

规范 ver0.5

目录

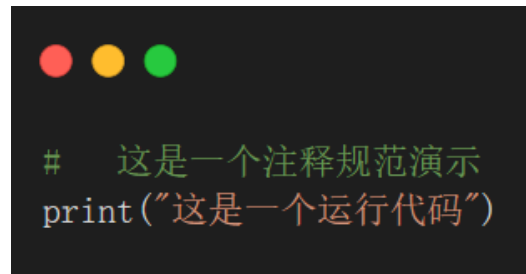
规范 ver0.5.....	1
1. 编码规范.....	2
1.1 关于注释	2
1.2 关于接口定义和合法性约束	2
1.3 关于缩进和换行.....	3
1.4 关于空格	3
2. Git 规范.....	4
2.1 关于提交和拉取.....	4
2.2 关于 fork	4
2.3 关于提交注释和 git 使用	4
2.4 关于 git 的本地管理	4
2.5 关于 git 远程仓库管理.....	5
2.6 关于 git 远程仓库的合并管理.....	6
2.7 关于本地仓库的更新	7
2.8 关于流程介绍	7
2.9 关于仓库的目录结构	8
3. 模型的使用	8
3.1 模型的绘图.....	8

1. 编码规范

1.1 关于注释

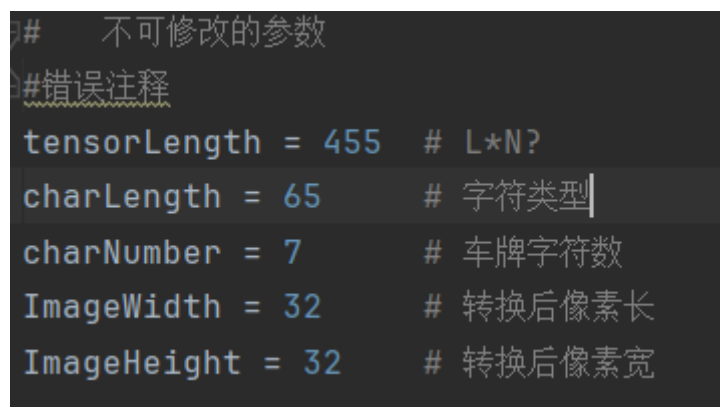
所有编码都需要进行注释，没有注释的代码经审核后不会同意合并分支并且拒绝 push 请求。

所有的注释使用#进行注释，在注释时使用 tab 进行对齐。不可以使用空格对齐，注释需要写在代码的上方，当一行不够的时候可以追加且需要注意对齐。



```
# 这是一个注释规范演示
print("这是一个运行代码")
```

如下图所示，一个好的注释应该做到对齐，否则下划线会标黄线。在对齐时，使用 tab 即可。会自动对齐的。假如不对齐，不使用制表符则会亮黄线，因为不符合 PEP8 编码规范。为了编写代码的时候眼睛舒服，尽量规范注释。



```
# 不可修改的参数
# 错误注释
tensorLength = 455 # L*N?
charLength = 65 # 字符类型
charNumber = 7 # 车牌字符数
ImageWidth = 32 # 转换后像素长
ImageHeight = 32 # 转换后像素宽
```

1.2 关于接口定义和合法性约束

在接口编写时，需要进行合法性检查。只有接口输入的数据合法才能进行下一步运行，否则抛出报错。规范如图所示。

```
1 def add_num(a, b):
2     if a == 0:
3         raise ValueError("不应该输入一个0")
4     return a + b
5
6
7 def main():
8     add_num(a=0, b=2)
9
10
11 if __name__ == '__main__':
12     main()
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
Traceback (most recent call last):
  File "D:\Unetfile\CelebA\test.py", line 12, in <module>
    main()
  File "D:\Unetfile\CelebA\test.py", line 8, in main
    add_num(0, 2)
  File "D:\Unetfile\CelebA\test.py", line 3, in add_num
    raise ValueError("不应该输入一个0")
ValueError: 不应该输入一个0
```

1.3 关于缩进和换行

Python 重视缩进，不要使用空格。当一条代码过长时，不要写在一行中，应该使用换行符换行。或者直接换行。

```
loss1, loss2, loss3, loss4, loss5, loss6, loss7 = criterion(y1, label1), criterion(y2, label2), \
    criterion(y3, label3), criterion(y4, label4), criterion(y5, label5), criterion(y6, label6), \
    criterion(y7, label7)
```

1.4 关于空格

在变量之间，赋值之时，判断之刻，应尽量使用空格。下图是一个示范。

```
def add_num(a, b):
    if a == 0:
        raise ValueError("不应该输入一个0")
    # 正确示范
    if b == 0:
        print(1)

    # 错误示范
    if b==1:
        print(2)
    # 错误示范
    c=a+b

    # 正确示范
    c = a + b

    # 正确示范
    a, b = b, a

    # 错误释放
    a_b = b_a
```

2. Git 规范

2.1 关于提交和拉取

对于 master 分支在未经允许时不可以进行修改。也就是不可以对 master 分支进行写操作，只允许读，也就是只允许抓取克隆到本地。

只有在单码经过审核后，也就是 merge 合并请求后且代码审核没有问题时，才可以进行合并。

2.2 关于 fork

在开发时需要将代码 fork 出来进行开发，这个概念会在日后进行更新并讲解。写好时会更新文档。在本地执行 pull 即可。

2.3 关于提交注释和 git 使用

每次 git 提交都需要进行注释. 并且 git add 时最好不要 git add . 也就是不要将所有的文件都 add 到缓存区, 这样提交到 git 仓库时是所有文件都提交到代码仓库的。这样的习惯不好，也危险。很容易造成本地错误的文件覆盖仓库的基线文件。因此 git add 时特定对应的文件，也就是 git add filename。

2.4 关于 git 的本地管理

1、你想创建新的项目库

创建一个文件夹（整个目录不能有中文，windows 系统编码问题），然后在此文件中，右键 git bash 输入 git init

2、你想用 git 管理你的项目

把项目复制到这个文件夹下，输入 `git add .` 再输入 `git commit -m "这是我项目的第一个版本"`

3、你想更新了你的代码

输入 `git add` 你的代码文件，再输入 `git commit -m "我刚修改了 x"`

4、你想删除你的没用的文件

输入 `git rm` 没用的文件路径和名称，再输入 `git commit -m "我刚删除了 x"`

5、你做了新代码，发现新功能不好，然后回溯到某个版本

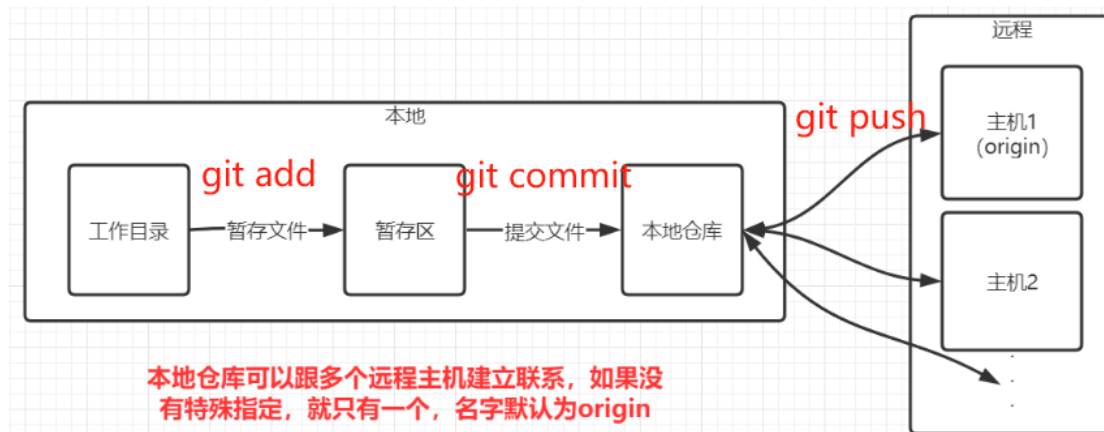
输入 `git log`，再输入 `git reset --hard` 版本号（通过 `git log` 看到的 hash 值就是版本号）

6、如果回溯也错了，不记得是哪个版本了

输入 `git reflog`，可以看到所有的版本，如 A 版本和 B 版本还有 C 版本，你回溯到了 B 版本，那么 A 版本的 id 不记得了，通过 `reflog` 就可以看到回溯的操作和所有版本的 commit，再通过 `git reset --hard` 版本号 回来。

7、如果你想看 git 的步骤到哪了，有没有需要提交的东西

输入 `git status`，可以看到目前各个区域的情况，如果想要对比这个文件和之前的文件的情况，可以通过 `git status` 发现哪些文件和版本库的分支不一样。在通过 `status` 列出的文件，进行 `git diff` 文件名 对比。



2.5 关于 git 远程仓库管理

远程主机名指本地连接的 git 仓库，可以通过 `git remote -v` 查看。比如我这里将远程仓库命名为 `salieri`，所以远程仓库名就是 `salieri`。首先，需要在本地创建自己的分支，也就是 `git branch yourname`。然后 `checkout` 到那个分支，`git checkout yourname`。然后就可以继续你的编码。

当然，你也可以不在本地创建新的分支。只使用本地的 `master` 分支，但是在 `push` 的时候需要 `push` 到你自己的分支。也就是 `git push reponame master:yourforkname`。

假如远程没有对应分支会自动创建一个分支。所以自己执行就行。最好以自己的名字缩写命名远程分支。因为到时候仓库会统计提交数，统计的时候会看的比较方便。

```
git push <远程 主机名> <本地分支名>:<远程分支名>
```

```
lenovo@Salieri MINGW64 /d/Unetfile/gitee_repo (qyk)
$ git remote -v
salieri git@gitee.com:salier1/imuproject_in_-dalian.git (fetch)
salieri git@gitee.com:salier1/imuproject_in_-dalian.git (push)

lenovo@Salieri MINGW64 /d/Unetfile/gitee_repo (qyk)
$
```

2.6 关于 git 远程仓库的合并管理

在你远程仓库自己的分支进行更新后，点击仓库的 pull request 可以申请远程仓库的合并。申请了的话我会查看代码，如果没有问题就合并到主分支。再次注意，提交到远程仓库时，提交到自己的分支上。不要提交到 master 分支，理论上来说也没有权限提交到 master 分支。更多的信息可以看[使用 Pull Request 功能进行代码审查 - Gitee.com](#)。

The screenshot shows the Gitee web interface for a repository named 'qyk'. The top navigation bar includes links for '代码' (Code), 'Issues', 'Pull Requests', 'Wiki', '统计' (Statistics), '流水线' (Pipeline), '服务' (Services), and '管理' (Management). The repository page shows the current branch 'master' is 2 commits behind. A dropdown menu is open, showing options to '新建 Pull Request' (Create Pull Request), '新建 Issue', '新建文件' (Create File), '新建 Diagram 文件', '新建文件夹' (Create Folder), '新建子模块', and '上传文件' (Upload File). The file list on the left includes files like 'model', 'model_result', 'pretrained_models/resnet', 'u3plus', 'utils', '配置管理', '.gitignore', 'LICENSE', 'README.en.md', 'README.md', 'pix_acc.png', 'predict.py', and 'test.csv'. The right sidebar shows the repository's introduction, release status, contributors, and recent activity.



2.7 关于本地仓库的更新

使用 `git pull remote branch` 这样的格式来更新本地仓库。假如仓库名通过 `git remote -v` 来查看。一般来说就执行 `git pull remote master`，只把 `master` 分支更新到本地。当然，你自己分支更新到本地也可以。但 `master` 分支是基线，通过冻结这个基线来保证全部人的代码一致。

```
lenovo@Salieri MINGW64 /d/Unetfile/gitee_repo (qyk)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 1021 bytes | 102.00 KiB/s, done.
From gitee.com:salier1/imuproject_in_-dalian
   fef2bd4..cbd62af  master    -> salieri/master
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

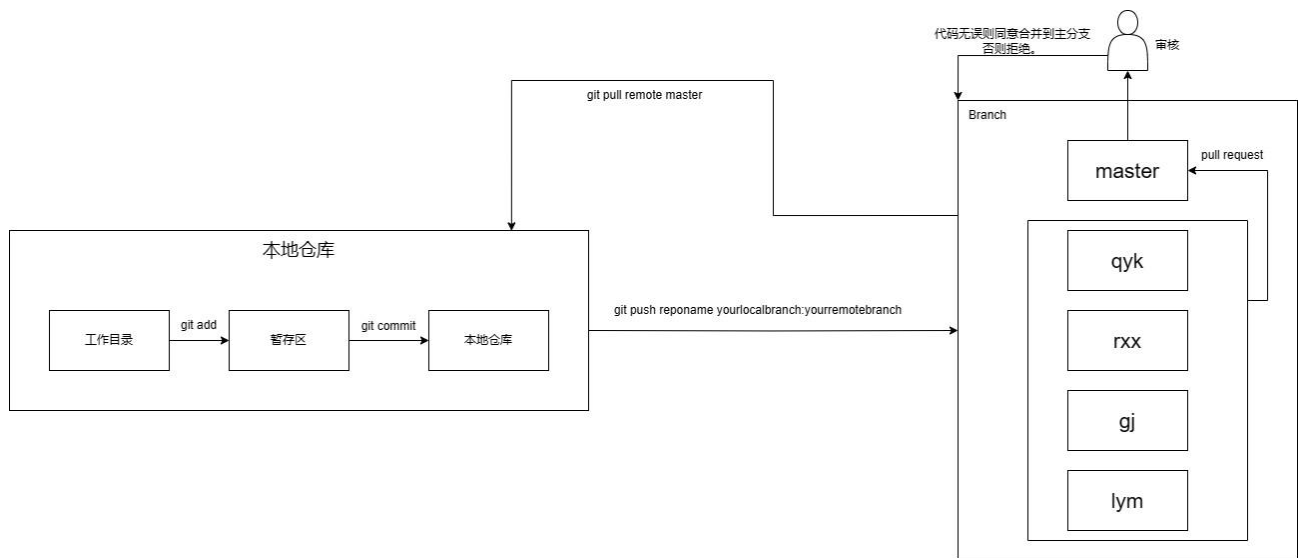
    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

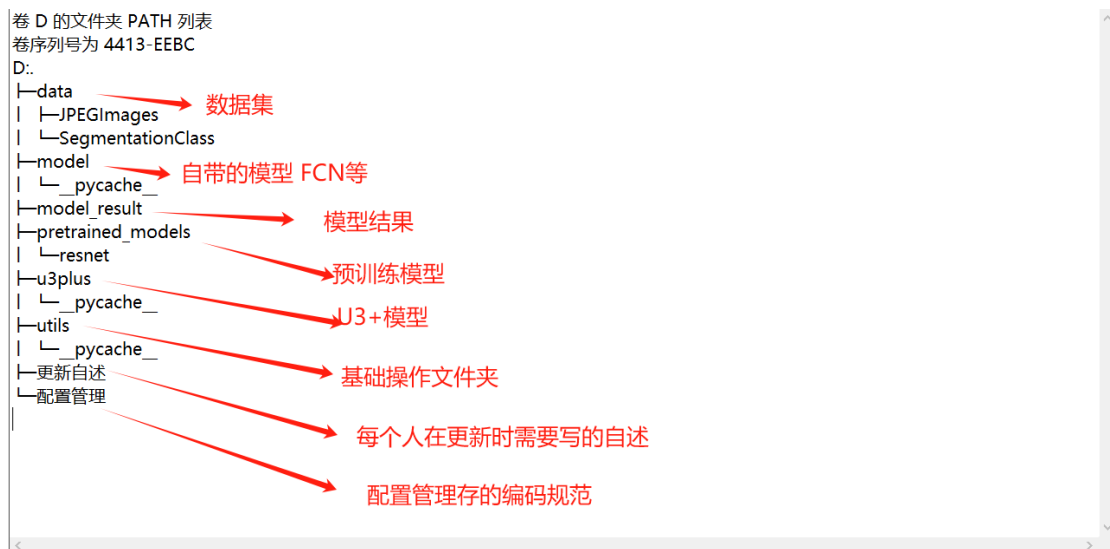
    git branch --set-upstream-to=salieri/<branch> qyk
```

2.8 关于流程介绍

如果你对上述的过程感到混乱，可以看下边这个图。代码每日进行一次提交，第二天每个人都需要更新自己的仓库。



2.9 关于仓库的目录结构



3. 模型的使用

3.1 模型的绘图

模型绘图要通过 Tensorboard 进行，相关教学会在后续更新。