

**基于PyTorch的前沿深度学习算法集成应用程序接口**

**测试需求说明书**

北京航空航天大学

2020-05

版本变更历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **版本** | **提交日期** | **主要编制人** | **审核人** | **版本说明** |
| 1.0 | 2020/5/14 | 全体成员 | 张崇智、高明骏 | 该文档的最初版本 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**注：全体成员包括张崇智、秦浩桐、黄涵、王茵迪、赵永驰、吴振赫、高明骏**

目 录

[1. 引言 5](#_Toc40433482)

[1.1 编写目的 5](#_Toc40433483)

[1.2 测试内容 5](#_Toc40433484)

[1.3 参考文献 5](#_Toc40433485)

[2. 测试计划 6](#_Toc40433486)

[2.1 测试目标 6](#_Toc40433487)

[2.2 测试工具及技术 7](#_Toc40433488)

[2.2.1 Python 7](#_Toc40433489)

[2.2.2 测试技术及应用 7](#_Toc40433490)

[2.3 测试用例设计要求及通过准则 8](#_Toc40433491)

[3. 运行配置测试 9](#_Toc40433492)

[3.1 概述 9](#_Toc40433493)

[3.2 测试用例 9](#_Toc40433494)

[4. 功能测试 11](#_Toc40433495)

[4.1 概述 11](#_Toc40433496)

[4.2 对抗样本生成模块 11](#_Toc40433497)

[4.2.1 FGSM调用运行 11](#_Toc40433498)

[4.2.2 PGD调用运行 13](#_Toc40433499)

[4.2.3 STEP-LL调用运行 14](#_Toc40433500)

[4.2.4 MI-FGSM调用运行 15](#_Toc40433501)

[4.2.5 C&W调用运行 16](#_Toc40433502)

[4.2.6 Fourier Corruption调用运行 17](#_Toc40433503)

[4.3 神经网络量化模块 18](#_Toc40433504)

[4.3.1 BNN调用运行 18](#_Toc40433505)

[4.3.2 XNOR调用运行 19](#_Toc40433506)

[4.3.3 HORQ调用运行 20](#_Toc40433507)

[4.3.4 BWN调用运行 21](#_Toc40433508)

[4.3.5 IR-Net调用运行 22](#_Toc40433509)

[4.4 目标检测模块 24](#_Toc40433510)

[4.4.1 Faster RCNN调用运行 24](#_Toc40433511)

[4.5 阅读理解模块 25](#_Toc40433512)

[4.5.1 RC调用运行 25](#_Toc40433513)

[4.6 主动学习模块 26](#_Toc40433514)

[4.6.1 BanditDiscreteSampler调用运行 26](#_Toc40433515)

[4.6.2 GraphDensitySampler调用运行 27](#_Toc40433516)

[4.6.3 HierarchicalClusterAL调用运行 28](#_Toc40433517)

[4.6.4 InformativeClusterDiverseSampler调用运行 29](#_Toc40433518)

[4.6.5 kCenterGreedy调用运行 31](#_Toc40433519)

[4.6.6 MarginAL调用运行 32](#_Toc40433520)

[4.6.7 MixtureOfSamplers调用运行 33](#_Toc40433521)

[4.6.8 RepresentativeClusterMeanSampling调用运行 34](#_Toc40433522)

[4.6.9 UniformSampling调用运行 35](#_Toc40433523)

[5. 非功能测试 36](#_Toc40433524)

[5.1 高效性 36](#_Toc40433525)

[5.1.1 测试策略描述 36](#_Toc40433526)

[5.1.2 测试用例描述 36](#_Toc40433527)

[5.2 用户友好性 37](#_Toc40433528)

[5.2.1 测试策略描述 37](#_Toc40433529)

[5.2.2 测试用例描述 37](#_Toc40433530)

[5.3 健壮性 38](#_Toc40433531)

[5.3.1 测试策略描述 38](#_Toc40433532)

[5.3.2 测试用例描述 38](#_Toc40433533)

[5.4 扩展性 39](#_Toc40433534)

[5.4.1 测试策略描述 39](#_Toc40433535)

[6. 总结 39](#_Toc40433536)

[7. 附录 40](#_Toc40433537)

# 引言

## 编写目的

软件需求是度量软件质量的基础，软件测试则是保证软件质量和可靠性的关键技术手段。由于修正一个软件错误所需的费用将随着软件生存期的进展而上升，错误发现得越早，修正它所需的费用就越少，而据统计60%以上的错误来自设计以前，所以应尽早地和不断地进行软件测试。

本文档就是在软件工程实验课程中对所选软件的功能需求的测试过程说明文档，主要描述了测试的目的、测试的计划、测试的方法、对测试需求的分析以及具体的测试用例说明等。测试人员可根据本文档中的测试计划和方法结合测试用例，对软件功能需求、非功能需求以及环境需求进行测试。

## 测试内容

小组成员在讨论后，决定依据需求分析阶段编写的《<基于PyTorch的前沿深度学习算法集成应用程序接口>需求规格说明书V5.0》对项目的功能需求、非功能需求和环境需求等进行测试。

根据软件测试规范，可以将测试过程分为以下几个过程：拟定软件测试计划；编制软件测试说明书；设计和生成测试用例；实施测试；生成软件测试报告；软件问题报告；测试结果报告；对整个测试过程进行有效的管理。

依据上述过程和项目的设计文档《<基于PyTorch的前沿深度学习算法集成应用程序接口>设计说明书V1.0》实现的内容，结合需求分析阶段的需求用例，逐一分析测试需求，利用相关测试技术，构建覆盖尽可能完全的测试用例，并完成测试任务，产出测试说明书和测试报告。

## 参考文献

[1] PyTorch官方文档[OL]<https://PyTorch.org/>

[2] 《<基于PyTorch的前沿深度学习算法集成应用程序接口>设计说明书V1.0》[3] 《<基于PyTorch的前沿深度学习算法集成应用程序接口>需求规格说明书V5.0》

# 测试计划

## 测试目标

本项目中系统的主要组成部分有两个：一个是所选择的软件PyTorch，这部分软件代码是开源的；另一个是根据实际需求，基于PyTorch框架的情况下，将一些深度学习领域的经典或优秀算法封装成API，这部分的代码全部由成员来完成。本次实验主要对后者进行测试。

在本项目中所选的PyTorch，是Torch的Python版本，是由Facebook开源的神经网络框架，专门针对 GPU 加速的深度神经网络（DNNs）编程。对于这个开源软件而言，主要针对其功能是否完整进行测试，即是否能在操作系统上进行部署，是否能使用其丰富并有效的工具包搭建神经网络，以及是否具有扩展并导入新包的能力。

开发部分的测试主要是验证软件的功能和性能及其他特性是否与需求说明书一致，主要从以下几个方面进行测试：

1. 针对需求规格说明书的需求用例进行尽可能全面的测试，即确定项目是否完全的实现了《需求文档》中的功能点。
2. 测试算法集成后是否能在PyTorch环境中，即确定集成后的PyTorch在Linux或Windows中是否能够被正常编译、安装；并测试各种算法是否有效，使用工具包搭建神经网络所需的功能实现的是否完整、稳定、可用。
3. 测试非功能需求和环境需求，即确定系统在一定条件下，在搭建神经网络的过程中，是否具有高效性、用户友好性、可扩展性等特性。
4. 在后续实验中，在不同数据集上，对比使用本项目提供的API和其他方法的优劣性。

## 测试工具及技术

本小节主要介绍在实际的测试过程中用到的语言，使用的测试技术以及应遵循的原则。

### 2.2.1 Python

Python 是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。PyTorch 则深入构建在 Python 之上，而不是简单地在整体 C++框架上绑定 Python。用户可以像使用 numpy、scipy 、 scikit-learn 那样轻松地使用 PyTorch，也可以用喜欢的库和包（如 Cython 和 Numba）在 Python 中编写新的神经网络层。从最新版本PyTorch1.5.0开始，不再支持Python 2，特别是2.7版。今后，PyTorch对Python的支持将仅限于Python 3，特别是Python 3.5、3.6、3.7和3.8（在PyTorch 1.4.0中首次启用）。

### 2.2.2 测试技术及应用

本项目中，成员分为CV和NLP两小组进行模块开发，所以在开发成员后，小组内部成员之间进行人工审查。代码审查即不执行程序代码，通过审查文档、代码的方式查找软件中的缺陷。人工走查与代码审查一样，首先通过资料研究程序，但不同的是在人工走查会上通过测试数据，人工运行程序来达到测试的目的。

代码审查是一种静态白盒测试方法，是早期发现软件缺陷最有效的方法。同时我们要求在开发过程中，利用程序插装技术，在关键代码段插入输出语句，以监控程序的运行状态。程序插装(Program Instrumentation)是实现白盒测试最基本的测试手段，借助于插装手段，能了解程序执行时的结构覆盖情况，如语句覆盖、分支覆盖及路径覆盖等信息。

在本次实验中，主要测试部分是依据需求规格说明书对功能需求进行黑盒测试。黑盒测试能站在用户立场上，从用户观点出发进行测试，又称功能测试、数据驱动测试或基于规格说明的测试。 测试者只需了解程序输入和输出之间的关系，或是程序的功能，完全依靠能够反映这一关系和程序功能的需求规格说明确定测试数据，判定测试结果的正确性。

除了对基本功能进行测试外，还需要对模块进行错误处理测试，判断出错的描述是否难以理解，是否能够对错误定位，显示的错误与实际的错误是否相符，对错误条件的处理是否正确等。

在本项目中，由多个模块集成一个高效API工具包，所以除对每个模块进行单元测试外，还应进行模块接口测试，即输入参数的数目、类型是否与模块形参数目相同；用于输入的变量有没有改变；在经过不同模块时，全局变量的定义是否一致;；在使用外部资源时，是否检查可用性关及时释放，如内存、文件、端口等上层模块和底层模块之间等。

之后对集成后的包进行初步的集成测试和特性测试。集成测试应考虑：模块之间是否相互影响，即一个单元模块的功能是否会对另一个单元模块的功能产生不利的影响；整个系统功能问题，即各个功能组合起来，是否能达至达预期的功能；全局数据结构是否有问题等。特性测试即测试系统在一定条件下，在搭建神经网络的过程中，是否具有高效性、用户友好性、可扩展性等特性。

同时还需要进行配置测试和回归测试。配置测试即测试项目在不同的软件和硬件配置中的运行情况，同时核实软件在配置后可立即正常运行，这通常是指运行大量为功能测试制定的测试。回归测试是指在修改了软件代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。

## 测试用例设计要求及通过准则

测试用例应具有代表性，能够代表并覆盖合理的和不合理、合法的和非法的、边界的和越界的，以及极限的输入数据、操作和环境设置等。测试结果应具有可判定性，即测试执行结果的正确性是可判定的，每一个测试用例都应有相应的期望结果。并且测试结果还应具有可再现性，即对同样的测试用例，系统的执行结果应当是相同的。

基于需求规格说明书和以上原则，在本项目的测试用例设计中，应符合以下要求：

1. 功能需求部分，每个需求至少应构建两个测试用例，分别是能够正确执行对应功能和能够触发捕捉错误功能的用例。
2. 对于环境测试，应至少有两个测试用例，分别是能在Linux和Windows平台上配置成功运行的用例。
3. 对于非功能需求中的高效性，应设计至少一个测试用例，使系统能够多线程、多GPU并行运行。
4. 对于非功能需求中的用户友好性，应设计至少一个测试用例，保证研究人员能够快速上手。

测试完成标准有很多，如常见的有：测试超过了预定的时间表则停止测试;执行了所有测试情况，但没有发现错误，则停止测试。对于本实验测试用例通过的准则，主要依据测试用例执行者测试结果与预期结果相符，或测试结果与预期结果虽有不符但不可归咎于PyTorch本身时为测试通过，反之测试失败。由于本项目大多算法输出结果不会完全一致，所以主要关注功能流和异常流是否执行正确，以及训练或预测结果在一定范围内是否一致。

# 运行配置测试

## 概述

测试集成本项目包后的PyTorch能够在多种操作系统（Linux/Windows）中被正常编译、安装。

## 测试用例

表3-1 TC401-Linux下配置测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Linux下配置测试 | **测试用例标识** | TC401 | **对应测试需求文档中的需求** | 4节的环境需求  TR401 | |
| **简要描述** | 本测试主要测试PyTorch在Linux操作系统中能否被正常编译、安装。 | | | | | |
| **前提和约束** | None | | | | | |
| **测试方法** | 黑盒测试 | | | | | |
| **测试过程描述** | | | | | | |
| **序号** | **测试步骤** | | **预期结果** | | **评价准则** | **测试**  **结论** |
| 1 | 安装git | | 安装成功 | | 实际结果与预期结果一致 |  |
| 2 | 安装PyTorch，配置安装组件 | | 安装成功 | | 实际结果与预期结果一致 |  |
| 3 | 启动PyTorch | | 启动成功 | | 实际结果与预期结果一致 |  |
| 4 | 导入项目包 | | 导入成功 | | 实际结果与预期结果一致 |  |

表3-2 TC402-Windows下配置测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Windows下配置测试 | **测试用例标识** | TC402 | **对应测试需求文档中的需求** | 4节的环境需求  TR401 | |
| **简要描述** | 本测试主要测试Torch在Windows操作系统中能否被正常编译、安装。 | | | | | |
| **前提和约束** | None | | | | | |
| **测试方法** | 黑盒测试 | | | | | |
| **测试过程描述** | | | | | | |
| **序号** | **测试步骤** | | **预期结果** | | **评价准则** | **测试结论** |
| 1 | 安装git | | 安装成功 | | 实际结果与预期结果一致 |  |
| 2 | 安装PyTorch，配置安装组件 | | 安装成功 | | 实际结果与预期结果一致 |  |
| 3 | 启动PyTorch | | 启动成功 | | 实际结果与预期结果一致 |  |
| 4 | 导入项目包 | | 导入成功 | | 实际结果与预期结果一致 |  |

# 功能测试

## 概述

功能测试就是对产品的各功能进行验证，根据功能测试用例，逐项测试，检查产品是否达到用户要求的功能

Functional testing（功能测试），也称为behavioral testing（行为测试），根据产品特性、操作描述和用户方案，测试一个产品的特性和可操作行为以确定它们满足设计需求。

功能测试也叫黑盒测试或数据驱动测试，只需考虑需要测试的各个功能，不需要考虑整个软件的内部结构及代码.一般从软件产品的界面、架构出发，按照需求编写出来的测试用例，输入数据在预期结果和实际结果之间进行评测，进而提出更加使产品达到用户使用的要求。

## 对抗样本生成模块

### FGSM调用运行

FGSM调用运行测试用例如表4-1所示。

表4-1 TC301-FGSM调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | FGSM调用运行 | **测试用例标识** | 用例TC301 | **测试需求标识** | TR301 |
| **简要描述** | 本测试验证是否可以导入fgsm()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入adversarial\_attack模块中的fgsm方法：  Import DeepBox.adversarial\_attack.fgsm as fgsm | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用fgsm：  X\_adv, Y = fgsm(model, data\_loader, criterion, epsilon, normalizer) | | 屏幕输出：  "Before FGSM the accuracy is xxx" (xxx由实际计算得出)  "After FGSM the accuracy is xxx" (xxx由实际计算得出)  同时返回的X\_adv, Y存放了生成的样本和正确的标签 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用fgsm，其中输入内容存在不符合函数规定的项目：  X\_adv, Y = fgsm(model, data\_loader, criterion, epsilon, normalizer) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### PGD调用运行

PGD调用运行测试用例如表4-2所示。

表4-2 TC302-PGD调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | PGD调用运行 | **测试用例标识** | 用例TC302 | **测试需求标识** | TR301 |
| **简要描述** | 本测试验证是否可以导入pgd()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入adversarial\_attack模块中的pgd方法：  Import DeepBox.adversarial\_attack.pgd as pgd | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用pgd：  X\_adv, Y = pgd(model, data\_loader, criterion, epsilon, normalizer, iteration) | | 屏幕输出：  "PGD alpha is xxx" (xxx由实际计算得出)  "Before PGD the accuracy is xxx" (xxx由实际计算得出)  "After PGD the accuracy is xxx" (xxx由实际计算得出)  同时返回的X\_adv, Y存放了生成的样本和正确的标签 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用pgd，其中输入内容存在不符合函数规定的项目：  X\_adv, Y = pgd(model, data\_loader, criterion, epsilon, normalizer, iteration) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### STEP-LL调用运行

STEP-LL调用运行测试用例如表4-3所示。

表4-3 TC303-STEP-LL调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | STEP-LL调用运行 | **测试用例标识** | 用例TC303 | **测试需求标识** | TR301 |
| **简要描述** | 本测试验证是否可以导入step\_ll ()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入adversarial\_attack模块中的step\_ll方法：  Import DeepBox.adversarial\_attack.step\_ll as step\_ll | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用step\_ll：  X\_adv, Y = step\_ll(model, data\_loader, criterion, epsilon, normalizer) | | 屏幕输出：  "Before Step-ll the accuracy is xxx" (xxx由实际计算得出)  "After Step-ll the accuracy is xxx" (xxx由实际计算得出)  同时返回的X\_adv, Y存放了生成的样本和正确的标签 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用step\_ll，其中输入内容存在不符合函数规定的项目：  X\_adv, Y = step\_ll(model, data\_loader, criterion, epsilon, normalizer) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### MI-FGSM调用运行

MI-FGSM调用运行测试用例如表4-4所示。

表4-2 TC304-MI-FGSM调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | MI-FGSM调用运行 | **测试用例标识** | 用例TC304 | **测试需求标识** | TR301 |
| **简要描述** | 本测试验证是否可以导入momentum\_ifgsm()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入adversarial\_attack模块中的momentum\_ifgsm方法：  Import DeepBox.adversarial\_attack. momentum\_ifgsm as momentum\_ifgsm | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用momentum\_ifgsm：  X\_adv, Y = momentum\_ifgsm(model, data\_loader, criterion, epsilon, normalizer, iteration, attack\_momentum) | | 屏幕输出：  "MI-FGSM alpha is xxx" (xxx由实际计算得出)  "Before Momentum IFGSM the accuracy is xxx" (xxx由实际计算得出)  "After Momentum IFGSM the accuracy is xxx" (xxx由实际计算得出)  同时返回的X\_adv, Y存放了生成的样本和正确的标签 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用momentum\_ifgsm，其中输入内容存在不符合函数规定的项目：  X\_adv, Y = momentum\_ifgsm(model, data\_loader, criterion, epsilon, normalizer, iteration, attack\_momentum) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### C&W调用运行

C&W调用运行测试用例如表4-5所示。

表4-5 TC305-C&W调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | C&W调用运行 | **测试用例标识** | 用例TC305 | **测试需求标识** | TR301 |
| **简要描述** | 本测试验证是否可以导入CarliniWagnerL2()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入adversarial\_attack模块中的CarliniWagnerL2方法：  Import DeepBox.adversarial\_attack.CarliniWagnerL2 as CarliniWagnerL2 | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用CarliniWagnerL2：  CarliniWagnerL2(model, data\_loader, steps, search\_steps, normalizer, debug=False) | | 屏幕不断输出：  "Batch: xxx, search step: xxx" (xxx由实际计算得出)  "Step: xxx, loss: xxx}, dist\_loss: xx ,average dist: xxx, pixel\_wise\_L2: xxx, modifier mean: xxx" (xxx由实际计算得出)  以上是在批次迭代过程中的中间输出结果。  同时返回的X\_adv, Y存放了生成的样本和正确的标签 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用CarliniWagnerL2，其中输入内容存在不符合函数规定的项目：  CarliniWagnerL2(model, data\_loader, steps, search\_steps, normalizer, debug=False) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### Fourier Corruption调用运行

Fourier Corruption调用运行测试用例如表4-6所示。

表4-6 TC306-Fourier Corruption调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Fourier Corruption调用运行 | **测试用例标识** | 用例TC306 | **测试需求标识** | TR301 |
| **简要描述** | 本测试验证是否可以导入Fourier\_based\_Corruption ()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入adversarial\_attack模块中的Fourier\_based\_Corruption方法：  Import DeepBox.adversarial\_attackFourier\_based\_Corruption as Fourier\_based\_Corruption | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用Fourier\_based\_Corruption：  X\_adv = Fourier\_based\_Corruption(dataset, imgsize, position) | | 屏幕输出：  " position(xxx,xxx)" (xxx由实际计算得出)  同时返回的X\_adv存放了生成的样本 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用Fourier\_based\_Corruption，其中输入内容存在不符合函数规定的项目：  X\_adv = Fourier\_based\_Corruption(dataset, imgsize, position) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

## 神经网络量化模块

### BNN调用运行

BNN调用运行测试用例如表4-7所示。

表4-7 TC307-BNN调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | BNN调用运行 | **测试用例标识** | 用例TC307 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入QNNs.bnn\_res20\_1w1a()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入QNNs.bnn\_res20\_1w1a模块：Import QNNs.bnn\_res20\_1w1a | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用bnn，其中数据路径不存在：QNNs.bnn\_res20\_1w1a().\_\_main\_\_().main() | | 屏幕输出：  “Downloading …”  CIFAR-10数据开始自动下载到指定位置 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用bnn，其中数据路径存在且其他一切正常：QNNs.bnn\_res20\_1w1a().\_\_main\_\_().main() | | 屏幕显示导入数据。  屏幕显示模型开始训练。 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用bnn，其中任意阶段异常：QNNs.bnn\_res20\_1w1a().\_\_main\_\_().main() | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### XNOR调用运行

XNOR调用运行测试用例如表4-8所示。

表4-8 TC308-XNOR调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | XNOR调用运行 | **测试用例标识** | 用例TC308 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入QNNs.xnor\_res20\_1w1a()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入QNNs. xnor\_res20\_1w1a模块：Import QNNs.bnn\_res20\_1w1a | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用xnor，其中数据路径不存在：QNNs. xnor\_res20\_1w1a().\_\_main\_\_().main() | | 屏幕输出：  “Downloading …”  CIFAR-10数据开始自动下载到指定位置 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用xnor，其中数据路径存在且其他一切正常：QNNs. xnor\_res20\_1w1a().\_\_main\_\_().main() | | 屏幕显示导入数据。  屏幕显示模型开始训练。 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用xnor，其中任意阶段异常：QNNs. xnor\_res20\_1w1a().\_\_main\_\_().main() | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### HORQ调用运行

HORQ调用运行测试用例如表4-9所示。

表4-9 TC309-HORQ调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | HORQ调用运行 | **测试用例标识** | 用例TC309 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入QNNs.horq\_res20\_1w2a()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入QNNs.horq\_res20\_1w2a模块：Import QNNs.horq\_res20\_1w2a | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用horq，其中数据路径不存在：QNNs.horq\_res20\_1w2a().\_\_main\_\_().main() | | 屏幕输出：  “Downloading …”  CIFAR-10数据开始自动下载到指定位置 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用horq，其中数据路径存在且其他一切正常：QNNs.horq\_res20\_1w2a().\_\_main\_\_().main() | | 屏幕显示导入数据。  屏幕显示模型开始训练。 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用horq，其中任意阶段异常：QNNs.horq\_res20\_1w2a().\_\_main\_\_().main() | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### BWN调用运行

BWN调用运行测试用例如表4-10所示。

表4-10 TC310-BWN调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | BNN调用运行 | **测试用例标识** | 用例TC310 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入QNNs.bwn\_res20\_1w32a()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入QNNs.bwn\_res20\_1w32a模块：Import QNNs.bwn\_res20\_1w32a | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用bwn，其中数据路径不存在：QNNs.bwn\_res20\_1w32a().\_\_main\_\_().main() | | 屏幕输出：  “Downloading …”  CIFAR-10数据开始自动下载到指定位置 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用bwn，其中数据路径存在且其他一切正常：QNNs.bwn\_res20\_1w32a().\_\_main\_\_().main() | | 屏幕显示导入数据。  屏幕显示模型开始训练。 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用bwn，其中任意阶段异常：QNNs.bwn\_res20\_1w32a().\_\_main\_\_().main() | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### IR-Net调用运行

IR-Net调用运行测试用例如表4-11所示。

表4-11 TC311-IR-Net调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | IR-Net调用运行 | **测试用例标识** | 用例TC311 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入QNNs.irnet\_res20\_1w1a()方法，并成功运行，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入QNNs.irnet\_res20\_1w1a模块：Import QNNs.bnn\_res20\_1w1a | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句调用irnet，其中数据路径不存在：QNNs.irnet\_res20\_1w1a().\_\_main\_\_().main() | | 屏幕输出：  “Downloading …”  CIFAR-10数据开始自动下载到指定位置 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用irnet，其中数据路径存在且其他一切正常：QNNs.irnet\_res20\_1w1a().\_\_main\_\_().main() | | 屏幕显示导入数据。  屏幕显示模型开始训练。 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用irnet，其中任意阶段异常：QNNs.irnet\_res20\_1w1a().\_\_main\_\_().main() | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

## 目标检测模块

### Faster RCNN调用运行

Faster RCNN调用运行测试用例如表4-1所示。

表4-12 TC312-Faster RCNN调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Faster RCNN调用运行 | **测试用例标识** | 用例TC312 | **测试需求标识** | TR303 |
| **简要描述** | 本测试验证是否可以导入launch\_interface()方法，成功构建Faster RCNN并运行。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入Detectron模块中的launch\_interface方法：  Import Detectron.launch\_interface | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 构建main函数，设置参数，输入语句调用launch\_interface：  launch\_interface(main\_function, args) | | 屏幕输出：  “Starting training from iteration 0” | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用launch\_interface，其中输入内容存在不符合函数规定的参数：  launch\_interface：  launch\_interface(main\_function, args) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

## 阅读理解模块

### RC调用运行

RC模块调用运行测试用例如表4-13所示。

表4-13 TC313-RC模块调用测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | RC调用运行 | **测试用例标识** | 用例TC313 | **测试需求标识** | TR304 |
| **简要描述** | 本测试验证是否可以导入RC库方法，并成功调用其中的方法。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, ujsin库； 2. config.py已经对运行需要的参数进行赋值，运行需要的文件路径填写正确 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入  from RC preproc import Preproc  from RC.model import QAModel  from RC.render import QATrain, QATest | | 能够成功导入方法，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入P = Preproc() | |  | |  |
| 3 | 输入train\_examples, train\_eval = P.process\_file(config.train\_file, "train", word\_counter, char\_counter) | | 屏幕输出“Generating xx examples”，返回值正确保存在。 | | 实际结果与预期结果一致。 |
| 4 | 输入word\_mat = np.array(json.load(open(config.word\_emb\_file, "r")), dtype=np.float32) | | 词向量矩阵被正确保存在一个ndarray中。 | | 实际结果与预期结果一致。 |
| 5 | 输入device = torch. device ("cuda"if torch. cuda .is\_ available () else "cpu") | | 为本次测试指定设备 | |  |
| 6 | 输入model = QANet(config, word\_mat).to(device) | |  | | 屏幕不提示任何异常 |
| 7 | 输入 QATrain(config, model, device, word\_mat, train\_eval) | | 程序正常运行，屏幕输出每轮损失函数，损失函数合理逐步降低，保存模型过程不提示错误 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

## 主动学习模块

### BanditDiscreteSampler调用运行

BanditDiscreteSampler调用运行测试用例如表4-14所示。

表4-14 TC314-BanditDiscreteSampler调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | BanditDiscreteSampler调用运行 | **测试用例标识** | 用例TC314 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入BanditDiscreteSampler类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的BanditDiscreteSampler类：  from active\_learning import BanditDiscreteSampler | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = BanditDiscreteSampler(X, Y, seed, reward\_function, gamma) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### GraphDensitySampler调用运行

GraphDensitySampler调用运行测试用例如表4-15所示。

表4-15 TC315- GraphDensitySampler调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | GraphDensitySampler调用运行 | **测试用例标识** | 用例TC315 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入GraphDensitySampler类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的GraphDensitySampler类：  from active\_learning import GraphDensitySampler | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = GraphDensitySampler (X, Y, seed) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### HierarchicalClusterAL调用运行

HierarchicalClusterAL调用运行测试用例如表4-16所示。

表4-16 TC316- HierarchicalClusterAL调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | HierarchicalClusterAL调用运行 | **测试用例标识** | 用例TC316 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入HierarchicalClusterAL类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的HierarchicalClusterAL类：  from active\_learning import HierarchicalClusterAL | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = HierarchicalClusterAL (X, Y, seed) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### InformativeClusterDiverseSampler调用运行

InformativeClusterDiverseSampler调用运行测试用例如表4-17所示。

表4-17 TC317- InformativeClusterDiverseSampler调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | InformativeClusterDiverseSampler调用运行 | **测试用例标识** | 用例TC317 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入InformativeClusterDiverseSampler类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的InformativeClusterDiverseSampler类：  from active\_learning import InformativeClusterDiverseSampler | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = InformativeClusterDiverseSampler (X, Y, seed) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### kCenterGreedy调用运行

kCenterGreedy调用运行测试用例如表4-18所示。

表4-18 TC318- kCenterGreedy调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | kCenterGreedy调用运行 | **测试用例标识** | 用例TC318 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入kCenterGreedy类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的kCenterGreedy类：  from active\_learning import kCenterGreedy | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = kCenterGreedy (X, Y, seed, metric) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### MarginAL调用运行

MarginAL调用运行测试用例如表4-19所示。

表4-19 T319- MarginAL调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | MarginAL调用运行 | **测试用例标识** | 用例TC319 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入MarginAL类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的MarginAL类：  from active\_learning import MarginAL | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = MarginAL (X, Y, seed) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### MixtureOfSamplers调用运行

MixtureOfSamplers调用运行测试用例如表4-20所示。

表4-20 TC320- MixtureOfSamplers调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | MixtureOfSamplers调用运行 | **测试用例标识** | 用例TC320 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入MixtureOfSamplers类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的MixtureOfSamplers类：  from active\_learning import MixtureOfSamplers | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = MixtureOfSamplers (X, Y, seed, mixture) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### RepresentativeClusterMeanSampling调用运行

RepresentativeClusterMeanSampling调用运行测试用例如表4-21所示。

表4-21 TC321- RepresentativeClusterMeanSampling调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | RepresentativeClusterMeanSampling调用运行 | **测试用例标识** | 用例TC321 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入RepresentativeClusterMeanSampling类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的RepresentativeClusterMeanSampling类：  from active\_learning import RepresentativeClusterMeanSampling | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = RepresentativeClusterMeanSampling (X, Y, seed) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

### UniformSampling调用运行

UniformSampling调用运行测试用例如表4-22所示。

表4-22 TC322- UniformSampling调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | UniformSampling调用运行 | **测试用例标识** | 用例TC322 | **测试需求标识** | TR305 |
| **简要描述** | 本测试验证是否可以导入UniformSampling类，并成功运行select\_batch\_方法，返回生成结果。 | | | | |
| **前提和约束** | 1. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 2. 系统环境及版本符合本API的要求； 3. 已经将符合要求的函数输入部分在PyThon环境中准备好(假设变量名和函数规定的输入名一致)。 | | | | |
| **测试方法** | 黑盒测试 | | | | |
| **测试过程描述** | | | | | |
| **序号** | **测试步骤** | | **测试结果** | | **评价准则** |
| **预期结果** | |  |
| 1 | 测试员在PyThon环境中输入命令导入active\_learning模块中的UniformSampling类：  from active\_learning import UniformSampling | | 能够成功导入采样器类，不会抛出异常。 | | 实际结果与预期结果一致。 |
| 2 | 输入语句构建采样器对象：  sampler = UniformSampling (X, Y, seed) | | 正确返回采样器对象 | | 实际结果与预期结果一致。 |
| 3 | 输入语句调用采样器方法select\_batch\_，其中输入内容满足函数规定：batch = sampler.select\_batch\_(labeled, N) | | 正确返回从X中采样的一组数据一组数据 | | 实际结果与预期结果一致。 |
| 4 | 输入语句调用采样器方法select\_batch\_，其中输入内容存在不符合函数规定的项目：  batch = sampler.select\_batch\_(labeled, N) | | 运行中抛出异常 | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）测试员测试设备的显存（使用GPU）或内存（使用CPU）应能满足算法运行的大小要求。 | | | | |

# 非功能测试

## 高效性

## 测试策略描述

在相同的硬件条件下，使用深度学习系统架构TensorFlow实现API中的对抗攻击方法FGSM，比较其和API中相同算法的计算效率。

## 测试用例描述

测试用例如表5-1所示。

表5-1 TC501-高效性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 高效性 | **测试用例标识** | 用例TC501 | | **测试需求标识** | TR501 | |
| **简要描述** | | 本测试测试TensorFlow实现API中的对抗攻击方法FGSM和API中FGSM在相同软硬件条件下的计算效率，验证API的高效性。 | | | | | | |
| **前提和约束** | | 1. PyTorch平台正常运行； 2. PyTorch平台神经网络搭建成功，并在PyTorch平台上成功加载数据； 3. TensorFlow平台正常运行； 4. TensorFlow平台神经网络搭建成功，并在TensorFlow平台上成功加载数据。 | | | | | | |
| **测试方法** | | 黑盒测试 | | | | | | |
| **测试过程描述** | | | | | | | | |
| **序号** | **测试步骤** | | | | **测试结果** | | | **评价准则** |
| **预期结果** | | |  |
| 1 | 使用API中的FGSM方法生成对抗样本 | | | |  | | |  |
| 2 | 使用TensorFlow实现的FGSM方法生成对抗样本 | | | |  | | |  |
| 3 | 比较两者的计算效率 | | | | API计算效率优于TensorFlow平台 | | | 实际结果与预期结果一致 |

## 用户友好性

## 测试策略描述

用户（初学者）导入API之后，能够结合相关教程和自身知识储备，通过简单的接口调用，完成功能的实现。

## 测试用例描述

测试用例如表5-2所示。

表5-2 TC502-用户友好性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 用户友好性 | **测试用例标识** | 用例TC502 | | **测试需求标识** | TR502 | |
| **简要描述** | | 本测试通过初学者用户的上手实操，验证API的用户友好性。 | | | | | | |
| **前提和约束** | | 1. PyTorch平台正常运行； | | | | | | |
| **测试方法** | | 黑盒测试 | | | | | | |
| **测试过程描述** | | | | | | | | |
| **序号** | **测试步骤** | | | | **测试结果** | | | **评价准则** |
| **预期结果** | | |  |
| 1 | 导入PyTorch及API工具包 | | | | 成功完成导入操作 | | | 实际结果与预期结果一致 |
| 2 | 运行API中的IR-Net算法 | | | | 调用简单，运行成功。 | | | 实际结果与预期结果一致 |

## 健壮性

## 测试策略描述

通过调用过程输入边界信息和错误信息，运行大规模深度神经网络，运行其他程序抢占资源，强行关闭程序等方式验证API健壮性。

## 测试用例描述

测试用例如表5-3所示。

表5-3 TC503-健壮性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 鲁棒性 | **测试用例标识** | 用例TC503 | | **测试需求标识** | TR503 | |
| **简要描述** | | 本测试主要验证API的健壮性。 | | | | | | |
| **前提和约束** | | 无 | | | | | | |
| **测试方法** | | 黑盒测试 | | | | | | |
| **测试过程描述** | | | | | | | | |
| **序号** | **测试过程** | | | | **测试结果** | | | **评价准则** |
| **预期结果** | | |  |
| 1 | 向平台输入边界信息和错误信息（实例：输入错误指令） | | | | 平台提示命令不合法 | | | 实际结果与预期结果一致 |
| 2 | 运行大规模神经网络 | | | | 运行正常 | | | 实际结果与预期结果一致 |
| 3 | 运行其他程序抢占资源 | | | | 运行正常 | | | 实际结果与预期结果一致 |
| 4 | 强行关闭程序 | | | | 保存现有数据状态 | | | 实际结果与预期结果一致 |

## 扩展性

## 测试策略描述

此软件库应该具有可扩展性，支持相关领域算法的添加。研究人员可以在此库的基础上，利用库中实验流程通用的方法，通过实现满足规范的领域算法，快速进行新算法的实验。

在测试环节中做测试，具体添加算法的步骤说明会在产品说明文档中详细介绍。

# 总结

本文提供的测试方案基本覆盖了需求文档中提到功能需求、非功能性需求以及环境需求。主要以功能测试为主，设计了覆盖本项目所有模块基本功能的测试用例。

对于非功能需求的测试，分为高效性、用户友好性、可修改性和扩展性分别进行测试。

对环境需求的测试，主要在不同操作系统上不同软件的环境下进行部署配置。

在每个测试用例中，给出了实现每个测试的基本步骤和测试的期望输出与评价准则。

综上所述，本测试一共设计27个测试用例，该测试方案较为全面地覆盖了需求规格说明书中对应需求。

# 附录

表7.1 测试用例覆盖表

|  |  |  |  |
| --- | --- | --- | --- |
| **测试用例覆盖表** | | | |
| 需求编号 | 对应需求文档  章节 | 用例编号 | 用例名称 |
| TR401 | 4 环境需求 | TC401 | TC401-Linux下配置测试用例 |
| TC402 | TC402-Windows下配置测试用例 |
| TR301 | 3.1.1 生成对抗样本模块 | TC301 | TC301-FGSM调用运行测试用例 |
| TC302 | TC302-PGD调用运行测试用例 |
| TC303 | TC303-STEP-LL调用运行测试用例 |
| TC304 | TC304-MI-FGSM调用运行测试用例 |
| TC305 | TC305-C&W调用运行测试用例 |
| TC306 | TC306-Fourier Corruption调用运行测试用例 |
| TR302 | 3.1.2 生成量化网络模块 | TC307 | TC307-BNN调用运行测试用例 |
| TC308 | TC308-XNOR调用运行测试用例 |
| TC309 | TC309-HORQ调用运行测试用例 |
| TC310 | TC310-BWN调用运行测试用例 |
| TC311 | TC311-IR-Net调用运行测试用例 |
| TR303 | 3.1.3调用目标检测模块 | TC312 | TC312-Faster RCNN调用运行测试用例 |
| TR304 | 3.1.5 调用阅读理解模块 | TC313 | TC313-RC模块调用测试用例 |
| TR305 | 3.1.4调用主动学习模块 | TC314 | TC314-BanditDiscreteSampler调用运行测试用例 |
| TC315 | TC315-GraphDensitySampler调用运行测试用例 |
| TC316 | TC316-HierarchicalClusterAL调用运行测试用例 |
| TC317 | TC317-InformativeClusterDiverseSampler调用运行测试用例 |
| TC318 | TC318-kCenterGreed调用运行测试用例 |
| TC319 | T319-MarginAL调用运行测试用例 |
| TC320 | TC320-MixtureOfSampler调用运行测试用例 |
| TC321 | TC321-RepresentativeClusterMeanSampling调用运行测试用例 |
| TC322 | TC322 UniformSampling调用运行测试用例 |
| TR501 | 3.2.1 高效性 | TC501 | TC501-高效性测试用例 |
| TR502 | 3.2.2 用户友好性 | TC502 | TC502-用户友好性测试用例 |
| TR503 | 3.2.3 健壮性 | TC503 | TC503-健壮性测试用例 |
| 无 | 3.2.4 扩展性 | 无 | 在测试环节中不做测试 |

表7.2 专业术语/缩略语描述表

|  |  |  |
| --- | --- | --- |
| 序号 | 术语/缩写 | 定义 |
| 1 | API | Application Programming Interface，应用程序接口 |
| 2 | UCM | Use Case Modeling，用例建模 |
| 3 | RUCM | Restricted Use Case Modeling，限制性用例建模 |
| 4 | CPU | Central processing unit，中央处理器 |
| 5 | GPU | Graphics Processing Unit，图形处理器 |
| 6 | DNNs | Deep Neural Networks,深度神经网络 |
| 7 | CUDA | Compute Unified Device Architecture，统一计算设备架构 |