《高级软件工程》课程项目报告
魔方复原系统

目录

1 项目简介

通过阅读《Solving the Rubik's Cube with Deep Reinforcement Learning and Search》和参考其展示网页 http://deepcube.igb.uci.edu，及研究其提供的源代码，做了一款基于 DeepCubeA 的魔方复原系统。

在复现原有魔方复原功能上，做出了一点改变和创新——人机对比魔方复原步骤系统，旨在对比人操纵魔方还原的步骤和机器步骤，来提高人玩魔方的技术。

本系统因基于 DeepCubeA，所以使用 Python Web 开发，这里选择了 Flask+uWSGI+Nginx 作为服务端，前端使用 HTML+CSS+JavaScript 开发。

2 需求分析

基本功能：魔方打乱和自动复原。

人机交互功能：

1 通过键盘可以输入步骤，人为打乱魔方或复原魔方；

2 通过鼠标可以拖动魔方，360 度观看魔方。

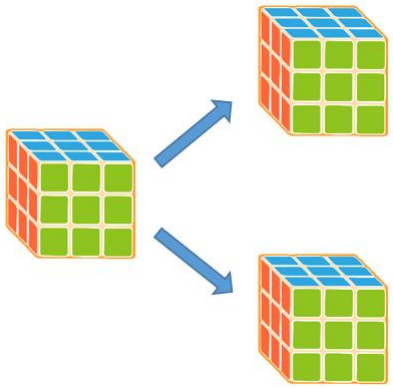
3 复原步骤控制，魔方打乱后可以逐步展示复原过程，并可停留到某一步上，细致观看魔方每一面。

以上是魔方复原系统的基础功能，肯定要实现的，如果缺少了其中一项，那么这个复原系统就不是很完整。

接下来让我设想一个场景：据统计顶级魔方玩家需要大约 50 次移动，才能在 4 秒内破解魔方，但 DeepCubeA 可以在 20 步内完成。如果有人想要训练进阶模仿技巧，目标是所用步数逼近 DeepCubeA 计算出的步数。这些人希望通过机器计算出的步骤，来启发和简化自己的复原步骤，接近或达到 20 步。

那么我们如何通过这个有了之前所说的基础功能的魔方系统来实现这个场景：

首先页面需要两个魔方——一个机器魔方一个玩家魔法，一个用来展示机器复原的步骤，另一个用来让玩家操纵。其次，两个魔方不能互相干扰，只有在打乱步骤时同时打乱，为了方便玩家练习。最后，需要一个计步器来纪录玩家的步骤，方便玩家对比。



3 原型设计

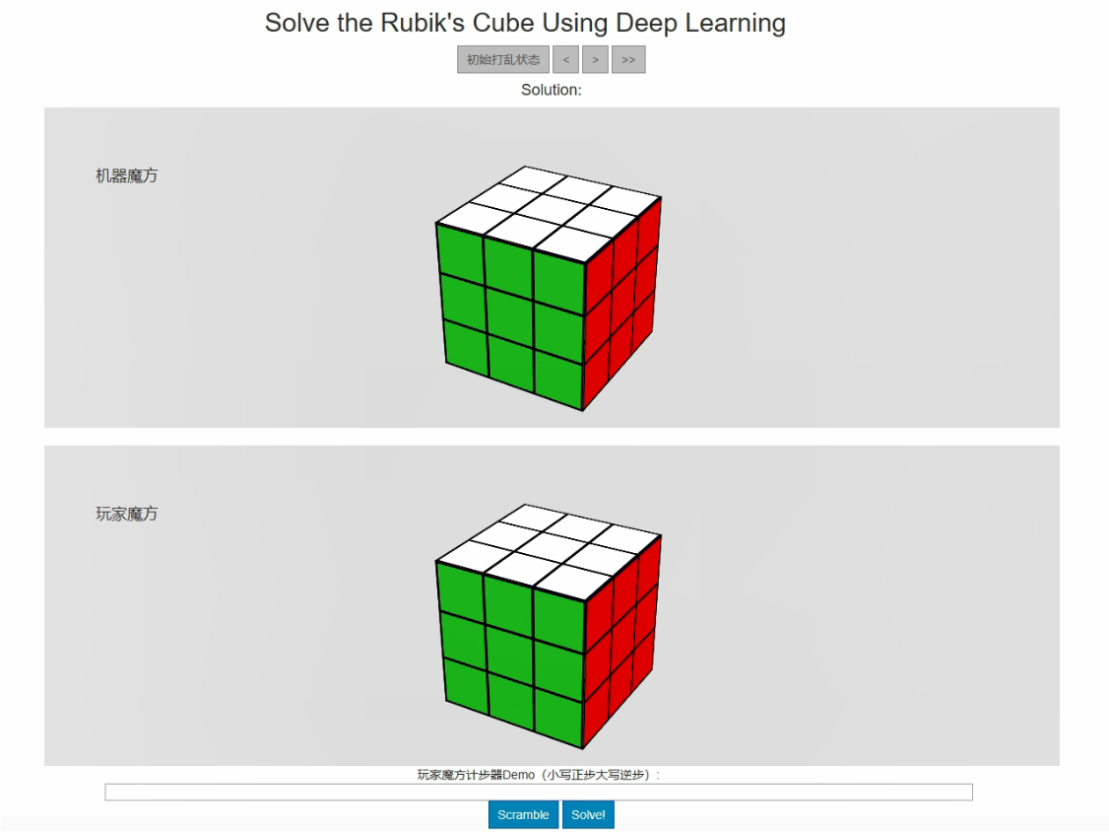
为了实现基础功能，一定要有魔方模块：这个魔方模块可以支持人机交互，用鼠标来旋转魔方观察，键盘输入手动打乱魔方；有打乱和复原按钮控制机器自动打乱和自动复原；有显示机器步骤的计步器，并且可以控制每一步，暂停快进回退等。

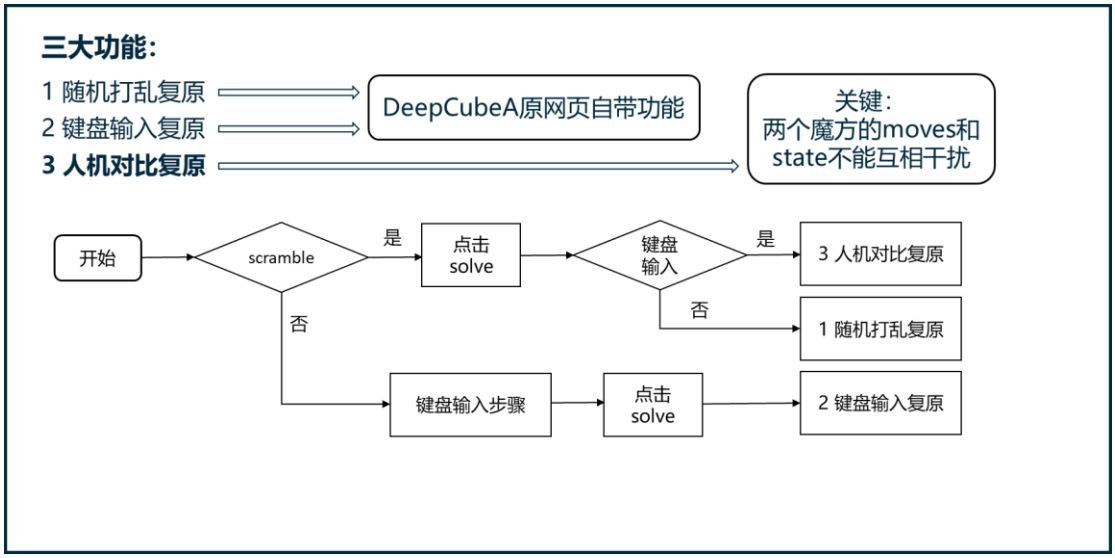
为了实现进阶的人机对比复原功能，则必须需要两个互不干扰的

魔方和玩家计步器。

总的来说一共三个功能：①随机打乱自动复原；②手动打乱自动复原；③人机步骤对比复原。

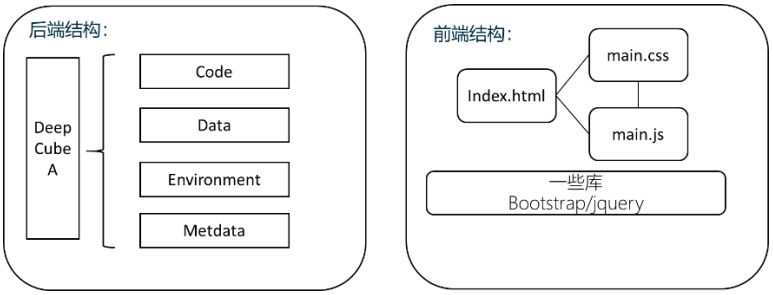
下图为主界面和功能逻辑



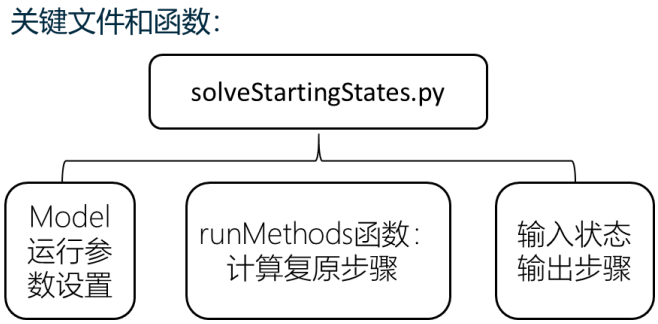


4 系统开发

首先要进行源码的分析，这边主要是两部分的源码，后端源码 DeepCubeA，和前端从展示网页上下载下来的源码。以下是两个代码的代码结构：



后端主要是神经网络搭建，模型训练，参数设置等。给予训练好的模型相应的初始状态，通过关键文件 solveStartingStates.py 中的一些方法即可输出魔方复原要用的步骤。



前端 HTML 和 CSS 用来搭建网页，显示魔方图形。JS 用来构建函数：打乱魔方，并将状态传给后端。复原魔方，通过接收后端计算好的复原步骤。还需用浏览器分析原网页的 POST：

通过这个我们可以分析出三点：

①前端传给后端的是一个代表魔方 6 个面 54 个 sticker 面的 state[54]数组；

②后端传给前端的是三个数组 moves，moves_rev，solve_text，分别代表根据①中的 state，计算出的复原步骤，反向复原步骤（回退用的），和步骤的文本；

③InitState.json 表示初始化时候的需要用到的数据。

通过以上的，后端关键源码，前端页面结构，和分析出的 POST 数据，我们可以轻易得出要构建的接口。

首先我们 python web 框架选择的是 Flask。Flask 是一个轻量级的框架，易于扩展和管理，从最简单的 Hello World 程序到复杂的大型应用都可以胜任，而更出名的 Django 框架太过冗余，虽然它有很多预设好的场景，如博客网站，论坛网站等，甚至 app 与数据库的接口都能自动设置，但是我们这个系统，没有数据库，没有复杂的应用逻辑，所以用 Flask 轻量级框架更省时省力。

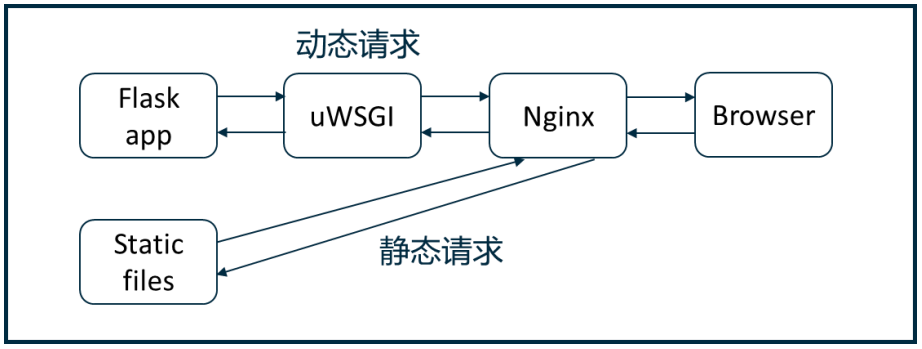
接下来，设计接口根据上面的分析，其实只用三个函数即可：

①Index():连接主页 html；

②Init():通过 InitState.json 初始化魔方；

③Solve():用 POST 方法接收到网页传来的 state[54]，调用 solveStartingStates.py 中的参数，函数，输入输出，计算好 moves、moves_rev、solve_text 数组，传到前端。

设计好接口后做本地测试，运行程序后访问设定的本机地址端口，没有问题后进行服务端部署。通过查阅网上资料，python web 现在最流行的就是 web 框架（Flask 或 Django）加 uWSGI 加 Nginx 的组合，所以这次选择了 Flask+uWSGI+Nginx。



uWSGI 是使用 C 语言编写的，显示了自有的 uWSGI 协议的 Web 服务器。它自带丰富的组件，其中核心组件包含进程管理、监控、IPC 等功能，实现应用服务器接口的请求插件支持多种语言和平台，比如 WSGI、Rack、Lua WSAPI，网管组件实现了负载均衡、代理和理由功能。只要 web 服务器和 web 框架满足 WSGI 协议，它们就能相互搭配。

Nginx (engine x) 是一个高性能的 HTTP 和反向代理。web 服务器访问量不大的话，当然可以由 uWSGI 和 Django 构成。但是一旦访问量过大，客户端请求连接就要进行长时间的等待。Nginx 可以分配

客户端的请求连接和 web 服务器这也就是由 Nginx 实现反向代理，即代理服务器。

通过配置好服务器，将代码部署到服务器上，再将域名解析到服务器的 IP，就可以用外网通过浏览器输入域名，访问我们开发的魔方复原网页了。

5 功能实现

项目主要包含三个主要功能：随机打乱复原、键盘输入复原和人机对比复原。随机打乱功能和键盘输入复原功能主要实现原 DeepCubeA 网页功能，可提供基本的魔方操作与复原。人机对比复原功能为本项目的主要创新点，用户可自行按照求解步骤操作魔方转动，达到更好的学习效果，并可以脱离实体魔方。

5.1 面临的问题

魔方的转动、求解等变化都与魔方此时的状态有关，而表达状态的参数多且复杂，函数间参数的正确传递是功能实现的关键所在。人机对比功能需要两个魔方在不同的操作下表达多样的状态，例如打乱与求解操作时，两个魔方必须同步进行，而键盘输入与查看步骤时又需要两个魔方相互保持独立状态。因此，如何保证复杂多变的状态参数既能够相互传递又能够互不干扰，是完成人机对比复原功能的最大问题。

5.2 解决方案

构造了两套参数以达到正确表达两个魔方的状态，并确定一套参

数为基准主参数，以确保在二者参数不匹配时不会出现状态混乱。

两个魔方模块：
将html、css、js文件中，关于控制魔方显示的代码，复制两份并修改关键变量名、函数名。

HTML

```
<div id="cube_div" class="container" s
    <h4 id="cube_name">机器魔方</h4>
    <section class="cube_container">
        <div id="cube"...>
    </section>
</div>

<div id="mycube_div" class="container"
    <h4 id="mycube_name">玩家魔方</h4>
    <section class="cube_container">
        <div id="mycube"...>
    </section>
</div>
```

JS

```
var moves = []
var mymoves = []
var solveIdx = null;
var mysolveIdx = null;
var faceNames = ["top", "botto
var myfaceNames = ["mytop", "m

function clearCube() {...}
function clearmyCube() {...}
function setStickerColors(ne
function setmyStickerColors(
```

CSS

```
#cube_name{...}
#mycube_name{...}
#cube_div {...}
#mycube_div {...}
#cube figure {...}
#mycube figure {...}
.sticker {...}
.mysticker {...}
```

在用户点击 solve 按钮前，该阶段主要为用户操作设定阶段，主要保证记录用户的输入得到正确的展示。此时以玩家魔方的参数为主状态参数，在状态输入、求解等函数中均以此参数为基准，机器魔方在求解与打乱的操作中接受玩家魔方的状态赋值。

在用户点击 solve 后，此阶段主要为用户的学习求解阶段，主要保证魔方得到正确的复原。此时以机器魔方的参数为主状态参数，在回到初始状态操作时以此参数为基准并向玩家魔方赋值，以达到用户转动时保证主参数的独立。

5.3 具体实现

针对魔方转动等状态改变中的两魔方参数独立性问题，可为操作设定选择参数，以此来控制不同情况下的魔方选取。其状态改变的主要函数为 nextState()，对其设定如下：

```
function nextState(moveTimeout=0, flag1=true, flag2=true)
```

flag1 和 flag2 分别表示机器魔方和玩家魔方，用于控制状态参数的独立性，其设定如下：

```
if ((moves.length > 0 && flag1)||(mymoves.length > 0 && flag2)) {
    disableInput();
    disableScroll();
    if (flag1){
        move1 = moves.shift() // get Move
        console.log("move1:",move1);
        state_rep1 = reOrderArray(state,FEToState)//convert to python representation
        newState_rep1 = JSON.parse(JSON.stringify(state_rep1))
        //swap stickers
        for (var i = 0; i < rotateIdxs_new[move1].length; i++) {
            newState_rep1[rotateIdxs_new[move1][i]] = state_rep1[rotateIdxs_old[move1][i]]
        }
    }
```

通过选择参数，可在各操作函数中根据情况选择不同魔方的操控，例如，键盘状态输入函数是调用 nextState()的参数为 flag1=false，flag2=true。

6 总结和展望

6.1 总结

通过这次课程项目的开发，小组协作完成课程项目：从需求分析，原型设计，系统开发部署，功能实现中得到了宝贵的经验。实现基础魔方复原算法，研究和重现 DeepCubeA 代码，学到了相关深度学习知识和经验。魔方系统接口设计和服务器部署，云服务器+Flask+uWSGI+Nginx，学到了服务端相关知识和经验。实现人机对比魔方复原功能，研究了相关 HTML+CSS+JS 代码，学到了前端设计实现相关知识。

6.2 展望

玩家计步器模块只完成了 Demo，仅用文本框记录玩家步骤——未来希望可以完成如机器魔方计步器的功能，不仅可以记录，还可以执行回到上一步等操作。

希望未来可以 Switch Mode：既三个功能可以通过切换模式，一键改变，不需要之前繁杂的步骤。如前两个基础功能只需要原版一个魔方模块即可实现，切换功能后再显示第二个魔方。

7 致谢