# “深度学习代码框架” 使用说明

注：

1. 本框架在于为组内成员养成较好的编程习惯、项目整体思维。，也便于之后的进一步合作，目前仅限于本组成员使用。非授权不得扩散；

2. 时间仓促，如有bug 望批评指正；也希望大家尽快上手，早日开始实战。

# 0. 数据集准备

本次作业准备了两个toys 数据集： MNIST & Fashion MNIST

百度盘链接：

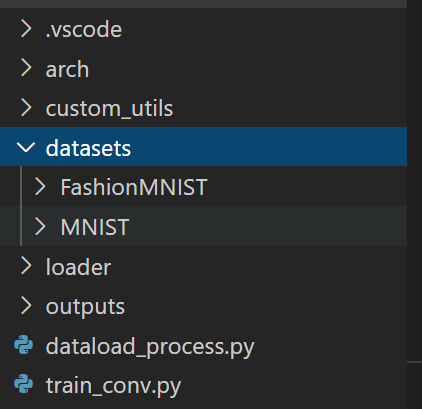
|  |
| --- |
| 链接：https://pan.baidu.com/s/1qmPFdUV78CP5LDoUX9KR5w  提取码：aw1l  复制这段内容后打开百度网盘手机App，操作更方便哦 |

下载数据集，解压到工作路径 $(root)/datasets 下，数据准备工作完成。

# 1. Project Overview

打开Vscode, 切换到深度学习conda 环境， File-> OpenFolder, 选择”rb\_BaseConv”这一路径。以下称该路径为项目的工作路径， 用$(root) 表示，基于它延申的路径即$(root)/XXX

首先了解整个项目的组织方式：在Vscode的左侧

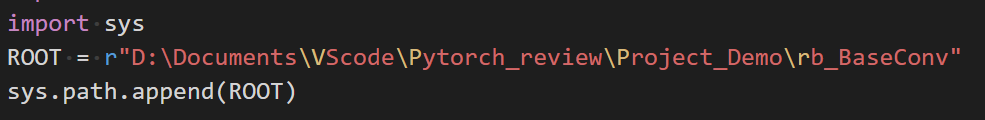


沿用 C++的project 组织方法 这里train\_conv.py 为训练程序的入口，其它子路经：

|  |
| --- |
| $(root)/arch 网络结构的定义  $(root)/custom\_utils 自定义的快速处理工具  $(root)/datasets 待加载的原始数据集文件，之前百度云下载的数据集放在此处  $(root)/loader自定义的数据集标准化方法(data.Dataset)  $(root)/loader 训练过程中产生的数据。包括log.txt, tensorbord 的文件， 以及训练过程中的权重文件导出 |

\*\*程序中部分脚本使用了绝对路径，运行前进行检查，更换成自己本地的对应路径，集中在basicLoader.py & train\_conv.py 中，请自行修改

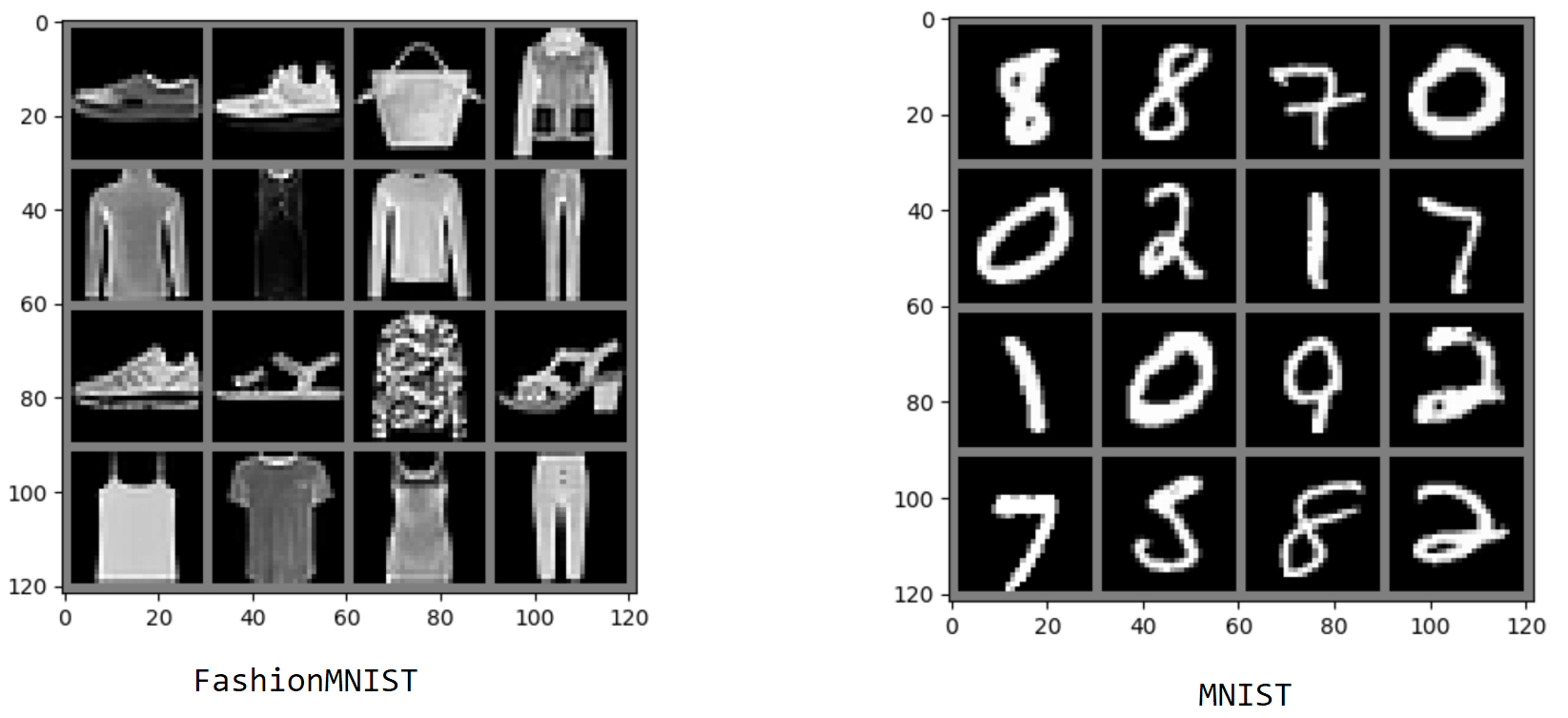
如：



# 2. Check dataLoading

在训练网络前，必须检查加载数据集的正确性， 称为一种“unit-test”, 该代码片段位于basicLoader.py内，从该脚本运行程序，将会执行

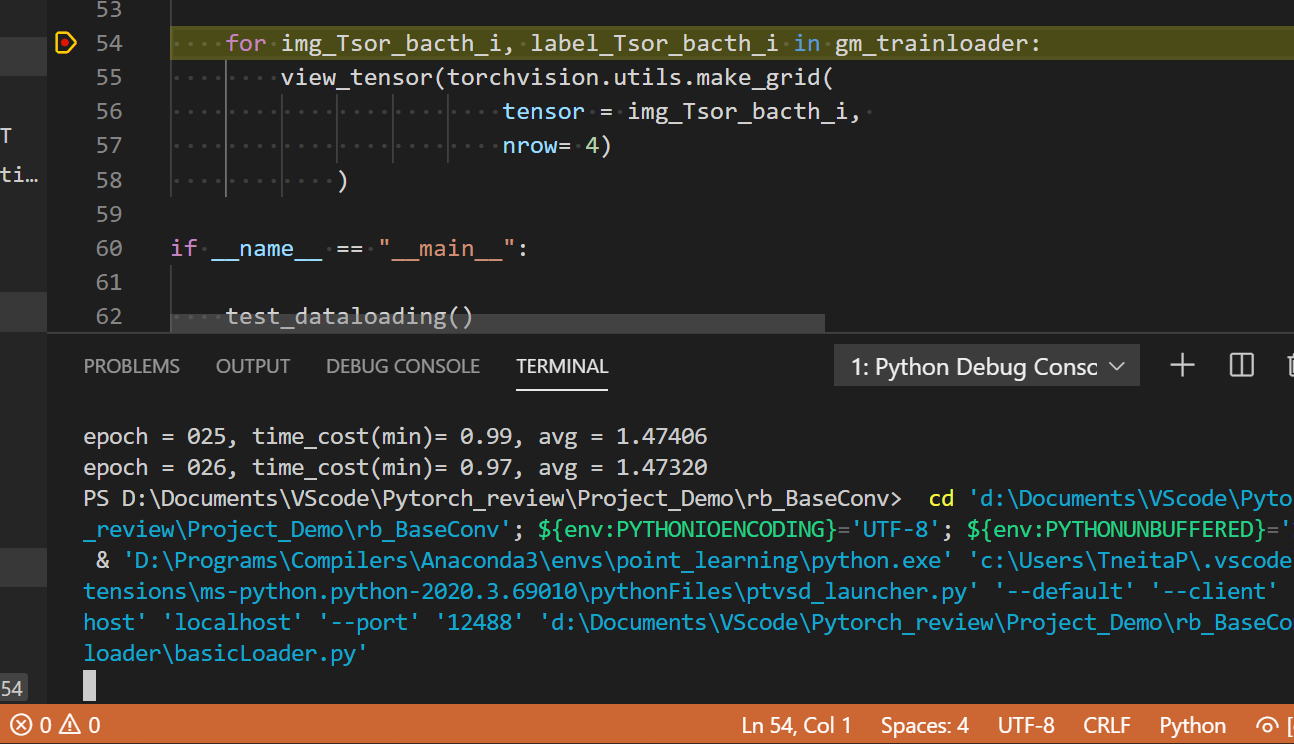
test\_dataloading()， 如果数据集和路径无误的话，将会如愿看到可视化的结果：



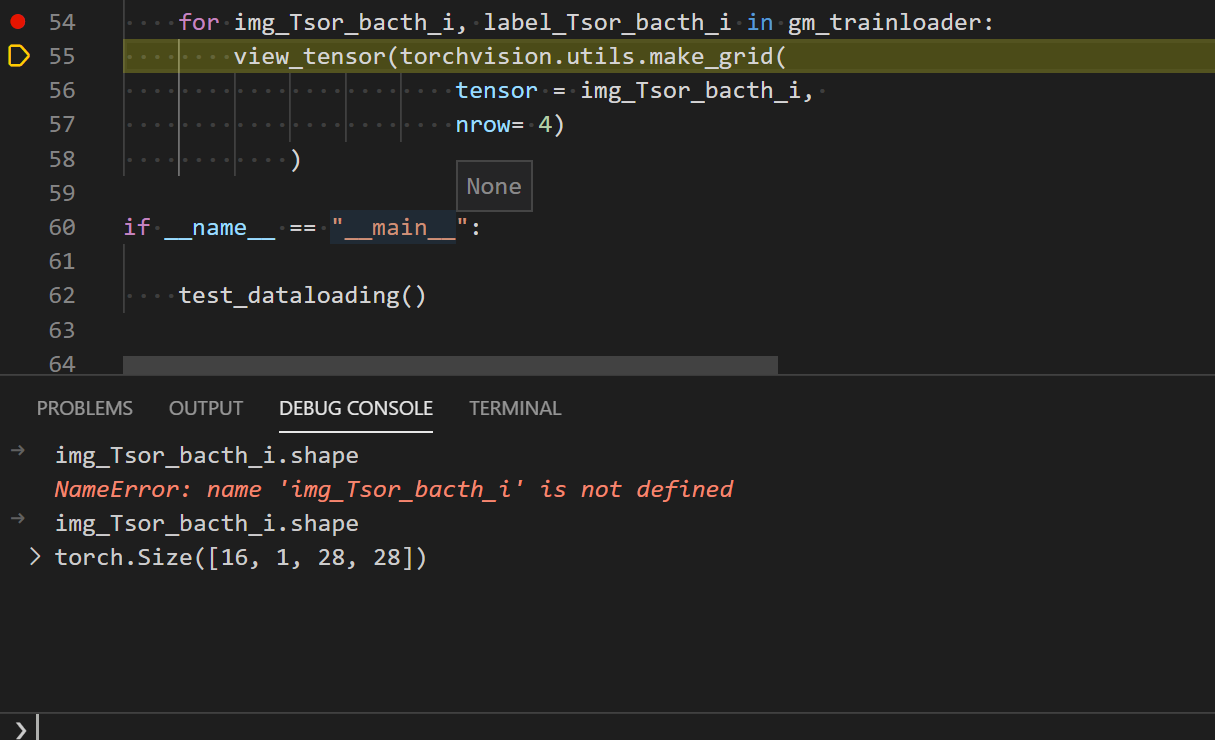
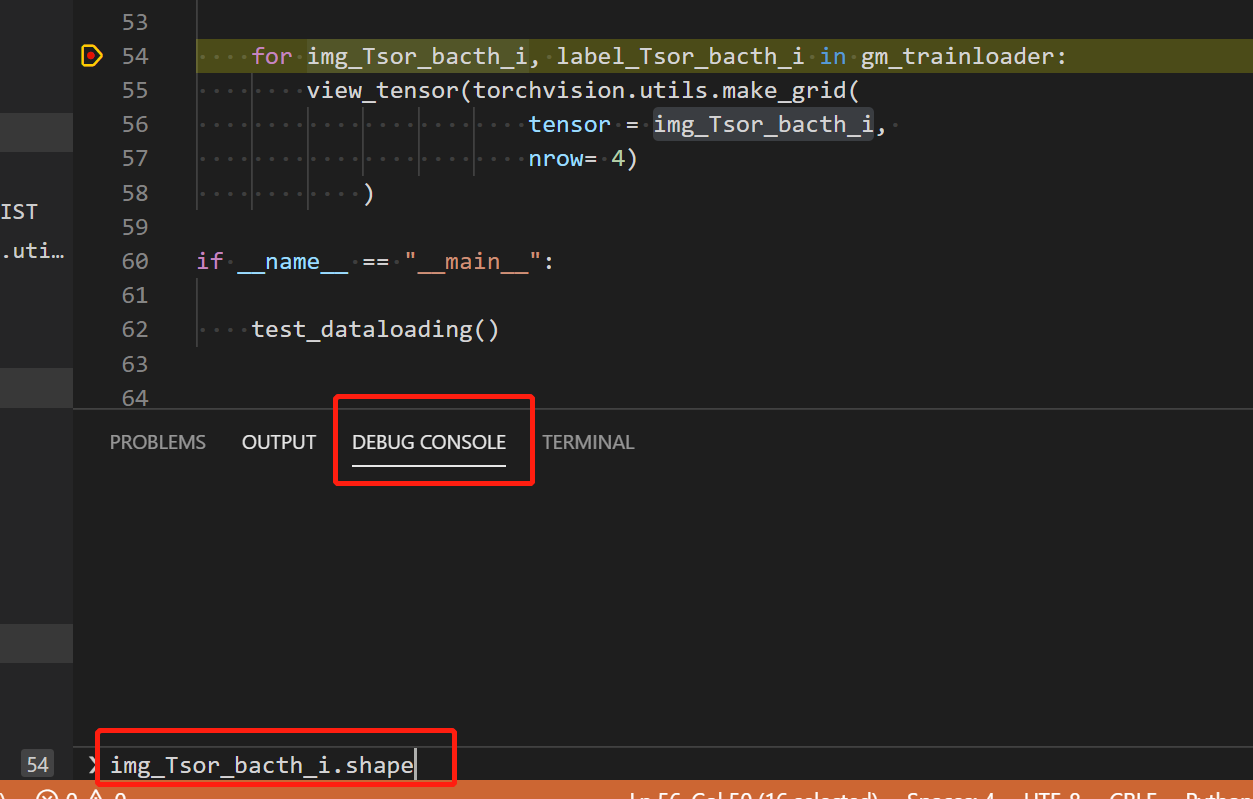
小作业：尝试使用Vscode 中的断点调试功能

具体：设置断点（单击对应行左侧）， F5运行程序到此处后， F10 单步运行。

由于python 为解释性语言，支持在运行中进行动态调试：



在Debug Console 中输入当前想要check 的内容，如：



从而可以检验加载tensor 的属性

# 3. 训练网络

在dataloader 中进行了待加载数据的确认之后，即可进入训练过程

本框架将训练过程抽象为以下几个对象之间的相互作用:

“gm\_”前缀的意思是 main 函数中的 global 变量。由于python 弱化了变量的类型，但在编程过程中，每一步都应该知道自己使用的变量的域、类型。

一些自定义函数的入参加入了“p\_“前缀，代表它是一个形参，与之后赋值的实参作区分

|  |  |
| --- | --- |
| config | 继承自custom\_utils.config 中定义的config 基类  负责掌管在训练过程中一切的超参数、路径问题、帮手函数…  具体见定义  cfg.writer 是一个tensorboard 对象，在训练过程中记录数据，可以实时呈现实验过程中的数据 |
| trainloader | 数据集加载器。该对象将入参dataset 加载到内存中，并转换为可迭代对象: 即可以在之后的for 循环中，逐个batch的获取训练数据 |
| net | 待训练的网络对象，内部参数需要进行初始化，否则进行随机初始化  初始化方法包括: kaiming, xavier, 以及用之前与训练的参数 |
| optimizer | 网络参数优化器，随机梯度下降的执行者  在之后的for 循环中，进行下面3个动作： |
| scheduler | 用于动态调节训练过程的学习率。 |
| criterion | 损失准则对象。常用的会在torch.nn 中定义好，自己任务的特定损失需要自定义。 |

torch.save 即对当前网络的状态进行存储，该框架预期了三种可能的weight 存储方法：

|  |  |
| --- | --- |
| KeyboardInterrupt | 在训练过程中发现了错误但是想保留当前的weight, 键盘Ctrl+ C ，可中断训练进行，保留最新的weight , 之后可以从这里加载。  这也是训练最外层嵌套“try”的目的 |
| processing | 训练过程中自动保存weight: |
| ending | 训练结束，保留最终的训练权重 |

通过上面的任意一种方式，都会将最新的weight保留到 outputs/ 路径下

# 4. 启动 tensorboard，观察训练结果

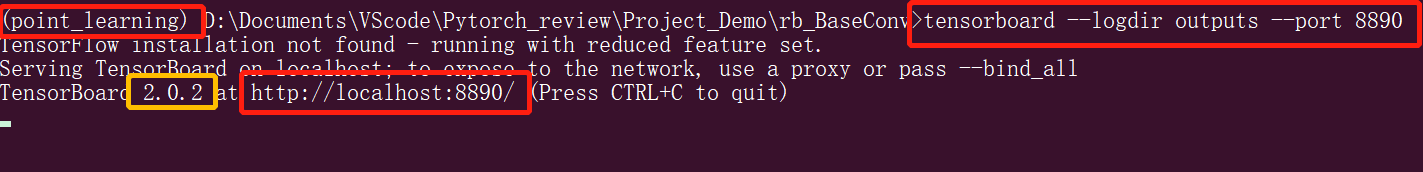
tensorboard 用于检测整个训练的进程。对应代码框架中的” SummaryWriter”行为。

在工作路径的root 位置打开命令行， 激活对应的conda 环境, 启动 tensorboard。

确保安装的 tensorboard 版本为2.0.2

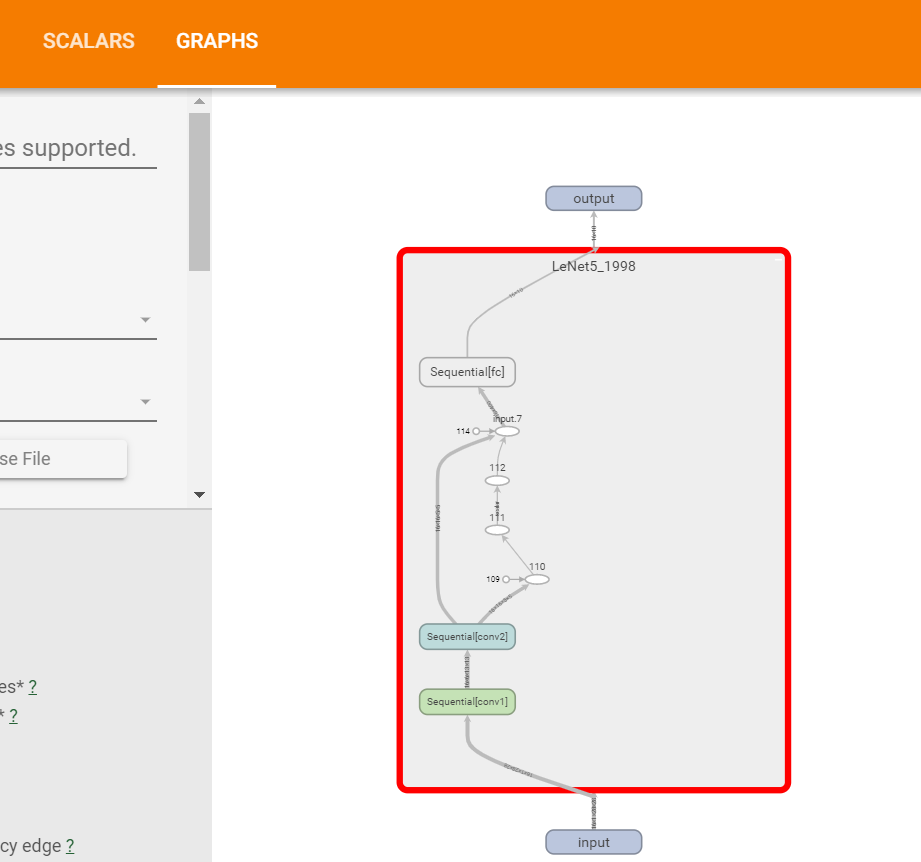
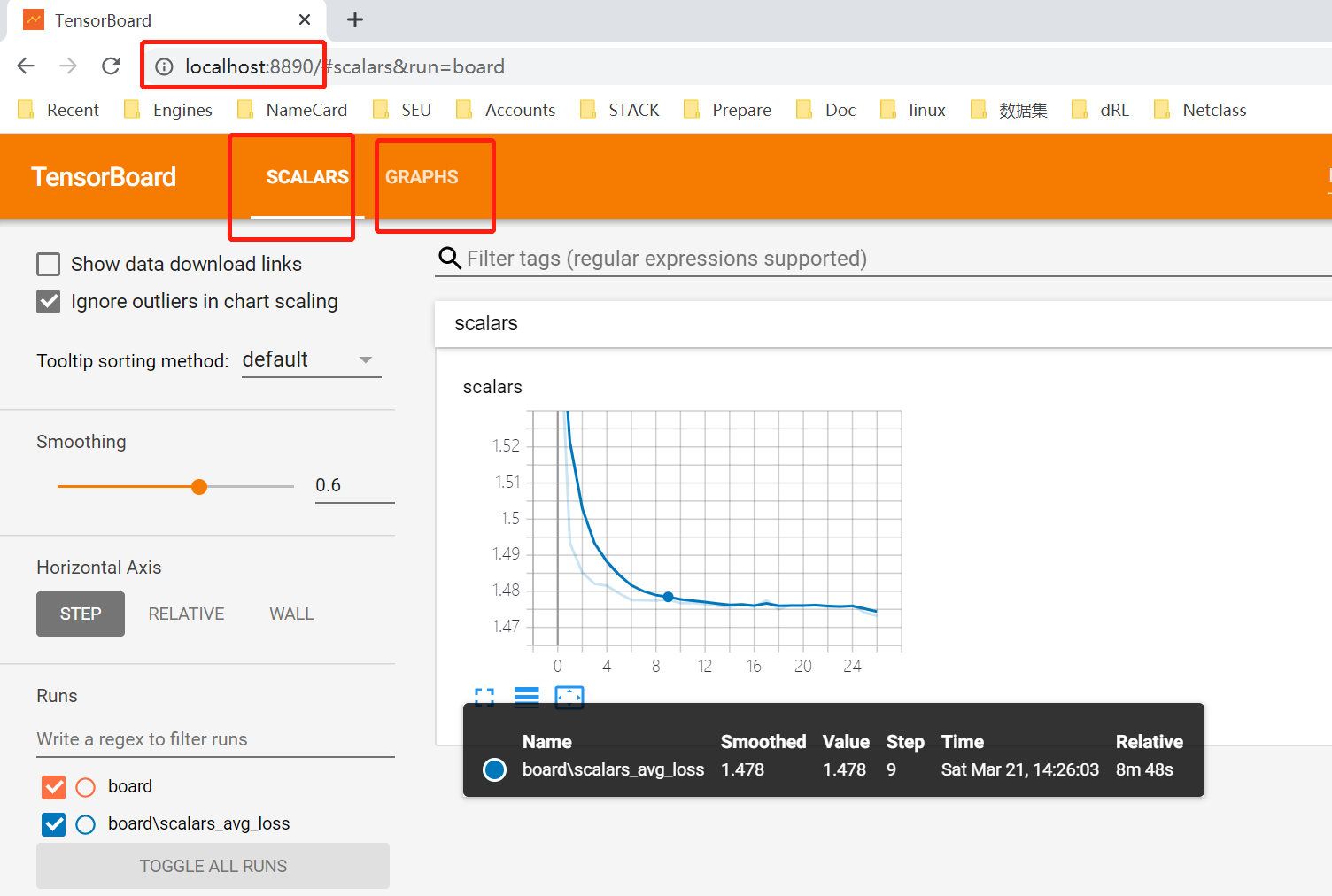
|  |
| --- |
| activate point\_learning  pip install tensorboard==2.0.2  tensorboard --logdir outputs --port 8890 |

结果：



tensorboard 利用存储event 事件，记录训练过程中产生的数据（见 outputs/board 路径下），每次开始新的训练，请清理之前的缓存文件，否则绘制的图像将发生混叠。

复制 ip 到浏览器中，即可打开tensorboard的localhost 界面



\*\* 更新：为了校验模型的过拟合，在训练过程中，每个epoch之后也会在eval 数据集【demo中简化使用test数据集，非实际情况】上进行准确率测试。

过拟合：在train数据集上的loss 降低，但是在eval数据集上的性能未发生变化甚至变差，此时可以判断模型已经在train数据集中过拟合，应当及时停止训练。

# 5. 在test 中评估训练结果

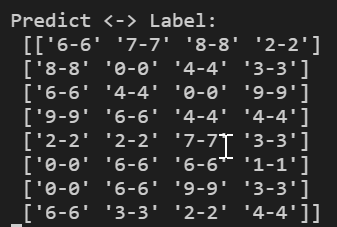
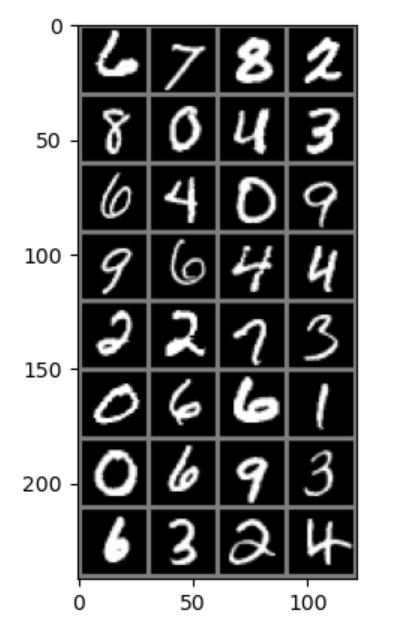
实现过程见新加入的test\_conv.py

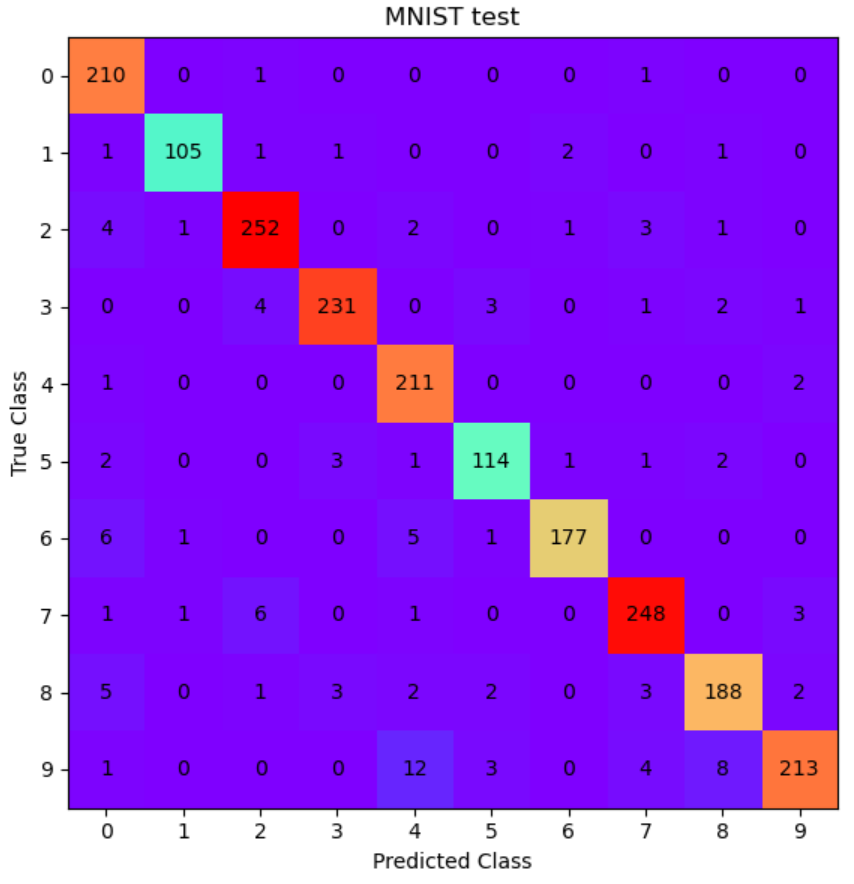
Evaluate 是对已训练模型的性能进行量化检验。

## 分类问题的指标

包括在测试集上的总体准确率、各个类的精确率和召回率（confusion matrix）描述。

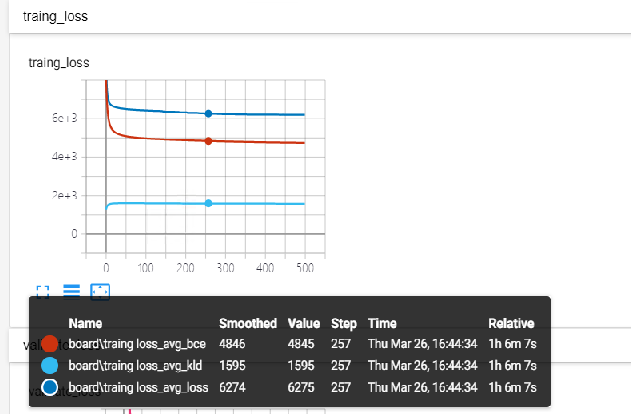


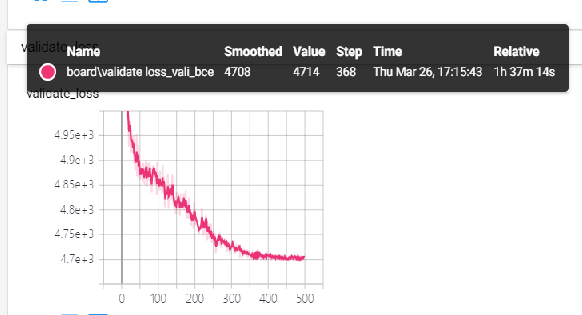


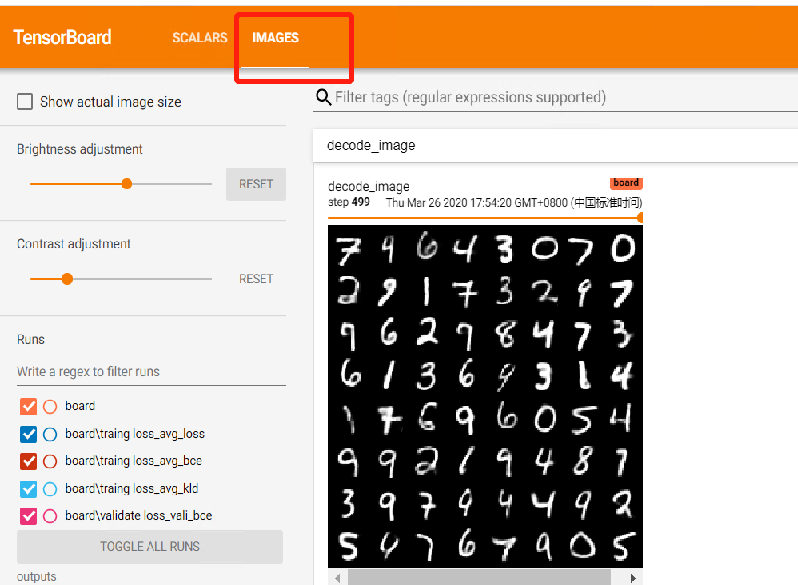


date0326 添加 VAE, 实现MNIST 的流形学习

1. 全连接performance

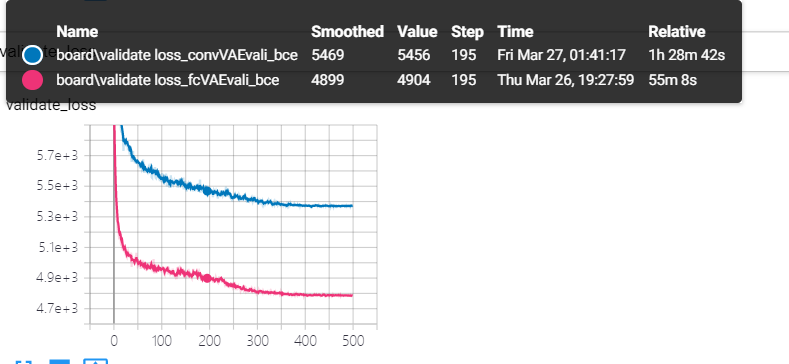






2) 卷积performance (并与全连接对比)





参数量的差异：

|  |  |
| --- | --- |
| fcVAE | 648,016 |
| convVAE | 167,464 |

可视化结果比较[ latent space dim =16]

|  |  |  |
| --- | --- | --- |
| input | convVAE output | fcVAE output |
|  |  |  |

|  |  |
| --- | --- |
| DCgan | Generator:[net info] para\_amount=2884608  Discriminator:[net info] para\_amount=2761728 |
| DCgan-mini | Generator:[net info] para\_amount=172608  Discriminator:[net info] para\_amount=142208 |