

**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 |
| 2.0 | 2020/5/17 | 全体成员 | 张崇智、高明骏 | 对测试用例的分类做了更详细和结构化的列举 |
| 3.0 | 2020/5/22 | 全体成员 | 张崇智、高明骏 | 针对D、E组的评审结果修改问题 |
| 4.0 | 2020/5/25 | 张崇智 | 高明骏 | 针对E组的评审结果延时修改的问题进行修正 |
| 5.0 | 2020/5/28 | 全体成员 | 高明骏 | 针对F和G组的评审结果修改问题 |

**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 1](#_Toc41584542)

[1.1 编写目的 1](#_Toc41584543)

[1.2 测试内容 1](#_Toc41584544)

[1.3 参考文献 1](#_Toc41584545)

[2. 测试计划 2](#_Toc41584546)

[2.1 测试目标 2](#_Toc41584547)

[2.2 测试工具及技术 2](#_Toc41584548)

[2.2.1 Python 3](#_Toc41584549)

[2.2.2 测试技术及应用 3](#_Toc41584550)

[2.3 测试用例设计要求及通过准则 4](#_Toc41584551)

[2.4 专业术语/缩略语 5](#_Toc41584552)

[3. 运行配置测试 6](#_Toc41584553)

[3.1 概述 6](#_Toc41584554)

[3.2 环境配置 6](#_Toc41584555)

[3.3 测试用例 7](#_Toc41584556)

[4. 单元测试 9](#_Toc41584557)

[4.1 概述 9](#_Toc41584558)

[4.2 对抗样本生成模块 9](#_Toc41584559)

[4.2.1 FGSM调用运行 10](#_Toc41584560)

[4.2.2 PGD调用运行 11](#_Toc41584561)

[4.2.3 STEP-LL调用运行 12](#_Toc41584562)

[4.2.4 MI-FGSM调用运行 13](#_Toc41584563)

[4.2.5 C&W调用运行 15](#_Toc41584564)

[4.2.6 Fourier Corruption调用运行 16](#_Toc41584565)

[4.3 神经网络量化模块 17](#_Toc41584566)

[4.3.1 BNN调用运行 17](#_Toc41584567)

[4.3.2 XNOR调用运行 18](#_Toc41584568)

[4.3.3 HORQ调用运行 20](#_Toc41584569)

[4.3.4 BWN调用运行 21](#_Toc41584570)

[4.3.5 IR-Net调用运行 22](#_Toc41584571)

[4.4 目标检测模块 23](#_Toc41584572)

[4.4.1 Faster RCNN调用运行 24](#_Toc41584573)

[4.4.2 Fast RCNN调用运行 25](#_Toc41584574)

[4.4.3 RetinaNet调用运行 26](#_Toc41584575)

[4.5 阅读理解模块 28](#_Toc41584576)

[4.5.1 PytorchPlus.RC.Preproc类单元测试 28](#_Toc41584577)

[4.6 主动学习模块 29](#_Toc41584578)

[4.6.1 BanditDiscreteSampler调用运行 30](#_Toc41584579)

[4.6.2 GraphDensitySampler调用运行 31](#_Toc41584580)

[4.6.3 HierarchicalClusterAL调用运行 32](#_Toc41584581)

[4.6.4 InformativeClusterDiverseSampler调用运行 33](#_Toc41584582)

[4.6.5 kCenterGreedy调用运行 35](#_Toc41584583)

[4.6.6 MarginAL调用运行 36](#_Toc41584584)

[4.6.7 MixtureOfSamplers调用运行 37](#_Toc41584585)

[4.6.8 RepresentativeClusterMeanSampling调用运行 38](#_Toc41584586)

[4.6.9 UniformSampling调用运行 40](#_Toc41584587)

[5. 集成测试 41](#_Toc41584588)

[5.1 概述 41](#_Toc41584589)

[5.2 测试用例 41](#_Toc41584590)

[6. 功能测试 43](#_Toc41584591)

[6.1 概述 43](#_Toc41584592)

[6.2 对抗样本生成模块 43](#_Toc41584593)

[6.2.1 FGSM调用运行 44](#_Toc41584594)

[6.2.2 PGD调用运行 45](#_Toc41584595)

[6.2.3 STEP-LL调用运行 46](#_Toc41584596)

[6.2.4 MI-FGSM调用运行 47](#_Toc41584597)

[6.2.5 C&W调用运行 48](#_Toc41584598)

[6.2.6 Fourier Corruption调用运行 50](#_Toc41584599)

[6.3 神经网络量化模块 51](#_Toc41584600)

[6.3.1 BNN调用运行 52](#_Toc41584601)

[6.3.2 XNOR调用运行 53](#_Toc41584602)

[6.3.3 HORQ调用运行 54](#_Toc41584603)

[6.3.4 BWN调用运行 55](#_Toc41584604)

[6.3.5 IR-Net调用运行 57](#_Toc41584605)

[6.4 目标检测模块 58](#_Toc41584606)

[6.4.1 Faster RCNN调用运行 58](#_Toc41584607)

[6.4.2 Fast RCNN调用运行 60](#_Toc41584608)

[6.4.3 RetinaNet调用运行 61](#_Toc41584609)

[6.5 阅读理解模块 62](#_Toc41584610)

[6.5.1 RC模块处理数据功能测试 63](#_Toc41584611)

[6.5.2 RC模块阅读理解算法功能测试 64](#_Toc41584612)

[6.6 主动学习模块 66](#_Toc41584613)

[6.6.1 BanditDiscreteSampler调用运行 66](#_Toc41584614)

[6.6.2 GraphDensitySampler调用运行 68](#_Toc41584615)

[6.6.3 HierarchicalClusterAL调用运行 69](#_Toc41584616)

[6.6.4 InformativeClusterDiverseSampler调用运行 70](#_Toc41584617)

[6.6.5 kCenterGreedy调用运行 72](#_Toc41584618)

[6.6.6 MarginAL调用运行 73](#_Toc41584619)

[6.6.7 MixtureOfSamplers调用运行 74](#_Toc41584620)

[6.6.8 RepresentativeClusterMeanSampling调用运行 75](#_Toc41584621)

[6.6.9 UniformSampling调用运行 77](#_Toc41584622)

[6.7 加载和保存模型模块 78](#_Toc41584623)

[6.7.1 加载模型测试用例 78](#_Toc41584624)

[6.7.2 保存模型测试用例 79](#_Toc41584625)

[7. 非功能测试 81](#_Toc41584626)

[7.1 用户友好性 81](#_Toc41584627)

[7.1.1 测试策略描述 81](#_Toc41584628)

[7.1.2 测试用例描述 81](#_Toc41584629)

[7.2 健壮性 82](#_Toc41584630)

[7.2.1 测试策略描述 82](#_Toc41584631)

[7.2.2 测试用例描述 82](#_Toc41584632)

[7.3 扩展性 83](#_Toc41584633)

[7.3.1 测试策略描述 83](#_Toc41584634)

[7.3.2 测试用例描述 83](#_Toc41584635)

[8. 总结 84](#_Toc41584636)

[9. 附录 84](#_Toc41584637)

# 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《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》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。

## 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，检查可用性和是否及时释放资源，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。

## 专业术语/缩略语

表2-1 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 | CPU | Central processing unit，中央处理器 |
| 3 | GPU | Graphics Processing Unit，图形处理器 |
| 4 | DNNs | Deep Neural Networks,深度神经网络 |
| 5 | CUDA | Compute Unified Device Architecture，统一计算设备架构 |
| 6 | FGSM | Fast Gradient Sign Method，快速梯度符号法 |
| 7 | PGD | Projected Gradient Descent，投影梯度下降 |
| 8 | STEP-LL | STEP Least Likelihood，最小可能性类别单步攻击 |
| 9 | MI-FGSM | Momentum Iterative Fast Gradient Sign Method，基于动量的迭代快速梯度符号法 |
| 10 | C&W | Carlini and Wagner，CW攻击，用两位发明者名字命名 |
| 11 | XNOR-Net | 使用XNOR操作的二值神经网络 |
| 12 | IR-Net | Information retention network |
| 13 | BWN | Binary Weight Network |
| 14 | BNN | Binary Neural Network |
| 15 | Fast RCNN | Fast Region Convolution Neural Network |
| 16 | Faster RCNN | Faster Region Convolution Neural Network |
| 17 | RetinaNet | Retina Network |
| 18 | seed | 用来随机采样的种子 |
| 19 | eval\_acc | 表示被评价模型的准确度 |
| 20 | N | 主动学习中需要标记的样本数 |
| 21 | CV | Computer Vision，计算机视觉 |
| 22 | NLP | Natural Language Processing ，自然语言处理 |

# 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。

## 环境配置

集成本项目包所需要的基本包版本要求如表3-1和表3-2所示。

表3-1 PytorchPlus环境及版本要求

|  |  |
| --- | --- |
| **环境要求** | **版本** |
| Python | >=3.6 |
| PyTorch | >=1.1.0 |
| Numpy | 1.18.4 |
| Pickle | 4.0 |
| Pillow | 6.0 |
| CUDA | 10.0 |
| scipy | 1.3.2 |
| Scikit-learn | 0.22.1 |
| spacy | >=2.0.11 |
| ujson | >=1.35 |

注：在安装完spacy模块后，需加装英文支持：python -m spacy download en

表3-2 附加模块detectron2环境及版本要求（在上表基础上）

|  |  |
| --- | --- |
| **环境要求** | **版本** |
| gcc | 5 |
| CMake | 3.5 |
| cuDNN | 7.6 |
| blas | 1.0 |
| mkl | 2020.1 |
| mkl-service | 2.3.0 |
| mkl\_fft | 1.0 |
| mkl\_random | 1.1 |

## 测试用例

针对需求规格说明书中的环境需求，分别构造在Windows和Linux下安装的配置测试，如表3-3和表3-4所示。

表3-3 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（Ubuntu 16.04）中能否被正常编译、安装。 | | | | | |
| **前提和约束** | PyTorch已经成功安装，且已安装相关依赖包，如numpy、CUDA、GCC等。 | | | | | |
| **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 | 启动PyTorch | | 启动成功 | | 实际结果与预期结果一致 |  |
| 2 | 导入项目包 | | 导入成功 | | 实际结果与预期结果一致 |  |
| 3 | 测试员将detection压缩包解压后输入指令：python -m pip install -e detection\_simply; | | detection2成功安装到python环境中；终端提示安装成功 | | 实际结果与预期结果一致 |  |

表3-4 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PyTorch在Windows操作系统中（Windows 10）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 |  |
| 5 | 测试员将detection压缩包解压后输入指令：python -m pip install -e detection\_simply; | | detection2成功安装到python环境中；终端提示安装成功 | | 实际结果与预期结果一致 |  |

# 单元测试

## 概述

单元测试（模块测试）是指开发者编写一小段代码，用于检验被测代码的一个很小的、很明确的功能是否正确。通常而言，一个单元测试是用于判断某个特定条件（或者场景）下某个特定函数的行为。在本软件中，将在开发过程中分别针对各个模块本身进行单元测试。

## 对抗样本生成模块

针对对抗样本生成模块进行单元测试，仅对该模块内部的功能性进行测试。该模块的代码结构如图4-1所示，顶层的文件夹adversarial\_attack下的adversarial\_attack.py文件存放着该模块所有算法的调用接口，在应用过程中，用户只需引用该部分即可实现算法运行。

该模块中各个算法涉及的输入类型规范，见实验三中的软件设计文档。



图4-1 对抗样本生成模块代码结构

### FGSM调用运行

FGSM调用运行测试用例如表4-1所示。

表4-1 TC101-1-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TC101-1 | **测试需求标识** | TR101-1 |
| **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用fgsm：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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 TC101-2-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TC101-2 | **测试需求标识** | TR101-2 |
| **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用pgd：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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 TC101-3-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TC101-3 | **测试需求标识** | TR101-3 |
| **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用step\_ll：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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4-4 TC101-4-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TC101-4 | **测试需求标识** | TR101-4 |
| **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用momentum\_ifgsm：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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 TC101-5-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TC101-5 | **测试需求标识** | TR101-5 |
| **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用CarliniWagnerL2：  Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  Attack.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 TC101-6-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TC101-6 | **测试需求标识** | TR101-6 |
| **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用Fourier\_based\_Corruption：  X\_adv = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv = Attack.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 TC102-1-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TC102-1 | **测试需求标识** | TR102-1 |
| **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PyTthon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** | 白盒测试 | | | | |
| **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（python运行路径/data）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（python运行路径/data）存在且其他一切正常(与2测试命令相同，数据输入路径存在且正确)：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 |
| 4 | 输入语句调用bnn，在运行过程中是用Ctrl+C强制打断：QNNs.bnn\_res20\_1w1a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\bnn\_res20\_1w1a\modules\bnn\_1w1a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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 TC102-2-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TC102-2 | **测试需求标识** | TR102-2 |
| **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** | 白盒测试 | | | | |
| **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（python运行路径/data）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（python运行路径/data）存在且其他一切正常(与2测试命令相同，数据输入路径存在且正确)：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 |
| 4 | 输入语句调用xnor，在运行过程中是用Ctrl+C强制打断：QNNs. xnor\_res20\_1w1a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ xnor\_res20\_1w1a \modules\xnor\_1w1a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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 TC102-3-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TC102-3 | **测试需求标识** | TR102-3 |
| **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** | 白盒测试 | | | | |
| **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（python运行路径/data）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（python运行路径/data）存在且其他一切正常(与2测试命令相同，数据输入路径存在且正确)：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 |
| 4 | 输入语句调用horq，在运行过程中是用Ctrl+C强制打断：QNNs.horq\_res20\_1w2a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ horq\_res20\_1w2a \modules\horq\_1w2a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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 TC102-4-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TC102-4 | **测试需求标识** | TR102-4 |
| **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** | 白盒测试 | | | | |
| **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（python运行路径/data）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（python运行路径/data）存在且其他一切正常(与2测试命令相同，数据输入路径存在且正确)：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 |
| 4 | 输入语句调用bwn，在运行过程中是用Ctrl+C强制打断：QNNs.bwn\_res20\_1w32a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ bwn\_res20\_1w32a \modules\bwn\_1w32a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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 TC411-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TC407 | **测试需求标识** | TR411 |
| **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** | 白盒测试 | | | | |
| **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（python运行路径/data）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（python运行路径/data）存在且其他一切正常(与2测试命令相同，数据输入路径存在且正确)：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 |
| 4 | 输入语句调用irnet，在运行过程中是用Ctrl+C强制打断：QNNs.irnet\_res20\_1w1a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ irnet\_res20\_1w1a \modules\irnet\_1w1a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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

由于目标检测模块对环境的依赖较多，与其他模块的差异较大，所以不强制用户安装，而是作为附加的模块单独打包，用户可选择另外安装。

针对目标检测模块进行单元测试，仅对该模块内部的功能性进行测试。该模块名称为detectron2，在顶层的文件夹下的train\_net.py文件存放着该模块调用的一个示例接口launch\_interface，在应用过程中，用户可以引用调用该接口实现算法运行，并通过args参数来构建不同的网络结构。

### Faster RCNN调用运行

Faster RCNN调用运行测试用例如表4-12所示。

表4-12 TC103-1-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TC103-1 | **测试需求标识** | TR103-1 |
| **简要描述** | 本测试验证是否可以导入launch\_interface方法，并成功传参建构Faster RCNN运行。 | | | | |
| **前提和约束** | 1. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员成功编译好安装包； 4. 测试员下载好COCO格式数据集:   http://images.cocodataset.org/zips/train2017.zip；   1. 测试员在终端中设置环境变量DETECTRON2\_DATASETS为数据集存储地址； 2. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import detectron2. train\_net | | 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 | 在detection\_simply的目录下，运行指令python detectron2/train\_net.py \  --config-file configs/COCO-Detection/faster\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 屏幕输出：  “Starting training from iteration 0”表示模型已经开始训练，未抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 在detection\_simply的目录下，运行包含错误参数名称的指令python detectron2/train\_net.py \  --CONFIG\_FILE configs/COCO-Detection/faster\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 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。 | | | | |

### Fast RCNN调用运行

Fast RCNN调用运行测试用例如表4-13所示。

表4-13 TC103-2-Fast RCNN调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Fast RCNN调用运行 | **测试用例标识** | 用例TC103-2 | **测试需求标识** | TR103-2 |
| **简要描述** | 本测试验证是否可以导入launch\_interface方法，并成功传参建构Fast RCNN运行。 | | | | |
| **前提和约束** | 1. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员成功编译好安装包； 4. 测试员下载好COCO格式数据集：http://images.cocodataset.org/zips/train2017.zip 5. 测试员在终端中设置环境变量DETECTRON2\_DATASETS为数据集存储地址； 6. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import detectron2. train\_net | | 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 | 在detection\_simply的目录下，运行指令python detectron2/train\_net.py \  --config-file configs/COCO-Detection/fast\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 屏幕输出：  “Starting training from iteration 0”表示模型已经开始训练，未抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 在detection\_simply的目录下，运行包含错误参数名称的指令python detectron2/train\_net.py \  --CONFIG\_FILE configs/COCO-Detection/fast\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 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。 | | | | |

### RetinaNet调用运行

RetinaNet调用运行测试用例如表4-14所示。

表4-14 TC103-3-RetineNet调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | RetinaNet调用运行 | **测试用例标识** | 用例TC103-3 | **测试需求标识** | TR103-3 |
| **简要描述** | 本测试验证是否可以导入launch\_interface方法，并成功传参建构RetinaNet运行。 | | | | |
| **前提和约束** | 1. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员成功编译好安装包； 4. 测试员下载好COCO格式数据集：http://images.cocodataset.org/zips/train2017.zip； 5. 测试员在终端中设置环境变量DETECTRON2\_DATASETS为数据集存储地址； 6. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import detectron2. train\_net | | 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 | 在detection\_simply的目录下，运行指令python detectron2/train\_net.py \  --config-file configs/COCO-Detection/retinanet\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 屏幕输出：  “Starting training from iteration 0”表示模型已经开始训练，未抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 在detection\_simply的目录下，运行包含错误参数名称的指令python detectron2/train\_net.py \  --CONFIG\_FILE configs/COCO-Detection/retinanet\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 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

因为深度学习算法在静止状态时无法判断其正确性，在运行时通过功能正常执行，观察预测结果，损失函数降低等现象判断算法正确，故本小节仅对PytorchPlus.RC.Preproc类进行了单元测试，该类负责对数据进行处理。

### 4.5.1 PytorchPlus.RC.Preproc类单元测试

因为需要共用一个Counter对象以验证数据处理结果的正确性，本用例串行process\_file，get\_embedding，build\_features三个函数单元进行测试。

表4.15 TC104-1-Preproc类单元测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | RC.Preproc类单元测试 | **测试用例标识** | 用例TC104-1 | **测试需求标识** | TR104-1 |
| **简要描述** | 本测试验证是否可以成功调用RC模块调用处理数据功能。 | | | | |
| **前提和约束** | 1. 当前路径下有config.py文件； 2. config.py文件中已经对的处理数据功能需要的参数进行赋值，具体要求如下：    1. 为char\_di, glove\_dim赋非负值，如果有词向量文件，赋值需要和使用的文件的向量长度一致；    2. 为para\_limitm, ques\_limit, answer\_limit, ques\_limit赋非负值。 | | | | |
| **测试方法** | 白盒测试 | | | | |
| **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  import config  from RC.preproc import Preproc  from collections import Counter  import ujson as json | | 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() | | 能成功创建对象，不会出现异常 | | 实际结果与预期结果一致 |
| 3 | 输入 word\_counter, char\_counter = Counter(), Counter() | | 成功创建两个计数器对象 | | 结果仅与python环境有关 |
| 4 | 定义并保存一个测试使用的数据，样例如下s={"data":[{"title":"Super\_Bowl\_50","paragraphs":[{"context":"this is a test context","qas":[{"answers":[{"answer\_start":0,"text":"this is"}],"question":"is this a test context?","id":"randomid0001"}]}]}],"version":"test\_simple"} | |  | |  |
| 5 | train\_examples, train\_eval = pproc.process\_file("test\_tmp\_rawdata\_file.json", "unitText", word\_counter, char\_counter) | | 查看 train\_examples, train\_eval的具体内容，应与输入测试数据（例如上述s）保持对应，并正确切割句子。 | | 正确处理文本条目，process\_file方法正常 |
| 6 | emb\_mat, word2idx\_dict = pproc.get\_embedding(word\_counter, "unitText", vec\_size=config.glove\_dim)  emb\_mat, char2idx\_dict = pproc.get\_embedding(char\_counter, "unitText", vec\_size=config.char\_dim) | | 查看 word2idx\_dict，char2idx\_dict的具体内容，字典应包含测试数据中的全部单词。 | | 正确构建字符序号和词序号词典， get\_embedding方法正常 |
| 7 | pproc.build\_features(config, train\_examples, "train", "test\_tmp\_record\_file.json",word2idx\_dict, char2idx\_dict) | | 特征保存在"test\_tmp\_record\_file.json"文件里，应与输入测试数据（例如上述s）保持对应，并正确切割句子。 | | 正确保存特征文件， build\_features方法正常 |
| **备注** | 可通过直接运行文件test\_unit.py进行测试 | | | | |

## 主动学习模块

针对主动学习模块进行单元测试，仅对该模块内部的功能性进行测试。模块顶层的文件夹active\_learning下定义了各个主动学习采样策略的类，和一些功能类。该模块所实现的所有采样策略的调用接口都在\_\_init\_\_.py文件中引入，在应用过程中，用户只需引用该部分即可实现算法运行。

该模块中各个算法涉及的输入类型规范，见实验三中的软件设计文档。

### BanditDiscreteSampler调用运行

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

表4-16 TC105-1-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T105-1 | **测试需求标识** | TR105-1 |
| **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** | 白盒测试 | | | | |
| **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) | | 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\_(N, labeled, eval\_acc, model) | | 正确返回从X中采样的一组数据 | | 实际结果与预期结果一致 |
| 4 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, eval\_acc, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, eval\_acc, model) | | 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4-17所示。

表4-17 TC105-2-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TC105-2 | **测试需求标识** | TR105-2 |
| **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** | 白盒测试 | | | | |
| **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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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4-18所示。

表4-18 TC105-3-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TC105-3 | **测试需求标识** | TR105-3 |
| **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** | 白盒测试 | | | | |
| **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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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4-19所示。

表4-19 TC105-4-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TC105-4 | **测试需求标识** | TR105-4 |
| **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** | 白盒测试 | | | | |
| **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\_(N, labeled, model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model) | | 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4-20所示。

表4-20 TC105-5-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TC105-5 | **测试需求标识** | TR105-5 |
| **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** | 白盒测试 | | | | |
| **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) | | 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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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4-21所示。

表4-21 TC105-6-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TC105-6 | **测试需求标识** | TR105-6 |
| **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** | 白盒测试 | | | | |
| **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\_(N, labeled, model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model) | | 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4-22所示。

表4-22 TC105-7-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TC105-7 | **测试需求标识** | TR105-7 |
| **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** | 白盒测试 | | | | |
| **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) | | 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\_(N, labeled, model=model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model=model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model=model) | | 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4-23所示。

表4-23 TC105-8-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TC105-8 | **测试需求标识** | TR105-8 |
| **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** | 白盒测试 | | | | |
| **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\_(N, labeled, model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model) | | 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4-24所示。

表4-24 TC105-9-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TC105-9 | **测试需求标识** | TR105-9 |
| **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** | 白盒测试 | | | | |
| **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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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。 | | | | |

# 集成测试

## 概述

实践表明，一些模块虽然能够单独地工作，但并不能保证连接起来也能正常的工作。一些局部反映不出来的问题，在全局上很可能暴露出来。

集成测试，也叫[组装测试](https://baike.baidu.com/item/%E7%BB%84%E8%A3%85%E6%B5%8B%E8%AF%95/3123153" \t "_blank)或联合测试。在[单元测试](https://baike.baidu.com/item/%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95/1917084" \t "_blank)的基础上，将所有模块按照设计要求（如根据结构图）组装成为子系统或系统，进行集成测试。

集成测试是[单元测试](https://baike.baidu.com/item/%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95" \t "_blank)的逻辑扩展。它最简单的形式是：把两个已经测试过的单元组合成一个组件，测试它们之间的[接口](https://baike.baidu.com/item/%E6%8E%A5%E5%8F%A3" \t "_blank)。从这一层意义上讲，组件是指多个单元的集成聚合。在现实方案中，许多单元组合成组件，而这些组件又聚合为程序的更大部分。方法是测试片段的组合，并最终扩展成进程，将模块与其他组的模块一起测试。最后，将构成进程的所有模块一起测试。

## 测试用例

项目模块PytorchPlus集成加载测试用例如表5-1所示。

表5-1 TC501-PytorchPlus集成加载测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | PytorchPlus集成加载测试用例 | **测试用例标识** | 用例TC501 | **测试需求标识** | TR501 |
| **简要描述** | 本测试验证是否可以成功导入PytorchPlus模块中所有子模块，即测试集成和加载是否完备无误。 | | | | |
| **前提和约束** | 1. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包。 | | | | |
| **测试方法** | 集成测试 | | | | |
| **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 | 测试员将PytorchPlus-1.0.0.tar.gz压缩包解压后进入文件夹PytorchPlus-1.0.0  输入指令python setup.py install | | PytorchPlus成功安装到python环境中。 | | 终端提示安装成功 |
| 2 | 测试员在python环境中输入命令：from detectron2.train\_net import launch\_interface | | 能成功导入模块，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 测试员在Python环境中输入命令：  import PytorchPlus | | 能够成功导入模块，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 测试员在Python环境中输入命令：  import PytorchPlus. adversarial\_attack | | 能够成功导入子模块adversarial\_attack，不会抛出异常。 | | 实际结果与预期结果一致 |
| 4 | 测试员在Python环境中输入命令：  import PytorchPlus. QNNs | | 能够成功导入子模块QNNs，不会抛出异常。 | | 实际结果与预期结果一致 |
| 5 | 测试员在Python环境中输入命令：  import PytorchPlus. RC | | 能够成功导入子模块RC，不会抛出异常。 | | 实际结果与预期结果一致 |
| 6 | 测试员在Python环境中输入命令：  import PytorchPlus. active\_learning | | 能够成功导入子模块active\_learning，不会抛出异常。 | | 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。 | | | | |

# 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，进而使产品达到用户使用的要求。

## 对抗样本生成模块

针对对抗样本生成模块进行功能测试，旨在对封装完毕的软件中该模块部分的功能性进行测试。由于进行了所有模块的封装，因此该模块的代码结构多了一级，具体如图6-1所示，顶层PytorchPlus下的文件夹adversarial\_attack下的adversarial\_attack.py文件存放着该模块所有算法的调用接口，在应用过程中，用户只需引用该部分即可实现算法运行。

该模块中各个算法涉及的输入类型规范，见实验三中的软件设计文档。



图6-1 封装后对抗样本生成模块部分代码结构

### FGSM调用运行

FGSM调用运行测试用例如表6-1所示。

表6-1 TC301-1-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TC301-1 | **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import PytorchPlus.adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用fgsm：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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6-2所示。

表6-2 TC301-2-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TC301-2 | **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import PytorchPlus.adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用pgd：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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6-3所示。

表6-3 TC301-3-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TC301-3 | **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import PytorchPlus.adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用step\_ll：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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6-4所示。

表6-4 TC301-4-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TC301-4 | **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import PytorchPlus.adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用momentum\_ifgsm：  X\_adv, Y = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv, Y = Attack.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6-5所示。

表6-5 TC301-5-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TC301-5 | **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import PytorchPlus.adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用CarliniWagnerL2：  Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  Attack.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6-6所示。

表6-6 TC301-6-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TC301-6 | **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. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import PytorchPlus.adversarial\_attack. adversarial\_attack as Attack | | 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 | 测试员调用外部方法处理得到符合要求的函数输入内容 （假设变量名和函数规定的输入名一致）。 | | 能够成功得到符合要求的函数输入内容，不会抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 输入语句调用Fourier\_based\_Corruption：  X\_adv = Attack.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 |
| 4 | 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（即对2中的某几个变量内容进行替换）：  X\_adv = Attack.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

针对神经网络量化模块进行功能测试，旨在对封装完毕的软件中该模块部分的功能性进行测试。由于进行了所有模块的封装，因此该模块的代码结构多了一级，具体如图6-2所示，顶层PytorchPlus下的文件夹QNNs下的bnn\_res20\_1w1a等文件夹下的\_\_main\_\_.py文件中，存放着该模块算法的调用接口，在应用过程中，用户只需引用该部分即可实现算法运行。

该模块中各个算法涉及的输入类型规范，见实验三中的软件设计文档。



图64-21 封装后神经网络量化bnn\_res20\_1w1a模块代码结构

### BNN调用运行

BNN调用运行测试用例如表6-7所示。

表6-7 TC302-1-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TC302-1 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入PytorchPlus.QNNs.bnn\_res20\_1w1a方法，并成功运行，返回训练结果。 | | | | |
| **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PytorchPlus.QNNs.bnn\_res20\_1w1a模块：import PytorchPlus.QNNs.bnn\_res20\_1w1a as bnn | | 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：import PytorchPlus.QNNs.bnn\_res20\_1w1a as bnn  bnn.\_\_main\_\_.main() | | 屏幕输出：  “Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data\cifar-10-python.tar.gz”  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，其中输入数据路径存在且其他一切正常：import PytorchPlus.QNNs.bnn\_res20\_1w1a as bnn  bnn.\_\_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 |
| 4 | 输入语句调用bnn，在运行过程中是用Ctrl+C强制打断：PytorchPlus.QNNs.bnn\_res20\_1w1a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\bnn\_res20\_1w1a\modules\bnn\_1w1a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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6-8所示。

表6-8 TC302-2-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TC302-2 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入PytorchPlus.QNNs.xnor\_res20\_1w1a方法，并成功运行，返回训练结果。 | | | | |
| **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PytorchPlus.QNNs.xnor\_res20\_1w1a模块：import PytorchPlus.QNNs.xnor\_res20\_1w1a as xnor | | 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：import PytorchPlus.QNNs.xnor\_res20\_1w1a as xnor  xnor.\_\_main\_\_.main() | | 屏幕输出：  “Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data\cifar-10-python.tar.gz”  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，其中输入数据路径存在且其他一切正常：import PytorchPlus.QNNs.xnor\_res20\_1w1a as xnor  xnor.\_\_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 |
| 4 | 输入语句调用xnor，在运行过程中是用Ctrl+C强制打断：PytorchPlus.QNNs.xnor\_res20\_1w1a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ xnor\_res20\_1w1a \modules\xnor\_1w1a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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6-9所示。

表6-9 TC302-3-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TC302-3 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入PytorchPlus.QNNs.horq\_res20\_1w2a方法，并成功运行，返回训练结果。 | | | | |
| **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PytorchPlus.QNNs.horq\_res20\_1w2a模块：import PytorchPlus.QNNs.horq\_res20\_1w2a as horq | | 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：import PytorchPlus.QNNs.horq\_res20\_1w2a as horq  horq.\_\_main\_\_.main() | | 屏幕输出：  “Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data\cifar-10-python.tar.gz”  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，其中输入数据路径存在且其他一切正常：import PytorchPlus.QNNs.horq\_res20\_1w2a as horq  horq.\_\_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 |
| 4 | 输入语句调用horq，在运行过程中是用Ctrl+C强制打断：PytorchPlus.QNNs.horq\_res20\_1w2a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ horq\_res20\_1w2a \modules\horq\_1w2a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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6-10所示。

表6-10 TC302-4-BWN调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | BWN调用运行 | **测试用例标识** | 用例TC302-4 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入PytorchPlus.QNNs.bwn\_res20\_1w32a方法，并成功运行，返回训练结果。 | | | | |
| **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PytorchPlus.QNNs.bwn\_res20\_1w32a模块：import PytorchPlus.QNNs.bwn\_res20\_1w32a as bwn | | 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：import PytorchPlus.QNNs.bwn\_res20\_1w32a as bwn  bwn.\_\_main\_\_.main() | | 屏幕输出：  “Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data\cifar-10-python.tar.gz”  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，其中输入数据路径存在且其他一切正常：import PytorchPlus.QNNs.bwn\_res20\_1w32a as bwn  bwn.\_\_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 |
| 4 | 输入语句调用bwn，在运行过程中是用Ctrl+C强制打断：PytorchPlus.QNNs.bwn\_res20\_1w32a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ bwn\_res20\_1w32a \modules\bwn\_1w32a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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6-11所示。

表6-11 TC302-5-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TC302-5 | **测试需求标识** | TR302 |
| **简要描述** | 本测试验证是否可以导入PytorchPlus.QNNs.irnet\_res20\_1w1a方法，并成功运行，返回训练结果。 | | | | |
| **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PytorchPlus.QNNs.irnet\_res20\_1w1a模块：import PytorchPlus.QNNs.irnet\_res20\_1w1a as irnet | | 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：import PytorchPlus.QNNs.irnet\_res20\_1w1a as irnet  irnet.\_\_main\_\_.main() | | 屏幕输出：  “Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data\cifar-10-python.tar.gz”  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，其中输入数据路径存在且其他一切正常：import PytorchPlus.QNNs.irnet\_res20\_1w1a as irnet  irnet.\_\_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，在运行过程中是用Ctrl+C强制打断：PytorchPlus.QNNs.irnet\_res20\_1w1a.\_\_main\_\_.main() | | 运行中抛出包含打断位置的异常， File "C:\Users\17398\Desktop\PytorchPlus-1.0.1\PytorchPlus\QNNs\ irnet\_res20\_1w1a \modules\irnet\_1w1a.py", line 22, in forward  self.dilation, self.groups)  KeyboardInterrupt | | 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

由于目标检测模块对环境的依赖较多，与其他模块的差异较大，所以不强制用户安装，而是作为附加的模块单独打包，用户可选择另外安装。

针对目标检测模块进行单元测试，仅对该模块内部的功能性进行测试。该模块名称为detectron2，在顶层的文件夹下的train\_net.py文件存放着该模块调用的一个示例接口launch\_interface，在应用过程中，用户可以引用调用该接口实现算法运行，并通过args参数来构建不同的网络结构。

### Faster RCNN调用运行

Faster RCNN调用运行测试用例如表6-12所示。

表6-12 TC303-1-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TC303-1 | **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，并成功传参建构Faster RCNN运行。 | | | | |
| **前提和约束** | 1. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员成功编译好安装包； 4. 测试员下载好COCO格式数据集：http://images.cocodataset.org/zips/train2017.zip； 5. 测试员在终端中设置环境变量DETECTRON2\_DATASETS为数据集存储地址； 6. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import detectron2. train\_net | | 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 | 在detection\_simply的目录下，运行指令python detectron2/train\_net.py \  --config-file configs/COCO-Detection/faster\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 屏幕输出：  “Starting training from iteration 0”表示模型已经开始训练，未抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 在detection\_simply的目录下，运行包含错误参数名称的指令python detectron2/train\_net.py \  --CONFIG\_FILE configs/COCO-Detection/faster\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 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。 | | | | |

### Fast RCNN调用运行

Fast RCNN调用运行测试用例如表6-13所示。

表6-13 TC303-2-Fast RCNN调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Fast RCNN调用运行 | **测试用例标识** | 用例TC303-2 | **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，并成功传参建构Fast RCNN运行。 | | | | |
| **前提和约束** | 1. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员成功编译好安装包； 4. 测试员下载好COCO格式数据集：http://images.cocodataset.org/zips/train2017.zip； 5. 测试员在终端中设置环境变量DETECTRON2\_DATASETS为数据集存储地址； 6. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import detectron2. train\_net | | 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 | 在detection\_simply的目录下，运行指令python detectron2/train\_net.py \  --config-file configs/COCO-Detection/fast\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 屏幕输出：  “Starting training from iteration 0”表示模型已经开始训练，未抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 在detection\_simply的目录下，运行包含错误参数名称的指令python detectron2/train\_net.py \  --CONFIG\_FILE configs/COCO-Detection/fast\_rcnn\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 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。 | | | | |

### RetinaNet调用运行

RetinaNet调用运行测试用例如表6-14所示。

表6-14 TC303-3-RetineNet调用运行测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | RetinaNet调用运行 | **测试用例标识** | 用例TC303-3 | **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，并成功传参建构RetinaNet运行。 | | | | |
| **前提和约束** | 1. 系统环境及版本符合本API的要求； 2. 测试员可以在Python环境中导入满足版本要求的torch, numpy, os等软件依赖包； 3. 测试员成功编译好安装包； 4. 测试员下载好COCO格式数据集：http://images.cocodataset.org/zips/train2017.zip； 5. 测试员在终端中设置环境变量DETECTRON2\_DATASETS为数据集存储地址； 6. 测试员可以处理得到符合要求的函数输入内容。 | | | | |
| **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torch, numpy, os等软件依赖包以及本模块中包含接口的部分：  import detectron2. train\_net | | 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 | 在detection\_simply的目录下，运行指令python detectron2/train\_net.py \  --config-file configs/COCO-Detection/retinanet\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 屏幕输出：  “Starting training from iteration 0”表示模型已经开始训练，未抛出异常。 | | 实际结果与预期结果一致 |
| 3 | 在detection\_simply的目录下，运行包含错误参数名称的指令python detectron2/train\_net.py \  --CONFIG\_FILE configs/COCO-Detection/retinanet\_R\_50\_FPN\_1x.yaml \  --num-gpus 1 SOLVER.IMS\_PER\_BATCH 2 SOLVER.BASE\_LR 0.0025 | | 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

阅读理解模块在PytorchPlus模块下的子模块名为RC，可调用的API如下图6.3所示，本小节描述的是处理数据功能和阅读理解算法的模型构建、训练、预测过程的测试步骤。



图6.3 RC模块结构

### 6.5.1 RC模块处理数据功能测试

处理数据功能测试用例如表6-15所示

表6-15 TC304-1-RC模块调用处理数据功能测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | RC模块处理数据功能测试 | **测试用例标识** | 用例TC304-1 | **测试需求标识** | TR304 |
| **简要描述** | 本测试验证是否可以成功调用RC模块调用处理数据功能。 | | | | |
| **前提和约束** | 1. 当前路径下有config.py文件和保存训练数据的文件夹，有待处理文件； 2. config.py文件中已经对的处理数据功能需要的参数进行赋值，具体要求如下：    1. target\_dir 为保存训练数据的文件夹；    2. train\_file, dev\_file, predict\_file 为原始训练、验证、预测文本；    3. word2idx\_file, char2idx\_file为词、字符序号文件；    4. word\_emb\_file, char\_emb\_file为词、字符编码文件；    5. [train, dev, predict]\_record\_file, [train, dev, predict]\_eval\_file 为处理数据的输出文件，用于模型训练、验证和预测。 | | | | |
| **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 PytorchPlus.RC.preproc import Preproc  import config | | 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() | | 能成功创建对象，不会出现异常 | | 实际结果与预期结果一致 |
| 3 | 输入 p.process\_train\_file(config) | | 屏幕输出“Generating xx examples”和处理进度条，结果保存在正确路径。 | | 实际结果与预期结果一致 |
| 4 | 输入  p.process\_predict\_file(config) | | 屏幕输出“Generating xx examples”和处理进度条，结果保存在正确路径。 | | 实际结果与预期结果一致 |
| 5 | 在可以浏览json文件的编辑器中，检查输出文件与输入文件的“context”“question”“answer”在内容与位置标签上是否对应 | | 处理后训练文件正确地将原始文本中的字符粒度的位置标签转换成了单词粒度的位置标签。 | | 正确处理全部文本条目 |
| **备注** | 可通过直接运行文件test\_case.py进行测试 | | | | |

### 6.5.2 RC模块阅读理解算法功能测试

调用阅读理解算法测试用例如表6-16所示

表6-16 TC304-2-RC模块调用阅读理解算法测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | RC模块阅读理解算法测试 | **测试用例标识** | 用例TC304-2 | **测试需求标识** | TR304 |
| **简要描述** | 本测试验证是否可以成功加载、运行RC模块阅读理解算法，完成模型训练、预测功能 | | | | |
| **前提和约束** | 1. 当前路径下有config.py文件和保存模型数据的文件夹结构如图6.5.2-1，有训练数据； 2. 已经加载了torch模块 3. config.py文件中已经对需要的参数进行赋值，具体要求如下：    1. target\_dir 为训练数据所在的文件夹；    2. save\_dir 为保存模型的文件夹；    3. [train, dev, predict]\_record\_file, [train, dev, predict]\_eval\_file 为模型训练、验证和预测的输入文件；    4. word2idx\_file, char2idx\_file为词、字符序号文件；    5. word\_emb\_file, char\_emb\_file为词、字符编码文件；    6. 其他定义模型的超参数和训练配置参数。 | | | | |
| **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 PytorchPlus.RC.renders import QATrain, QAPredict  from PytorchPlus.RC.models import QANet  import config | | 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model = QANet(config).to(device) | | 能成功创建对象，不会出现异常 | | 实际结果与预期结果一致 |
| 3 | 输入  QATrain(config,model,device) | | 屏幕输出训练进度条和验证结果，模型保存在正确路径。 | | 屏幕输出的“loss”数据随训练进行有明显下降 |
| 4 | 输入  QAPredict(config, model, device) | | 屏幕输出处理进度条，预测结果保存在正确路径。 | | 实际结果与预期结果一致 |
| 5 | 在可以浏览json文件的编辑器中打开预测结果 | | 查看预测结果是否有一定的合理性 | | 有一定合理性 |
| **备注** | 可通过直接运行文件test\_case.py进行测试 | | | | |



图6.4 测试用例的文件夹结构

## 主动学习模块

针对主动学习模块进行功能测试，旨在对封装完毕的软件中该模块部分的功能性进行测试。由于进行了所有模块的封装，因此该模块的代码结构多了一级，顶层PytorchPlus下的文件夹active\_learning下定义了各个主动学习采样策略的类，和一些功能类。该模块所实现的所有采样策略的调用接口都在\_\_init\_\_.py文件中引入，在应用过程中，用户只需引用该部分即可实现算法运行。

该模块中各个算法涉及的输入类型规范，见实验三中的软件设计文档。

### BanditDiscreteSampler调用运行

BanditDiscreteSampler调用运行测试用例如表6-17所示。

表6-17 TC305-1-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TC305-1 | **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 PytorchPlus.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) | | 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\_(N, labeled, eval\_acc, model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, eval\_acc, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, eval\_acc, model) | | 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6-18所示。

表6-18 TC305-2-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TC305-2 | **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 PytorchPlus.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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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6-19所示。

表6-19 TC305-1-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TC305-3 | **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 PytorchPlus.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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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6-20所示。

表6-20 TC305-4-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TC305-4 | **测试需求标识** | TR305-4 |
| **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 PytorchPlus.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\_(N, labeled, model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model) | | 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6-21所示。

表6-21 TC305-5-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TC305-5 | **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 PytorchPlus.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) | | 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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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6-22所示。

表6-22 TC305-6-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TC305-6 | **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 PytorchPlus.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\_(N, labeled, model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model) | | 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6-23所示。

表6-23 TC305-7-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TC305-7 | **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 PytorchPlus.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) | | 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\_(N, labeled, model=model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model=model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model=model) | | 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6-24所示。

表6-24 TC305-8-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TC305-8 | **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 PytorchPlus.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\_(N, labeled, model) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled, model) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled, model) | | 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6-25所示。

表6-25 TC305-9-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TC305-9 | **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 PytorchPlus.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\_(N, labeled) | | 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 | 进行边界测试，当N > 剩余未标注样本数目时，运行batch = sampler.select\_batch\_(N, labeled) | | 返回None，并提示调用者未标注样本不足 | | 实际结果与预期结果一致 |
| 5 | 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\_(N, labeled) | | 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。 | | | | |

## 加载和保存模型模块

### 加载模型测试用例

本测试主要针对需求规格说明书中的加载模型需求进行测试，由于每个模块下的每种算法模型加载时需要的参数类型和数目不一致，所以使用每个模块中较具有代表性的算法进行测试，分别构造输入。

测试用例如表6-26所示。

表6-26 TC306-1-加载模型测试用例

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 加载模型 | | **测试用例标识** | 用例TC306-1 | | **测试需求标识** | TR306 | |
| **简要描述** | | 本测试通过测试加载各个模块内的代表性算法模型，验证本项目加载模型需求实现是否存在缺陷。 | | | | | | | |
| **前提和约束** | | 1. 系统环境及版本符合本API的要求； 2. PyTorch平台正常运行； 3. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 4. 已有模型被保存至本地或服务器中。 | | | | | | | |
| **测试方法** | | 黑盒测试 | | | | | | | |
| **测试过程描述** | | | | | | | | | |
| **序号** | **测试步骤** | | **输入** | | | **测试结果** | | | |
| **预期结果** | | | **评价准则** |
| 1 | 测试员在PyThon环境中输入命令导入相应模块中的目标方法 | | Import DeepBox.xxx.xxx | | | 成功导入目标包，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 2 | 测试员加载目标模型 | | 加载对抗样本生成模块FGSM算法模型，输入语句：model = torch.load(path) | | | 成功导入FGSM算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 加载神经网络量化模块QNNs算法模型，输入语句：model.load\_state\_dict(state\_dict) | | | 成功导入QNNs算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 加载目标检测模块Faster RCNN算法模型，输入语句：model = build\_model(config)  checkpointer =DetectionCheckpointer(model)  checkpointer.load(config.MODEL.WEIGHTS) | | | 成功导入Faster RCNN算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 加载阅读理解模块QANet算法模型，输入语句：model = torch.load(path) | | | 成功导入QANet算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 3 | 输入语句中加载QANet模型路径不存在 | | model = torch.load(“”) | | | 抛出路径错误异常 | | | 实际结果与预期结果一致 |
| 4 | 输入语句任意字段异常 | | 无 | | | 抛出参数错误异常 | | | 实际结果与预期结果一致 |
| 备注 | 由于主动学习模型是对输入数据进行处理，输出是处理后的数据，不存在加载模型的概念，所以这里不包括主动学习模块加载模型的测试。 | | | | | | | | |

### 保存模型测试用例

本测试主要针对需求规格说明书中的保存模型需求进行测试，由于每个模块下的每种算法模型保存时需要的参数类型和数目不一致，且返回类型也不一致，所以以每个模块的代表性算法进行测试，分别构造输入。

测试用例如表6-27所示。

表6-27 TC308-1-保存模型测试用例

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 保存模型 | | **测试用例标识** | 用例TC308-1 | | **测试需求标识** | TR308 | |
| **简要描述** | | 本测试通过测试保存各个模块内的代表性算法模型，验证本项目保存模型需求实现是否存在缺陷。 | | | | | | | |
| **前提和约束** | | 1. 系统环境及版本符合本API的要求； 2. PyTorch平台正常运行； 3. 测试员已成功在PyThon环境中中导入torch, numpy, os等基础库； 4. 成功导入项目包，且目标算法模型已经训练完成。 | | | | | | | |
| **测试方法** | | 黑盒测试 | | | | | | | |
| **测试过程描述** | | | | | | | | | |
| **序号** | **测试步骤** | | **输入** | | | **测试结果** | | | |
| **预期结果** | | | **评价准则** |
| 1 | 测试员保存目标模型 | | 保存对抗样本生成模块FGSM算法模型，输入语句：  torch.save({'static\_dict':model.state\_dict()}, path) | | | 成功保存FGSM算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 保存神经网络量化模块QNNs算法模型，输入语句：  state = {  'state\_dict': model.state\_dict(),  'best\_prec1': best\_prec1,  }  torch.save(state, filename) | | | 成功保存QNNs算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 保存目标检测模块Faster RCNN算法模型，输入语句：  torch.save({'static\_dict':model.state\_dict()}, path) | | | 成功保存Faster RCNN算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 保存阅读理解模块QANet算法模型，输入语句：  torch.save({'static\_dict':model.state\_dict()}, path) | | | 成功保存QANet算法模型，不会抛出异常。 | | | 实际结果与预期结果一致 |
| 2 | 输入语句中保存QANet模型路径不存在 | | torch.save({'static\_dict':model.state\_dict()}, “”) | | | 抛出路径错误异常 | | | 实际结果与预期结果一致 |
| 3 | 输入语句被保存模型存在异常 | | 无 | | | 抛出参数错误异常 | | | 实际结果与预期结果一致 |
| 备注 | 由于主动学习模型是对输入数据进行处理，输出是处理后的数据，不存在保存模型的概念，所以这里不包括主动学习模块保存模型的测试。 | | | | | | | | |

# 非功能测试

## 用户友好性

### 7.1.1 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。与之对比的是，使用PyTorch平台自身搭建网络和构造同样的算法过程，对比耗时。

### 7.1.2 测试用例描述

测试用例如表7-1所示。

表7-1 TC701-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TC701 | | **测试需求标识** | TR701 | |
| **简要描述** | | 本测试通过初学者用户的上手实操，验证本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PytorchPlus工具包，索引到API中的各个算法，并记录索引过程平均用时 | | | | 正常运行 | | | 实际结果与预期结果一致 |
| 2 | 通过PyTorch语法手动完成算法的构建，记录构建过程的平均用时 | | | | 正常运行 | | | 实际结果与预期结果一致 |
| 3 | 对比前两个步骤用时 | | | | 步骤1入操作且调用简单，耗时较少，步骤2构建过程复杂，相比步骤1耗时较长 | | | 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。

### 7.2.2 测试用例描述

测试用例如表7-2所示。

表7-2 TC702-健壮性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 鲁棒性 | **测试用例标识** | 用例TC702 | | **测试需求标识** | TR702 | |
| **简要描述** | | 本测试主要验证本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API中的各个算法 | | | | 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

### 7.3.1 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。

### 7.3.2 测试用例描述

测试用例如表7-3所示。

表7-3 TC703-扩展性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 扩展性 | **测试用例标识** | 用例TC703 | | **测试需求标识** | TR703 | |
| **简要描述** | | 本测试主要验证本API的扩展性。 | | | | | | |
| **前提和约束** | | 准备好对抗样本生成模块待扩展的算法源码，并保证符合软件环境依赖 | | | | | | |
| **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 | 测试员将准备好的源码放入对抗样本生成模块的adversarial\_attack.py文件中，并将其转化为函数接口的形式。 | | | | 成功构造扩展函数接口 | | | 实际结果与预期结果一致 |
| 2 | 仿照4.2中模块单元测试的用例对新扩展内容进行单元测试。 | | | | 通过单元测试 | | | 实际结果与预期结果一致 |
| 3 | 将整个新构造好的API进行打包，得到新版本的软件安装包。 | | | | 打包封装过程顺利执行 | | | 实际结果与预期结果一致 |
| 4 | 仿照6.2中功能测试的用例对新扩展内容进行功能测试。 | | | | 通过功能测试 | | | 实际结果与预期结果一致 |

# 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。本文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，分为用户友好性、健壮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9.1 功能测试用例覆盖表

|  |  |  |  |
| --- | --- | --- | --- |
| **功能测试用例覆盖表** | | | |
| 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 | TC301-1 | TC301-1-FGSM调用运行测试用例 |
| TC301-2 | TC301-2-PGD调用运行测试用例 |
| TC301-3 | TC301-3-STEP-LL调用运行测试用例 |
| TC301-4 | TC301-4-MI-FGSM调用运行测试用例 |
| TC301-5 | TC301-5-C&W调用运行测试用例 |
| TC301-6 | TC301-6-Fourier Corruption调用运行测试用例 |
| TR302 | 3.1.2 生成量化网络模块 | TC302-1 | TC302-1-BNN调用运行测试用例 |
| TC302-2 | TC302-2-XNOR调用运行测试用例 |
| TC302-3 | TC302-3-HORQ调用运行测试用例 |
| TC302-4 | TC302-4-BWN调用运行测试用例 |
| TC302-5 | TC302-5-IR-Net调用运行测试用例 |
| TR303 | 3.1.3调用目标检测模块 | TC303-1 | TC303-1-Faster RCNN调用运行测试用例 |
| TC303-2 | TC303-2-Fast RCNN调用运行测试用例 |
| TC303-3 | TC303-3-RetinaNet调用运行测试用例 |
| TR304 | 3.1.5 调用阅读理解模块 | TC304-1 | TC304-1-RC模块调用处理数据功能测试用例 |
| TC304-2 | TC304-2-RC模块调用阅读理解算法测试用例 |
| TR305 | 3.1.4调用主动学习模块 | TC305-1 | TC305-1-BanditDiscreteSampler调用运行测试用例 |
| TC305-2 | TC305-2-GraphDensitySampler调用运行测试用例 |
| TC305-3 | TC305-3-HierarchicalClusterAL调用运行测试用例 |
| TC305-4 | TC305-4-InformativeClusterDiverseSampler调用运行测试用例 |
| TC305-5 | TC305-5-kCenterGreed调用运行测试用例 |
| TC305-6 | TC305-6-MarginAL调用运行测试用例 |
| TC305-7 | TC305-7-MixtureOfSampler调用运行测试用例 |
| TC305-8 | TC305-8-RepresentativeClusterMeanSampling调用运行测试用例 |
| TC305-9 | TC305-9-UniformSampling调用运行测试用例 |

表9.2 非功能需求测试用例覆盖表

|  |  |  |  |
| --- | --- | --- | --- |
| **非功能需求测试用例覆盖表** | | | |
| 需求编号 | 对应需求文档  章节 | 用例编号 | 用例名称 |
| TR701 | 3.2.2 用户友好性 | TC701 | TC701-用户友好性测试用例 |
| TR702 | 3.2.3 健壮性 | TC702 | TC702-健壮性测试用例 |
| TR703 | 3.2.4 扩展性 | TC703 | TC703-扩展性测试用例 |