实验报告

机器学习之基于深度迁移学习的人脸识别器

邹天舒

指导老师：魏祥

目录

1. 摘要

深度学习作为一种新的分类模型，近年来越来越受到研究者的重视，并已成功地应用于许多领域。在生物信息学和机器人技术等领域，由于数据采集和标注成本高，构建大规模的、注释良好的数据集非常困难，这限制了数据集的发展。迁移学习不要求训练数据必须与测试数据独立且同分布，激发了我们使用迁移学习来解决训练数据不足的问题。

2. 需求分析

基本需求：

1. 能够自定义输入类别个数。
2. 能够自定义输入类别名称，如张三、李四、王五，以及实时显示每类加载图片数。
3. 一个具有保存功能的 html 和一个能够加载模型的 html（保存和加载功能）。
4. 打开能够加载模型的 html 后可直接进行人脸识别。

根据需求，首先需要一个训练样本的页面进行训练然后保存模型；还需要一个加载模型进行识别的页面

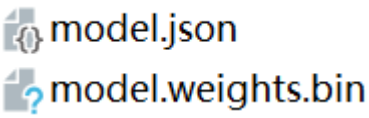
3. 代码

1) Save()

根据需求可知，我们需要在训练结束之后获得训练好的模型，那么此时我们需要调用save()方法来进行操作。代码示例如下：

```
saveButton = createButton('保存模型');
saveButton.mousePressed(function () {
   classifier.save();
});
```

我们设置一个保存按钮，当按钮按下的时候，设置事件将训练好的模型进行保存。当点击保存按钮之后，就会自动下载两个文件，如下图：



利用这两个文件，我们就可以通过训练好的一些模型直接进行特定物品的识别。

2) Load()

保存了训练模型之后，我们开始进行识别，但是在识别之前我们需要将训练的模型加载

出来，否则我们就不能进行识别。想要加载出模型，我们就需要 load()方法。关于 load()方法的使用有两种。

第一种:

```
const modelInfo = {
  model: 'path/to/model.json',
  metadata: 'path/to/model_meta.json',
  weights: 'path/to/model.weights.bin',
};
neuralNetwork.load(modelInfo, modelLoadedCallback);
```

第二种:

```
neuralNetwork.load('path/to/model.json', modelLoadedCallback);
```

注意，我们这里的路径需要写相对路径。执行完这个方法之后，我们得到的就是训练好的模型，就可以开始我们的识别工作了。

3) 自定义类训练

在本次实验中,还有一项需求就是可以自定义类名进行输入并且训练。首先我的思路是,当输入了需要训练的类别后，除了显示出相应个数的添加训练样本按钮之外，同时也显示出相应个数的输入框个数，然后获取每一个输入框的值，将其赋值给每一个类的名字，并在 video 上面显示出来。

如下输入函数的代码:

```
function myInputEvent(){
   classifier.numClasses = inp.value();
   for (let i = 1; i <= inp.value(); i++){
      num[i]=0;
      inps[i]=createInput();
      na = createButton('添加训练样本'+i);
      na.mousePressed(function(){
         classifier.addImage(inps[i].value());
         num[i]++;
         //console.log(num[i]);
         index=i;
      })
   }
}
```

代码中，循环添加了输入框的个数，然后通过 value()将获取到的输入值赋值给每一类。添加样本完毕之后，开始训练，当训练结束之后，获取结果:

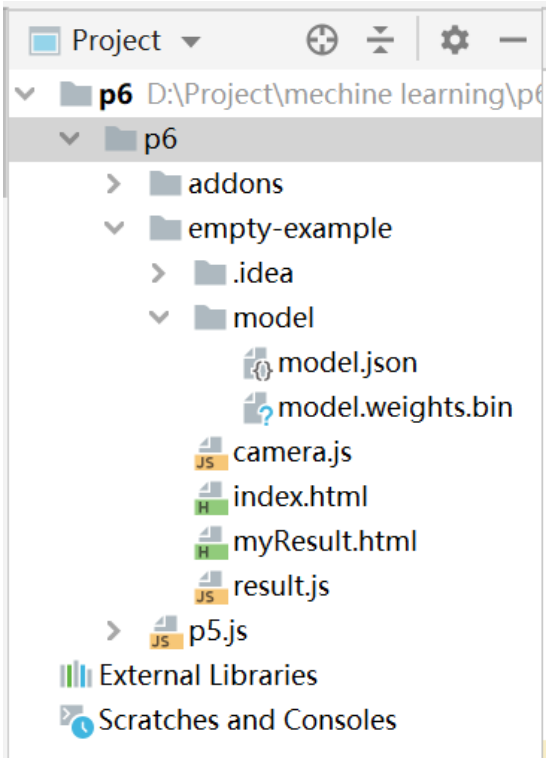
```
function GetResult(error, results) {
   if (error) {
      console.error(error);
   } else {
      console.log(results);
```

```
        name = results;
        //prob = results[0].probability;
        fill([255,0,0]);
        textSize(30);
        text(name, 10, height - 20);
          //createP(name);
          //createP(prob);
    }
}
```

当我们打印结果之后发现，输出的值就是类别的名字，那么此时就可以将其显示到 video 上了。

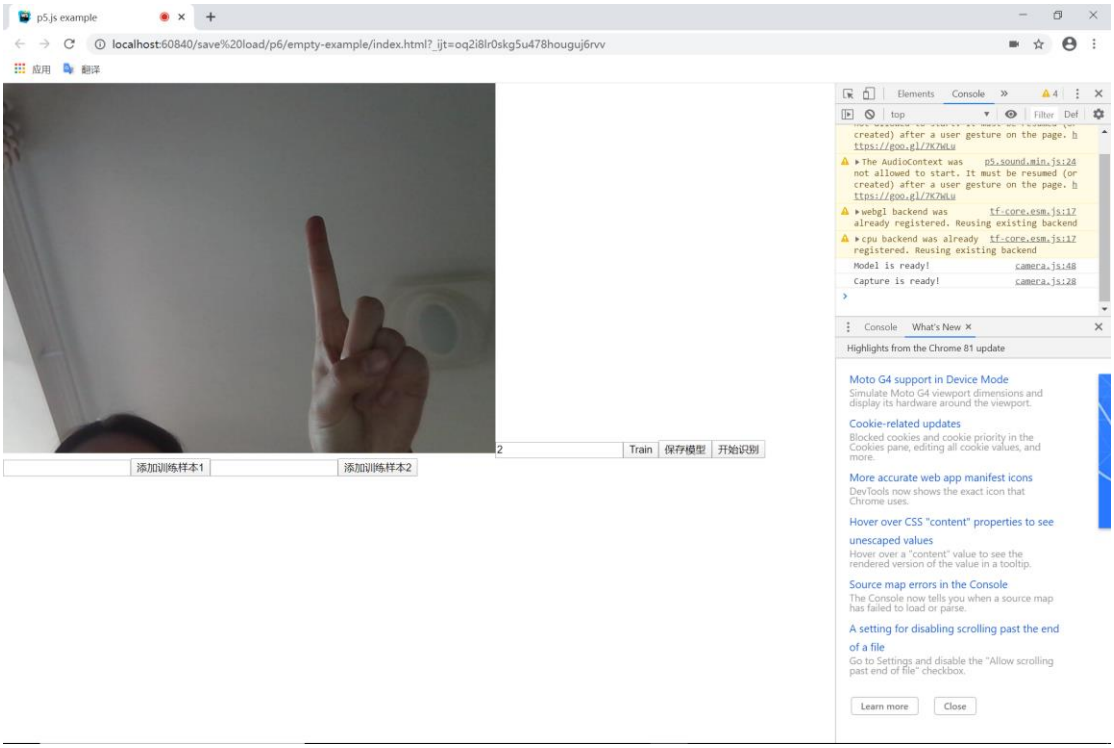
4) 目录结构

本次实验的目录结构如下图：



model 文件夹中存放了保存的训练模型，index.html 和 camera.js 是实现自定义训练和保存训练模型的功能，myResult.html 和 result.js 实现的是加载模型进行识别的功能。

4. 运行截图



主要运行见文件中的录屏。

5. 实验总结

通过这次实验，我主要学习了 save()和 load()这两个函数，进行保存训练好的模型和加载训练好的模型，还了解了基于深度迁移学习的一些相关知识。