

**基于Torch平台的神经网络压缩**

**研究与应用**

**测试结果报告**

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 | 2017/6/5 | 付强 | 陈伟民 曹进 李恬霖 | 该文档的最初版本 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[1. 引言 5](#_Toc484548290)

[1.1 编写目的 5](#_Toc484548291)

[1.2 参考文献 5](#_Toc484548292)

[2. 运行要求测试 5](#_Toc484548293)

[2.1 测试结果 5](#_Toc484548294)

[2.2 测试结论 7](#_Toc484548295)

[3. 功能测试 7](#_Toc484548296)

[3.1 导入工具包 7](#_Toc484548297)

[3.2 文件数据读取 8](#_Toc484548298)

[3.3 搭建MLP神经网络 9](#_Toc484548299)

[3.4 搭建卷积神经网络 10](#_Toc484548300)

[3.5 训练神经网络模型 11](#_Toc484548301)

[3.6 测试神经网络模型 12](#_Toc484548302)

[3.7 测试结论 12](#_Toc484548303)

[4. 扩展功能测试 13](#_Toc484548304)

[4.1 压缩MLP网络测试 13](#_Toc484548305)

[4.2 压缩卷积网络测试 14](#_Toc484548306)

[4.3 测试结论 15](#_Toc484548307)

[5. 非功能测试 16](#_Toc484548308)

[5.1 高效性 16](#_Toc484548309)

[5.1.1 测试策略描述 16](#_Toc484548310)

[5.1.2 测试结果描述 16](#_Toc484548311)

[5.2 用户友好性 17](#_Toc484548312)

[5.2.1 测试策略描述 17](#_Toc484548313)

[5.2.2 测试结果描述 17](#_Toc484548314)

[5.3 可修改性 18](#_Toc484548315)

[5.3.1 测试策略描述 18](#_Toc484548316)

[5.3.2 测试结果描述 18](#_Toc484548317)

[5.4 鲁棒性 19](#_Toc484548318)

[5.4.1 测试策略描述 19](#_Toc484548319)

[5.4.2 测试用例描述 19](#_Toc484548320)

[5.5 测试结论 21](#_Toc484548321)

# 引言

## 编写目的

本文档为基于Torch平台的神经网络压缩研究与应用的测试报告，本报告目的在于总结和分析各项测试需求用例的相应测试结果，描述系统是否达到需求的目的。预期参考人员包括用户、测试人员、开发人员、项目管理者和其他质量管理人员。

## 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）

[4]《基于Torch平台的神经网络压缩研究与应用测试需求说明书》（V1.2）

# 运行要求测试

## 测试结果

表2-1 TC001-Linux下安装测试结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Linux下安装测试 | **测试用例标识** | TC001 | **对应需求文档中的需求** | 4.2 软件运行要求 | |
| **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 | 通过 |
| 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 | 通过 |
| 3 | 启动Torch | | 启动成功 | | 实际结果与预期结果一致 | 通过 |
| **测试人员** | 李恬霖 | | | | | |
| **测试时间** | 2017/05/23 | | | | | |

表2-2 TC002-Mac OS下安装测试结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **测试用例名称** | Mac OS下安装测试 | **测试用例标识** | TC002 | **对应需求文档中的需求** | 4.2 软件运行要求 | |
| **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 | 通过 |
| 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 | 通过 |
| 3 | 启动Torch | | 启动成功 | | 实际结果与预期结果一致 | 通过 |
| **测试人员** | 李恬霖 | | | | | |
| **测试时间** | 2017/05/23 | | | | | |

## 测试结论

通过本章测试，可以得到结论如下：Torch系统在linux系统和Mac OS系统中都能够正确的编译、安装和启动。

# 功能测试

## 导入工具包

表4-1 TC401-导入工具包测试结果

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试结论** |
| **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 | 通过 |
| 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 | 通过 |
| 3 | Torch7测试员在Torch平台上导入一个路径不存在文件 | | Torch平台报告“路径不存在”错误信息 | | 实际结果与预期结果一致 | 通过 |
| **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。 | | | | | |
| **测试人员** | 李恬霖 | | | | | |
| **测试时间** | 2017/05/25 | | | | | |

## 文件数据读取

表3-2 TC402-文件数据读取测试结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试结论** |
| **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 | 通过 |
| **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。 | | | | | | |
| **测试人员** | | 李恬霖 | | | | | | |
| **测试时间** | | 2017/05/25 | | | | | | |

## 搭建MLP神经网络

表3-3 TC403-搭建MLP神经网络测试结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试结论** |
| **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 | 通过 |
| **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。 | | | | | | |
| **测试人员** | 付强 | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | |

## 搭建卷积神经网络

表3-4 TC404-搭建卷积神经网络测试结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试结论** |
| **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 | 通过 |
| **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。 | | | | | | | |
| **测试人员** | 付强 | | | | | | | |
| **测试时间** | | 2017/05/27 | | | | | | |

## 训练神经网络模型

表3-5 TC405-训练神经网络模型测试结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试结论** |
| **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 | 通过 |
| **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； | | | | | | | |
| **测试人员** | 付强 | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | |

## 测试神经网络模型

表3-6 TC406-测试神经网络模型测试结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | **测试结论** |
| **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 | | 通过 |
| **测试人员** | 付强 | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | |

## 测试结论

通过本章测试，可以得到结论如下：测试需求说明书中的各个功能需求测试用例的实际测试结果均与预期结果一致。

# 扩展功能测试

## 压缩MLP网络测试

表4-1 TC501-压缩MLP神经网络测试结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | **测试结论** |
| **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 | | 通过 |
| **测试人员** | 陈伟民 | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | |

## 压缩卷积网络测试

表4-2 TC502-压缩卷积神经网络测试结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | **测试结论** |
| **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 | | 通过 |
| **测试人员** | 陈伟民 | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | |

## 测试结论

因为扩展功能需求是本次测试的重点，下面结合数据对测试结果进行进一步的解释。

在5层MLP网络中，只有中间三个隐层是经过压缩的——即第（4）、（6）、（8）层，而且都是压缩的相同倍数。因为只涉及到全连接层，所以我们只检验全连接层的压缩效果，变换压缩因子的大小，检查改进后网络的准确率变换，得到下表，其中中间的数字表示的是1-Top-1 error rate，即分类得到的结果与原本的标签的是相等的占比：

表4-3 MLP网络在不同压缩率下的准确率表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 压缩倍数 | 8 | 12 | 16 | 24 | 32 | 48 | 64 |
| 未压缩 | 0.960 | | | | | | |
| 已压缩 | 0.941 | 0.9496 | 0.9437 | 0.9477 | 0.9493 | 0.9449 | 0.948 |

在LeNet-5网络中，我们只对卷积层进行压缩——即第（3）、（5）层，来检测不同的压缩因子对于准确率的影响。因为第一层的压缩前的基数只有25，所以极大的影响了压缩的倍数，所以我们只对其进行压缩倍数5倍或者12.5倍的尝试，第二层的压缩前的基数是500，所以，相对来说压缩倍数可以大一些，实验中选择5、10、25、50四个数来做尝试。同理，中间的数字是1-Top-1 error rate。

表4-4 LeNet-5网络在不同压缩率下的准确率表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 压缩倍数 | 5/5 | 5/10 | 5/25 | 5/50 | 12.5/10 | 12.5/25 | 12.5/50 |
| 未压缩 | 0.9805 | | | | | | |
| 已压缩 | 0.9588 | 0.9639 | 0.9609 | 0.9496 | 0.9578 | 0.9493 | 0.9422 |

总的来说，改进的压缩的网络对比标准的不压缩的网络Standard Neural Network或者Standard Convolutional Neural Network，准确率是有所下降的，但是性能损失的最后都在2%左右，所是可以接受的性能损失。但是在可允许的性能损失下，网络的压缩倍数高达几十倍，网络的计算消耗明显减小。

此外，目前的实验表明，在全连接层上面的压缩效果要好于在卷积层上的压缩效果，算法性能更好，网络模型也更加稳定。

# 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LeNet-5神经网络，比较两者的计算效率。

## 测试结果描述

表5-1 TC601-高效性测试结果

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试结论** |
| **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 | 通过 |
| **测试人员** | 曹进 | | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | | |

## 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。

## 测试结果描述

表5-2 TC602-用户友好性测试结果

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | **测试结果** |
| **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 | | 通过 |
| 2 | 搭建入门级神经网络MLP | | | 网络搭建简单，运行成功。 | | | 实际结果与预期结果一致 | | 通过 |
| **测试人员** | 曹进 | | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | | |

## 可修改性

## 测试策略描述

根据对应的需求文档可知，本系统的扩展需求为通过压缩Torch平台中卷积层和线性层的训练参数，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。

## 测试结果描述

表5-3 TC603-可修改性测试结果

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | | **测试结论** |
| **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章扩展功能测试） | | | |  | |  | |  |
| 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 | | 通过 |
| **测试人员** | 曹进 | | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | | |

## 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

表5-4 TC604-鲁棒性测试结果（1）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试过程** | | | | **测试结果** | | **评价准则** | | **测试结论** |
| **预期结果** | |  | |  |
| 1 | 打开Torch平台 | | | |  | |  | |  |
| 2 | 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 | | 通过 |
| **测试人员** | 曹进 | | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | | |

表5-5 TC604-鲁棒性测试结果（2）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试过程** | | | | **测试结果** | | **评价准则** | | **测试结论** |
| **预期结果** | |  | |  |
| 1 | 打开Torch平台 | | | |  | |  | |  |
| 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 | | 通过 |
| **测试人员** | 曹进 | | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | | |

表5-6 TC604-鲁棒性测试结果（3）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试过程** | | | **测试结果** | | | **评价准则** | | **测试结论** |
| **预期结果** | | |  | |  |
| 1 | 打开Torch平台 | | |  | | |  | |  |
| 2 | 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 | | 通过 |
| **测试人员** | 曹进 | | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | | |

表5-7 TC604-鲁棒性测试结果（4）

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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** | **测试过程** | | | **测试结果** | | | **评价准则** | | **测试结论** |
| **预期结果** | | |  | |  |
| 1 | 打开Torch平台 | | |  | | |  | |  |
| 2 | Kill当前程序运行进程 | | | 保存现有数据状态 | | | 实际结果与预期结果一致 | | 未保存当前数据状态 |
| 3 | 再次打开Torch平台 | | | 恢复意外关闭前的数据状态 | | | 实际结果与预期结果一致 | | 未恢复意外关闭前的数据状态 |
| **测试人员** | 曹进 | | | | | | | | |
| **测试时间** | 2017/05/27 | | | | | | | | |

## 测试结论

通过本章测试，可以得到以下结论：系统的非功能性需求在大部分情况下均按照预想的情况执行。只有1个测试用例出现了偏差。

出现偏差的测试用例为用例604的第四种场景，用户强行Kill当前程序运行进程，我们期望系统会保存现有数据状态，并在下次启动系统时自动恢复但是实际运行中再次打开Torch平台时系统并未恢复意外关闭前的数据状态。这是Torch平台本身设计存在的问题，我们无法修改，只能在之后的使用中尽量避免这种情况的发生。