**标准化和构建实验数据库**

将 RLA 集成到您的项目中后，我们将创建一个由隐式配置的"数据库"。我们运行的每个实验都将被索引并作为项目存储到"表"中。特别是，RLA包括以下elelment来构建"RL实验数据库"。rla\_config.yaml

1、**数据库：**数据库由 YAML 文件 配置。在我们的实践中，我们只在一个研究主题中创建一个数据库。rla\_config.yaml

2、**表：**我们在研究过程中将标准数据库系统中的Table概念映射到"任务"的概念中。这两个概念之间有许多相似之处。例如，在以下情况下，我们将经常创建另一个表/任务：

2.1表的结构是不同的，例如，它们具有不同的键。在研究过程中，不同的任务通常有完全不同类型的日志要记录。例如，在基于脱机模型的 RL 中，第一个任务可能预训练动态模型，第二个任务可能是使用学习的模型进行策略学习。在模型学习中，我们关注的是模型的MSE;在政策学习中，我们关心的是政策的回报。

2.2表的内容太大，可能会损害查询速度。表分区是一种常见的解决方案。在研究过程中，如果logdir有很多次运行，我们需要大内存来加载Tensorboard。我们可以通过将运行拆分到不同的子目录中来解决此问题。

在 RLA 中，我们需要为每个要运行的实验分配一个"任务"。

3、**数据项：**我们将数据项的概念映射到实验中生成的复杂数据。对于每个数据项，我们需要为每个项定义索引和值。

3.1**索引：**我们需要一个唯一索引来定义表中的项目，以便添加、删除、修改和查询项目。在 RLA 中，我们将每个实验的索引定义为：。datetime of the experiment (for uniqueness) + ip address (for keeping uniqueness in distributed training) + tracked hyper-parameters (easy to identify and easy to search)

3.2**价值：**在运行实验时，我们会生成许多具有不同结构的数据。基于我们的研究实践，目前，我们制定并存储以下数据

a.代码和超参数：每行代码并选择超参数来运行实验。这是实验可重复性的备份。

b.记录变量：我们经常在实验过程中记录许多中间变量，例如奖励，一些损失或学习率。我们将变量记录在键值公式中，并将它们存储在张量板事件和 CSV 文件中。

c.模型检查点：我们支持在Tensorflow和Pytorch框架中保存神经网络和相关自定义变量的权重。我们可以使用检查点来恢复实验，或者使用实验结果来完成下游任务。

d.其他数据，如数字或视频：它本质上是实验过程中的非结构化中间变量。我们可能会绘制您的代理行为的帧到帧视频或一些曲线来检查训练过程。我们在 RL 方案中提供了一些常用工具来生成相关变量并将其存储在目录中。

目前，我们将数据项存储在标准文件系统中，并通过预定义的目录结构管理数据项、表和数据库之间的关系。运行一些实验后，数据库将如下所示：免疫接种.png（插入对应图片）

在这里，我们在"sb\_ppo\_example"项目中构建一个数据库。

目录"archive\_tester"用于存储超参数和相关变量，以便恢复实验。

目录"结果"用于存储其他数据，如数字或视频。

目录"log"用于存储记录的变量。

目录"code"是实验可重复性的代码备份。

目录"检查点"保存神经网络的权重。

我们有一个名为"demo\_task"的表，它是log/archive\_tester/checkpoint/code/results的根目录。

实验的"指数"在表述中命名。${%Y}/${%m}/${%d}/${%H-%M-%S-%f} ${ip address} ${tracked hyper-parameters}

**用于管理数据库的工具**

在标准数据库系统中，管理数据库的常用操作是添加、删除修改和查询。我们还提供了类似的工具来管理RLA数据库。

添加：

1. RLA.easy\_log.tester.exp\_manager 是一个全局对象，用于创建实验并在实验过程中管理数据。
2. RLA.easy\_log.logger 是一个用于添加记录变量的模块。
3. RLA.easy\_log.simple\_mat\_plot是一个用于构建其他数据（如数字或视频）的模块

删除：

1. rla\_scripts.delete\_log：通过正则表达式删除数据项的工具;

修改：

1. resume： RLA.easy\_log.exp\_loader.ExperimentLoader：一个类，用于恢复具有不同灵活设置的实验。

查询：

1. 张量板：记录的变量被添加到张量板事件中，可以通过标准张量板工具加载。免疫接种.png
2. easy\_plot：我们提供了一些 API 来加载和可视化 CSV 文件中的数据。结果将如下所示：

**其他原则**

第二种设计原则易于集成。要实现这一目标，还有很长的路要走。我们在目录示例中给出了几个与 RLA 集成的示例项目。

1.基于stable\_baselines（张量流）的 RLA 的 PPO：示例/sb\_ppo\_example

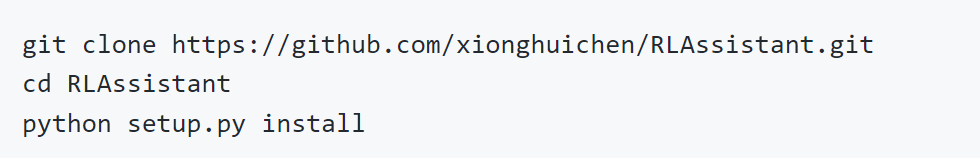
2.基于stable\_baselines3（pytorch）的带有RL的PPO：示例/sb3\_ppo\_example

我们还列出了使用RLA的RL研究项目，如下所示：

1.槭

2.科达斯

**安装**



**工作流程**

我们构建了一个用于集成 RLA 的示例项目，这可以在 ./example/simplest\_code 中看到。现在，我们总结了使用它的步骤。

**步骤1：配置。**

1. 我们在 中定义数据库的属性。您可以基于 ./example/simplest\_code/rla\_config.yaml 中的模板构造 YAML 文件。rla\_config.yaml

2. 我们在 exp\_manager.config 中定义表的属性。在开始实验之前，应按如下所示配置全局对象 RLA.easy\_log.tester.exp\_manager。

from RLA.easy\_log.tester import exp\_manager

kwargs = {'env\_id': 'Hopper-v2', 'lr': 1e-3}

exp\_manager.set\_hyper\_param(\*\*kwargs) # kwargs are the hyper-parameters for your experiment

exp\_manager.add\_record\_param(["env\_id"]) # add parts of hyper-parameters to name the index of data items for better readability.

task\_name = 'demo\_task' # define your task

rla\_data\_root = '../' # the place to store the data items.

exp\_manager.configure(task\_name, private\_config\_path='../../../rla\_config.yaml', data\_root=rla\_data\_root)

exp\_manager.log\_files\_gen() # initialize the data items.

exp\_manager.print\_args()

以上代码截图不全，用图片代替

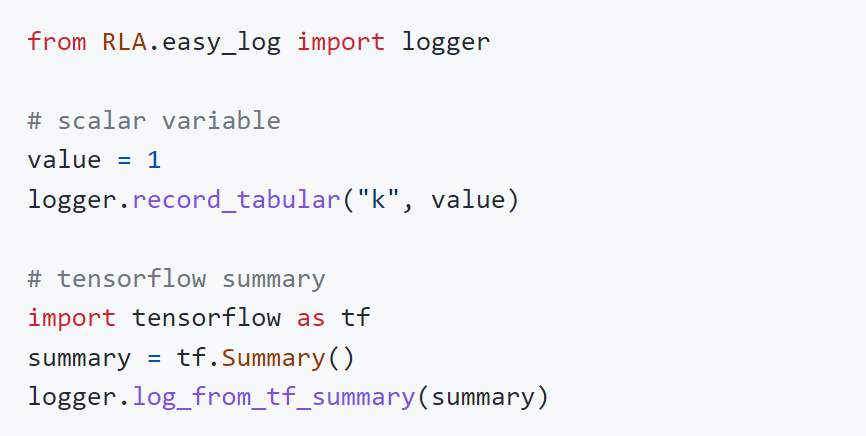
3. 我们将生成的数据项添加到 .gitignore 中，以避免将它们推送到我们的 git 存储库中。



**步骤2：记录中间变量/检查点/其他类型的数据。**

**记录中间标量**

我们通过以下方式记录标量：RLA.easy\_log.logger



**记录检查点**

我们通过保存神经网络的检查点。exp\_manager.save\_checkpoint

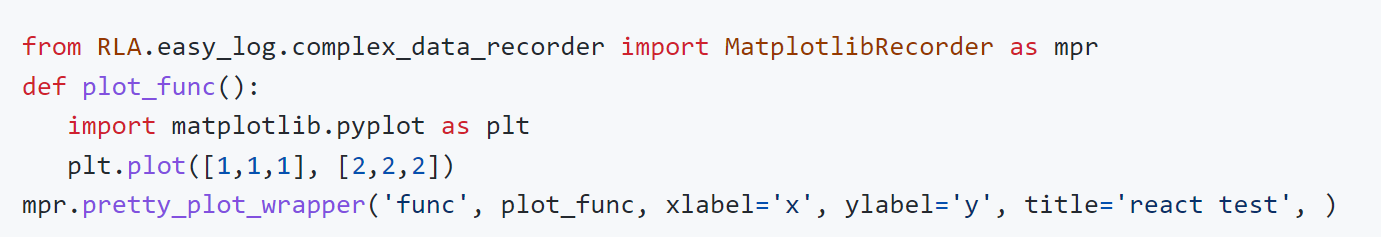


**记录其他类型的数据[正在开发中]**

目前我们可以基于张量板记录复杂结构数据：



我们将开发API来记录RLA.easy\_log.complex\_data\_recorder中常用的复杂结构数据。现在我们给出了一个MatplotlibRecorder工具来管理matplotlib生成的数字：



plot\_func绘制的数字将保存在"结果"目录中。

**步骤3：处理您的历史实验。**

处理实验的方法可分为以下模块：

**查询**

目前，我们基于两种常见场景开发查询工具：结果可视化和实验审查。

结果可视化：

1. Tensorboard：我们可以使用tensorboard/tensorboardX来查看记录的日志。张量板事件将保存在 中。我们可以通过以下方式查看张量板中的结果：。例如，lanuch tensorboard by .我们可以看到结果：${data\_root}/log/${task\_name}/${index\_name}/tb/eventstensorboard --logdir ${data\_root}/log/${task\_name}tensorboard --logdir ./example/simplest\_code/log/demo\_task/2022/03免疫接种.png

2. Easy\_plot工具包：中间标量变量保存在 中的 CSV 文件中。我们开发了高级 API 来加载来自多个实验的 CSV 文件，并按自定义键对行进行分组。我们在 example/plot\_res.ipynb 中给出了一个使用工具包easy\_plot示例。结果将是这样的：${data\_root}/log/${task\_name}/${index\_name}/progress.csv免疫接种.png

3.直接查看"结果"目录中的数据：其他类型的数据存储在${data\_root}/results/${task\_name}/${index\_name} 免疫接种.png

实验回顾：

1. 给定任何名为 的实验，我们可以在 中找到逐行代码。我们可以将文件配置为存储在rla\_config.yaml中的BACKUP\_CONFIG中。${task\_name}/${index\_name}${data\_root}/code/${task\_name}/${index\_name}
2. 相应的超参数（由 记录）可以在 中找到。exp\_manager.set\_hyper\_param${data\_root}/log/${task\_name}/${index\_name}/backup.txt

**修改**

通常，无需更改试验日志的内容。在我们的实践中，一个常见的场景是加载历史实验/结果以恢复训练或提供给下游任务。在RLA中，我们开发了一种工具，用于实验加载的不同要求，该工具位于.它可以很容易地用于：RLA.easy\_log.exp\_loader

1.为另一项任务加载预训练模型（例如，验证）;

2.恢复实验;

3.使用其他设置恢复实验。

**浴室管理**（翻译有误）

我们通过rla\_scripts工具包管理数据库中的项目。目前，rla\_scripts包括

1.存档：将一些重要结果存档到另一个表中。

2.删除：一次删除所有无用的实验。

3.发送到远程 [待办事项]

4.从远程下载 [待办事项]

在将rla\_scripts复制到我们的研究项目并修改DATA\_ROOT config.py 以找到RLA数据库的根目录后，我们可以使用上述工具。