**<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

**作 者： TeamA全体组员**

**完成日期： 2020.03.25**

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

修订历史记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **修改人员** | **审核人员** |
| 2020.03.25 | V1.0 | 无 | 王茵迪，赵永驰，秦浩桐，黄涵，吴振赫，张崇智，高明骏 | 张崇智，高明骏 |
| 2020.03.31 | V2.0 | 针对上周老师的意见进行针对性修改 | 王茵迪，赵永驰，秦浩桐，黄涵 | 张崇智，高明骏 |
| 2020.04.04 | V3.0 | 针对参考文献等细节部分进行了修改 | 张崇智 | 高明骏 |
|  |  |  |  |  |

目录

[1. 引言 4](#_Toc36972205)

[1.1 背景说明 4](#_Toc36972206)

[1.1.1 技术背景 4](#_Toc36972207)

[1.1.2 软件定位 4](#_Toc36972208)

[1.2 参考资料 4](#_Toc36972209)

[1.3 术语和缩略语 5](#_Toc36972210)

[2. 软件总体概述 6](#_Toc36972211)

[2.1 软件描述 6](#_Toc36972212)

[2.1.1 软件属性 6](#_Toc36972213)

[2.1.2 开发背景 6](#_Toc36972214)

[2.1.3 软件功能 7](#_Toc36972215)

[2.2 假设与约束 7](#_Toc36972216)

[3. 具体需求 8](#_Toc36972217)

[3.1 功能需求 8](#_Toc36972218)

[3.1.1调用对抗样本生成模块 9](#_Toc36972219)

[3.1.2调用模型量化模块 10](#_Toc36972220)

[3.1.3调用目标检测模块 10](#_Toc36972221)

[3.1.4调用主动学习模块 11](#_Toc36972222)

[3.1.5调用阅读理解模块 12](#_Toc36972223)

[3.1.6加载模型 13](#_Toc36972224)

[3.1.7处理数据 14](#_Toc36972225)

[3.1.8保存模型 15](#_Toc36972226)

[3.2 性能需求 15](#_Toc36972227)

[3.2.1 高效性 15](#_Toc36972228)

[3.2.2 用户友好性 16](#_Toc36972229)

[3.2.3 安全性 16](#_Toc36972230)

[3.2.4 扩展性 16](#_Toc36972231)

[4.环境需求 16](#_Toc36972232)

[4.1设备环境 16](#_Toc36972233)

[4.2支持软件环境 16](#_Toc36972234)

[4.3接口 17](#_Toc36972235)

[4.3.1 外部接口 17](#_Toc36972236)

[4.3.2 内部接口 17](#_Toc36972237)

[4.3.3 硬件接口 17](#_Toc36972238)

[4.3.4 软件接口 17](#_Toc36972239)

[4.4 安全和保密 18](#_Toc36972240)

# 1. 引言

## 1.1 背景说明

### 1.1.1 技术背景

随着计算资源算力的快速发展以及学术界的深入探索，深度学习已经在计算机视觉、自然语言处理和语音识别等各个领域取得了惊人的成就。深度学习使机器能够模仿视听和思考等人类的活动，解决了很多复杂的模式识别难题，使得人工智能相关技术取得了很大进步。

与此同时，以谷歌公司的TensorFlow和Facebook公司的PyTorch为代表的深度学习框架也取得了长足的进步，助力学术界的研究以及工业界的部署。其中，PyTorch平台以其易于理解的抽象层次、灵活易用的接口设计以及对初学者友好的学习难度，使得其在全球学术界中逐渐成为最受欢迎的深度学习框架。

然而，由于人工智能领域的更新速度过快，当前框架集成的高层算法及模型已经不能满足科研人员的实际需求。因此，对于深度学习多领域最新成果的集成是必要且紧迫的，完成后可以有效地减少科研人员对过往算法和模型的复现工作，使科研人员集中精力在研发工作上，提高科研的效率和质量。

### 1.1.2 软件定位

本次待开发的软件为基于PyTorch的前沿深度学习算法集成应用程序接口，该应用程序接口可在任何支持规定版本Python环境的计算终端进行安装和调用。目标用户为深度学习领域的科研人员，用户通过使用该应用程序接口在个人计算终端完成前沿深度学习算法的调用和模型及其他结果的生成。通过使用该软件，相关科研人员可以减少复现过往论文的工作，提高科研的效率和质量。

## 1.2 参考资料

[1] GB-T8567-2006, 计算机软件文档编制规范[S].

[2] Roger S.Pressman著, 郑人杰等译.软件工程[M].第七版.北京：机械工业出版社,2011.

[3] PyTorch官方文档[OL]https://PyTorch.org/

[4] Siddhant A, Lipton Z C. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study[J]. arXiv preprint arXiv:1808.05697, 2018.

[5] Sener O, Savarese S. Active learning for convolutional neural networks: A core-set approach[J]. arXiv preprint arXiv:1708.00489, 2017.

[6] Wang, Yizhong, et al. Multi-passage machine reading comprehension with cross-passage answer verification. [R]: Baidu Research, 2018

[7] Carlini N, Wagner D. Towards evaluating the robustness of neural networks[C]//2017 ieee symposium on security and privacy (sp). IEEE, 2017: 39-57.

[8] Dong Y, Liao F, Pang T, et al. Boosting adversarial attacks with momentum[C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 9185-9193.

## 1.3 术语和缩略语

**表1.1 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 | UCM | Use Case Modeling，用例建模 |
| 3 | RUCM | Restricted Use Case Modeling，限制性用例建模 |
| 4 | CPU | Central processing unit，中央处理器 |
| 5 | GPU | Graphics Processing Unit，图形处理器 |
| 6 | DNNs | Deep Neural Networks,深度神经网络 |
| 7 | CUDA | Compute Unified Device Architecture，统一计算设备架构 |

# 2. 软件总体概述

## 2.1 软件描述

### 2.1.1 软件属性

API是指一些预先定义的函数，或指软件系统不同组成部分衔接的约定。目的是提供应用程序与开发人员基于某软件或硬件得以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。这使得使用者可以充分利用到前人所做出的成果，不必再将已有的工作重复做一遍。

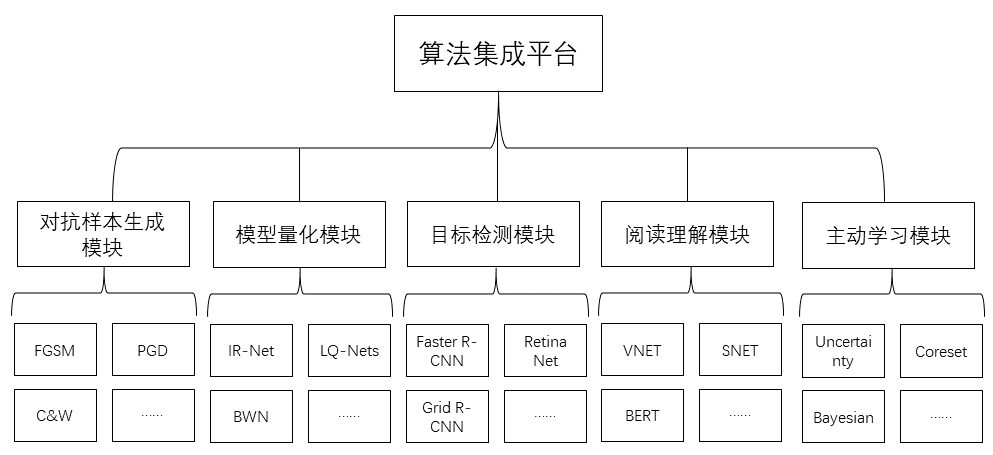
本平台在基于PyTorch框架的情况下将一些深度学习领域的经典或优秀算法封装成了API，以便使用者可以简单的调用这些已有的算法或工作。它对于学生、科研人员、教师、企业员工等工作群体具有较高的使用价值，可以提高深度学习实验的效率，有助于推动研究和工程项目的开展进行。

### 2.1.2 开发背景

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。本平台基于PyTorch框架进行算法开发。

1. 开发目的：帮助科研人员提高深度学习实验效率。
2. 应用目标：通过封装的API提高已有代码的复用性，以提高深度学习实验效率。
3. 使用范围：科研人员、教师、学生、程序员等编程人员。

### 2.1.3 软件功能



**图2.1 系统架构图**

本平台主要包含了多个子模块，分别对应四个常用的深度学习领域。它们分别是对抗样本生成模块、模型量化模块、阅读理解模块、主动学习模块。我们为每一个子模块都提供了多种实用算法的API，通过调用相应的函数以及输入参数，可以快速的实现对应的算法内容。对抗样本生成模块中提供了多种深度学习对抗攻防领域中的前沿算法，包括FGSM、PGD、C&W等。模型量化模块中提供了多种在线量化压缩算法，包括IR-Net、LQ-Nets、BWN等。阅读理解模块中提供了多种自然语言文本问答算法，包括VNET、SNET、BERT等。主动学习模块提供了多种主动学习方向的经典算法，包括Uncertainty、Coreset、Bayesian等。

## 2.2 假设与约束

为了保证平台的正常运行或发布，对其运行环境和过程做如下假设和约束：

1. 法律政策

软件设计、制造、发布等遵循相关的法律政策。

1. 硬件限制

平台软件需要运行在达到一定条件的运行平台上，一般指性能达到或超过以下配置的个人电脑（PC）、训练集群或服务商提供的云端平台：英特尔i7-6700H CPU，英伟达GTX-1080TI GPU，16G运行内存，500G硬盘存储。平台软件的产物（浮点模型、量化模型、对抗样本等）应根据实际需要部署、应用在指定性能的训练平台上，但要求可以使用平台的硬件进行相关测试。

1. 与其他应用的接口

平台设计中预计使用、集成一部分Python和C++接口，平台接口定义清晰、输入输出明确。

1. 编程语言

使用Python3作为编程语言。

1. 工具约束

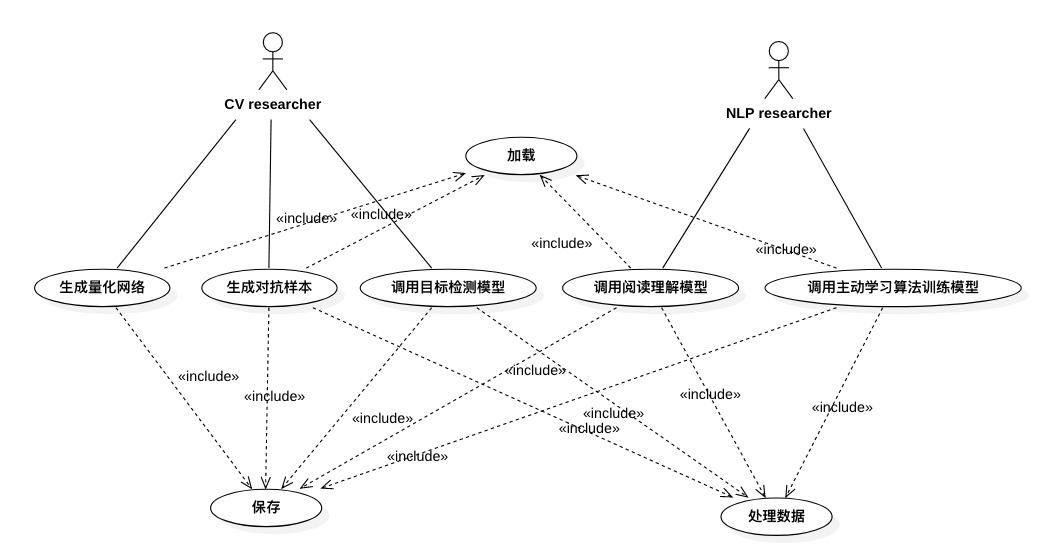
使用Python3解释器作为平台的运行工具，同时使用conda、CUDA等工具作为运行环境。

1. 代码体积

代码体积（不包括输入、输出的产物，以及运行代码使用的工具）控制在100M以内，以方便工具的编译、发布和使用。

# 3. 具体需求

## 3.1 功能需求



**图3.1 系统用例图**

不同研究领域的用户可以根据需求的不同直接调用本系统的五个用例，分别是生成对抗样本、生成量化网络、调用主动学习算法训练模型、调用阅读理解模型和调用目标检测模型。

生成对抗样本用例包含了加载和保存用例，加载作为算法输入的模型文件和用于生成目标输出的数据文件，保存生成的对抗样本。

生成量化网络用例包含了加载和保存用例，加载作为算法输入的模型文件，保存生成的量化网络。

调用主动学习算法训练模型包含了加载、处理数据和保存用例，加载作为算法输入的模型文件，处理数据用例的功能是将文本数据向量化，保存优化后的网络和预测结果。

调用阅读理解模型和调用目标检测模型也包含了处理数据和保存用例，保存优化后的网络和预测结果。

### 3.1.1调用对抗样本生成模块

对抗样本生成模块是为深度学习对抗攻防领域的研究人员提供的当今经典以及前沿对抗攻击算法库。用户可以调用该模块完成多种对抗样本的生成，减少科研人员复现方法的时间，提高科研效率。具体地，用户应将使用的模型和数据集处理成符合要求的格式，作为函数的输入。对应的对抗攻击算法将针对模型信息，在数据集的每一个样本上生成对抗样本，并将新生成的样本作为一个数据集输出给用户。

对该模块构造的RUCM如下所示：

**表3.1 调用对抗攻击方法生成对抗样本用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 调用对抗攻击方法生成对抗样本 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的对抗攻击方法对提供的模型在指定数据集上生成对抗样本 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 加载,INCLUDE USE CASE 保存 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户载入模型 | |
| 3 | 用户载入数据处理成满足模型量化模块输入要求的格式 | |
| 4 | 用户调用工具包中的指定对抗攻击算法 | |
| 5 | 用户将模型和数据集传入函数中，并返回生成的样本 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flow | **RFS** 2, 3,4 | | |
| 1 | 导入工具包失败 | |
| 2 | 系统报告命令不合法信息 | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 3.1.2调用模型量化模块

模型量化模块是为神经网络模型量化领域的研究人员提供的当今经典以及前沿的在线量化压缩算法库。用户可以调用该模块完成多种尺寸、位宽的模型量化，减少科研人员复现方法的时间，提高科研效率。具体地，用户应将使用的模型和数据集处理成符合要求的格式，作为函数的输入。对应的模型量化算法将针对模型的具体信息，针对每个模型以及指定的算法、位宽生成替换模型，并将新的模型作为一个输出返还给用户以便用户使用。接着，用户调用量化模块生成的模型，将量化生成的模型进行部署或训练。

对该模块构造的RUCM如下所示：

**表3.2 调用模型量化方法生成量化模型用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 调用模型量化方法生成量化模型 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的量化方法对提供的模型在指定位宽上进行量化，并保存到本地 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 加载,INCLUDE USE CASE 保存 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户载入模型 | |
| 3 | 用户载入数据处理成满足模型量化模块输入要求的格式 | |
| 4 | 用户调用工具包中的指定模型量化算法， | |
| 5 | 将模型传入模型量化模块中，并返回量化后的模型 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flows | **RFS** 2, 3,4 | | |
| 1 | 导入工具包失败 | |
| 2 | 系统报告命令不合法信息 | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 

### 3.1.3调用目标检测模块

目标检测模块是为深度学习目标检测领域的研究人员提供当今经典及前沿的目标检测算法库。用户可以调用该模块以多种算法检测图像中的前景物体，减少科研人员复现方法的时间，提高科研效率。具体地，用户应将使用的数据集处理成符合要求的格式并选择预训练模型，作为函数的输入。对应的目标检测算法将根据指定的参数信息，载入预训练模型并对数据集的每个图片生成一系列物体的候选框，并输出给用户。

对该模块构造的RUCM如下所示：

**表3.3 调用目标检测算法生成物体候选框用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 调用目标检测算法生成物体候选框 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的目标检测方法使用指定的预训练模型在指定数据集上生成物体候选框，并保存到本地 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 加载模型，INCLUDE USE CASE 处理数据 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户加载模型 | |
| 3 | 用户载入数据处理成满足函数输入规格要求的格式 | |
| 4 | 用户调用工具包中的指定目标检测算法，将模型和数据集传入函数中 | |
| 5 | 用户得到生成的物体候选框 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flow | **RFS** 2, 3, 4 | | |
| 1 | 使用工具包失败 | |
| 2 | 系统报告命令不合法信息 | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 3.1.4调用主动学习模块

用户调用文本处理工具对文本数据预处理，转化为Token向量形式，将分类标签转化为标签ID。为了后续执行主动学习算法的方便，每一条数据处理为字典。训练数据包括三个Key：Tokens、Label、IsLabeled。其中Tokens为文本对应的Token向量，Label为数据对应的分类标签，IsLabeled代表样本是否处于标注池中；而测试数据只包含Tokens和Label两个Key。

用户可以指定数据处理的模式为训练模式或者测试模式。在训练模式下，数据加载算法会随机选择小部分数据作为主动学习算法的初始训练数据。数据文件的格式规定如下：每行为一条数据，包括文本和类标签。文本和类标签之间用逗号隔开。用户可以选择本模块提供的对模型进行主动学习训练的算法，用户需要输入需要训练的模型。模块中的主动学习算法支持任意的序列建模模型，模型的输入需要为Token向量，输出要包括预测的概率分布以及序列的深层次特征向量。

用户构建主动学习算法对象时，除了提供训练数据以及需要训练的模型外，还需要提供对应主动学习算法的参数配置信息。用户调用主动学习对象的训练方法，可以应用提供的输入，利用对应的主动学习算法对模型进行训练。用户需要配置训练的相关信息，包括损失函数、评价指标、batch size以及epoch等。

用户调用主动学习对象的test方法，可以对训练得到的模型进行测试。用户需要提供测试指标以及测试数据，测试数据的格式要求与数据加载算法中描述的测试模式下的数据处理结果相同。

该用例的RUCM如下所示：

**表3.4 调用模型进行主动学习用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 调用模型进行主动学习 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的主动学习方法对提供的模型进行训练和测试。 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | NLP 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 加载模型，INCLUDE USE CASE 处理数据 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户调用数据处理工具 | |
| 3 | 用户选择主动学习算法 | |
| 4 | 用户配置训练相关参数开始训练网络 | |
| 5 | 用户定义测试指标 | |
| 6 | 用户测试训练完成后的网络性能 | |
| Postcondition | | 系统不报告错误信息，正常运行 |
| Bounded  Alternative  Flow | **RFS** 3, 4, 6 | | |
| 1 | 程序运行异常 | |
| 2 | 系统报告命令不合法信息 | |
| Postcondition | | 程序中断，用户依据报告信息修改代码 |

### 3.1.5调用阅读理解模块

自然语言阅读理解模块可以调用先进的阅读理解模型，调用模型从文本内容中寻找问题答案。用户可以通过不同的参数设定，将不同格式的文本和问题读入进行分词等预处理，以获得模型可以直接识别的规范数据集。用户可以设定不同的参数，选择不同的经典模型。其中，该模型的具体形态、大小等参数也通过用户决定。用户可以决定训练该模型的方式方法、优化函数、训练集大小，用户通过不同的参数设定，对模型进行训练。对测试集数据进行测试，回答预测集给出的问题。

该用例的RUCM如下所示：

**表3.5 调用自然语言阅读理解模型用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 调用自然语言阅读理解模型 | | |
| Brief Description | 用户导入工具包，并使用工具包中选定的阅读理解模型在指定数据集上训练网络，使用训练好的网络回答指定数据集的问题。 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | NLP 研究人员 | | |
| Secondary Actor | None | | |
| Dependency | INCLUDE USE CASE 处理数据,INCLUDE USE CASE 保存 | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户使用数据处理工具处理指定数据集 | |
| 3 | 用户指定网络模型和参数 | |
| 4 | 用户使用训练工具对模型在指定训练集上进行训练 | |
| 5 | 用户输入文本和问句 | |
| 6 | 用户使用预测工具预测答句 | |
| Postcondition | | 整体过程不报告错误信息，运行结束退出 |
| Bounded  Alternative Flow | **RFS** 1, 2, 3, 4, 5, 6 | | |
| 1 | 程序运行异常 | |
| 2 | 系统提示运行时出错位置 | |
| Postcondition | | 中断退出 |

### 3.1.6加载模型

模型是深度学习基础的操作对象。无论是在算法中使用预训练的模型，还是对模型进行量化等操作，用户均需要进行加载模型的操作。具体而言，用户可以设定具体的模型路径，然后调用工具包中模型加载算法将存储在本地的模型参数载入到内存中，以供后续程序利用模型进行训练、测试等操作。

该用例的RUCM如下所示：

**表3.6 加载模型用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 加载模型 | | |
| Brief Description | 用户导入工具包，并使用工具包中针对特定模型的载入算法，将模型参数读入内存，以供后续程序对于模型的使用。 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV/NLP研究人员 | | |
| Secondary Actor | None | | |
| Dependency | None | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户准备满足模型载入算法格式要求的模型 | |
| 3 | 用户使用数据加载工具载入模型参数 | |
| 4 | 系统 VALIDATES THAT 模型路径和参数合法 | |
| 5 | 用户获得可进行后续处理的模型 | |
| Postcondition | | 整体过程不报告错误信息，运行结束退出 |
| Bounded  Alternative Flow | **RFS** 4 | | |
| 1 | 程序运行异常，模型不合法 | |
| 2 | 系统提示运行时出错位置 | |
| Postcondition | | 中断退出 |

### 3.1.7处理数据

目前的深度学习方法大多是数据驱动的，因此数据是深度学习研究的基础。而处理数据是各种学习任务的第一步，研究人员要将数据处理为模型可以理解的形式，然后才可以将其作为模型的输入。明确地说，研究人员需要准备满足对应数据处理算法格式地数据，然后导入工具包进行数据处理，最终得到可供模型输入的数据，以供之后模型的训练或测试。

该用例的RUCM如下表所示：

**表3.7 处理数据用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 处理数据 | | |
| Brief Description | 用户导入工具包，并使用工具包中针对特定数据的处理算法处理数据，供网络训练或测试使用。 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV/NLP研究人员 | | |
| Secondary Actor | None | | |
| Dependency | None | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户准备满足数据处理算法格式要求的数据集 | |
| 3 | 用户使用数据处理工具处理数据集 | |
| 4 | 用户获得可供网络输入的数据 | |
| Postcondition | | 整体过程不报告错误信息，运行结束退出 |
| Bounded  Alternative Flow | **RFS** 1, 2, 3, 4 | | |
| 1 | 程序运行异常 | |
| 2 | 系统提示运行时出错位置 | |
| Postcondition | | 中断退出 |

### 3.1.8保存模型

模型是深度学习基础的操作对象。无论是希望在算法中重新加载预训练的模型，还是对模型进行分析等操作，用户均需要进行保存模型的操作。具体而言，用户可以设定具体的模型保存路径，然后调用工具包中模型保存工具将训练好的模型保存到本地中，以供后续程序复用该模型。

该用例的RUCM如下表所示：

**表3.8 保存模型用例描述表**

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case Specification | | | |
| Use Case Name | 保存模型 | | |
| Brief Description | 用户导入工具包，并使用工具包中针对特定模型的算法处理、保存模型，最终使模型参数保存到本地。 | | |
| Precondition | 系统正常运行 | | |
| Primary Actor | CV/NLP研究人员 | | |
| Secondary Actor | None | | |
| Dependency | None | | |
| Generalization | None | | |
| Basic Flow | **Steps** | | |
| 1 | 用户导入工具包 | |
| 2 | 用户准备满足模型保存模块格式要求的模型 | |
| 3 | 用户使用模型保存工具保存模型参数 | |
| 4 | 用户在本地获得可加载的保存后的模型 | |
| Postcondition | | 整体过程不报告错误信息，运行结束退出 |
| Bounded  Alternative Flow | **RFS** 1, 2, 3, 4, 5, 6 | | |
| 1 | 程序运行异常 | |
| 2 | 系统提示运行时出错位置 | |
| Postcondition | | 中断退出 |

## 3.2 非功能需求

### 3.2.1 高效性

1. 软件库需要能够支持多线程、多GPU并行运行，以保证大规模深度神经网络的高效训练和使用。
2. 使用向量化进行计算加速，消除显示循环语句，通过使用PyTorch支持的矩阵和向量运算，保证矩阵运算的效率。

### 3.2.2 用户友好性

为了保证深度学习研究者和开发者可以尽快熟悉使用此扩展库，库API的设计应该充分考虑相关领域研究的实验流程，符合研究者的实验习惯，并满足研究者实验分析等各方面的需求，使得研究人员能够快速上手。

相应的API应具有对应的说明文档，以满足易学习性和易操作性。

### 3.2.3 安全性

系统应该在异常和危险情况下都能保持健壮的表现和稳定的性能，有健全的容错机制和方法，能够处理系统运行过程中出现的各种异常情况，如：人为操作错误、输入非法数据、硬件设备失败等。系统应该具有保存操作、记录日志的能力，在程序发生故障后，应具有恢复直接受影响数据和模型的能力，或能够回滚用来紧急处理异常。在并行处理时，系统能够保证在进行多线程计算时不出现死锁，模型和数据能够正确分配对应的CPU和GPU，不会出现卡死或内存显存被持续占用的情况。

### 3.2.4 扩展性

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。

# 4.环境需求

## 4.1设备环境

1. 操作系统：Linux为Ubuntu 16及以上版本、Windows7及以上版本、MacOS 10及以上版本。
2. CPU：英特尔i7-6700H等性能相当或更高配置的CPU。
3. GPU：英伟达GTX-1080TI等性能相当或更高配置的GPU。
4. 内存：16G或更高。
5. 硬盘存储：500G或更高。

## 4.2支持软件环境

选择PyTorch版本为1.1或以上版本，同时按照设备的显卡驱动版本安装CUDA 9.0或以上的版本。PyTorch使用的编程语言为Python和C++，其使用的编译程序是结合Python的setuptools搭配cmake进行构建。PyTorch所依赖的第三方库包括：gloo、pybind11、cpuinfo、onnx、QNNPACK、fbgemm。PyTorch当前对于Linux、Windows和MacOSX操作系统均能支持。

## 4.3接口

### 4.3.1 外部接口

对于用户，PyTorch以Python为前端框架，在借助Python这个动态语言本身的特性基础上使用动态图来定义模型，可以说PyTorch很好地借助了Python编译器的一些特性（例如垃圾回收，语言本身的动态性等），这使得用户在使用时会觉得好像比tensorflow这样使用静态图的框架容易。

### 4.3.2 内部接口

目前大部分深度学习系统（TensorFlow、PyTorch等）都是基于 C、C++ 构建的后端，因此这些系统基本都存在 C、C++ 的扩展接口。C++ 扩展是一种以允许用户创建一些包含的资源之外的PyTorch 运算符，为基于PyTorch的项目提供高度的灵活性。PyTorch中的拓展模块定义代码主要在torch/csrc/Module.cpp中。

PyTorch内部结构上大约有两部分，一部分是Python，另外一部分是C/C++写的后端。C/C++后端在v0.4开始用Aten拓展专用库作为上层封装，通过torch/csrc中的部分代码接入Python（但慢慢在改为pybind11）。一般计算的运算符都是通过Aten在c++实现然后绑定到Python上。JIT编译机制通过调用 PyTorch API中一个名为torch.utils.cpp\_extension.load()的简单函数，能够提供了一种动态编译和加载扩展的方式。

PyTorch的ATen后端也可以抽象化正在运行的计算设备。这意味着为CPU编写的相同代码也可以在GPU上运行，并且各个操作将相应地分派到GPU优化的实现。

### 4.3.3 硬件接口

参考4.1小节的设备环境描述。

### 4.3.4 软件接口

1. Anaconda3 (with Python 3.5/3.6/3.7) 。
2. 英伟达驱动、GPU版本需要任意版本的 CUDA （包内置了CUDA 8 / 9 的部分主要二进制文件）。
3. Anaconda3 x64 (with Python 3.5/3.6)。
4. opencv，visdom等相关库。
5. 以上所有列举接口都是通过程序内部的调用来交互的。