### 优化

针对于这个模型的优化部分：

1. 模型部分，使用nn.Sequential() 来重建了模型。

self.model1 = nn.Sequential(

nn.Conv2d(1, 6, 5, padding=2),

nn.ReLU(),

nn.MaxPool2d(2, 2),

nn.Conv2d(6, 16, 5, padding=2),

nn.ReLU(),

nn.MaxPool2d(2, 2),

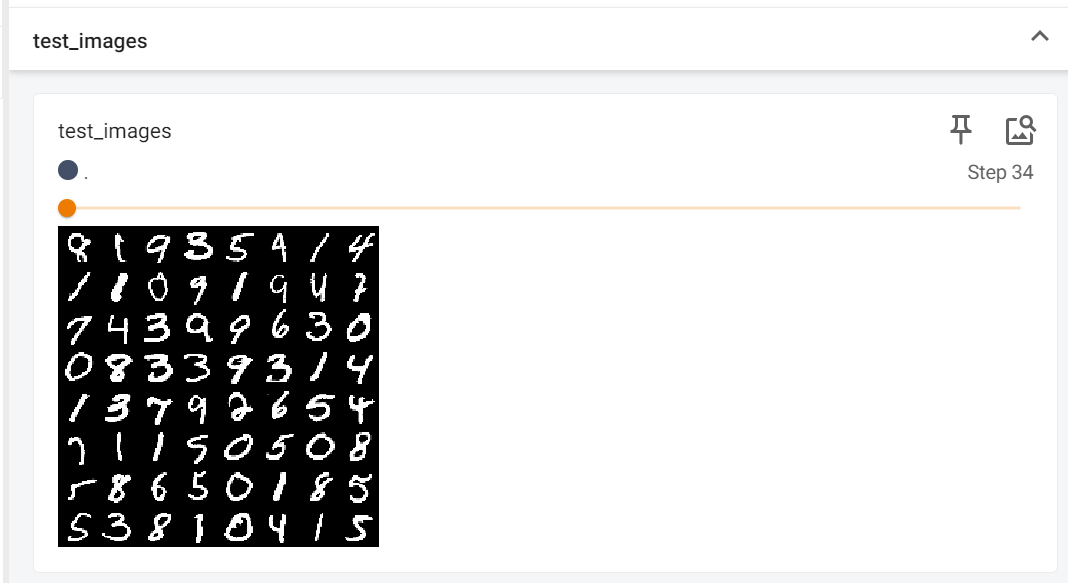
nn.Flatten(),

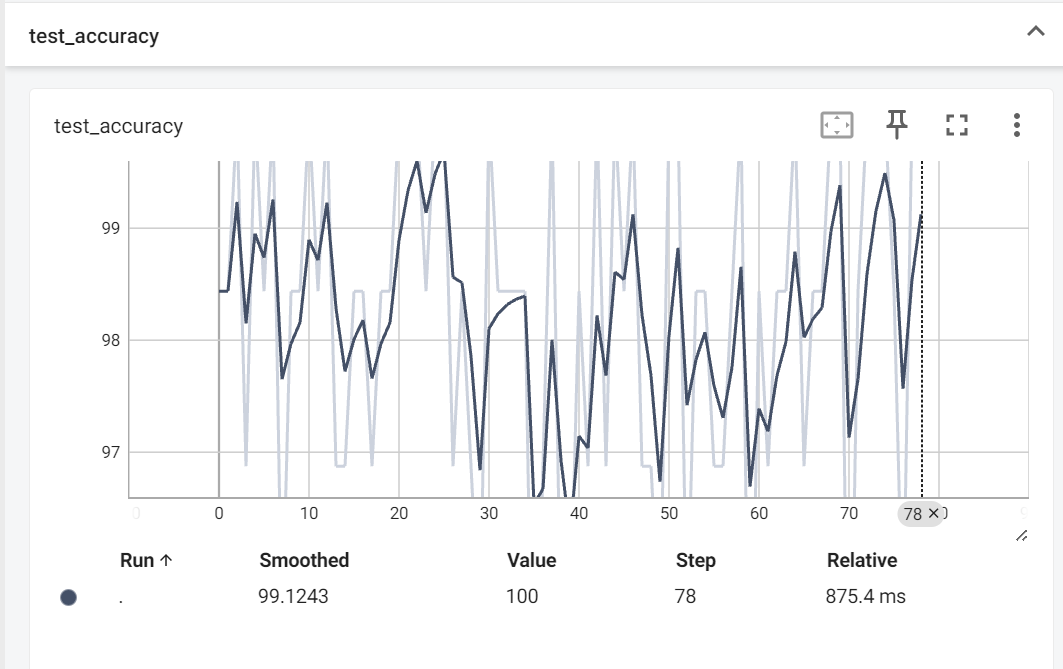
nn.Linear(16 \* 7 \* 7, 512),

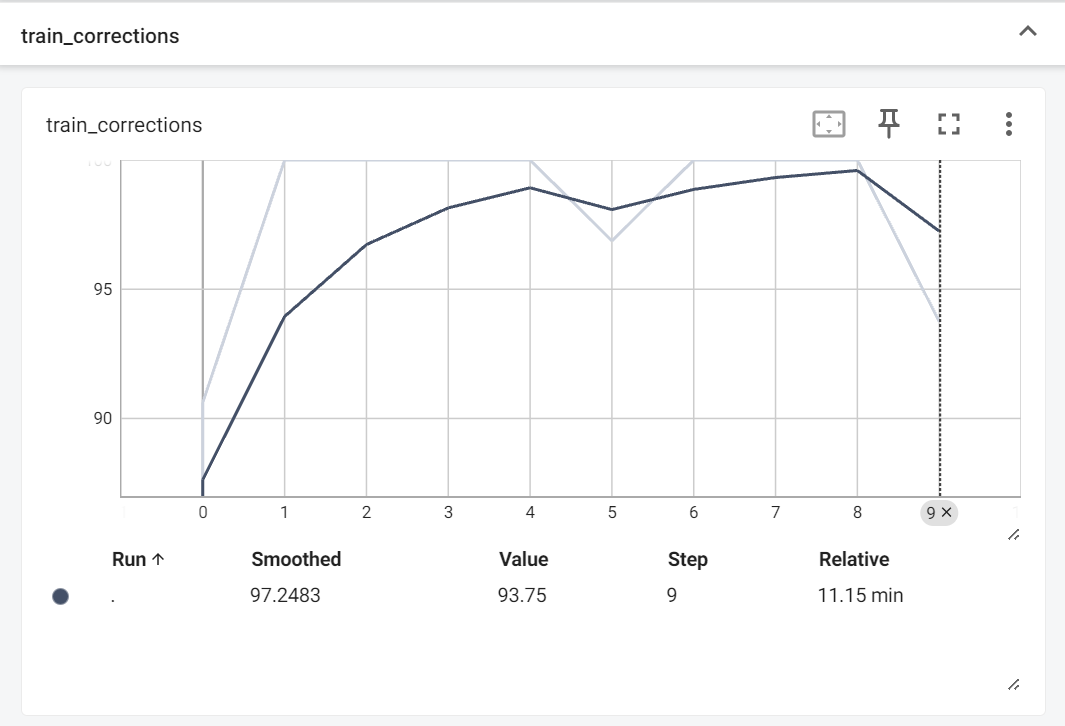
nn.ReLU(),

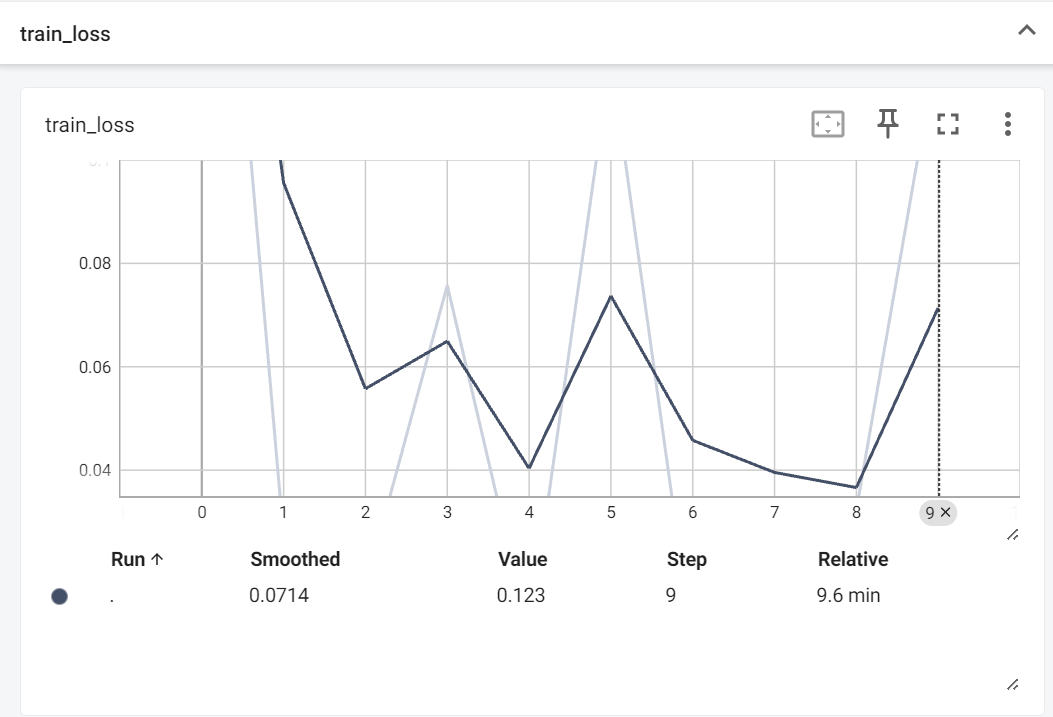
nn.Linear(512, 10)

1. 加入tensorboard来收集训练的batch，并且记录train\_loss, test\_loss, test\_accuracy. 便于后续检查模型输出结果。









1. debug了几个error
2. 没有英伟达GPU，去除了cuda的部分使用GPU的部分。

### 熟悉pytorch

pytorch几个重要的部分：

nn: 是用来写模型的主要模块，需要继承nn.Module. 并且overload init, forward。 其中也包含各种layer： conv2D, Linear, Pooling, flatten 以及其他的activation functions: sigmoid, relu, tanh

dataset: torchvision.datasets; MINST也是其中一个database； 在load时就可以进行download, 或者直接转换图片的格式。一般格式为： tensor, np.array, PIL. 训练时一般使用tensor格式

models: torchvison.model; audio, text, vision; 比如处理图形的VGG系列；某些模型可以直接import 使用

dataloader: 可以直接分割dataset按照自己想要的batch\_size. 也可以选择是否shuffle, 并且可以选择是否drop\_offset.

### 熟悉CNN架构

1. 构建CNN model: 继承nn.Module. 写layer, overload forward （正向传播）function
2. 确定train, test data: torchvision.datasets有许多开源dataset. 可以直接下载，分开train,test data.如果是图像的话，确保图像load之后需要保证是tensor格式： PIL，np.array都可以直接调用function来转化成tensor格式
3. 确定loss function: classification可以使用entropyloss。其他的regression之类的可以使用RMSE, MSE 之类的。
4. 确定optimizer。 我们这里使用的是SGD(stochastic gradient descent). 一般常用的也有GD。LR之类的可以调整。最好是前期训练大一点，后面小一点。（想快一点找到最优解，但是LR在后面过大会导致在最优解左右反复横跳）
5. 每次计算完loss后进行back propagation (反向传播)，算SGD/GD之类的来不断更新parameters的值。
6. 使用tensorboard来可视化结果：包括训练图片，输出图片，train loss, test loss, accuracy来方便分析结果。