

**基于Torch平台的神经网络压缩**

**研究与应用**

**测试需求说明书**

北京航空航天大学

2017-05

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 | 2017/5/19 | 付强 陈伟民 曹进 李恬霖 | 付强 陈伟民 曹进 李恬霖 | 该文档的最初版本 |
| 1.1 | 2017/5/22 | 付强 陈伟民 | 曹进 李恬霖 | 修改文档格式，添加扩展需求、非功能需求部分 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1. 引言 5](#_Toc483226520)

[1.1 编写目的 5](#_Toc483226521)

[1.2 过程分析 5](#_Toc483226522)

[1.3 参考文献 5](#_Toc483226523)

[2. 测试计划 6](#_Toc483226524)

[2.1 目标 6](#_Toc483226525)

[2.2 测试工具及相关部件简介 6](#_Toc483226526)

[2.2.1 Lua 6](#_Toc483226527)

[2.2.2 C语言 7](#_Toc483226528)

[2.3 测试数据 7](#_Toc483226529)

[2.4 测试策略 8](#_Toc483226530)

[2.5 测试通过准则 8](#_Toc483226531)

[3. 安装测试 8](#_Toc483226532)

[3.1 概述 8](#_Toc483226533)

[3.2 测试用例 9](#_Toc483226534)

[4. 功能测试 10](#_Toc483226535)

[4.1 概述 10](#_Toc483226536)

[4.2 导入工具包 11](#_Toc483226537)

[4.3 文件数据读取 12](#_Toc483226538)

[4.4 搭建MLP神经网络 12](#_Toc483226539)

[4.5 搭建卷积神经网络 13](#_Toc483226540)

[4.6 训练神经网络模型 14](#_Toc483226541)

[4.7 神经网络模型输出 15](#_Toc483226542)

[5. 扩展功能测试 16](#_Toc483226543)

[6. 非功能测试 16](#_Toc483226544)

[6.1 高效性 16](#_Toc483226545)

[6.1.1 测试策略描述 16](#_Toc483226546)

[6.1.2 测试用例描述 16](#_Toc483226547)

[6.2 用户友好性 17](#_Toc483226548)

[6.2.1 测试策略描述 17](#_Toc483226549)

[6.2.2 测试用例描述 17](#_Toc483226550)

[6.3 可修改性 18](#_Toc483226551)

[6.3.1 测试策略描述 18](#_Toc483226552)

[6.3.2 测试用例描述 18](#_Toc483226553)

[6.4 鲁棒性 18](#_Toc483226554)

[6.4.1 测试策略描述 18](#_Toc483226555)

[6.4.2 测试用例描述 19](#_Toc483226556)

[7. 测试方案的自我评价 19](#_Toc483226557)

# 引言

## 编写目的

本文档是软件工程实验课程中对所选软件及其扩展功能的测试需求分析的过程说明文档，主要描述了测试计划的制定，测试需求的分析，测试用例产生原因和项目中的测试用例设计说明等。测试人员可根据本文档中的测试计划和方案结合测试用例，对软件基本功能和扩展功能进行测试。

## 过程分析

小组成员在讨论后，决定依据编写的《基于Torch平台的神经网络压缩研究与应用需求规格说明书》对Torch平台的功能需求、非功能需求和扩展需求等进行测试。

为了使测试用例更加规范，我们小组约定了几条在编写测试用例时需要遵守的基本规则：首先，测试用例的前置条件应该与对应需求用例中的条件相对应；其次，测试用例的主流程应该与对应需求用例的主流程相对应，至少包括用例中参与者与系统的交互过程；最后，测试用例中应该给出明确的判断测试用例是否通过的标准。

## 参考文献

[1] Torch官方关于nn包的文档[OL].https://github.com/torch/nn/tree/master/doc

[2] Han S, Mao H, Dally W J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding[J]. Fiber, 2015.

[3]《基于Torch平台的神经网络压缩研究与应用需求规格说明书v2.2》

# 测试计划

## 目标

本次软件工程实验中，涉及到的“系统”主要有两个：一个是所选择的软件Torch，这部分软件代码是开源的；另一个是根据实际需求对软件Torch的扩展，这部分的代码全部由组员来完成。对于这两个“系统”所做的测试，其目的应该是不一样的。

具体在本项目中，对于所选的Torch，是一个用Lua编写的支持机器学习算法的计算框架。对于这个开源软件而言，测试的目的有两类：

1. 测试其功能是否完整，即是否能够为搭建神经网络提供丰富并有效的工具包，以及是否能够通过模块分割等方式来简化用户操作，方便用户搭建神经网络。
2. 测试其性能是否优于其他神经网络框架。

在本课程中，实现了Torch的基本配置、神经网络的搭建和测试、神经网络压缩后的测试。对于扩展部分的测试，目的是验证其是否满足《基于Torch平台的扩展：神经网络压缩算法研究》中提到的两种压缩算法是否有效，包括修改工具包后搭建神经网络所需的各层功能实现的是否完整、稳定、可用，以及在不同压缩比下神经网络在MNIST数据集上的表现，其中测试的结果可以反馈给开发人员作进一步的改进。

## 测试工具及相关部件简介

本小节主要介绍在以后的实际的测试过程中可能用到的语言和工具。

## Lua

Lua 是一种轻量小巧的脚本语言，用标准C语言编写并以源代码形式开放， 其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。

Lua具有几个重要的特性。

* 轻量级: 它用标准C语言编写并以源代码形式开放，编译后仅仅一百余K，可以很方便的嵌入别的程序里。
* 可扩展: Lua提供了非常易于使用的扩展接口和机制：由宿主语言(通常是C或C++)提供这些功能，Lua可以使用它们，就像是本来就内置的功能一样。
* 其它特性:
  + 支持面向过程(procedure-oriented)编程和函数式编程(functional programming)；
  + 自动内存管理；只提供了一种通用类型的表（table），用它可以实现数组，哈希表，集合，对象；
  + 语言内置模式匹配；闭包(closure)；函数也可以看做一个值；提供多线程（协同进程，并非操作系统所支持的线程）支持；
  + 通过闭包和table可以很方便地支持面向对象编程所需要的一些关键机制，比如数据抽象，虚函数，继承和重载等。

## C语言

C语言是一门通用计算机编程语言，应用广泛。C语言的设计目标是提供一种能以简易的方式编译、处理低级存储器、产生少量的机器码以及不需要任何运行环境支持便能运行的编程语言。

尽管C语言提供了许多低级处理的功能，但仍然保持着良好跨平台的特性，以一个标准规格写出的C语言程序可在许多电脑平台上进行编译，甚至包含一些嵌入式处理器（单片机或称MCU）以及超级电脑等作业平台。

## 测试数据

本测试实践的测试数据集是MNIST。

MNIST是一个手写数字数据库，它是NIST数据库的一个子集，现在已成为图像识别领域用来测试自己的算法的一个基准数据库，它的训练集由60000张手写数字图片样本组成，测试集包含了10000个样本，其中所有的图片样本都经过了尺寸标准化和中心化，图片的大小固定为28\*28。这个数据集由四部分组成，一个训练图片集，一个训练标签集，一个测试图片集，一个测试标签集。从官网获取的并不是并不是普通的文本文件或是图片文件，而是一个压缩文件，下载并解压出来，是二进制文件，这个文件里面包含了60000个标签内容，每一个标签的值为0到9之间的一个数。

## 测试策略

我们准备进行测试的过程为Torch系统测试，系统测试的测试策略如下：

系统测试的主要目的是在于验证软件的功能和性能及其他特性是否与用户的要求一致，主要是下列类型的测试:

1. 安装测试：确定Torch在Linux中能够被正常编译，安装。
2. 功能测试：确定Torch是否完全的实现了《需求文档》中的功能点。主要内容包括提供丰富的工具包和模块化搭建神经网络，确定Torch平台满足搭建神经网络所需的基本功能，以及所搭建神经网络在MNIST数据集的表现。
3. 非功能测试：确定Torch在一定条件下，在搭建神经网络的过程中，是否具有高效性、可修改性等特性。
4. 扩展需求测试：对Torch平台的工具包进行修改后，测试搭建的神经网络在MNIST数据集上的表现，并与修改前在准确率的表现上进行比较。

## 测试通过准则

Torch系统通过测试的准则，即当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Torch本身时为测试通过，反之测试失败。

# 安装测试

## 概述

确定Torch在多种操作系统（Linux/MacOS）中都能够被正常编译，安装。

## 测试用例

表3-1 TC001-Linux下安装测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Linux下安装测试 | **测试用例标识** | TC001 | **对应测试需求文档中的需求** | 2.4节的第一条 | |
| **简要描述** | 本测试主要测试Torch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Torch，配置安装组件 | | 安装成功 | | 实际结果与预期结果一致 |  |
| 3 | 启动Torch | | 启动成功 | | 实际结果与预期结果一致 |  |

表3-2 TC002-Mac OS下安装测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Mac OS下安装测试 | **测试用例标识** | TC002 | **对应测试需求文档中的需求** | 2.4节的第一条 | |
| **简要描述** | 本测试主要测试Torch在Mac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Torch，配置安装组件 | | 安装成功 | | 实际结果与预期结果一致 |  |
| 3 | 启动Torch | | 启动成功 | | 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。本地化软件的功能测试，用于验证应用程序或网站对目标用户能正确工作。使用适当的平台、浏览器和测试脚本，以保证目标用户的体验将足够好，就像应用程序是专门为该市场开发的一样。功能测试是为了确保程序以期望的方式运行而按功能要求对软件进行的测试，通过对一个系统的所有的特性和功能都进行测试确保符合需求和规范。

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。

## 导入工具包

导入工具包测试用例如表4-1所示。

表4-1 TC401-导入工具包测试用例

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **测试用例名称** | 导入工具包 | **测试用例标识** | 用例TC401 | **测试需求标识** | TR401 |
| **简要描述** | 本测试验证是否可以在Torch平台上导入path包，torch包，nn包。 | | | | |
| **前提和约束** | 1）Torch7测试员打开Torch平台；  2）Torch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 | 输入命令导入path包和nn包：  require ‘path’  require ‘nn’  paths.filep(“exapmle.zip”) | | Torch平台输出：  true | | 实际结果与预期结果一致。 |
| 2 | Torch7测试员在Torch平台上输入：  require ‘torch’  trainset = torch.load(‘cifar10-train.t7’)  print(trainset) | | Torch平台输出：  {  data:ByteTensor-size:10000\*3\*32\*32 label:ByteTensor-size:10000  } | | 实际结果与预期结果一致。 |
| **备注** | 说明：  1）Torch 7测试员在Torch平台输入的命令符合Lua语言语法。 | | | | |

## 文件数据读取

文件数据读取测试用例如表4-2所示。

表4-2 TC402-文件数据读取测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | 文件数据读取 | | **测试用例标识** | 用例TC402 | **测试需求标识** | | TR402 | |
| **简要描述** | 本测试验证是否可以在Torch平台上成功读取本地文件中的数据。 | | | | | | | |
| **前提和约束** | 1）Torch7测试员打开Torch平台；  2）Torch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mnist包：require 'mnist' | | | |  | |  |
| 2 | | 导入mnist训练数据mnist. traindataset()和测试数据mnist.testdataset() | | | |  | |  |
| 3 | | 数据预处理，按指定的大小和数据（带标签）构建训练集trainset和验证集validationset | | | |  | |  |
| 4 | | 打印训练集和验证集的前10张图像 | | | | Torch平台输出按照格式输出的数据与文件中的数据一致。 | | 实际结果与预期结果一致。 |
| **备注** | | 说明：   1. Torch 7测试员在Torch平台输入的命令符合Lua语言语法。 | | | | | | |

## 搭建MLP神经网络

搭建MLP神经网络测试用例如表4-3所示。

表4-3 TC403-搭建MLP神经网络测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | 搭建MLP神经网络 | **测试用例标识** | 用例TC403 | | **测试需求标识** | TR403 |
| **简要描述** | 本测试验证是否可以在Torch平台上成功搭建MLP神经网络模型。 | | | | | |
| **前提和约束** | 1）Torch7测试员打开Torch平台；  2）Torch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torch、nn包 | | |  | |  |
| 2 | 建立线性容器nn.sequential() | | |  | |  |
| 3 | 添加线性层nn.Linear() | | |  | |  |
| 4 | 添加激活函数 | | |  | |  |
| 5 | 添加分类器nn.LogSoftMax() | | |  | |  |
| 6 | 添加代价函数nn.ClassNLLCriterion() | | |  | |  |
| 7 | 打印网络结构print(model) | | | Torch平台输出与设计的网络结构一致 | | 实际结果与预期结果一致 |
| **备注** | 说明：   1. Torch 7测试员在Torch平台输入的命令符合Lua语言语法； 2. Torch7测试员在Torch平台上建立的MLP神经网络模型，遵循合法的MLP神经网络搭建流程； 3. MLP神经网络：一种前向结构的人工神经网络。 | | | | | |

## 搭建卷积神经网络

搭建卷积神经网络测试测试用例如表4-4所示。

表4-4 TC404-搭建卷积神经网络测试用例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 搭建卷积神经网络 | **测试用例标识** | 用例TC404 | **测试需求标识** | | TR404 |
| **简要描述** | | 本测试验证是否可以在Torch平台上成功搭建卷积神经网络模型。 | | | | | |
| **前提和约束** | | 1）Torch7测试员打开Torch平台；  2）Torch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torch、nn包 | | | | |  |  |
| 2 | 建立线性容器nn.sequential() | | | | |  |  |
| 3 | 添加线性层nn.SpatialConvolution() | | | | |  |  |
| 4 | 添加池化层nn.SpatialMaxPooling() | | | | |  |  |
| 5 | 添加激活函数nn.ReLU() | | | | |  |  |
| 6 | 添加全连接层nn.Linear() | | | | |  |  |
| 7 | 添加分类器nn.LogSoftMax() | | | | |  |  |
| 8 | 添加代价函数nn.ClassNLLCriterion() | | | | |  |  |
| 9 | 打印网络结构print(model) | | | | | Torch平台输出与设计的网络结构一致 | 实际结果与预期结果一致 |
| **备注** | 说明：   1. Torch7测试员在Torch平台输入的命令符合Lua语言语法； 2. Torch7测试员在Torch平台上建立的卷积神经网络模型，遵循合法的卷积神经网络搭建流程。 | | | | | | |

## 训练神经网络模型

训练神经网络模型测试用例如表4-5所示。

表4-5 TC405-训练神经网络模型测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 训练神经网络模型 | **测试用例标识** | 用例TC405 | | **测试需求标识** | TR405 | |
| **简要描述** | | 本测试验证是否可以在Torch平台上成功训练神经网络模型，以得到模型的各个参数。 | | | | | | |
| **前提和约束** | | 1. Torch平台正常运行； 2. Torch平台神经网络搭建成功，并在Torch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 | 定义网络训练参数sgd\_params | | | |  | | |  |
| 2 | 定义单个迭代过程step()： | | | |  | | |  |
| 3 | 定义求导过程feval()：  定义前向过程：model:forward()  定义后向过程：model:backward()  计算损失criterion:forward() | | | |  | | |  |
| 4 | 调用优化算法optim.sgd(feval, x, sgd\_params) | | | |  | | |  |
| 5 | 进行神经网络训练 | | | | Torch平台输出模型训练过程中，每次迭代的输出误差以及迭代次数 | | | 实际结果与预期结果一致 |
| **备注** | 说明：  1）Torch 7测试员在Torch平台输入的命令符合Lua语言语法； | | | | | | | |

## 测试神经网络模型

测试神经网络模型测试用例如表4-6所示。

表4-6 TC406-测试神经网络模型测试用例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 测试神经网络模型 | **测试用例标识** | 用例TC406 | **测试需求标识** | TR406 | |
| **简要描述** | | 本测试验证在Torch平台上成功训练得到的神经网络模型是否有理想的输出结果。 | | | | | |
| **前提和约束** | | 1. Torch平台正常运行；   2）Torch平台神经网络搭建成功，并在Torch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 | 定义输入数据inputs | | | |  | |  |
| 2 | 定义输出数据（标签）label | | | |  | |  |
| 3 | 前向过程得到网络输出model:forward(inputs) | | | |  | |  |
| 4 | 将网络输出与标签做对比，求得准确率 | | | | 得到网络对所求问题的预测准确率 | | 实际结果与预期结果一致 |

# 扩展功能测试

## 压缩MLP网络测试

压缩MLP网络模型测试用例如表5-1所示。

表5-1 TC501-压缩MLP神经网络测试用例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 压缩MLP神经网络 | **测试用例标识** | 用例TC501 | **测试需求标识** | TR501 | |
| **简要描述** | | 本测试验证在Torch平台上扩展的压缩神经网络算法在MNIST数据集上的时间和准确度的表现 | | | | | |
| **前提和约束** | | 1. Torch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torch、nn包 | | | |  | |  |
| 2 | 建立线性容器nn.sequential() | | | |  | |  |
| 3 | 添加修改线性层nn.RPLinear() | | | |  | |  |
| 4 | 添加激活函数y = x / y = Htan(x) | | | |  | |  |
| 5 | 添加分类器nn.LogSoftMax() | | | |  | |  |
| 6 | 添加代价函数nn.ClassNLLCriterion() | | | |  | |  |
| 7 | 按照用例TC405训练网络 | | | |  | |  |
| 8 | 按照用例TC406测试网络并求得准确率与时间 | | | | 得到网络对所求问题的预测准确率和时间 | | 在基本不损失准确率的情况下所用时间减少 |

## 压缩卷积网络测试

压缩卷积网络模型测试用例如表5-2所示。

表5-2 TC502-压缩卷积神经网络测试用例

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 压缩卷积神经网络 | **测试用例标识** | 用例TC502 | **测试需求标识** | TR502 | |
| **简要描述** | | 本测试验证在Torch平台上扩展的压缩神经网络算法在MNIST数据集上的时间和准确度的表现 | | | | | |
| **前提和约束** | | 1. Torch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torch、nn包 | | | |  | |  |
| 2 | 建立线性容器nn.sequential() | | | |  | |  |
| 3 | 添加修改线性层nn.RPSpatialConvolution() | | | |  | |  |
| 4 | 添加池化层nn.SpatialMaxPooling() | | | |  | |  |
| 5 | 添加激活函数y = x / y = Htan(x) | | | |  | |  |
| 6 | 添加修改全连接层nn.RPLinear() | | | |  | |  |
| 7 | 添加分类器nn.LogSoftMax() | | | |  | |  |
| 8 | 添加代价函数nn.ClassNLLCriterion() | | | |  | |  |
| 9 | 按照用例TC405训练网络 | | | |  | |  |
| 10 | 按照用例TC406测试网络并求得准确率与时间 | | | | 得到网络对所求问题的预测准确率和时间 | | 在基本不损失准确率的情况下所用时间减少 |

# 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torch和tensorflow两种主流深度学习系统架构实现相同结构的神经网络，比较两者的计算效率。

## 测试用例描述

测试用例如表6-1所示。

表6-1 TC601-高效性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 高效性 | **测试用例标识** | 用例TC601 | | **测试需求标识** | TR601 | |
| **简要描述** | | 本测试测试torch和tensorflow两种主流深度学习系统架构在相同软硬件条件下的计算效率，验证torch的高效性。 | | | | | | |
| **前提和约束** | | 1. Torch平台正常运行； 2. Torch平台神经网络搭建成功，并在Torch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 | 在Torch上定义神经网络模型，并进行训练 | | | |  | | |  |
| 2 | 在Tensorflow上定义相同的神经网络模型，并进行训练 | | | |  | | |  |
| 3 | 比较两者的计算效率 | | | | Torch平台计算效率优于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）打开Torch平台之后，能够结合相关教程和自身知识储备，通过图形操作界面，完成简单的神经网络搭建。

## 测试用例描述

测试用例如表6-2所示。

表6-2 TC602-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TC602 | | **测试需求标识** | TR602 | |
| **简要描述** | | 本测试通过初学者用户的上手实操，验证torch的用户友好性。 | | | | | | |
| **前提和约束** | | 1. Torch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 | 打开Torch平台 | | | | 平台正常启动运行 | | | 实际结果与预期结果一致 |
| 2 | 搭建入门级神经网络 | | | | 网络搭建简单，运行成功。 | | | 实际结果与预期结果一致 |

## 可修改性

## 测试策略描述

根据对应的需求文档可知，本系统的扩展需求即为修改Torch平台中卷积层和线性层，可以体现Torch平台的可修改性。所以扩展需求的测试运行成功就验证了Torch平台的可修改性。

## 测试用例描述

测试用例如表6-3所示。

表6-3 TC603-可修改性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 可修改性 | **测试用例标识** | 用例TC603 | | **测试需求标识** | TR603 | |
| **简要描述** | | 本测试通过测试扩展需求是否成功运行，验证torch的用户友好性。 | | | | | | |
| **前提和约束** | | 1. Torch平台正常运行； 2. Torch平台神经网络搭建成功，并在Torch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 | 根据扩展需求修改Torch平台卷积层、线性层代码（具体修改见第5章） | | | |  | | |  |
| 2 | 运行修改代码 | | | | 代码可运行，网络修改成功。 | | | 实际结果与预期结果一致 |

## 鲁棒性

## 测试策略描述

通过向平台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系统鲁棒性。

## 测试用例描述

测试用例如表6-4所示。

表6-4 TC602-用户友好性测试用例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | | 鲁棒性 | **测试用例标识** | 用例TC604 | | **测试需求标识** | TR604 | |
| **简要描述** | | 本测试主要验证torch的鲁棒性。 | | | | | | |
| **前提和约束** | | 无 | | | | | | |
| **测试方法** | | 黑盒测试 | | | | | | |
| **测试过程描述** | | | | | | | | |
| **序号** | **测试步骤** | | | | **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 |

# 测试方案的自我评价

本次测试方案基本覆盖了需求文档中提到功能性、扩展功能和非功能性需求。对于功能性需求以功能测试为主，设计了覆盖Torch建立神经网络基本功能的测试用例。

对于扩展功能需求的测试，分为MLP神经网络和卷及神经网络分别进行测试。

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

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，本测试一共设计了14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。