**GraphNAS Sample分析**

**目录：**

**1. sample过程代码执行流程分析**

**2. sample原理分析(基于策略的强化学习机制)**

**3. 实验验证**

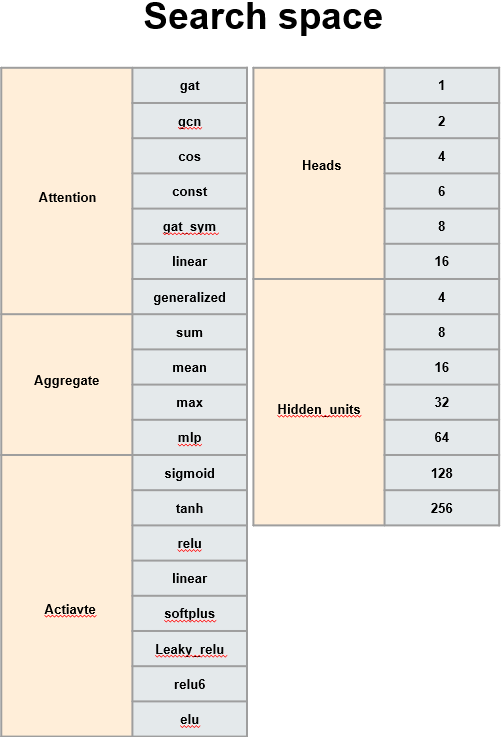
**参考**

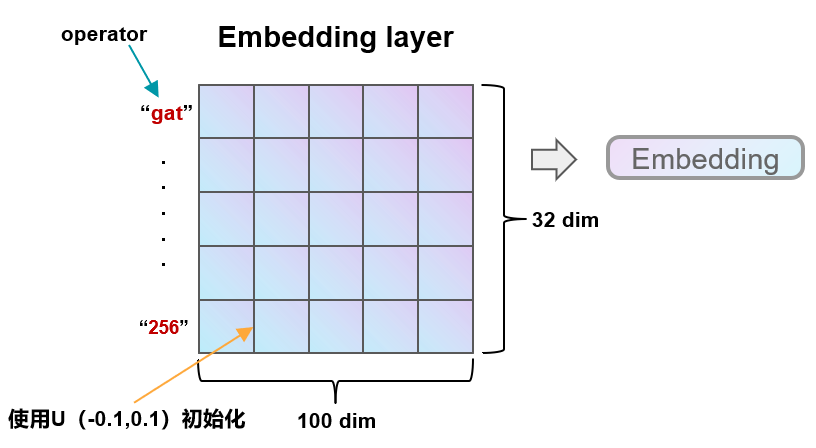
**附录**

**一、sample过程代码执行流程分析**

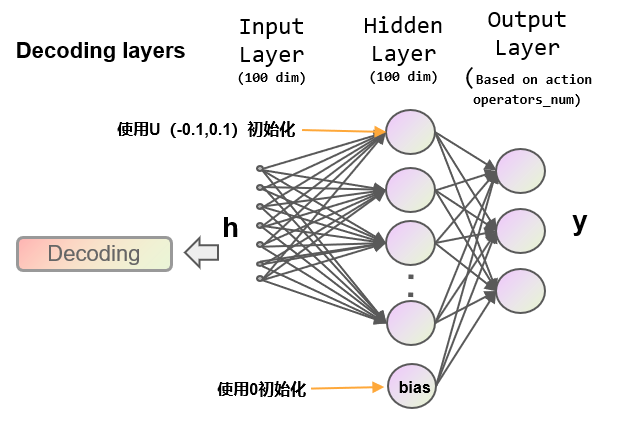
**1.controller定义：**

（1）.embedding、search space定义：

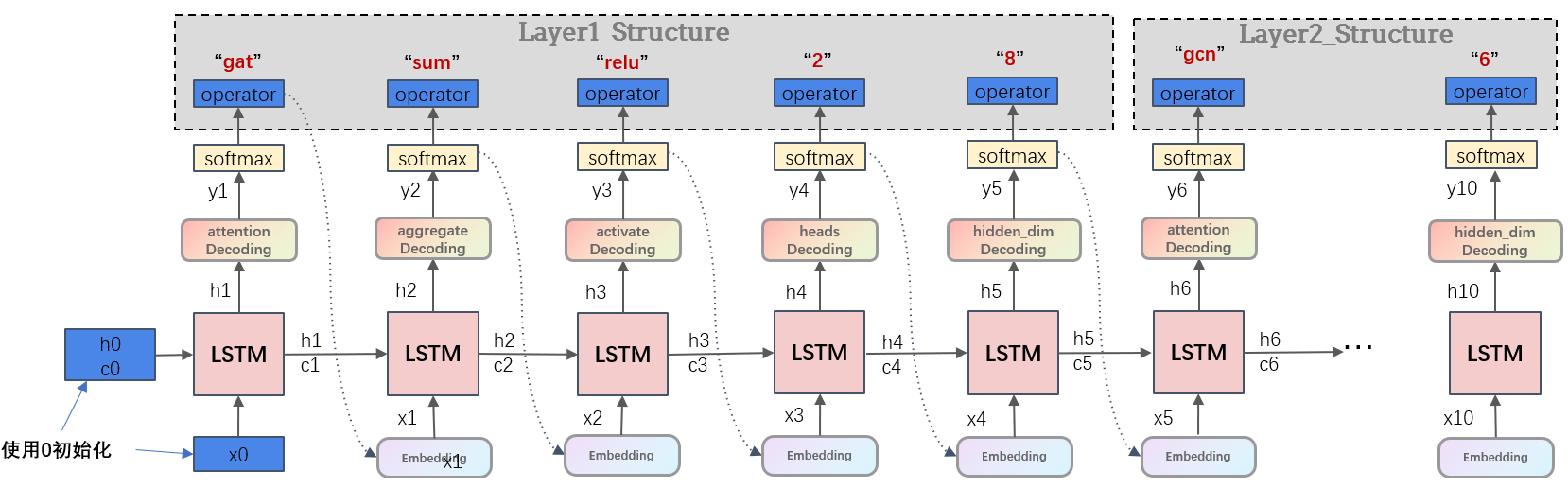




（2）.decoder定义：



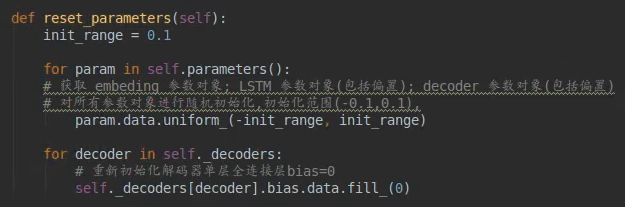
（3）.LSTM编码、解码过程

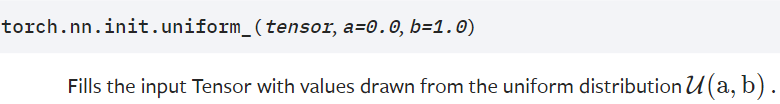


（4）.初始化操作的代码部分

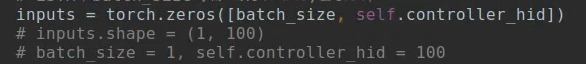
脚本graphnas\_controller.py

4.1、embedding、decoder、LSTM使用范围定义在上下定义在[-0.1,0.1]之间的均匀分布初始化，其中decoder的偏置使用0初始化



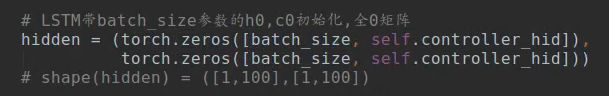


4.2、x0使用100维0向量初始化：

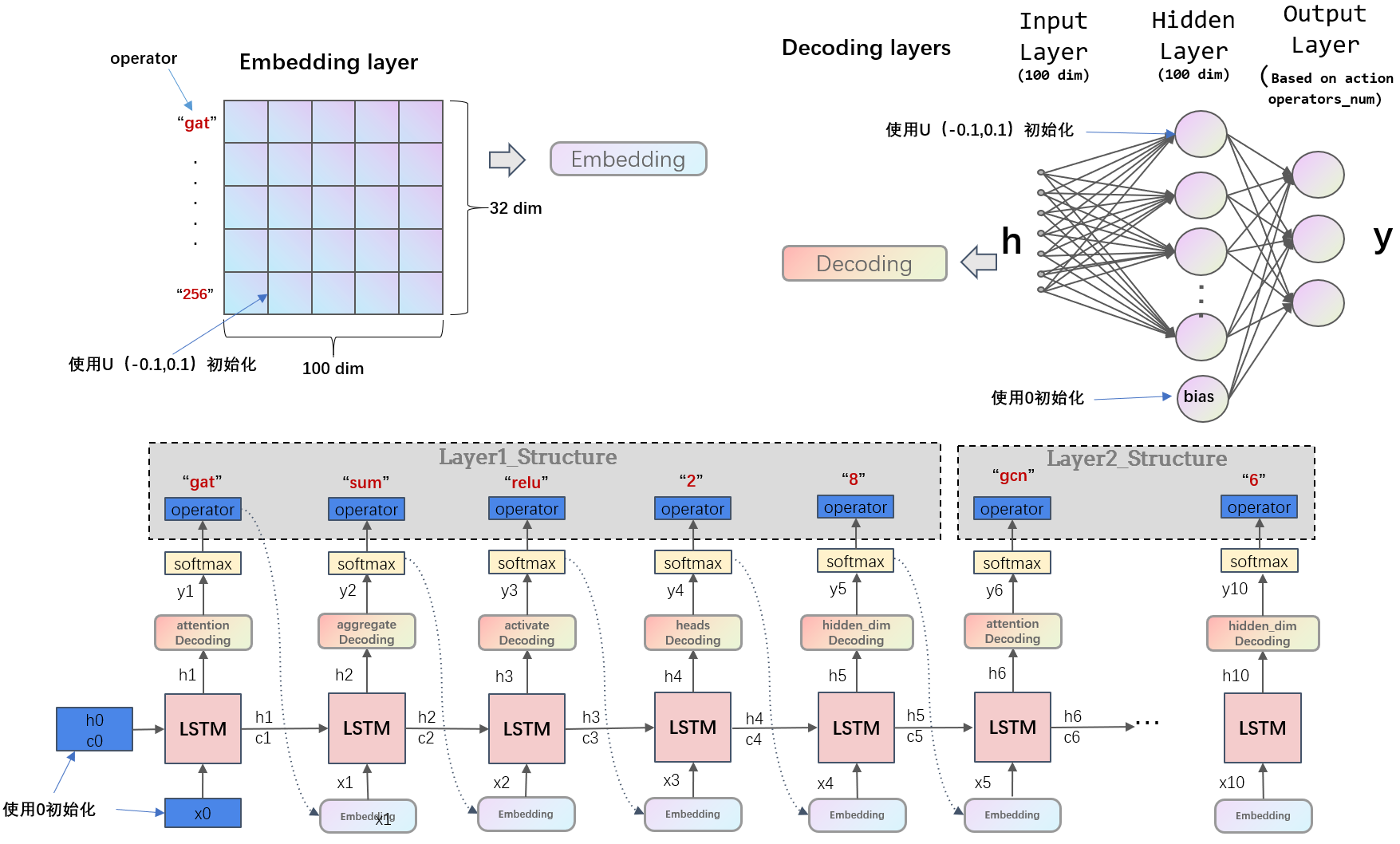


4.3、h0、c0使用100维0向量

初始化：



（5）.sample过程：



(1).按照attention到hidden\_units的action顺序每一个LSTM时间步选择一个operator；

(2).第一个时间步输入的向量是x0,是全零向量

(3).LSTM输出100维度的h1向量

(4).对h1进行decoding,将h1输入单层mlp进行线性映射,得到相应维度输出y1，attention\_decoding输出维度为7（operator的数量）；

(5).对向量y1进行softmax概率化得到y1的概率向量probs;

(6).计算向量probs的自信息向量log\_prob;

(7).基于probs与log\_prob计算本次输出向量y1的熵entroy;

(8).基于向量y1构建多项式概率分布，对向量y1中的向量值进行概率采样，获取采样值的索引编号index，通过索引号获取对应相应的operator（向量值越大，被采到的概率越大）;



(9).基于index选出自信息向量log\_prob中自信息值selected\_log\_prob;

(10).使用entropies列表收集selected\_log\_prob;

(10).使用emcoding对选择出的operator进行编码作为LSTM第二个时间步的输入;

(11).将此次选出的operator放入actions列表中保存

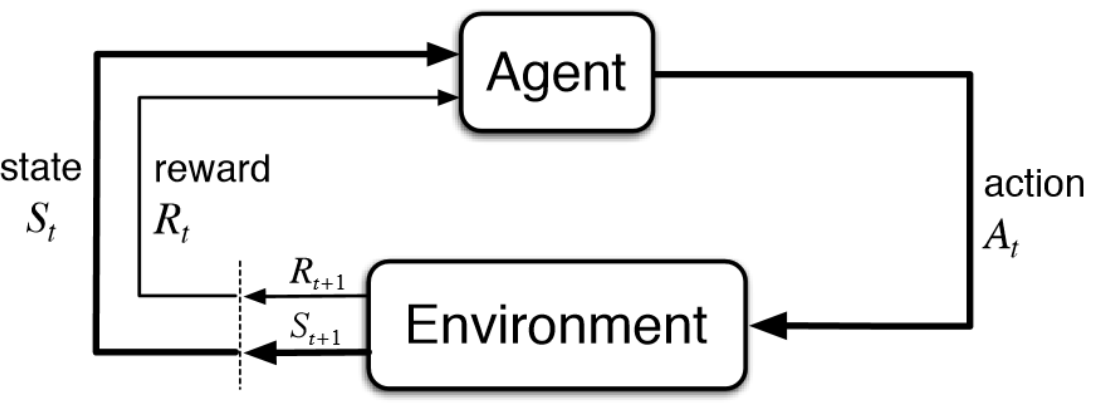
(11).循环上述过程直至选出两层网络中10个operators组成一个GNN结构，收集10次的

entroy值与selected\_log\_prob值为计算LSTM的loss值做准备;

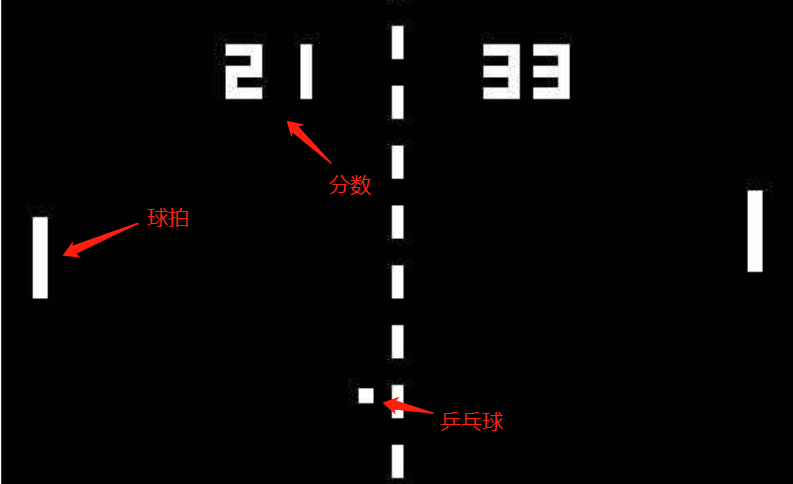
**二、sample原理分析(基于策略的强化学习机制)**

**1.基于策略的强化学习机制**

**1.1.强化学习的基本要素**



**强化学习逻辑流程**



**游戏Atari PONG**

**智能体**（Agent）：**指能够通过动作与环境交互的对象**，强化学习中智能体通常是带参数的算法，比如在游戏中的主体是用于控制本方球拍的算法

**环境**（Environment）：指**智能体动作作用的对象**， 比如游戏本身。

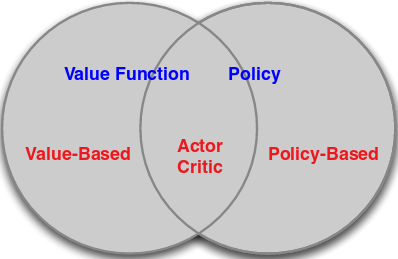
**动作**（Action）: 指**所有可能作用于环境上的操作**，比如游戏中算法控制球拍上下移动。

**状态**（State）: 指**可被主体感知的关于环境的信息**，比如游戏中屏幕显示的球和球拍的位置以及移动方向和速度信息。

**奖励**（Reward）: 指由**环境反馈给智能体描述上一个动作效果的信息**，比如游戏中球拍动作导致双方的得分变化

**1.2.强化学习目标与分类**

目标：**学习一个策略来获得最大的奖励**



**强化学习分类**

**1.3.基于策略的强化学习**

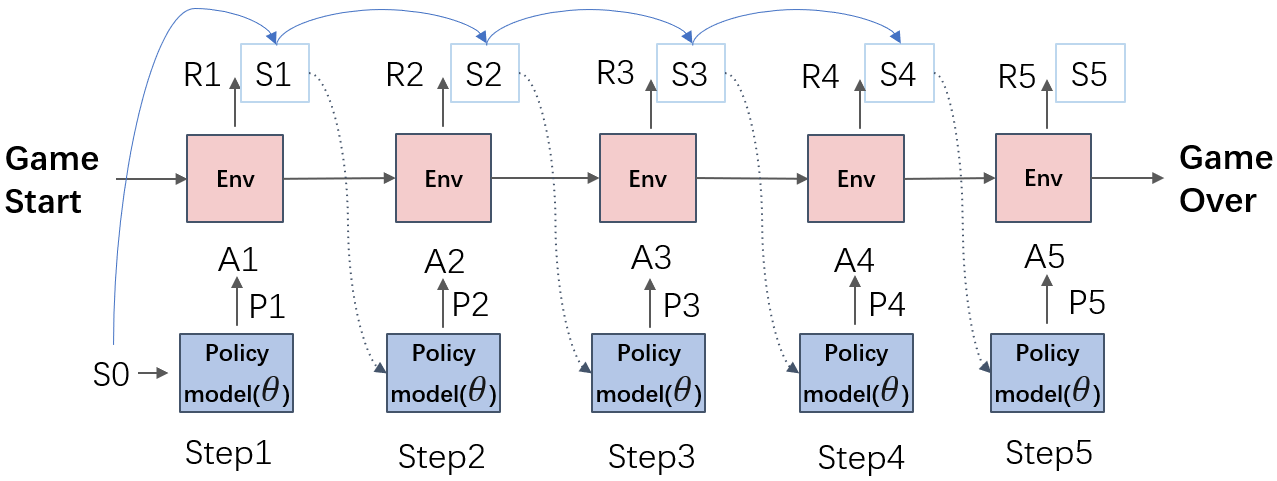
(1).**策略(Policy)**:表示在状态S和策略模型参数的条件下，发生动作a的概率P



ps:策略模型是基于策略强化学习中训练的对象

(2).**轮次(Episode)**:表示从起始状态开始使用某种策略(由策略模型的决定)，产生一系列动作与环境进行交互，直到某个状态终结。例如:在围棋游戏中的一个轮次就是从棋盘中的第一个落子开始直到对弈分出胜负。

(3).**策略(Policy)模型的目标函数**：



**策略(Policy)强化学习一个轮次(Episode)逻辑流程**

(3.1).**策略模型的目标函数:**

对目标函数求导，得到策略梯度(policy gradient)，使用梯度上升算法优化策略模型参数，使得目标函数最大化——>实现强化学习的目标:

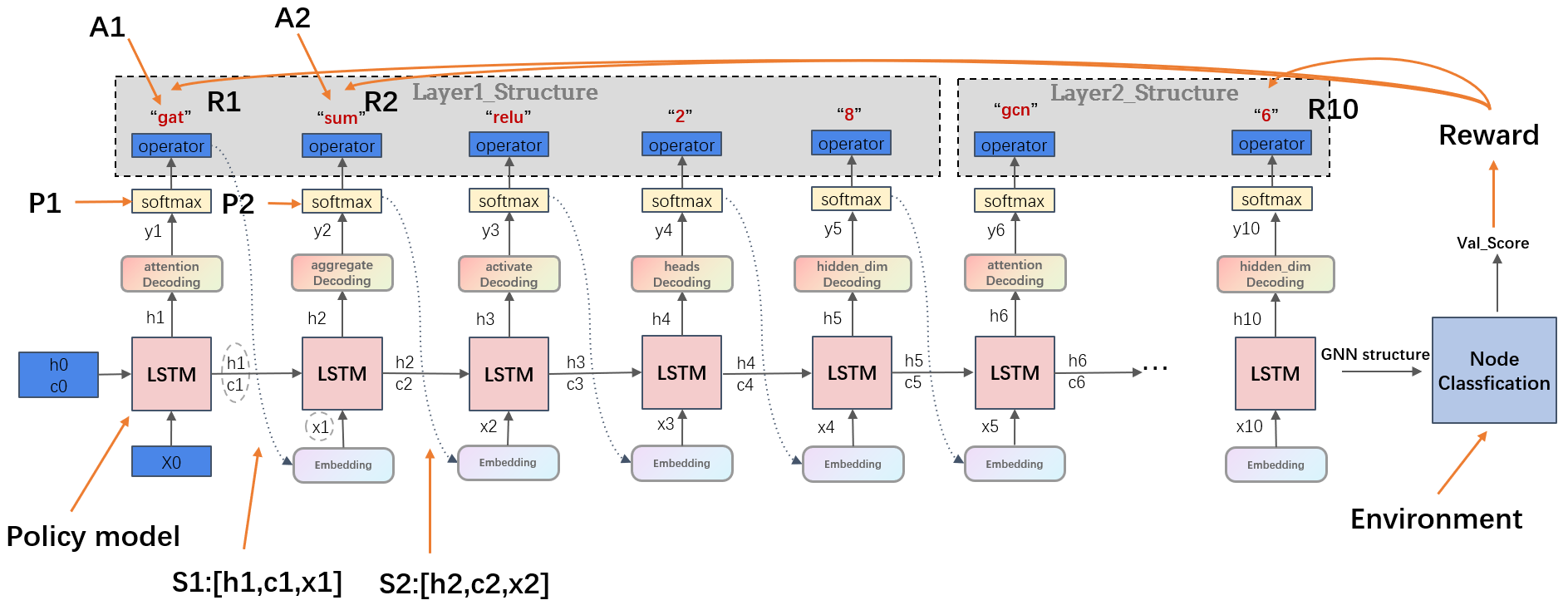
(3.2).目标函数理解：

<1>.全局理解：通过训练找到一组参数，使得在一轮(或多轮游戏取平均)

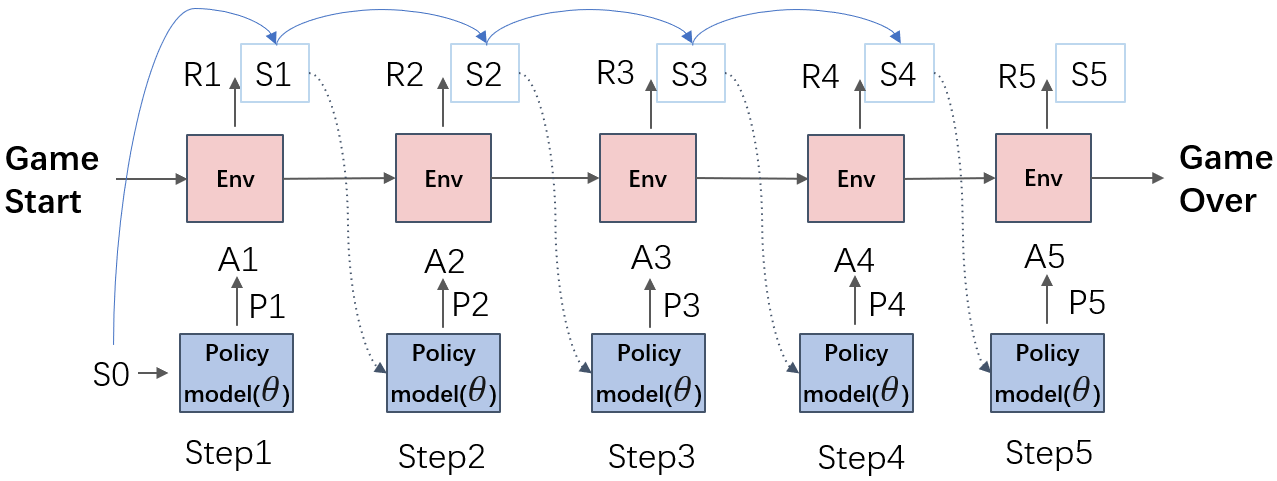
游戏中，**其获得的奖励的期望()最大**

<2>.局部理解：从某一次采取行动获得奖励上看，，如果某阶段采取**action1**获得较小的，则模型会在训练更新参数后，降低此阶段选择此**action1**的概率，增大此阶段选择其他**action**的概率，以保证更大概率的获得更高奖励，同理，当本次选取的**action1**获得较大的时，模型在更新参数后，会提高本阶段选取**action1**的概率，策略模型运行到此阶段具备更高概率选择产生高奖励的动作**action1**。

**2.GraphNas强化学习机制解析**



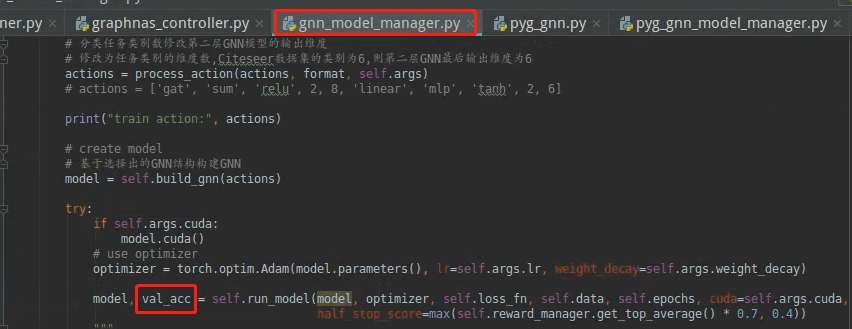
**graphnas基于策略强化学习机制**

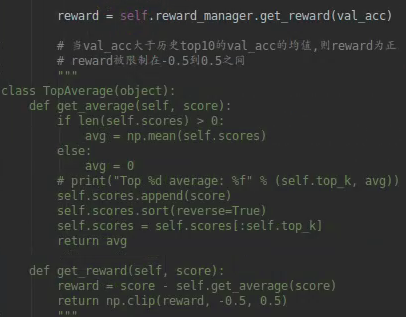


**策略(Policy)强化学习一个轮次(Episode)逻辑流程**

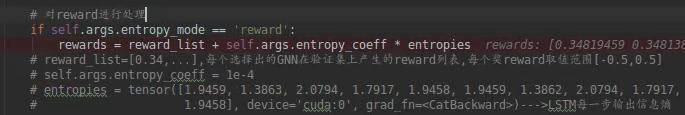
**3获取reward、计算策略损失、更新controller代码实现细节：**

(1).基于GNN\_structure获取val\_score，计算reward





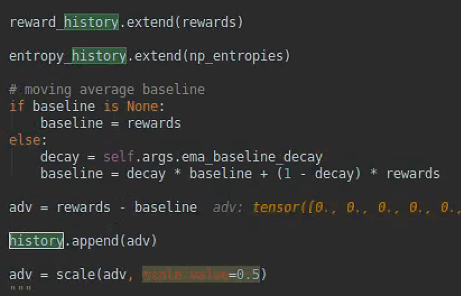
(2).对10维entropies数组乘上熵系数coeff，然后对熵数组entropies(熵越大信息的不确定度越高，信息量越大)每个位置加上本次计算得到reward(0.348)值，得到一个10维的rewards\_list，类型array。——>计算每一个action产生的reward



**每个action产生reward的设计？**

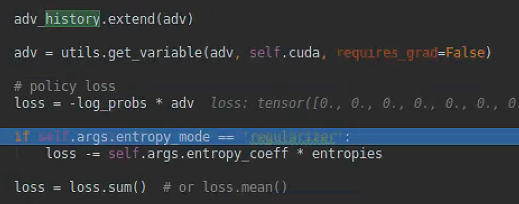


(3).基于滑动平均baseline计算reward



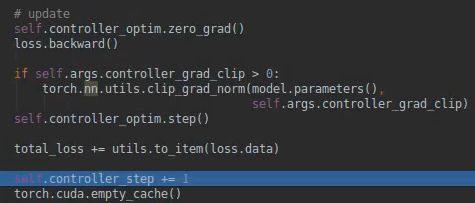
**baseline随着controller选择GNN的表现越来越好，在不断动态提高baseline的值，对每一轮trajectory产生的rewards\_list中每个action导致的reward是正反馈还是负反馈的考察也越来越严格，提高了controller模型达到奖励期望最大化的训练速度**

(4).计算本轮次获得奖励期望，添加负号将其转化为梯度下降优化



* 由于policy gradient需要使用梯度上升优化，而代码给loss添加了负号转变为使用adam梯度下降优化
* baseline=baseline\*0.95+0.05\*rewards
* baseline继承上次baseline的0.95信息，融入本次训练得到的0.05rewards信息

(5).使用adam优化controller参数

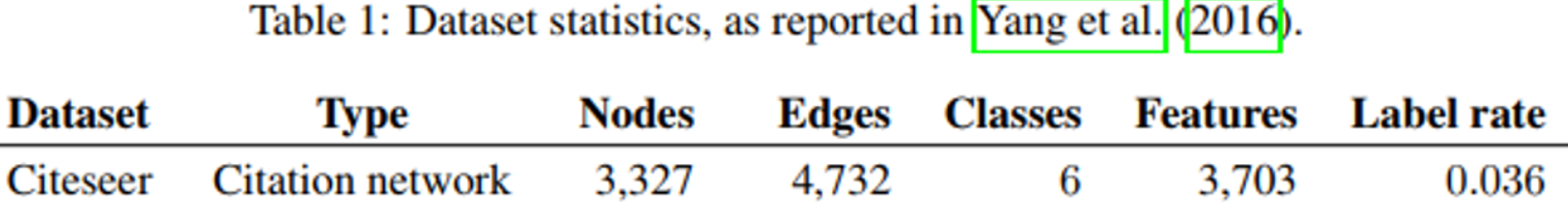


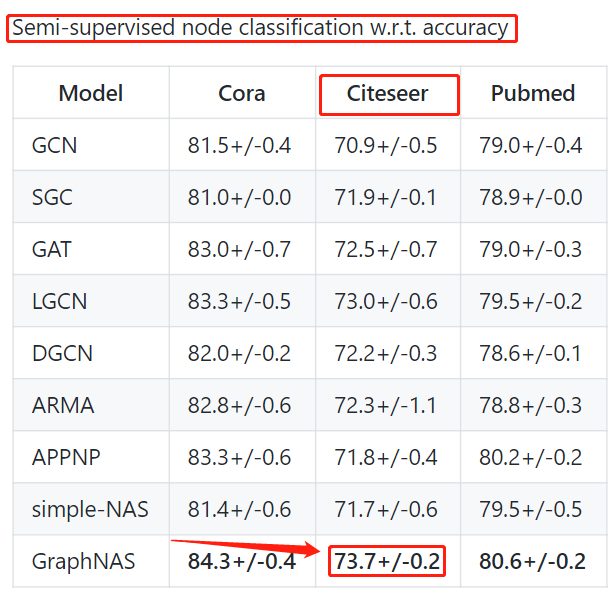
**三、实验验证**

**1.验证**

**数据集：**

**使用数据集中120个节点的标签信息修正网络参数**

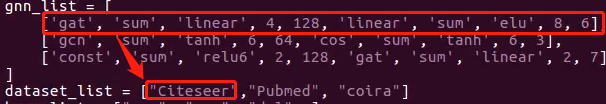




**验证训练获得的最佳GNN结构与测试效果：**



**论文给出的给出的最佳GNN结构：**



**2.搜索效率分析：**

**按照定义的搜索空间**

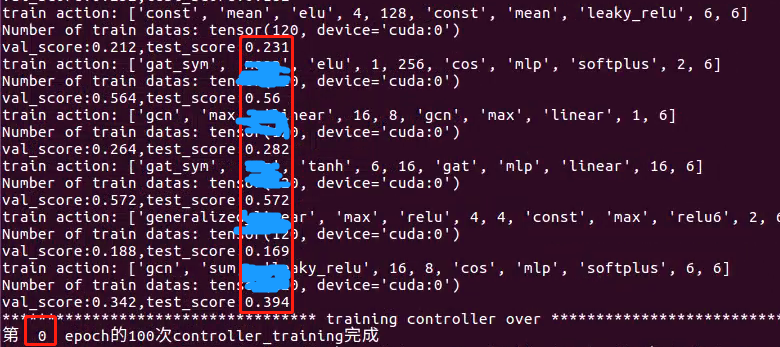
**穷举搜索空间大小9408**

**GraphNas搜索空间小于2000（可能包含重复GNN结构）**

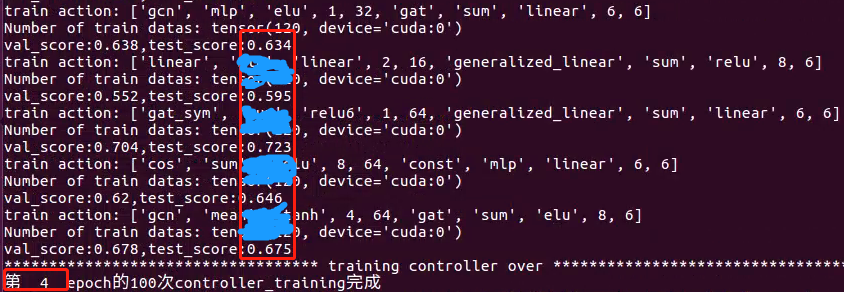
**(1000个搜索结果基于controller训练过程产生，1000基于derive过程产生)**

**空间搜索比例小于21.2%**

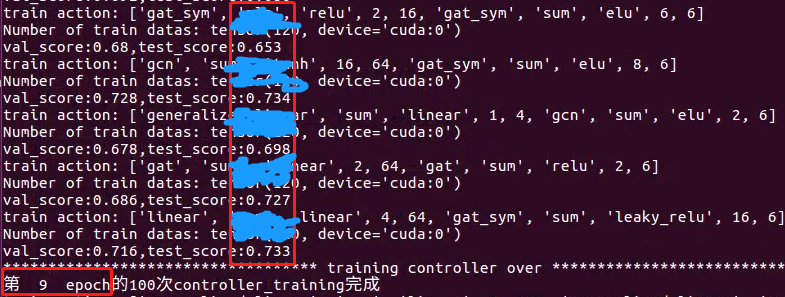
**3.controller训练test\_score变化趋势**



第一轮controller训练中，最后五轮**mean\_test\_score:0.3954，std\_test\_score:17.5%**



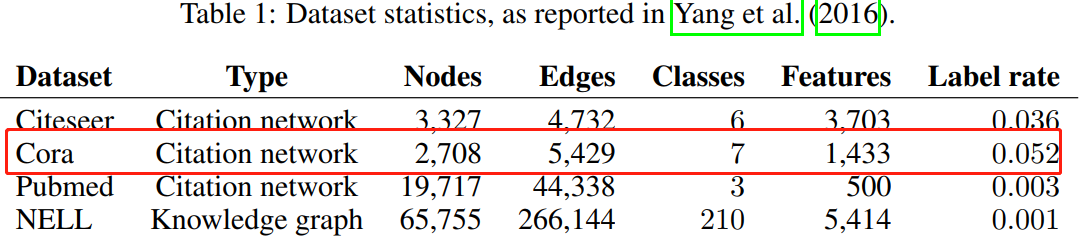
第五轮controller训练中，最后五轮**mean\_test\_score:0.6546，std\_test\_score:4.78%**

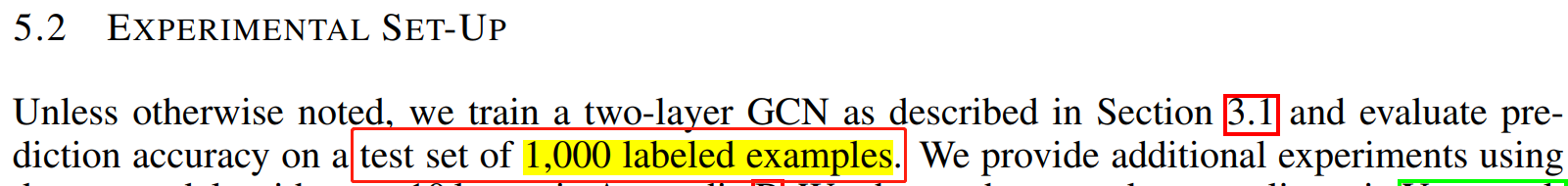


第十轮controller训练中，最后五轮**mean\_test\_score:0.709，std\_test\_score:0.347%**

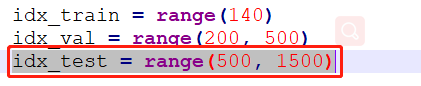
**半监督任务数据集划分：**

**Label\_rate:表示使用带标签节点训练的个数**





cora数据集划分



参考：

(基于策略强化学习)：

[1].<https://developer.ibm.com/zh/articles/ba-lo-deep-introduce-policy-gradient/>

[2].<https://blog.csdn.net/u013010889/article/details/82385592>

[3].<https://spinningup.openai.com/en/latest/spinningup/rl_intro3.html#deriving-the-simplest-policy-gradient>

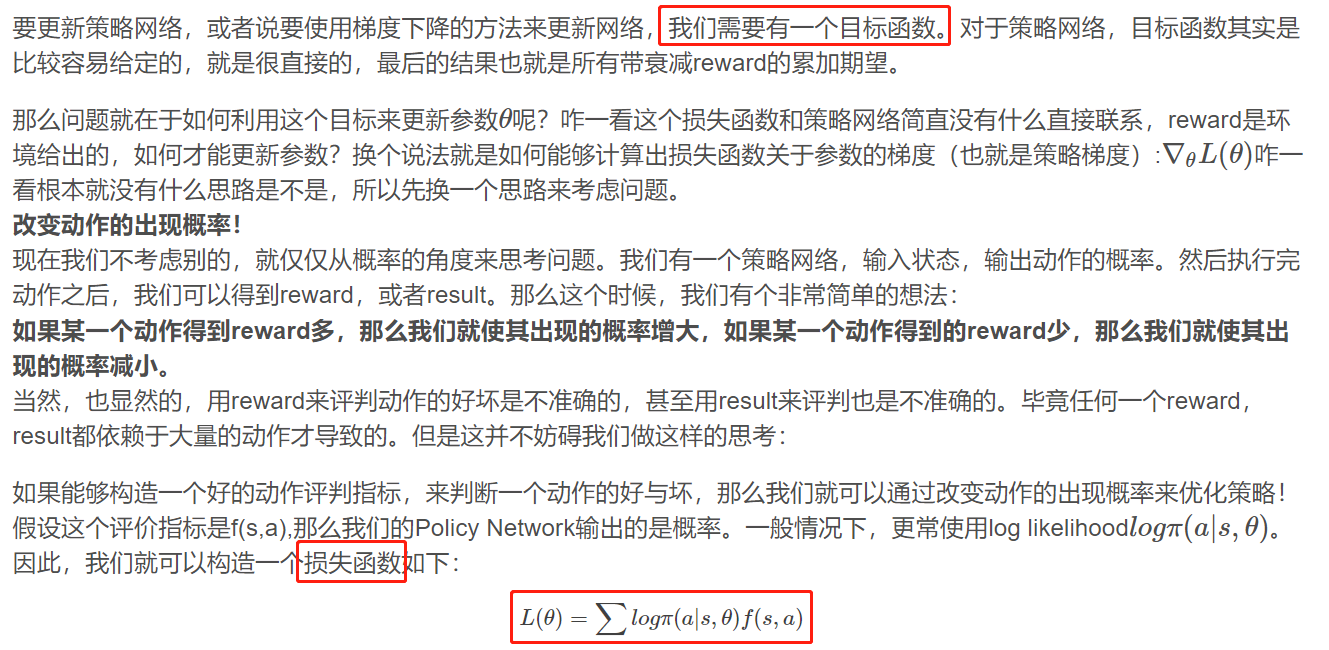
(baseline)：

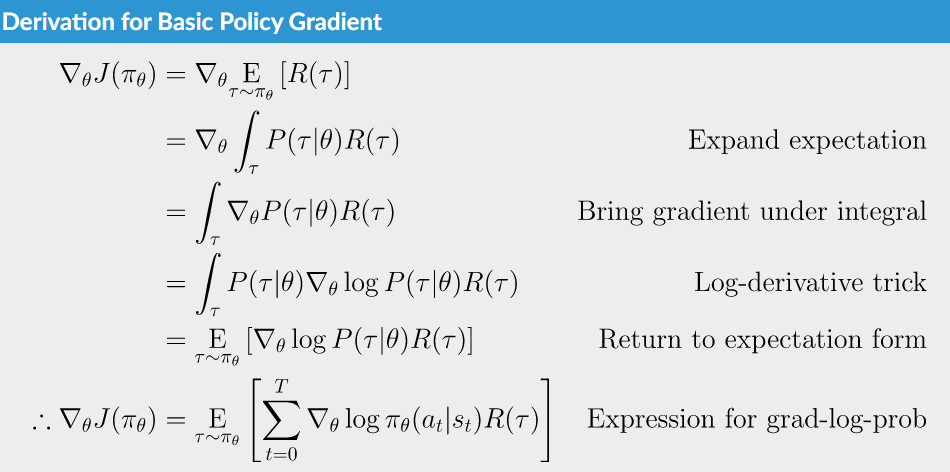
[4].<https://zhuanlan.zhihu.com/p/70273391>

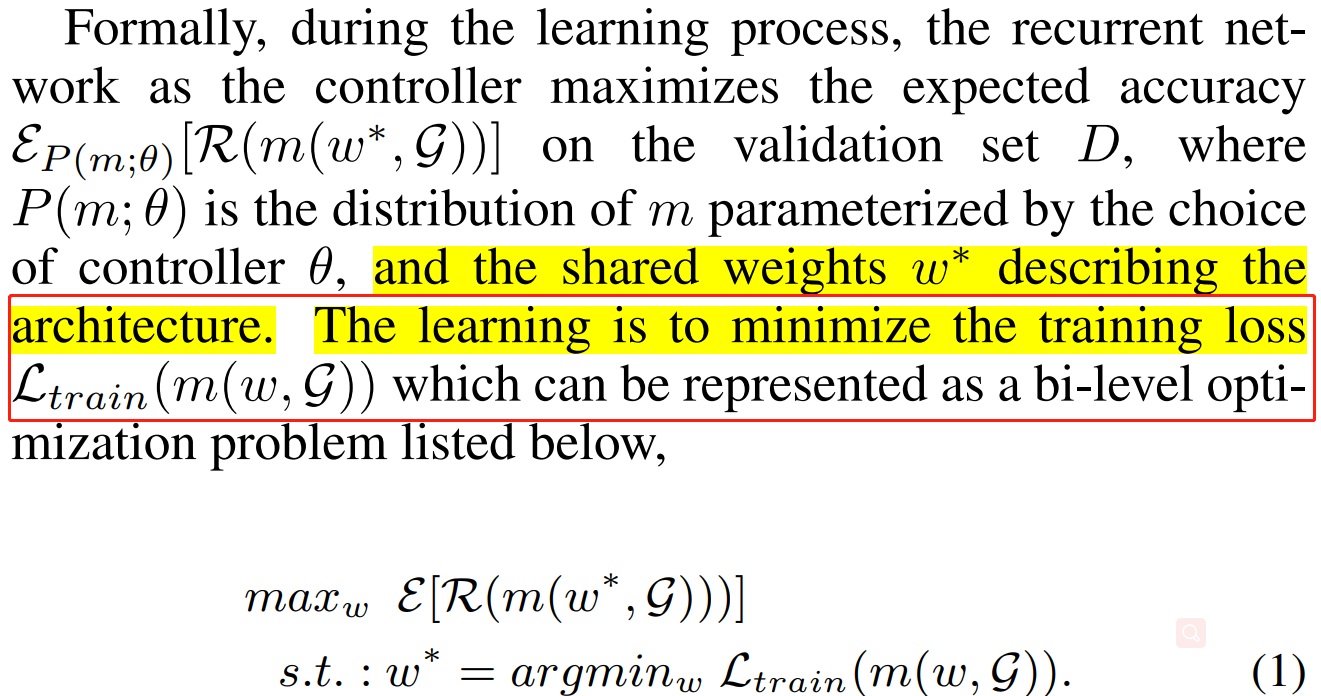
[5].<https://blog.csdn.net/u013010889/article/details/82385592>

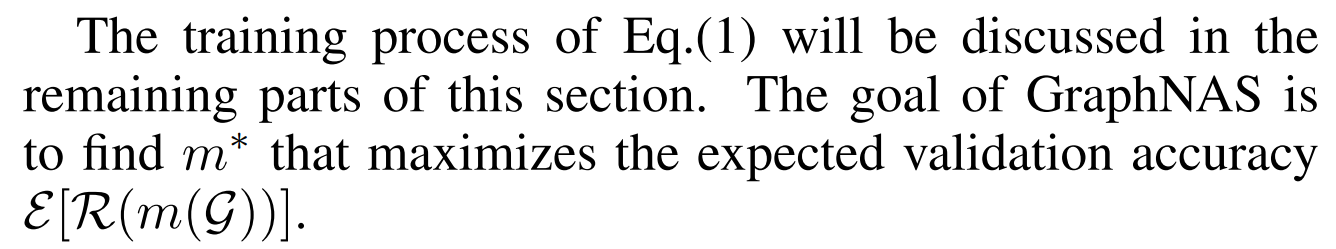
[6].<https://blog.csdn.net/mmc2015/article/details/79874219>

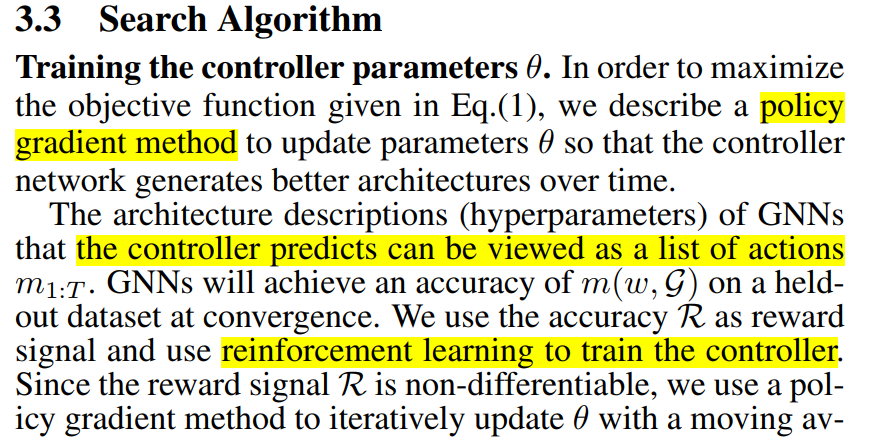
**笔记附录：**

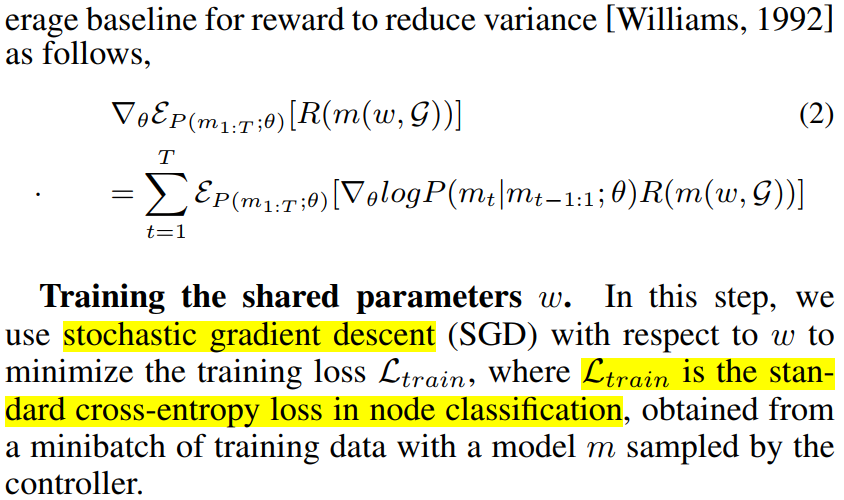






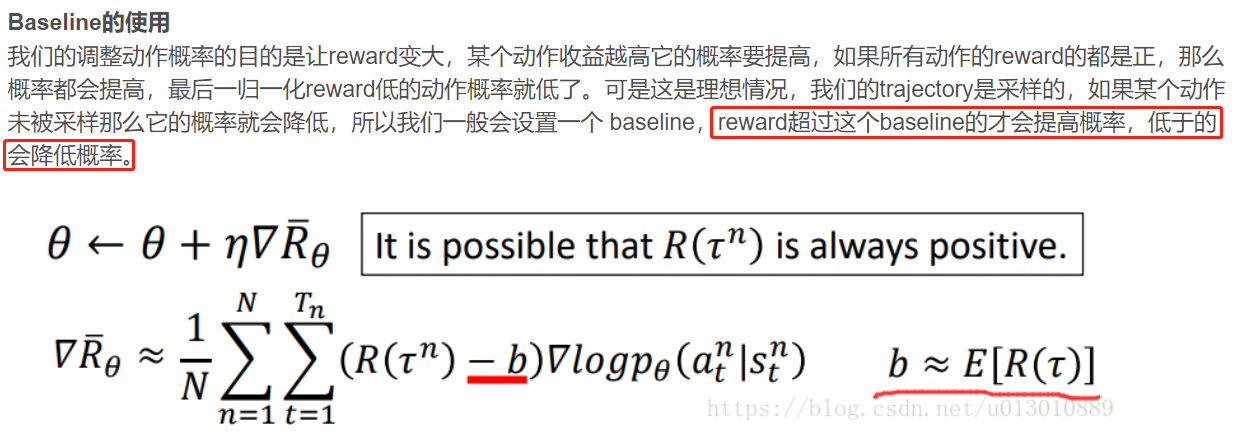








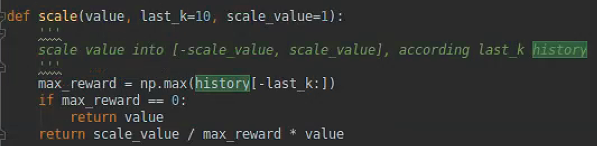
**baseline解释：**





本次sample每一步获得最终正负反馈由adv（adv = rewards - baseline ）决定,

history存放每一次的adv(adv=baseline - rewards)



1.挑选最近10次的adv列表，在10个列表中选出最大的值max\_reward

2.使用1/(2\*max\_reward)去规范化本次获得的adv列表

第二次训练的total\_loss与GNN测试结果:





第三次训练的total\_loss与GNN测试结果:





第四次：



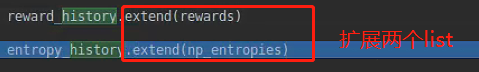


第五次：





**结论：当val\_score越高，loss的绝对值越小**



强化学习基本要素：

