tiny-training流程

1. 下载*tiny-traning*

整个实验中不涉及到开发板的所有部分都在这个文件夹里了。

```
git clone https://github.com/mit-han-lab/tiny-training.git
```

2. 来到*tiny-traning/algorithm*下，按照*README.md*中的步骤执行

2.1. *Setups*

2.1.1. 环境设置

这里按照它给的指令执行即可。

2.1.2. 数据集准备

*Dataset preparation*这一栏的第二点中蓝色*here*链接，跳转之后就是这个→，点进去后，下载、执行*make_all_datasets.sh*之后就可以得到aircraft, cub200, flowers102, food, pets, stanford_cars的数据集，也可以分别下载。

对于VWW数据集，*Dataset preparation*这一栏的第三点中蓝色*here*链接，跳转之后就是这个→给出了操作方法，这里我没看懂也没弄明白，在tiny-traning这里也是标了“**TODO**”。

2.2. *Usage*

总的来说，（按我的理解）*tiny-traning/algorithm*的核心是*train_cls.py*文件，它会调用*assets, configs, core, quantize*这4个文件夹下面的相关代码，

- 其中*configs*文件夹中存放的是我们可以自行设置的配置文件（以.yaml格式存储）；
- *assets*文件夹下预先存放了准备好了（指用ImageNet数据集进行了预训练，并执行了PTQ）的3个模型文件（分别是*mbv2-w0.35, mcunet-5fps. proxyless-w0.3*这3个模型，以.pkl的格式存储）；
- *core*和*quantize*文件夹更像是自行编写的可供*train_cls.py*调用的库；
- 还有一个*scripts*文件夹，里面的文件算是一种“官方配置”，运行之后是为了展示QAS和稀疏更新的效果的，有一定参考意义，但是太过固定，也就是说碰上自己的数据集和应用要求的话，还是要自己设置配置文件的数值；
- *train_cls.py*应该是模拟实机，预先展示一下实机上的效率。

2.2.1. 模型量化

实验已经给了预训练好且PTQ过的3个模型的.pkl文件，当然也可以自己拿模型从预训练开始做，最终得到.pkl文件。（要这样做的话可以参考*quantize*文件夹，这边我没有做这个操作所以没仔细看里面文件的内容）

2.2.2. QAS

这个应该也不需要有什么改动，执行*train_cls.py*的时候会自动判断调用的.yaml文件里面是否设置了执行QAS，如果设置了执行QAS，会调用相关文件（大概是*quantize*文件夹中的某个文件）、执行相关操作。

2.2.3. 稀疏更新

这个应该是要自己找合适的配置，这边我没做出来ㅠㅠ
所有的配置设置好后，运行*train_cls.py*之后，会在运行文件夹（自己设置）中生成*checkpoint*文件夹，*config.yaml*和*exp.log*文件。

3. 来到*tiny-traning/compilation*下，按照*README.md*中的步骤执行

我猜这边是用tvm这个框架给模型做了优化，对2.2.1.得到的模型执行.yaml里面的配置，最终得到可以下载到实机的深度学习网络。

3.1. *Setup forked tvm*

他的意思是从蓝色链接进去，从源码安装tvm（这也是个用在深度学习上的框架）
第二个蓝色链接即tvm官网的教程→

3.1.1 下载tvm源码

这里是将官网教程的源码链接换成了自己的，即*README.md*中的第一个蓝色链接→

```
git clone https://github.com/Lyken17/tvm-hack.git
```

这里下载完了还要更新一下子模块（教程里写：”对于使用GitHub工具的win用户，可以打开git shell，输入以下命令“）：

```
git submodule update --init --recursive
```

3.1.2. 安装GCC/Clang，LLVM(<14.0)和python(<3.10)

这边我装的是clang-11.0.0.src.tar + llvm-11.0.0.src.tar + visualstudio-2019
跟着这个教程做的→

3.1.3. 编译tvm

我跟着上面那个教程做到底，完成了编译，但是时间可能有点长。

3.1.4. 把tvm放进路径

用合适自己系统的方式执行给出的两个命令即可：

```
export TVM_HOME=<DIR to third_party/tvm-hack>
export PYTHONPATH=$TVM_HOME/python:${PYTHONPATH}
```

3.2. 把2中得到的pytorch模型转换成.ir中间格式

执行下面的文件即可：

```
python mcu_ir_gen.py
```

3.3. 把.ir中间格式转换成.json格式

就是里面给的这个命令：

```
python ir2json.py <target IR path>
```