|  |  |  |
| --- | --- | --- |
| 文档状态 | 保密级别 |  |
| [√]草稿  [ ]修订  [ ]发布 | 文档编号 |  |
| 管理部门 |  |
| 修订年月 |  |
| 版本号 |  |

**深度学习调试器项目**

**详细设计说明书**

|  |  |  |
| --- | --- | --- |
| **修订人签字** | **审核人签字** | **批准人签字** |
| 日期： | 日期： | 日期： |

变更履历

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **变更日期** | **版本** | **变更位置** | **变更原因** | **修订人** | **审核人** | **批准人** |
| 1 | 3/26 | V1.0.0 |  | 1 | jiajunlong |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

说明：“变更原因”主要是分为：

1. 建立初稿
2. 内容修订
3. 正式发布

目录

[1. 文档介绍 6](#_Toc550928616)

[1.1. 编写目的 6](#_Toc1706745589)

[1.2. 读者对象 6](#_Toc1009107130)

[1.3. 术语与缩写解释 6](#_Toc825016193)

[1.4. 参考资料 7](#_Toc881816863)

[2. 总体设计 7](#_Toc1700131766)

[2.1. 系统使用说明 7](#_Toc2012953151)

[2.2. 系统模块划分 8](#_Toc605179273)

[2.3. 系统模块要求 8](#_Toc65707038)

[2.3.1. 整体要求 8](#_Toc1052578855)

[2.3.2. 针对界面（Taiyi） 9](#_Toc274317674)

[2.3.3. 针对组件 9](#_Toc924844534)

[3. 程序设计详细描述 10](#_Toc1799272728)

[3.1. 界面模块 10](#_Toc539172941)

[3.1.1. 配置文件解析 10](#_Toc1389439378)

[3.1.2. 模块扩展注册 11](#_Toc243226094)

[3.1.3. 监测信息追踪 12](#_Toc195432869)

[3.1.4. 结果输出功能 13](#_Toc994255105)

[3.2. 中间结果计算模块 14](#_Toc44104595)

[3.2.1. 需要的扩展 15](#_Toc469368342)

[3.2.2. 指标计算 15](#_Toc1802759672)

[3.2.3. 指标是否需要计算 16](#_Toc1887777869)

[3.2.4. 计算 17](#_Toc740312939)

[3.2.5. 结果输出功能 18](#_Toc1036603877)

[3.3. 底层信息生成模块 18](#_Toc1783047762)

[3.3.1. hook功能 19](#_Toc1848586490)

[3.3.2. 扩展计算功能 20](#_Toc681324307)

# 文档介绍

## 编写目的

本文档描述深度学习调试器详细设计说明书的目的是：

1. 定义软件总体要求，作为用户和软件开发人员之间相互了解的基础；
2. 说明程序系统的各个层次中的每个软件对象（包括每个模块和程序）的设计考虑，以向系统实现（编码和测试）阶段提供关于程序系统实现方式的详细描述；
3. 作为软件总体测试的依据。

## 读者对象

编写详细设计人员及程序开发人员。

## 术语与缩写解释

|  |  |
| --- | --- |
| 缩写、术语及符号 | 解释 |
| Module | pytorch中的nn.module()，是调试工具的基本观察单元 |
| Model | 训练的模型 |
| Quantity | 指标，是针对module想要观察的值（Module和quantity是调试工具中最重要的两个维度，一个module可以有多个观察值，quantity可以对不同的module有相同的计算流程） |
| Extension | 在quantity计算过程中，需要module提供quantity需要的基本信息，如果是module本身没有的属性或者数值，需要module通过注册hook来进行获取 |
| Taiyi | 调试工具与使用者进行交互的接口，输入是用户想要观察module的quantity，输出是具体的计算值 |
| Config | 由用户编写，Taiyi进行读取，基本格式是{module:[quantity1,quantity2]}。 |
| Regisiter | 对module进行hook注册 |
| ty | 太一 |
| Jm | 界面 |
| Qt | 指标 |
| Inf | 信息 |

## 参考资料

| **序号** | **文档名称** | **文档编号** | **版本** | **发布日期** |
| --- | --- | --- | --- | --- |
| 1 | https://github.com/f-dangel/cockpit |  | 285 |  |
| 2 | https://github.com/pytorch/captum |  | V0.6.0 |  |

# 总体设计

## 系统使用说明

用户使用过程如下：

1. 准备数据，准备模型，准备优化器，准备loss；
2. 准备config文件，config形式为{module1:[quantity1,quantity2], module2:[quantity1,quantity3]}；
3. 通过模型和config文件初始化taiyi；
4. 进入batch训练阶段，在执行forward和backward之后，Taiyi对配置文件中的每个module进行quantity计算，将计算结果进行存储后进行下一轮的训练和计算。

## 系统模块划分

各模块之间的关系如图1所示。

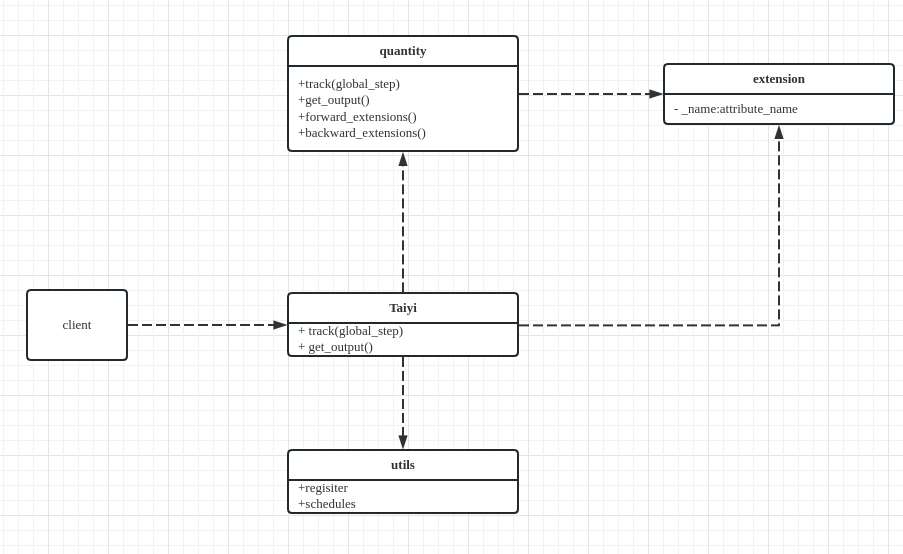


表 1 系统功能分布图

## 系统模块要求

### 整体要求

1. 规范

1）每个类的命名规范；

2）代码编写用有表达力的名字表达含义search、extract这种，循环迭代器用带名字例如xxx\_i来代替I；

3）名字不能有歧义，尤其是针对TRUE 和 FALSE这种，要明确当前函数名称返回的值跟名称的真假性。

1. 注释

1）注释风格应该一致；

2）每个函数都应该编写注释，要理解的准确无误；

3）能直接看出含义的代码不需要写注释；

4）可以用注释记录采用当前解决方法的思考过程以及提醒一些特殊情况：比如TODO（待做），hack（粗糙的解决方案）等等。

1. 代码编写

1）尽可能使用少的长表达式；

2）减小变量作用域（全局变量这种，如果是必要的，可以写个静态类专门存储全局变量）；

3）大问题拆分成小问题，在解决具体方法时先写伪代码，所有伪代码中涉及的细节用专门的小函数进行计算（要注意拆分力度）；

4）多用标准库实现。

### 针对界面（Taiyi）

尽可能粗线条（抽象），类似于只涉及功能的伪代码，所有的运行过程能通过肉眼看出来在做什么事情，所有涉及到细节的东西（如何计算，如何注册hook，如何读取配置文件，如何存储结果）不应该体现在Taiyi的运行过程中。

### 针对组件

所有的组件应该尽可能的功能封闭加功能单一：（1）修改单一组件的bug足够精细，不影响其他组件，可以被其他功能复用；（2）与其他模块的交流应该尽可能的少。

# 程序设计详细描述

## 界面模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模块名称** | | 界面模块 | | | |
| **模块简介** | | 调试工具与使用者进行交互的接口，输入是用户想要观察module的quantity，输出是具体的计算值 | | | |
| **模块功能列表** | | | | | |
| **序号** | **一级功能** | | | **二级功能** | |
| **功能名称** | | **功能编号** | **功能名称** | **功能编号** |
| 1 | 模型监测功能 | | tyJm001 | 配置文件解析 | tyJm001001 |
| 模块扩展注册 | tyJm001002 |
| 监测信息追踪 | tyJm001003 |
| 2 | 结果输出功能 | | tyJm002 | 结果输出功能 | tyJm002001 |

### 配置文件解析

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 配置文件解析 | **需求编号** | tyJm001001 |
| **所属类** | Taiyi | **方法名称** | \_config\_parser |
| **方法类型** | 私有方法 | | |
| **类属性** | model：nn.module 包含监测模块的模型  parse\_quantity:dict 解析得到的quantity实例  parse\_module:dict 解析得到的module实例  output：dict 计算得到的结果输出 | | |
| **功能描述** | 将初始化得到的配置文件进行解析。 | | |
| **依赖模块** | Quantity | | |
| **输入** | 无 | | |
| **流程逻辑** | 1. 首先判断配置文件的module是不是model的属性    1. 是 2. 通过getattr方法从model中获取module 3. 将获取到的module放入解析模块中    1. 否 4. 从model的named\_modules()方法中进行模型查找 5. 将查找得到的符合条件的module放入解析模块中 6. 在Quantity模块中进行quantity查找，将查找得到的符合条件的quantity放入解析模块中 | | |
| **扩展及异常** | 配置文件中的module和quantity不符合规定 | | |
| **输出** | 解析得到的module，解析得到的quantity | | |
| **补充说明** | 如果配置文件中module给的是nn.Module的类名，则将此类全部进行监测 | | |

### 模块扩展注册

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 模块扩展注册 | **需求编号** | tyJm001002 |
| **所属类** | Taiyi | **方法名称** | \_register |
| **方法类型** | 私有方法 | | |
| **类属性** | model：nn.module 包含监测模块的模型  parse\_quantity:dict 解析得到的quantity实例  parse\_module:dict 解析得到的module实例  output：dict 计算得到的结果输出 | | |
| **功能描述** | 将解析得到的module进行属性扩展 | | |
| **依赖模块** | Regisiter，Extension | | |
| **输入** | 无 | | |
| **流程逻辑** | 1. 从parse\_module中获取需要扩展的module 2. 从parse\_quantity中获取当前quantity计算需要module进行扩展的extension 3. 对同一个module需要的所有quantity的extension进行去重 4. 使用Regisiter模块对module进行extension注册 | | |
| **扩展及异常** | 无 | | |
| **输出** | 无 | | |
| **补充说明** | 1. 如果配置文件中module给的是nn.Module的类名，则将此类全部进行监测 2. parse\_module key=module\_name:str value=module:nn.module 3. parse\_quantity key=module\_name:str value=[modules:nn.module] | | |

### 监测信息追踪

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 监测信息追踪 | **需求编号** | tyJm001003 |
| **所属类** | Taiyi | **方法名称** | track |
| **方法类型** | 公有方法 | | |
| **类属性** | model：nn.module 包含监测模块的模型  parse\_quantity:dict 解析得到的quantity实例  parse\_module:dict 解析得到的module实例  output：dict 计算得到的结果输出 | | |
| **功能描述** | 执行quantity计算 | | |
| **依赖模块** | Quantity | | |
| **输入** | global\_step：int 判断当前迭代是否需要指标监测 | | |
| **流程逻辑** | 1. 从parse\_quantity中获取需要监测的quantity 2. 调用quantity的track()方法进行指标计算 | | |
| **扩展及异常** | 无 | | |
| **输出** | 无 | | |
| **补充说明** | 无 | | |

### 结果输出功能

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 结果输出功能 | **需求编号** | tyJm002001 |
| **所属类** | Taiyi | **方法名称** | get\_output |
| **方法类型** | 公有方法 | | |
| **类属性** | model：nn.module 包含监测模块的模型  parse\_quantity:dict 解析得到的quantity实例  parse\_module:dict 解析得到的module实例  output：dict 计算得到的结果输出 | | |
| **功能描述** | 将quantity计算的结果值进行输出 | | |
| **依赖模块** | Quantity | | |
| **输入** | 无 | | |
| **流程逻辑** | 1. 从parse\_quantity中获取需要监测的quantity 2. 调用quantity的get\_output()方法获取quantity的计算结果 3. 将计算结果存储到output属性中 4. 将output进行返回 | | |
| **扩展及异常** | 无 | | |
| **输出** | output：dict 计算得到的结果输出 | | |
| **补充说明** | output key=module\_name:str value=quantity\_output:dict  quantity\_output key=quantity\_class\_name value=q\_output:dict  q\_output key=global\_step:int value=result:Tensor | | |

## 中间结果计算模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模块名称** | | 中间结果计算模块 | | | |
| **模块简介** | | 指标，是针对module想要观察的值  Module和quantity是调试工具中最重要的两个维度，一个module可以有多个观察值，quantity可以对不同的module有相同的计算流程 | | | |
| **模块功能列表** | | | | | |
| **序号** | **一级功能** | | | **二级功能** | |
| **功能名称** | | **功能编号** | **功能名称** | **功能编号** |
| 1 | 指标计算功能 | | tyQt001 | 需要的扩展 | tyQt001001 |
| 指标计算 | tyQt001002 |
| 指标是否需要计算 | tyQt001003 |
| 计算 | tyQt001004 |
| 2 | 结果输出功能 | | tyQt002 | 结果输出功能 | tyQt002001 |

### 需要的扩展

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 需要的扩展 | **需求编号** | tyQt001001 |
| **所属类** | Quantity | **方法名称** | extensions |
| **方法类型** | 公有方法 | | |
| **类属性** | module：nn.module 需要监测的模块  track\_schedule:callable用来判断当前迭代是否需要进行计算  output：dict 计算得到的结果输出 | | |
| **功能描述** | 返回计算quantity计算过程中需要module执行的扩展 | | |
| **依赖模块** | extensions | | |
| **输入** | 无 | | |
| **流程逻辑** | 1. 返回extensions的列表 | | |
| **扩展及异常** | 无 | | |
| **输出** | extensions：list 扩展列表 | | |
| **补充说明** | extensions分为前向过程和反向过程的扩展 | | |

### 指标计算

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 指标计算 | **需求编号** | tyQt001002 |
| **所属类** | Quantity | **方法名称** | track |
| **方法类型** | 公有方法 | | |
| **类属性** | module：nn.module 需要监测的模块  track\_schedule:callable用来判断当前迭代是否需要进行计算  output：dict 计算得到的结果输出 | | |
| **功能描述** | 进行指标计算 | | |
| **依赖模块** | 无 | | |
| **输入** | global\_step:int | | |
| **流程逻辑** | 1. 通过\_should\_compute方法判断当前global\_step是否需要进行计算    1. 是 2. 通过\_comput方法进行计算得到结果result 3. 将计算结果保存到output中    1. 否 | | |
| **扩展及异常** | 无 | | |
| **输出** | 无 | | |
| **补充说明** | 计算得到的结果需要进行格式转化 | | |

### 指标是否需要计算

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 指标是否需要计算 | **需求编号** | tyQt001003 |
| **所属类** | Quantity | **方法名称** | \_should\_compute |
| **方法类型** | 私有方法 | | |
| **类属性** | module：nn.module 需要监测的模块  track\_schedule:callable用来判断当前迭代是否需要进行计算  output：dict 计算得到的结果输出 | | |
| **功能描述** | 返回当前global\_step是否需要进行计算 | | |
| **依赖模块** | Schedule | | |
| **输入** | global\_step:int | | |
| **流程逻辑** | 1. track\_schedule进行计算 2. 返回计算结果 | | |
| **扩展及异常** | 无 | | |
| **输出** | True/False:Bool | | |
| **补充说明** | 子类需要实现该方法 | | |

### 计算

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 计算 | **需求编号** | tyQt001004 |
| **所属类** | Quantity | **方法名称** | \_compute |
| **方法类型** | 私有方法 | | |
| **类属性** | module：nn.module 需要监测的模块  track\_schedule:callable用来判断当前迭代是否需要进行计算  output：dict 计算得到的结果输出 | | |
| **功能描述** | 对当前指标进行计算并返回结果 | | |
| **依赖模块** | 无 | | |
| **输入** | global\_step:int | | |
| **流程逻辑** | 1. 进行指标计算 | | |
| **扩展及异常** | 无 | | |
| **输出** | result：Tensor | | |
| **补充说明** | 子类需要实现该方法 | | |

### 结果输出功能

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 结果输出功能 | **需求编号** | tyQt002001 |
| **所属类** | Quantity | **方法名称** | get\_output |
| **方法类型** | 公有方法 | | |
| **类属性** | module：nn.module 需要监测的模块  track\_schedule:callable用来判断当前迭代是否需要进行计算  output：dict 计算得到的结果输出 | | |
| **功能描述** | 返回计算结果 | | |
| **依赖模块** | 无 | | |
| **输入** | 无 | | |
| **流程逻辑** | 1. 返回属性output | | |
| **扩展及异常** | 无 | | |
| **输出** | output：dict 计算得到的结果输出 | | |
| **补充说明** | output key=global\_step:int value=result:Tensor | | |

## 底层信息生成模块

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **模块名称** | | 底层信息生成模块 | | | |
| **模块简介** | | 作用是将hook得到的结果值作为module的属性进行保存，不同的module在获取相同值的时候对应的hook操作不同。子类在继承基类时，要实现\_default hook：当子类没有实现未知的nn.module时，返回的默认的hook | | | |
| **模块功能列表** | | | | | |
| **序号** | **一级功能** | | | **二级功能** | |
| **功能名称** | | **功能编号** | **功能名称** | **功能编号** |
| 1 | hook功能 | | tyInf001 | hook功能 | tyInf001001 |
| 2 | 扩展计算功能 | | tyInf002 | 扩展计算功能 | tyInf002001 |

### hook功能

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | hook功能 | **需求编号** | tyInf001001 |
| **所属类** | Extension | **方法名称** | \_\_call\_\_ |
| **方法类型** | 魔法方法 | | |
| **类属性** | name:str module扩展的属性值 | | |
| **功能描述** | 将quantity计算的结果值进行输出 | | |
| **依赖模块** | 无 | | |
| **输入** | 无 | | |
| **流程逻辑** | 1. module\_hook方法进行属性值计算返回result 2. 将返回的result作为module的属性值进行注册 | | |
| **扩展及异常** | 无 | | |
| **输出** | 无 | | |
| **补充说明** | 无 | | |

### 扩展计算功能

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | 扩展计算功能功能 | **需求编号** | tyInf002001 |
| **所属类** | Extension | **方法名称** | \_module\_hook |
| **方法类型** | 私有方法 | | |
| **类属性** | name:str module扩展的属性值 | | |
| **功能描述** | 属性值计算 | | |
| **依赖模块** | 无 | | |
| **输入** | 无 | | |
| **流程逻辑** | 1. 扩展属性计算 2. 返回结果result | | |
| **扩展及异常** | 无 | | |
| **输出** | result：Tensor 计算得到的结果 | | |
| **补充说明** | 不同的nn.module对应的\_module\_hook计算过程不同，需要编写默认计算过程，对于不同的nn.module需要编写独有的方法 | | |