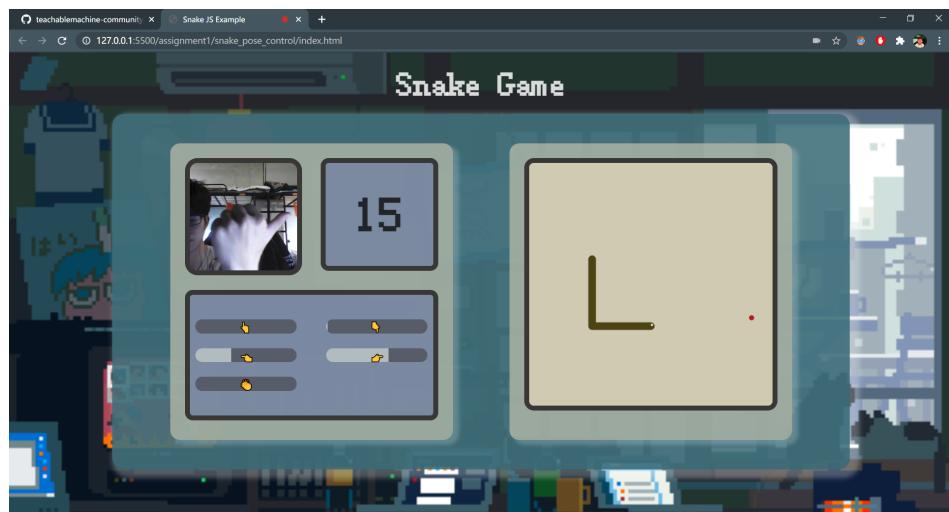
- 下↓



- 左



- 右



总结与思考

项目的不足

该小游戏在实际进行的过程中，有几个不足之处。

第一，模型的稳定性。虽然模型经过对比与修改，但还是不怎么稳定，而且识别的范围有限，只能识别我自己的手势，他人的手势虽然也可以识别但是准确度不如自己的高。

第二，实际在游戏中的运用。手势识别方向会有延迟，因此我调慢了蛇的速度以适应游戏模式，使用手势进行游戏只是为了体验与学习机器学习和增强游戏的趣味性，在实际使用中并没有带来便利。