Pytorch避坑：

1. 列表操作什么的一定要转化成numpy进行运算 运算时间将大大节约
2. 对于超参数务必提前定义，否则网络修改将为极其繁琐。包括一些seq\_length类的。
3. 注意数据的维度，在进行loss计算的时候一定要确保数据维度相同，必要时可以采取data.View(-1,1)这种操作，不一定非得按照batch\_size来，因为很可能dataloader最后一组数据不满足batch\_size
4. 注意cuda、cpu、numpy、tensor数据之间的转化 有些时候需要将tensor强制转化 或者只取data进行计算。例：data.cpu().numpy()直接在gpu上训练的tensor不能直接转化成numpy
5. 加载预训练模型在一定程度上能节约模型训练时间，此时往往需要将预训练模型中的参数freeze 只需要将parameters requires\_grad=False
6. 各个激活函数的比较
7. 各个损失函数的比较(target是否需要处理成one-hot，output是否需要经过softmax层)（如何设置不同维度loss的权重）
8. 关于数据不平衡的处理方法 各个类别所占比例相差较大
9. 关于动态调整学习率的方法 先大后小
10. 能用符号就用符号表示，最好少在模型中直接使用数字
11. 列表添加操作，采用np.append耗时最少
12. Loss计算可以采用+=loss.item()的方法，从而获得总体的平均误差
13. 获得模型的参数名字以及对应的值或者对应的梯度更新可以选择

For name,param in model.named\_parameters():

print(name,param.requires\_grad)

1. 神经网络的可解释性 《The Mythos of Model Interpretability》线性模型的可解释性不一定高于深度神经网络。第一点：透明度，表现形式为可模拟性，人类可以手动模拟机器学习模型。第二点：可分解性，模型的每个部分都有直观的解释。第三点：算法，算法具有独一无二的解。模型的可转移性 是否能推广用在未见过的数据上，模型的安全性 是否能够防止欺骗。

对于线性模型，直观的感受到模型的权重与偏置的具体实际含义，例如房价与房屋面积、距市中心距离的线性模型，因此可以产生很多明确的优化方案，比如增加某个权重的比例。

深度神经网络 第一个特点就是神经元多，参数众多；第二个特点就是结构分层，且隐含层众多；第三个特点就是神经网络的参数对应的特征不是人工设计的，是神经网络在学习过程中自己选择的。