教育部高等学校大学计算机课程教学指导委员会

中国大学生计算机设计大赛

软件开发类作品文档简要要求

作品编号：61826

作品名称：alphaHex

作　　者：何远杰，马跃，沙昊

版本编号：0.2

填写日期：2019 年 5 月 3 日

填写说明：

1、本文档适用于**所有**涉及软件开发的作品，包括：软件应用与开发、大数据、人工智能、物联网应用；

2、正文一律用五号宋体，一级标题为二号黑体，其他级别标题如有需要，可根据需要设置；

3、本文档为简要文档，不宜长篇大论简明扼要为上；

4、提交文档时，以 PDF 格式提交本文档；

5、本文档内容是正式参赛内容组成部分，务必真实填写。如不属实，将导致奖项等级降低甚至终止本作品参加比赛。

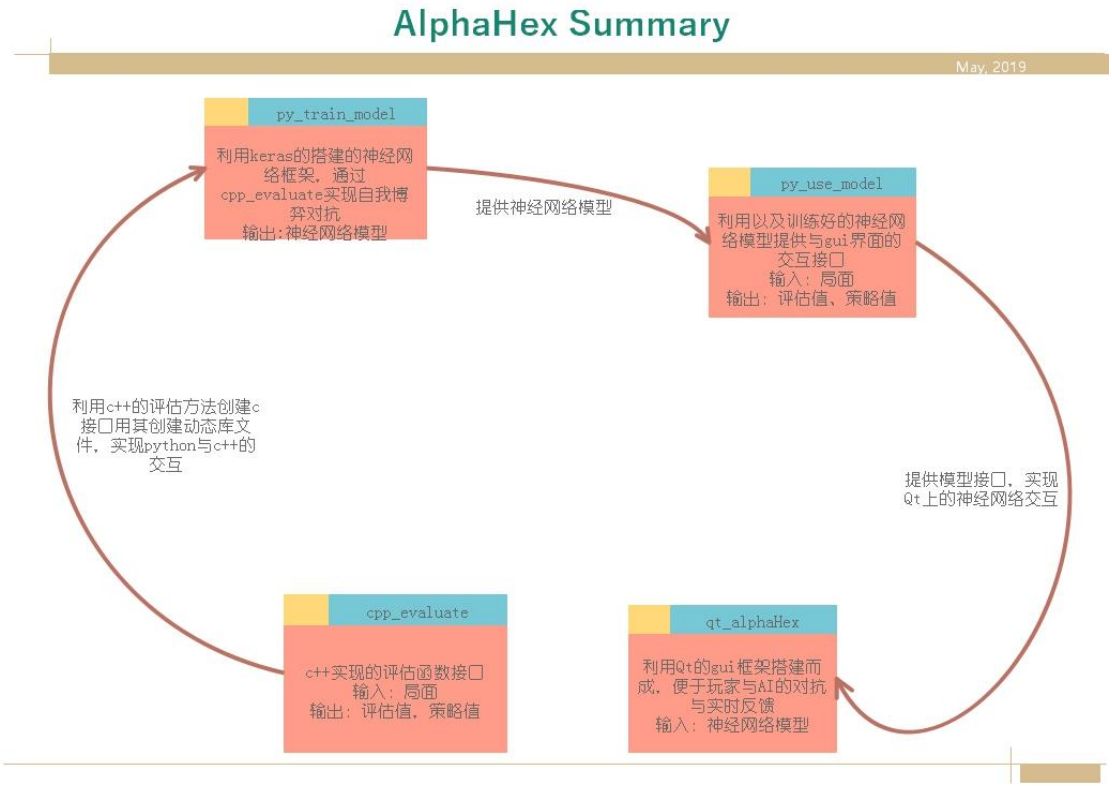
目　录

第一章 需求分析

我们设计这款会下海克斯棋的 AI 的初衷是想制作初步的一个属于自己的人工智能，一方面源于对人工智能技术的好奇，另一部分是被 AlphaGo 系列掀起的计算机博弈浪潮所吸引。另一部分发现如今普遍存在的 AlphaGo Zero 框架使用的是 UCT（上限置信区间算法）作为对局面的评估策略，而对于较为简单易用的 Hex 棋。我们考虑到目前已经有较为成熟的评估策略各方面都有不错的效果。因此我们便决定用海克斯这款相对比比较简单的棋做起，综合各方面分析得到的需求如下：

- 博弈树评估具有较快的速度与准确性
- 强化学习模型简洁易使用
- 具有用户交互性的 GUI
- 可快捷地迁移到其他棋类游戏
- 可实现训练的模型易对比分析

第二章 概要设计



- py_train_model： 利用 keras 框架实现的自我博弈深度学习框架
- cpp_evaluate： c++实现的 hex 棋的评估函数，向 py_train_model 提供接口
- py_use_model 使用模型提供下子策略，向 qt_alphaHex 提供接口
- qt_alphaHex 利用 qt 实现的 gui 博弈窗口

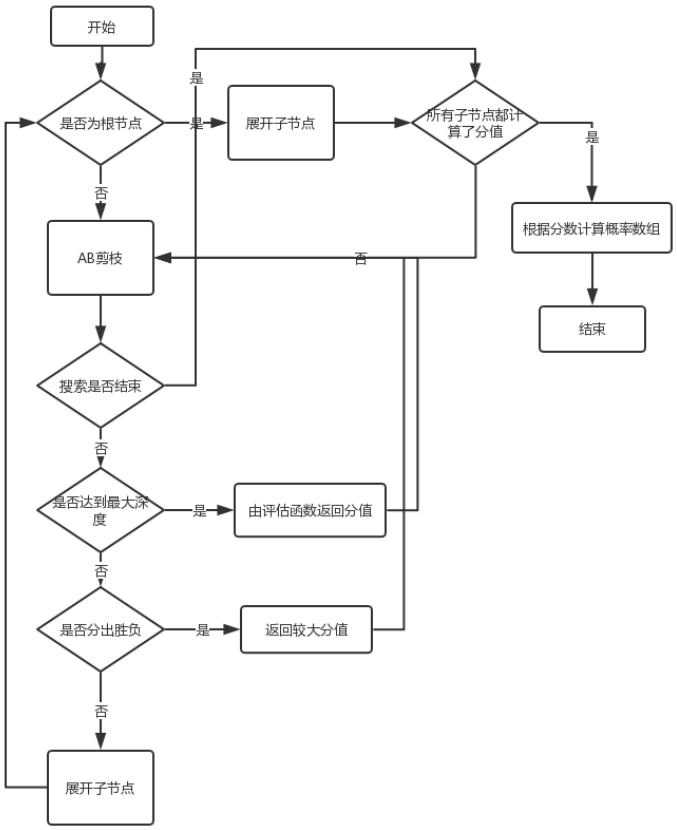
如图所示，我们利用 py_train_model 调用由 cpp_evaluate 生成的动态库文件接口，实现

python 向 c++的交互，这样在利用 c++的高效性来保证其运算速度的同时，也充分利用到了 python 语言对深度学习框架的高度兼容性。通过 py_train_model 进行强化学习迭代，实现模型的更新与强化。此时便产生了我们需要的模型，便可以通过 py_use_model 轻松的将其嵌入到 qt_alphaHex 的 c++程序之中了。

第三章 详细设计

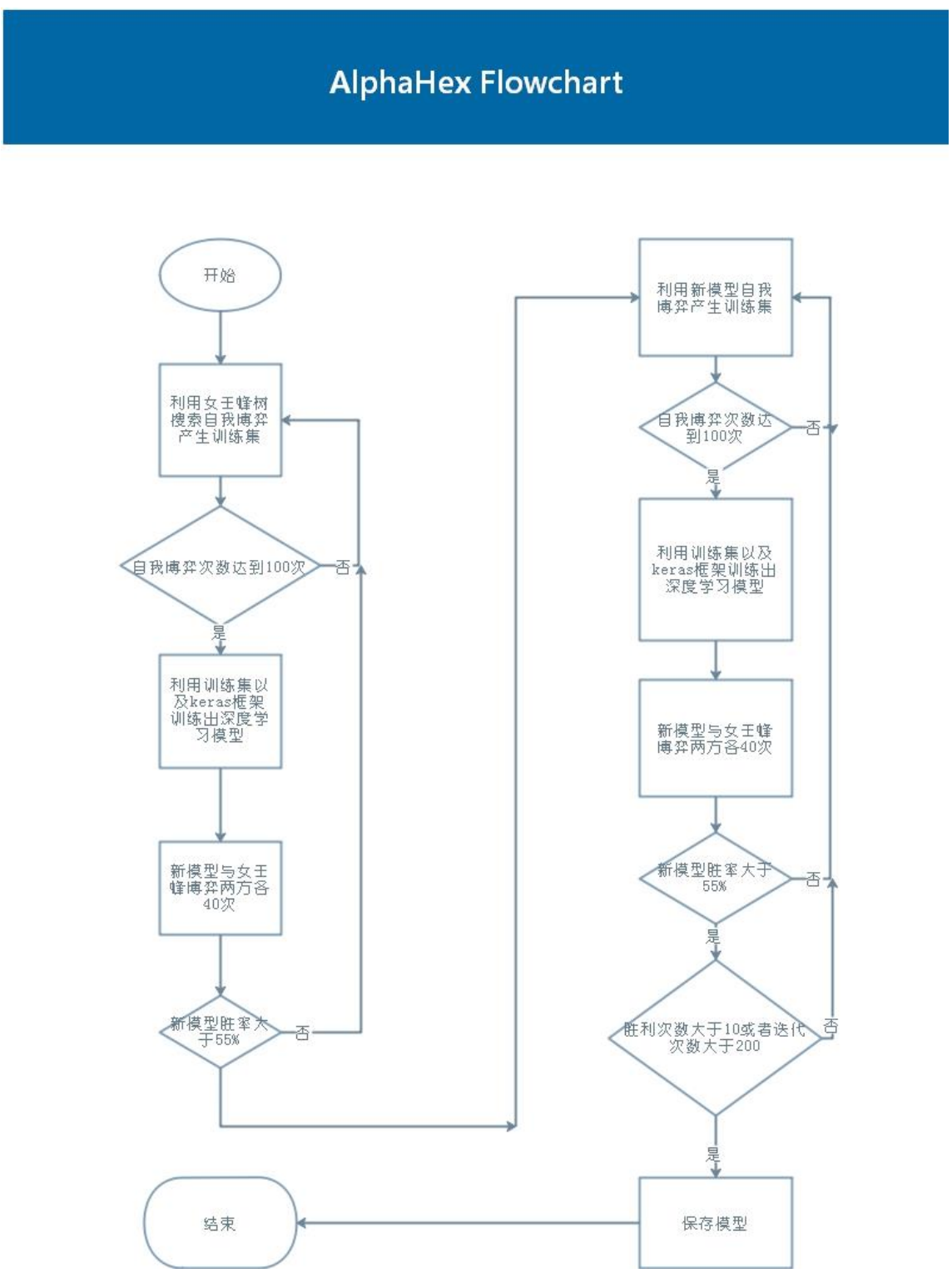
该工程可以主要分为三大部分：

一，博弈树基本流程



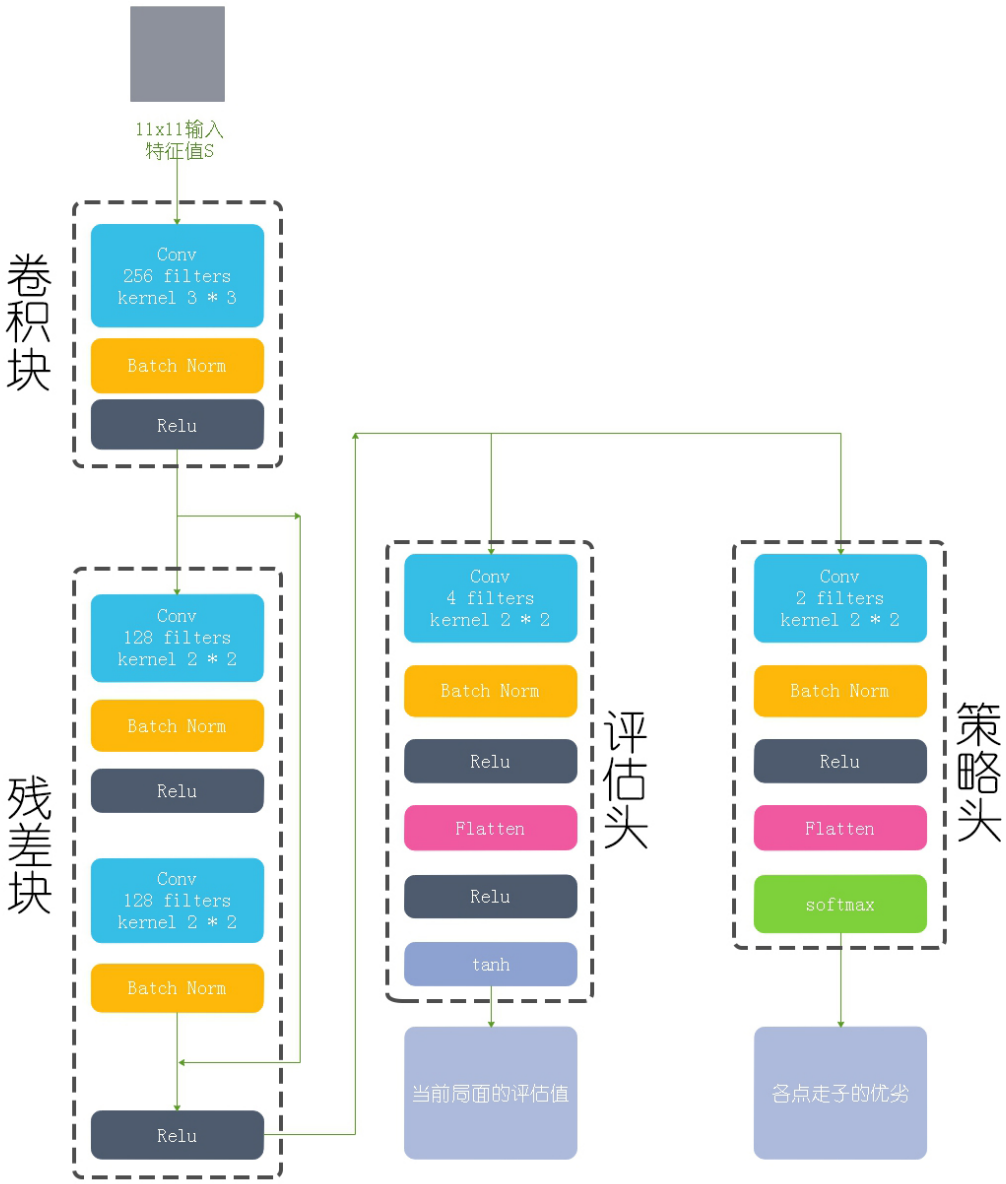
二，神经网络思路

a） 强化学习的基本流程

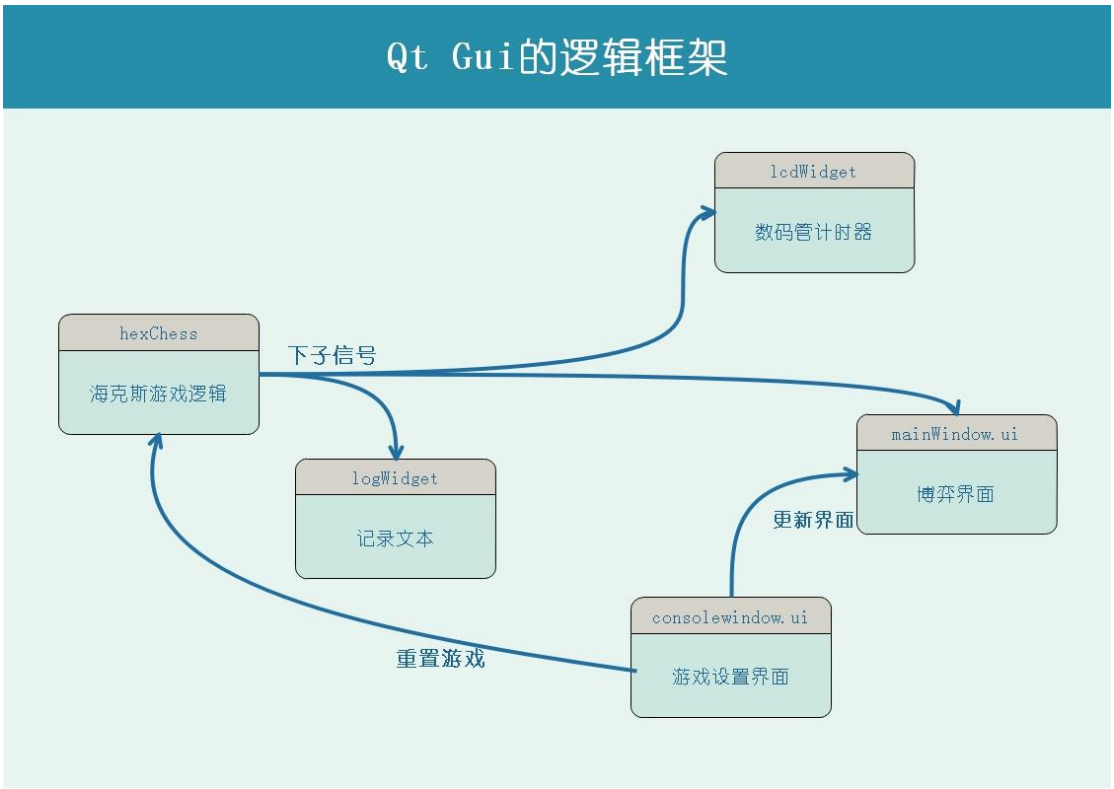

AlphaHex Flowchart
开始
利用女王蜂树搜索自我博弈产生训练集
自我博弈次数达到100次
否
是
利用训练集以及keras框架训练出深度学习模型
新模型与女王蜂博弈两方各40次
新模型胜率大于55%
否
利用新模型自我博弈产生训练集
自我博弈次数达到100次
否
是
利用训练集以及keras框架训练出深度学习模型
新模型与女王蜂博弈两方各40次
新模型胜率大于55%
否
是
胜利次数大于10或者迭代次数大于200
否
是
保存模型
结束

b） 神经模型的构架

AlphaHex 神经网络构建图



三：界面逻辑：



第四章 测试报告

一：运行速度：

- 运行环境：

  cpu:Intel Core-i5 6300HQ @2.30GHZ

  gpu:nvidia gtx950m

  cuda version:9.1

  keras version:1.0

- 训练过程：

  棋盘大小 5 * 5，迭代次数 150 次，45min

  棋盘大小 8 * 8 迭代次数 150 次 13h

  棋盘大小 11 * 11 迭代次数 80 次 32h

- 运行过程：

  博弈树：

  棋盘大小 5 * 5，< 0.5s

  棋盘大小 8 * 8 1s

  棋盘大小 11 * 11 15s

  深度学习：

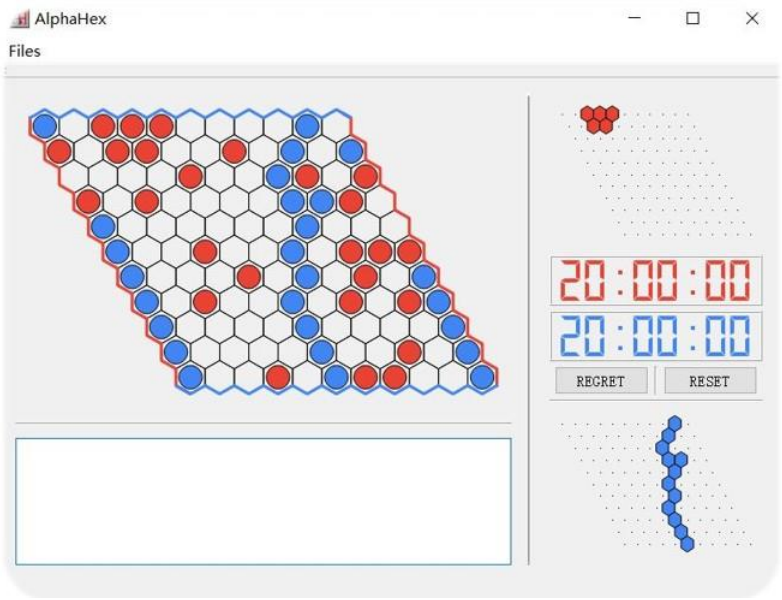
  棋盘大小 5 * 5，< 0.5s

  棋盘大小 8 * 8 < 0.5s

  棋盘大小 11 * 11 < 0.5s

二：可用性：

可以便捷的利用 gui 界面与 AI 进行博弈



三：部署方便性：

为了方便 python 环境的部署以及利用我们利用 freeze 导出了运行该项目所需的所有环境，并且提过了 README.MD 中的部署文档，帮助用户的轻松的实现环境部署。并且编写了一键部署脚本，可以在联网条件下完成对 python 依赖项的一键安装。而对于 python 调用 c++部分，若修改了评估参数，可以直接利用 produce_cppdll 脚本实现动态库的更新。

四：拓展性：

可以轻易的更换 Hex_game 为其他完全对抗博弈棋类，并更改评估过程便可以将其运用在其他棋类的部署上。

第五章 安装及使用

如果想运行该项目的核心部分（强化学习），安装 keras 框架是必不可少的，我们已经将 python 需要的依赖包列举了出来，你可以通过执行'bash install_pyenv.sh'实现依赖项的一键安装。但是如果需要使用 nvidia gpu 加速模型的训练、预测,安装 cuda 是必不可少的<https://developer.nvidia.com/cuda-downloads>。但是如果只是执行女王蜂树搜索部分，这些并不是必须的。你可以直接在'''./qt_alphaHex'''文件夹中运行该 qt 项目，并且利用 qt 良好的 gui 从而实现对抗的可视化。我们训练了一个 11x11 的 keras 模型（'''./py_train_model/keras_model/hex11.h5'''）,由于算力有限只实现了 100 次迭代。默认使用的模型是该模型，你可以将你训练的模型替换它，以得到更好的效果。

----------基本安装步骤----------

#linux(推荐):

1.安装 python3

-创建文件夹

mkdir /usr/local/python3

-官网下载 python3

'https://www.python.org/downloads/'

-安装依赖包

yum install gcc -y \
yum install zlib -y \
yum install zlib-devel -y \
yum install openssl-devel -y

-编译安装
cd /opt/Python-(version) \
./configure --prefix= /usr/local/python3 --with-ssl \
make && make install

-添加 python3 到环境变量
vim ~/.bashrc \
export PATH="/usr/local/python3/bin:$PATH" \
source ~/.bashrc

2.安装依赖项（若需要训练模型）
-'bash install_pyenv'

3.安装 Qt5（若需要部署到 qt）

4.修改 qt pro 文件（若需要部署到 qt）
- 将 LIBS += -L/home/reget/anaconda3/envs/keras/lib/ -lpython3.6m
修改为 LIBS += -L'本地 python3 lib 路径'

#windows:
1.安装 python3
-官网下载安装 python3
2.安装依赖项（若需要训练模型）
-根据'py_train_model/requirements.txt'文件 install 相关依赖包
3.安装 Qt5（若需要部署到 qt）
4.修改 qt pro 文件（若需要部署到 qt）
- 将 LIBS += -L/home/reget/anaconda3/envs/keras/lib/ -lpython3.6m
修改为 LIBS += -L'本地 python3 lib 路径'

----------基本运行方法----------

一：实现女王蜂评估体系的可视化运行:
在 window 环境下直接打开'win_exe/alphaHex.exe'文件，即可实现运行

二：训练新的模型：
-python py_train_model/play.py
tips：
-可以修改'cpp_evaluate/eval.h'文件实现评估策略参数的修改，
并执行'bashe produce_cppdll.sh'可实现'libpymcts.so'动态库的更新。
-可以修改'py_train_model/play.py'中的 size 实现不同大小棋盘的训练
-训练好的模型可以在'py_train_model/keras_model'中找到

三：将模型可视化调用：

-修改 pro 链接库到本地'python.h'路径

-修改'qt_alphaHex/src/policy/ai_policy'中的

PolicyNet::PolicyNet()

''Py_SetPythonHome(L"/home/reget/anaconda3/envs/keras")''

为本地 keras python 环境路径

-修改 int PolicyNet::getAiMoveNum(GameCore *core)中

''if(size != 5){

cerr<<"bad size"<<endl;

exit(0);

}''

修改 5 为你的模型大小。

第六章 项目总结

一：项目协调：

该作品是我们小组三人的共同完成的成果，无论是在初期的想法构思的讨论、辨析乃至后期的实现分工，都离不开每个人的付出。在该项目的实现过程中我们选择了每个人负责其中的一个部分，并且规定好各个模块的接口。以方便在最后的拼接的过程。

二：任务分解：

即使是一个模块的编写也需要考虑多个方面的过程，就比如说 gui 界面的编写，涉及到了多种模式的转换以及各处信号与槽函数的建立。于是我们将其划分为多个界面，利用下子的过程在其中产生各部分控制型号。

三：面对困难

博弈树算法的实现、与调试并没有花费过多的时间。但是存在运行时间的难题，如果一步的计算时间为 1min，由于训练集的生成需要成百上千次的迭代，那么我们认为其在强化学习中是完全没有利用价值的。于是我们在这方面下了很大的功夫才使得其具有如今较快的反应速度。

四：水平提升

通过本次的作品，为了加快博弈树的运算时间，我们选择了将其部署在 c++语言中执行，但是这样便涉及到了之前很少用到的语言间的交互调用。无论是到 python 对 c++的调用还是利用在 c++中对脚本语言的执行。都是一种全新的尝试。并且通过该次作品的完成工程理解到了团队协作的重要性。

五：后续升级

在博弈树方面，会引入多线程，使得其在第一层不采用 α－β 剪枝时充分利用 cpu 的性能。

在强化学习的过程中，由于博弈树利用 cpu 性能，而训练过程中目前利用的是 gpu。便可以在此过程中提前进行下一次的自我博弈迭代以节约时间，另一方面修改部分网络结构，使其更快的达到收敛。Gui 界面部分，将会开放网络接口使得人人可以在自己的电脑上训练模型，或者开发评估算法。实现一个检验、竞技的平台。