第五章- GVGAI学习

佳林刘

前几章主要讨论了GVGAI框架的设计和GVGAI中的规划，其中每个游戏都有一个前向模型(FM)。GVGAI也可以用作强化学习(RL)平台。在这种情况下，agent不允许与FM交互，提前计划在真实游戏中执行的动作，而是需要通过经验来学习，从而多次进行实际游戏(如RL中的情节)，以逐步提高其性能。

进行了两个研究问题。首先，我们能否设计一个学习型智能体a_1在经过对同一款游戏的几个已知级别的训练后，能够在未知级别的游戏G上玩得很好$_1$?然后，在一个新的游戏G_2上，是否有可能训练一个agent a_2，基于一个$_1$，比从头开始更有效率的e?rst问题的重点是提高一个智能体在相同目标和规则下执行类似任务的能力。一个应用是仓库中的自主订单拣选机器人，它可以优化路线并通过di不同的拣选区域旅行，以最大限度地提高拣选e效率。第二个应用旨在通过转移在不同任务(例如，具有di rent规则的任务)上学到的知识来提高新任务的学习效率。例如，网球运动员可能比两种运动都不玩的人更快地掌握羽毛球。

本章首先提出了GVGAI学习的挑战(第2.1节)。第2节提供了GVGAI学习平台的总体视图，然后介绍了已经实现的两个环境。第3节介绍了比赛中使用的规则和游戏，提交到比赛的代理已经使用的方法，以及di erent方法的性能分析。竞赛的参赛作品在第4节中描述。

1 GVGAI学习的挑战

规划和学习任务共享一些相同的挑战，包括但不限于，缺乏先验知识，对通用性的要求，延迟奖励(通常与在奖励景观一起)和实时约束。

先验知识的缺乏先验知识的缺乏是双重的。第一，游戏规则没有提供。代理的目标是赢得一场游戏，但它不知道如何得分，赢/输或任何终止条件。只有代理人的

可以访问当前游戏tick的状态观察，包括当前状态下的合法行为。但是，这种状态观察并不一定覆盖整个游戏状态。其次，在相应的比赛中，只发布部分游戏关卡进行训练;agents将在未见过的关卡上进行测试。

由于电子游戏的实时性，学习智能体的构建和初始化时间不应该太长，在游戏过程中，智能体需要足够快地决定一个动作(在GVGAI比赛中，分别不超过15毫秒和40毫秒)。这保证了更流畅的游戏体验，类似于人类玩家玩实时视频游戏时的体验。出于这个原因，在第3部分后面介绍的比赛中，一个学习智能体在比赛中被取消资格⊠ed，或者根据选择一个动作所需的实际时间，执行一个doNothing动作而不是它的选择。

一般游戏玩法另一个问题是，一个经过训练，在一个特定的⊠c级别的特定⊠c游戏中表现良好的智能体，可能在类似的游戏级别上表现得相当好，但在非常不同的⊠不同的游戏或级别上很容易失败。例如，人类级别的超级马里奥代理可能无法在吃豆人游戏中存活很长时间;在《太空入侵者》中占主导地位的智能体很可能无法解决《推箱子》。设计一个在一套di⊠erent未知游戏上表现出色的单一智能体并非易事。
在GVGAI框架下的所有游戏(以及比赛)中，主要任务是赢得游戏。但是，除了当前章节的即时游戏分数外，没有提供获胜条件。在益智游戏和桌游中，游戏过程中的即时分数景观通常是⊠at直到最后一集，因此设计一些启发式和提前规划是必要的。无论是对于一款游戏的未见过的关卡还是一套游戏来说，适当的启发式都不是微不足道的。

2 框架

与规划轨迹不同的是，在学习环境中没有给予agent FM，因此无法模拟游戏。为了避免访问前向模型，2017年在主要GVGAI框架之上实现了一个新的接口。然后，Philip Bontrager和Ruben Rodriguez Torrado在2018年将itt与OpenAI Gym进行了接口。前者已经在IEEE 2017年计算智能与游戏大会(IEEE CIG 2017)上组织的第一届GVGAI学习竞赛中使用，该竞赛支持用Java和Python编写的代理。然后在IEEE 2018年CIG上组织了第二场比赛

与OpenAI Gym接口的框架，它只支持用Python编写的代理。由于使用了VGDL, GVGAI框架可以很容易地在大量的游戏上训练和测试学习代理。由于所需的通用性和有限的在线决策时间，这是一个di□cult强化学习问题。

2.1 GVGAI学习框架

在每个游戏集合中提供多个游戏，每个游戏集合包含多个关卡(在框架中通常是□ve)。由于VGDL和GVGAI框架提供的集成游戏，这使得可以选择设计不同的游戏和关卡集。

在一个集合中的执行将有两个阶段:使用N_L可用的N级别的学习阶段和使用$N_T = N - N_l$的其他N_T级别的测试阶段。下面给出这两个阶段的大图，尽管在2017年和2018年的比赛中有一些di□的细节。

学习阶段:一个智能体有有限的时间T_L来玩一组M个游戏，每个游戏都有N_L个学习关卡。它将玩N_L的每个关卡一次，然后可以自由选择下一个关卡来玩，如果还有时间的话。除了正常可用的游戏动作之外，代理被允许在任何时刻将动作中止发送到□nish游戏。每次游戏结束时都会调用方法result，无论游戏是否□已正常结束，还是被使用abort的代理破坏。因此，只要遵守时间限制T_L，代理可以玩任意多的游戏，可能选择玩的关卡。

测试阶段:在学习阶段之后，训练好的智能体在N_T测试关卡上进行测试，这些关卡在学习期间从未见过。测试试验中的胜率和分数和游戏长度的统计数据被用来评估代理。

请注意，在这两个阶段都无法访问FM，但是学习代理在每个游戏时刻以不同的事件形式接收当前游戏状态的观察，这取决于使用的是哪种GVGAI学习环境。这两个环境都可在线使用，并可用于一般视频游戏学习的研究。

2.2 GVGAI学习环境

本节将简要介绍IEEE CIG 2017单人学习竞赛中使用的GVGAI学习环境，有关如何设置框架和竞赛程序的更多技术细节请参见技术手册[12]。

GVGAI学习环境与规划环境[1]是同一个项目的一部分。它支持用Java或Python编写的学习代理。在每个游戏tick，代理被允许发送一个有效的游戏动作来在游戏中执行，同时，在序列化的JSON游戏观察(字符串)和游戏屏幕的截图(PNG)之间选择下一个状态观察的格式来接收。在任何时候，代理可以选择在next game tick接收的游戏观察的格式，使用以下类型之一:

```
lastSsoType = Types.LEARNING_SSO_TYPE.JSON;   //request for a JSON
lastSsoType = Types.LEARNING_SSO_TYPE.IMAGE;  //request for a screen-shot
lastSsoType = Types.LEARNING_SSO_TYPE.BOTH;   //request for both
```

这个选择将被记住，直到智能体使用上述命令做出另一个选择。图1给出了屏幕截图的一个。

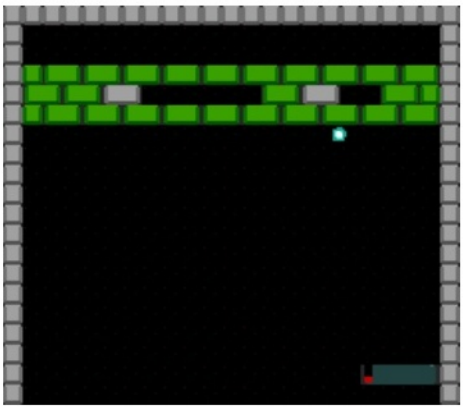

图1:示例:游戏屏幕截图。

下图是一个序列化状态观察的例子。

```
SerializableStateObservation{
    phase=ACT, isValidation=false, gameScore=8.0,
    gameTick=150, gameWinner=NO_WINNER, isGameOver=false,
    worldDimension=[230.0, 200.0], blockSize=10, noOfPlayers=1, ...
    ...
    availableActions=[ACTION_USE, ACTION_LEFT, ACTION_RIGHT],
    avatarResources={},
```

[1] https://github.com/GAIGResearch/GVGAI

```
8       observationGrid={
9           Observation{category=6, itype=0, obsID=578, position=0.0 : 0.0,
10          reference=-1.0 : -1.0, sqDist=2.0}
11          ...
12          Observation{category=6, itype=11, obsID=733, position=150.0 : 40.0,
                reference=-1.0 : -1.0, sqDist=24482.0}
13      },
14      resourcesPositions=null, portalsPositions=null,
            fromAvatarSpritesPositions=null}
15  }
```

与规划环境中一样，Java或Python代理应该继承抽象类AbstractPlayer，实现构造函数和三个方法:act、init和result。这个类必须命名为Agent.java或Agent.py。

学习代理的实现

Agent()构造函数在每个游戏中被调用一次，并且必须finish在不超过CPU时间的开始时间(默认是1s)，因此它在每个游戏的学习和测试阶段被调用一次。

init(SerializableStateObservation sso, ElapsedCpuTimer elapsedTimer)创建代理后，init()在每个游戏运行之前被调用。它应该在不超过CPU时间的初始化时间(默认为1s)内完成。

在每个游戏时刻，act()被调用并确定agent在规定的CPU时间动作时间内的下一个动作(默认为40ms)。可能的动作有动作左、动作右、动作上、动作下、动作使用和动作NIL(什么都不做)。如果超过动作时间DISQ(默认为50ms)，代理将立即被取消资格。否则，应用NIL动作(什么都不做)。请注意，在游戏或游戏状态中，并非上述所有操作都是可用的(合法的)操作，但动作 NIL总是可用的。

这个方法在每场比赛结束时被调用。它没有时间限制，所以除了指定的总学习时间外，代理不会因为超支而受到惩罚。智能体可以利用它在结果调用上花费的时间来做更多的学习或玩更多的游戏。在每次调用结果时，都应该返回一个动作或一个等级编号。

当玩家赢/输游戏或达到最大游戏时间(最大时间步骤)时，游戏结束。

超时如果代理在动作时间之后返回一个动作，但不超过动作时间DISQ，则执行动作 NIL动作。

如果代理在动作时间DISQ之后返回一个动作，代理将被取消资格并输掉比赛。

表1总结了框架中的表示法和相应的参数，以及默认值。

客户端(代理)参数			
变量		默认值Usage	
		1	
		1	init()的时间
Start_time	ACTION_TIME	40毫秒	代理构造器的时间每滴答返回一个动作的
initialization_time		50毫秒	时间每滴答取消资格的阈值允许学习一个
action_time_disq		5分钟	游戏的时间额外的学习时间用于通信的
total_learning_time		1	Socket端口
extra learning_time		8080	
socket_port			
服务器参数			
变量		默认的	使用
MAX_TIMESTEPS		1000	最大游戏次数一个游戏可以运行的情节数
VALIDATION_TIMES		10	验证

表1:学习框架中的主要参数。

2.3 GVGAI健身环境

GVGAI Gym是纽约大学工程学院博士候选人Ruben Rodriguez Torrado和Philip Bontrager将GVGAI框架与OpenAI Gym环境相结合的结果[15]。除了更加友好的界面，一个学习代理仍然会收到当前游戏屏幕和游戏分数的截图，然后在每个游戏tick返回一个有效的动作。使用GVGAI Gym播放第一级外星人的随机代理示例如图2所示。我们在表2中比较了GVGAI Gym实现和原始GVGAI环境。

	GVGAI规划 头号选手的双人	GVGAI学习GVGAI健身房	
相似之处	• 玩看不见的游戏，没有游戏规则可用 • 获取游戏分数，打勾，如果终止 • 获得法律诉讼 • 有权观察当前游戏状态		
提出模型?	是的	没有	没有
历史事件?	是的	没有	饱
状态的观察?	Java对象 Java	字符串或PNG Java python	PNG Python

表2:规划和学习环境的比较。

```python
import gym
import gym_gvgai

env = gym.make('gvgai-aliens-lvl0-v0')
env.reset()

score = 0
for i in range(2000):
    action_id = env.action_space.sample()
    state, reward, isOver, info = env.step(action_id)
    score += reward
    print("Action " + str(action_id) + " played at game tick " + str(i+1) + ", reward=" + str(reward) + ", new score=" + str(score))
    if isOver:
        print("Game over at game tick " + str(i+1) + " with player " + info['winner'])
        break
```

图2:使用GVGAI Gym随机播放外星人第1关的示例代码。

2.4 与其他学习框架的对比

其他通用框架，如OpenAI Gym [5]， Arcade Learning Environment (ALE)[3]或Microsoft Malm¨o[7]包含大量的单人/多人，无模型或基于模型的任务。与这些系统的接口将大大增加所有GVGAI代理可以通过公共API玩的可用游戏的数量。这也将使框架向3D游戏开放，这是当前基准没有涵盖的环境的一个重要部分。

在撰写本文时，ALE[3]提供了比GVGAI更高质量的游戏，因为它们是几十年前的家庭主机商业游戏，而GVGAI提供了结构化的API(通过Java对象，JSON接口或屏幕捕获获得的信息);代理在未见过的游戏上进行测试;而且有可能在⊠夜供应游戏。在GVGAI术语中，ALE只有两条轨道:单人学习和计划，其中学习轨道使用得更广泛。GVGAI框架有可能通过增加双人学习轨道来扩展，这将带来更多开放式挑战。这超出了当前ALE的范围。再次感谢VGDL，使用GVGAI学习环境或为这些游戏创建新关卡变得更加容易

GVGAI体育馆。它也很容易在现有的VGDL游戏及其关卡上自动生成变化。因此，用户可以应用程序内容生成来生成游戏和关卡变化，用于训练和测试他们的学习代理。GVGAI比ALE更容易扩展，并提供了一个解决方案。

3 GVGAI学习竞赛

在撰写本书时，只组织了两次GVGAI学习竞赛。在这一节中，除了2.1节中已经提出的两者的共同挑战外，我们描述了个人竞争的竞争规则和核心挑战。

3.1 使用GVGAI学习环境的竞赛

第一次GVGAI学习竞赛是在IEEE 2017年计算智能与游戏大会(IEEE CIG 2017)上组织的。

比赛程序和规则本次比赛使用了10个游戏，其中3个等级用于训练，2个私人等级用于测试。本次学习竞赛中使用的10个游戏的集合是2017 GVGAI单人游戏规划竞赛的训练集1。

学习阶段包括两个步骤，称为学习阶段1和学习阶段2。整个过程如算法1和图3所示。一个智能体在两个训练阶段总共有5分钟的有限时间(合法学习时长)。计时器不包括通信时间。如果5min已经用完，游戏的结果和观察结果仍然会发送给代理，代理在测试前的时间不会超过1秒。

图3:2017单人学习大赛中，一款游戏的学习和测试阶段。在比赛中，一个agent将在一组(通常为10个)未知游戏上执行，因此这个过程将重复10次。

在每个游戏的学习阶段1(算法1的第2-5行)，智能体按顺序完成三个训练关卡。在每一关结束时，无论游戏是否正常终止，或者智能体强制终止游戏(使用abort)，服务器都会在终止前将游戏结果(可能是un⊠)发送给智能体。

5 - GVGAI学习

Algorithm 1 Main procedure of the 2017 learning competition.

Require: $\mathcal{G}$ set of games
Require: $\mathcal{L}$ set of training levels (per game)
Require: $\mathcal{T}$ set of training levels (per game)
Require: π and agent

```
1: for each game G ∈ 𝒢 do
2:     for each game level ∈ G_ℒ do
3:         Let π play level once
4:         nextLevel ← π.result()
5:     while Time is not elapsed do
6:         Let π play nextLevel once
7:         nextLevel ← π.result()
8: for each game G ∈ 𝒢 do
9:     for each game level ∈ G_𝒯 do
10:        RES_G ← Results of 10 games of π on level
11: return RES_𝒢
```

在经历了⊠Learning phase 1之后，代理可以通过调用int result()方法(详见2.2节)自由选择下一个等级去玩(从三个训练等级中)。如果所选等级返回的索引不是有效索引，那么将从有效索引中传递一个随机索引，并开始一个新的游戏。重复这个步骤(学习阶段2)(算法1的第6-9行)，直到法定学习时间过期。

在循环整个游戏集后，测试阶段(算法1的第11-15行)开始。经过训练的代理依次重复玩10次私人测试关卡。没有更多的总时间限制，但代理仍然需要尊重init、act和result方法的时间限制，并可以在游戏过程中继续学习。

除了在GVGAI(2.1节)中学习的挑战之外，由于比赛规则，还有一些其他关键问题，例如如何选择下一步训练的级别;如何分配总学习时长;接收哪种类型的状态观察;Training di⊠erent models for di⊠erent games or Training one unique model;等。这些都不是微不足道的决定，没有一个参赛作品在比赛中表现合理。因此，2018年的比赛规则发生了变化。

在2017年版的GVGAI学习竞赛中，控制器的执行分为学习和验证两个阶段。在

学习阶段，每个控制器都有一个有限的时间，5分钟，用于学习□rst三个关卡的每个游戏。只要尊重5分钟的时间限制，代理可以想玩多少次就玩多少次，在这三个关卡中进行选择。在验证阶段，控制器按顺序玩10次4级和5级。在这些验证级别中获得的结果就是比赛中用来对参赛作品进行排名的结果。除了两个用Java和Python编写的样本随机代理和一个使用Java编写的Sarsa的样本代理外，第一个GVGAI单人学习赛道收到了三个用Java编写的提交和一个用Python编写的提交[11]。结果如表3所示。这条赛道的赢家是Q-Learning算法的一个简单实现(第4.5节)。

代理	训练集分数	排名	测试集分数	排名
kkunan sampleRandomt	125	6	184	1
DontUnderestimateUchiha	154	2	178	2
sampleLearnert	149	3	158	3
ercumentilhan	149	4	152	4
YOLOBOT	179	1	134	5
	132	5	112	6

表3:2017年GVGAI学习竞赛中提交agent的得分和排名。†表示一个样本控制器。

表4比较了单机规划和学习智能体在同一测试集上的最佳得分。请注意，由于在游戏本身发现的bug，测试集中的一个游戏被从□nal排名中删除。学习智能体在测试游戏上的最佳表现远不如规划智能体所能达到的效果。

3.2 GVGAI Gym比赛

GVGAI Gym已用于IEEE 2018年计算智能与游戏大会(IEEE CIG 2018)组织的学习竞赛。

比赛程序和规则根据第一届GVGAI学习比赛的经验对比赛规则进行了修改。主要程序如下。

1. 在提交截止日期前一个月，我们给了三款游戏，每个游戏都有两个公共关卡进行训练。参与者可以自由地私下训练他们的代理，没有任何限制，并且可以使用他们拥有的任何计算资源。

游戏	1-P计划最高分	1-P学习 最好的分数	代理
G2	109.00±38.19	31.5±14.65	sampleRandomt
G3	1.00 ±0.00	0±0	*
G4	1.00 ±0.00	0.2 ±0.09	库南
G5	216.00±24.00	1±0	*
G6	5.60±0.78	3.45±0.44	DontUnderestimateUchiha
七国集团(G7)	31696.10 ±6975.78	29371.95±2296.91	库南
八国集团	1116.90 ±660.84	35.15±8.48	库南
国	1.00±0.00	0.05±0.05	sampleRandomt
十国集团	56.70±25.23	2.75±2.04	sampleLearnert

表4:表比较了单机规划和学习智能体在同一测试集上的最佳得分。注意，由于游戏本身存在bug，测试集中有一款游戏被从⊠nal排名中删除。†表示一个控制器样本。

2. 在提交截止日期之后，三款游戏各自的三个新的私人关卡被用于验证。验证阶段由主办方使用一台笔记本电脑运行。每个实验都重复了10次。在验证期间，每个智能体每个游戏周期有100ms来选择一个动作。

为本次比赛设计了三个不同方面的测试游戏⊠。游戏1改编自《外星人》，唯一的区别是角色可以在四个方向射击和移动，而不是一个方向和两个方向。游戏2是一个名为“灯亮”的益智游戏，在这个游戏中，角色如果打开游戏屏幕右端的灯就赢了(例如，图4)。游戏3是由一个名为“欺骗币”的欺骗性游戏[1]改编而来的。

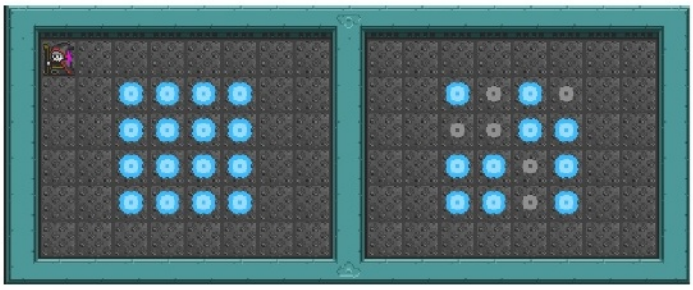

图4:第二届GVGAI学习比赛中使用的解谜游戏《灯亮》截图。截图说明了一个关卡的初始状态。

表5:2018年GVGAI学习竞赛中提交agent的得分和排名。† 表示一个样本控制器。

游戏	第一场比赛			第二场比赛		第三场比赛			排名
水平	3	4	5	34	5	3	4	5	
fraBot-RL-萨尔萨	-2	1	-1	00	0	2	3	**2**	1
fraBot-RL-QLearning	-2	-1	-200		0	1	02		2
随机	-0.5	0.2-0.1	0 0 0			3.5	0.7	2.7	3
的	61.5	-1	0.300	0		-	-	-	-
优先决斗DQNt A2Ct	36.8	-1	-200	0		-	-	-	-
	8.1	-1	-200		0	-	-	-	-
OLETS规划代理	41.7	48.6	3.1	00	2.2	4.2	8.1	14	-

竞争的挑战新的竞争规则和更加用户友好的框架使用户更加关注学习算法，包括那些已经与Open AI Gym兼容的算法。用户可以自由地私下训练他们的代理，使用尽可能多的学习时间和计算资源，这使得更多的机会和可能性成为可能。然而，一些经典的学习算法无法处理不同游戏所具有的di 不同维度的屏幕。

本次竞赛仅收到来自法兰克福应用科学大学的同一组投稿者提交的fraBot-RL-QLearning和fraBot-RL-Sarsa两个参赛作品。条目和样本代理(random, DQN, 优先决斗DQN和A2C[15])的结果汇总在表5中。为了进行比较，包括了规划代理OLETS(可以访问前向模型)。DQN和优先决斗DQN在游戏1的第3个关卡(测试关卡)中表现突出，因为第3个关卡与第2个关卡(训练关卡)非常相似。有趣的是，样本学习代理DQN在游戏1的第三个关卡上的表现优于OLETS。例如，基线代理DQN、优先决斗DQN和A2C由于不同关卡的游戏屏幕尺寸不同而无法学习欺骗币游戏，尽管它们在由外星人改装的游戏测试关卡中表现出色。

4 竞争条目

本节介绍 rst解决2017年和2018年比赛中单人学习轨道中设置的挑战的方法，然后转向其他方法。

4.1 随机代理(样本代理)

一个样本随机代理，它在每次游戏tick时均匀随机地选择一个动作，被包含在框架中(Java和Python都有)，以用于测试。这个智能体也意味着被作为一个基线:一个学习者被期望比一个随机行动且不进行任何学习的智能体表现得更好。

4.2 DRL算法(样本智能体)

Torrado等人使用新的GVGAI Gym[15]比较了OpenAI Gym的三种实现的深度强化学习算法，Deep Q-Network (DQN)、优先决斗DQN和Advance Actor-Critic (A2C)，在8种具有不同难度和游戏规则的GVGAI游戏上。所有三个RL代理在大多数游戏中都表现良好，然而，dqn和A2C在游戏过程中没有给出游戏分数(游戏邦注:只有在游戏结束时才会给出输赢)时表现不佳。在IEEE CIG 2018组织的学习竞赛中，这三个智能体被用作样本智能体。

4.3 Multi-armed bandit算法

K. Kunanusont的DontUnderestimateUchiha基于两种流行的多武装强盗(MAB)算法,即递减贪婪算法和上置信限(UCB)。在任意游戏tick T时，选择概率为$1-\boxtimes_T$的当前最佳动作，否则均匀随机选择一个动作。在时刻T的最佳行动是使用UCB以分数增量作为奖励来确定的。这是一个非常有趣的组合，因为ucb风格的选择和递减贪婪算法都旨在平衡利用更多最佳到目前为止的行动和探索其他行动之间的权衡。此外，$\boxtimes_0$被设置为0.5，它随着时间缓慢减少，形式化为$\boxtimes_T=\boxtimes_0-0.0001T$。根据比赛设定，所有游戏的持续时间都将超过2 000个游戏tick，因此$\forall T\in\{1,\cdots,2000\}$，$0.5\geqslant\boxtimes_T\geqslant0.3$。因此，随机决策的时间大约为40%。

4.4 撒尔沙

sampleLearner、ercumentilhan和frabet - rl -Sarsa是基于Sarsa (State-Action-Reward-State-Action)算法[13]。sampleLearner和ercumentilhan可以使用整个游戏状态信息的子集来构建新状态，以减少需要保存的信息量，并考虑到类似的情况。主要的di⊠erence是前者使用一个以⊠固定大小为中心的正方形区域

化身的位置，而后者使用⊠固定距离的⊠rst-person视图。frabet - rl -Sarsa使用Sarsa，它使用GVGAI Gym提供的游戏画面的整个截图作为输入。每个游戏的每个级别使用1000集来训练agent，总训练时间为48小时。

4.5 Q-learning

kkunan，由K. Kunanusont设计，是一个简单的Q-learning智能体[13]，它使用角色的大部分当前信息作为特征，只有少数例外(如角色的生命值和屏幕大小，因为这些元素在不同的游戏中差异很大)。游戏tick t + 1的奖励是de⊠ned作为在t + 1的分数和在t的分数之间的di⊠erence。学习率α和折现因子γ被手动设置为0.05和0.8。在学习阶段，以概率⊠= 0.1执行一个随机动作，否则，选择最佳动作。在验证阶段，总是选择最佳动作。虽然很简单，但它还是在2017年赢得了the⊠rst赛道。fraBot-RL-QLearning使用Q-Learning算法。它在每一场比赛的每一关都使用1000集进行训练，总训练时间为48小时。

4.6 树搜索方法

YOLOBOT是对YOLOBOT规划代理(如前面第4章所述)的一种改进。由于FM在学习轨道上不再可访问，MCTS被贪心算法取代，以选择最多使所选对象距离最小的动作。根据作者的说法，YOLOBOT在学习轨道上的表现不佳，而不是在规划轨道上的成功，这是由于他们自己创建的碰撞模型没有很好地工作。

4.7 其他学习智能体

使用此框架作为学习环境的⊠rst工作之一是由Samothrakis等人[14]进行的，他们在基准的10个游戏中使用了神经进化。具体来说，作者使用两个di⊠不同的策略(⊠-greedy vs Softmax)和一个线性函数逼近器vs一个作为状态评估函数的神经网络，对可分离的自然进化策略(S-NES)进行了实验。分数、游戏状态、头像和其他精灵信息等特征被用于进化1000集的学习者。结果显示，⊠-greedy与线性函数近似器是学习如何最大化每个游戏分数的更好组合。

Braylan和Miikkulainen[4]进行了一项研究，其目标是在30场比赛中学习一个前向模型。目标是从。学习下一个状态

当前一加一个动作，其中状态是de⊠ned作为精灵属性值的集合(刷出，方向，移动等)，通过逻辑回归的方式。此外，作者将学习到的对象模型从一个游戏转移到另一个游戏，前提是假设许多机制和行为在它们之间是可转移的。实验表明，对象模型迁移在学习前向模型的准确性方面的实际价值，导致这些智能体在探索方面更强。

同样在学习设置中，Kunanusont等人[9][10]开发了能够通过屏幕捕获玩几个游戏的代理。特别地，作者在7个日益复杂的游戏框架中使用了Deep Q-Network，并对GVGAI进行了一些增强，以处理不同的屏幕尺寸和非可视化游戏模式。结果表明，该方法允许代理学习如何在确定性和随机游戏中玩，随着剧集数量的增加，实现更高的胜率和游戏分数。

Apeldoorn和Kern-Isberner[2]提出了一种学习智能体，通过使用混合符号/子符号智能体模型，在未知环境中快速确定和利用启发式。所提出的基于agent的模型使用子符号学习方法学习加权状态-动作对。该智能体已经在GV-GAI框架的单人随机游戏Camel Race上进行了测试，并且在剩下的100个游戏周期内，在不同的关卡中赢得了一半以上的游戏，而标准Q-Learning智能体在相同的游戏长度下从未获胜。Dockhorn和Apeldoorn[6]在[2]的基础上，使用容错层次知识基础(HKBs)来学习近似正演模型，并在2017 GVGAI学习轨迹框架上测试该方法，并尊重竞争规则。提出的智能体击败了CIG 2017[6]组织的学习竞赛中的最佳参赛选手，但表现仍然远逊于最好的规划智能体，这些智能体可以获得真实的前向模型。

最后，Justesen等人[8]在训练环境中实现了gvgae - gym接口中的A2C，该环境允许通过程序生成新关卡进行学习。通过改变智能体所玩的级别，由此产生的学习更加通用，不会超过⊠t到特定⊠c级别。关卡生成器在每一集创建关卡，根据观察到的智能体表现，以di⊠culty缓慢增加的水平产生关卡。

4.8 讨论

所呈现的代理di⊠er彼此之间在输入游戏状态(JSON字符串或屏幕截图)，学习时间的多少，所使用的算法。此外，一些代理已经在di⊠erent游戏集上进行了测试，有时使用di⊠erent游戏长度(即允许的最大游戏滴答数)。没有一个

提交给2017年学习竞赛的代理使用了经典的GVGAI框架，并使用了屏幕捕获。

基于sarsa的智能体在比赛中的表现令人惊讶地糟糕，可能是由于任意选择的参数和非常短的学习时间。此外，学习三个等级并在两个更多的di□cult等级上进行测试，只给了5分钟的学习时间，这是一个di□cult任务。一个agent应该负责学习预算的分配，并决定何时停止学习一个关卡并继续下一个关卡。

使用异常容忍HKBs[6]的学习代理学习速度快。然而，当允许更长的学习时间时，则由深度强化学习(DRL)代理主导。在Torrado等人[15]测试的8个游戏中，三种DRL算法在6个游戏中都没有优于计划代理。然而，在高度随机的Seaquest游戏中，A2C的得分几乎是最佳规划代理MCTS的两倍。

5 总结

在本章中，我们提出了两个用于GVGAI学习挑战的平台，它们可用于测试强化学习算法，以及一些基线代理。本章还回顾了使用每个平台组织的2017年和2018年GVGAI学习竞赛。由于VGDL的使用，这些平台具有由人类和AIs设计用于训练强化学习智能体的新游戏的潜力。特别是，GVGAI Gym易于实现和比较代理。我们相信这个平台可以用于多个研究方向，包括为一个特定的□c任务设计强化学习代理，调查arti□社会通用智能，以及评估di□不同的算法如何学习和进化来理解各种变化的环境。

参考文献

1. D. Anderson, M. Stephenson, J. Togelius, C. Salge, J. Levine, and J. Renz, □Deceptive Games,□ arXiv preprint arXiv:1802.00048, 2018.
2. D. Apeldoorn and G. Kern-Isberner, □An Agent-Based Learning Approach for Finding and Exploiting Heuristics in Unknown Environments,□ in COMMONSENSE, 2017.
3. M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, □The arcade learning environment: an evalua-tion platform for general agents,□ Journal of Arti□cial Intelligence Research, vol. 47, no. 1, pp. 253–279, 2013.
4. A. Braylan and R. Miikkulainen, □Object-Model Transfer in the General Video Game Domain,□ in Twelfth Arti□ cial Intelligence and Interactive Digital Entertainment Conference, 2016.
5. G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, □Openai Gym,□ arXiv preprint arXiv:1606.01540, 2016.

6. A. Dockhorn and D. Apeldoorn, ☒Forward Model Approximation for General Video Game Learning,☒ in Computational Intelligence and Games (CIG), IEEE Conference on, 2018.

7. M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, ☒The Malmo Platform for Arti☒cial Intelligence Experimentation,☒ in IJCAI, 2016, pp. 4246–4247.

8. N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, ☒Illuminating Gener-alization in Deep Reinforcement Learning through Procedural Level Generation,☒ arXiv:1806.10729, 2018.

9. K. Kunanusont, ☒General Video Game Arti☒cial Intelligence: Learning from Screen Capture,☒ Master' s thesis, University of Essex, 2016.

10. K. Kunanusont, S. M. Lucas, and D. P´erez-Li´ebana, ☒General Video Game AI: Learning from Screen Capture,☒ in 2017 IEEE Conference on Evolutionary Computation (CEC). IEEE, 2017.

11. J. Liu, ☒GVGAI Single-Player Learning Competition at IEEE CIG17,☒ 2017. [Online]. Available: https://www.slideshare.net/ljialin126/gvgai-singleplayer-learning-competition-at-ieee-cig17

12. J. Liu, D. Perez-Liebana, and S. M. Lucas, ☒The Single-Player GVGAI Learning Framework - Technical Manual,☒ 2017. [Online]. Available: http://www.liujialin.tech/publications/GVGAISingleLearningmanual.pdf

13. S. J. Russell and P. Norvig, Arti☒cial Intelligence: a Modern Approach. Malaysia; Pearson Education Limited,, 2016.

14. S. Samothrakis, D. Perez-Liebana, S. M. Lucas, and M. Fasli, ☒Neuroevolution for General Video Game Playing,☒ in Conference on Computational Intelligence and Games (CIG). IEEE, 2015, pp. 200–207.

15. R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, ☒Deep Reinforcement Learning in the General Video Game AI framework,☒ in IEEE Conference on Computational Intelligence and Games (CIG), 2018.