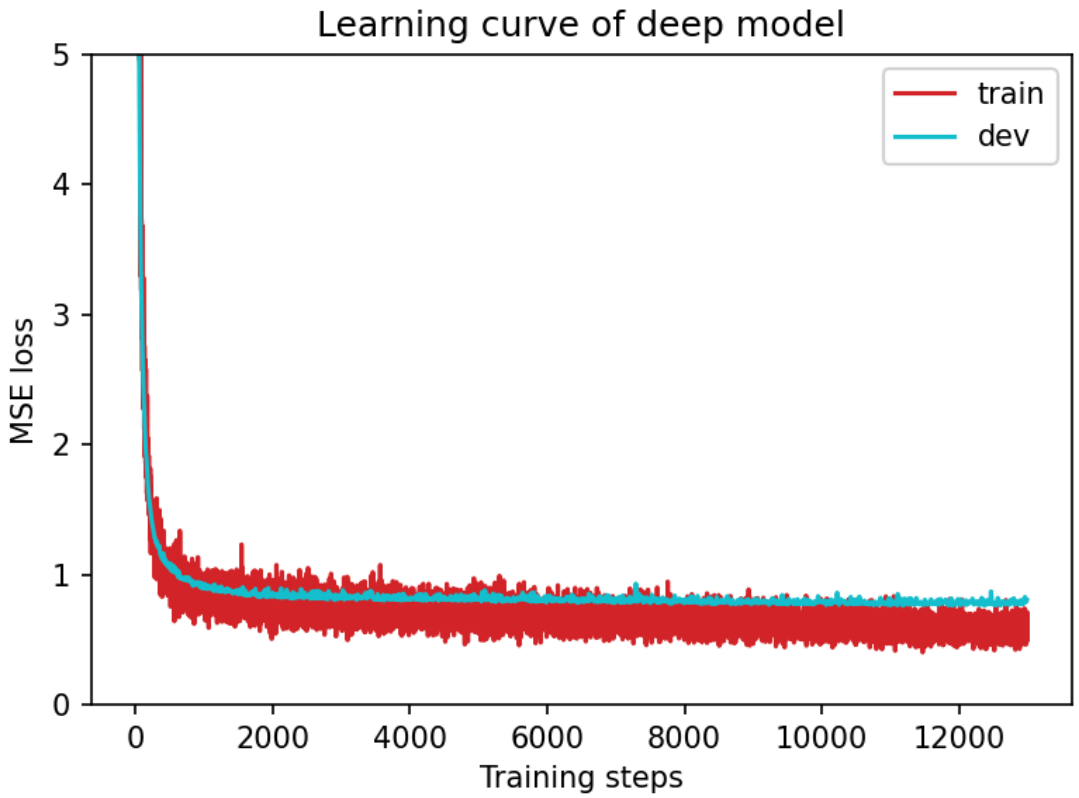
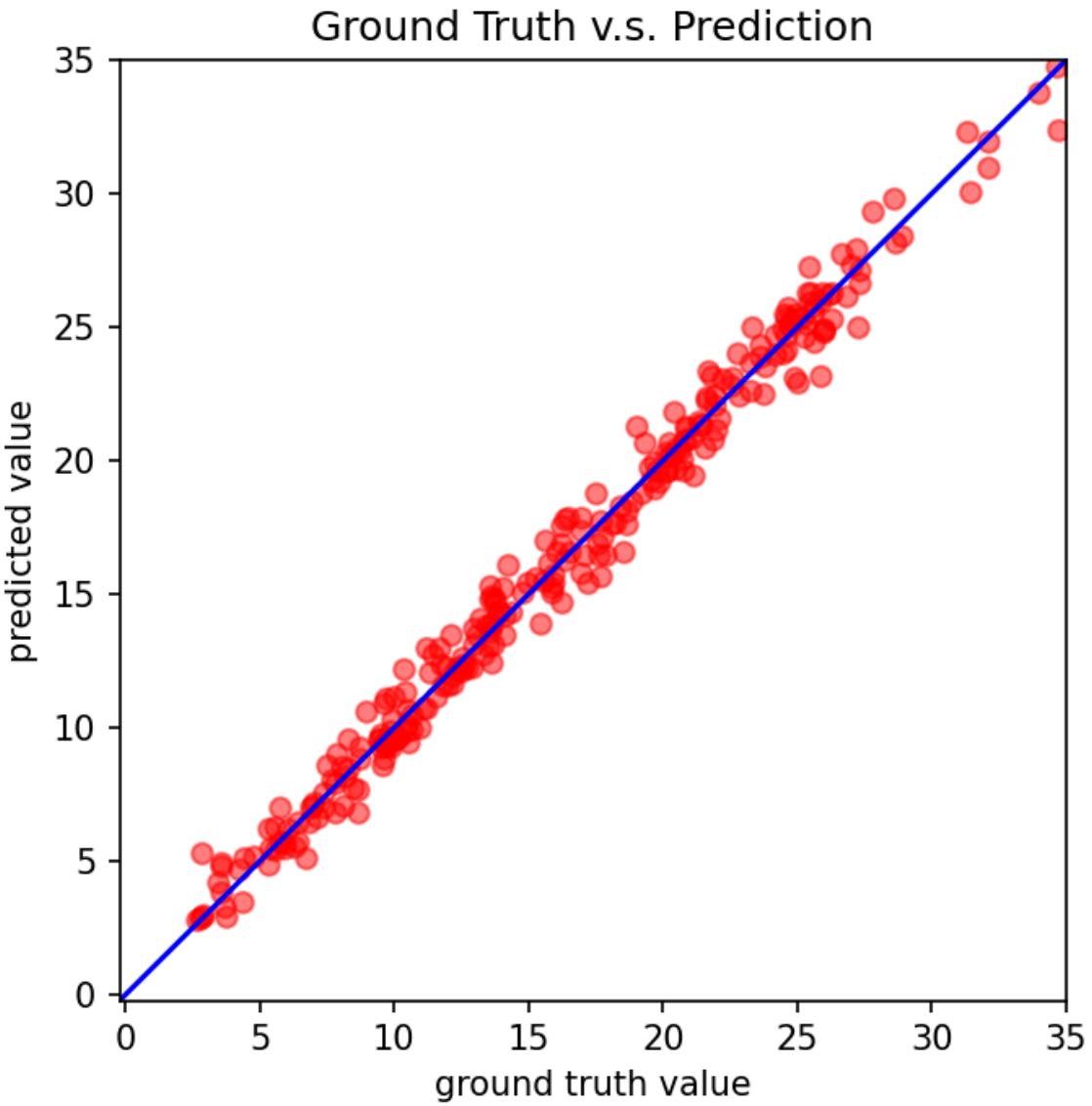
HW1实验

最初代码实验结果：






```
Saving model (epoch =  933, loss = 0.7749)
Saving model (epoch =  965, loss = 0.7712)
Saving model (epoch = 1027, loss = 0.7683)
Saving model (epoch = 1119, loss = 0.7669)
Saving model (epoch = 1196, loss = 0.7640)
Saving model (epoch = 1234, loss = 0.7629)
Saving model (epoch = 1243, loss = 0.7602)
Finished training after 1444 epochs
```

其实可以看到，loss还是很大的

```
Finished reading the train set of COVID19 Dataset (2430 samples found, each dim = 93)
Finished reading the dev set of COVID19 Dataset (270 samples found, each dim = 93)
Finished reading the test set of COVID19 Dataset (893 samples found, each dim = 93)
```

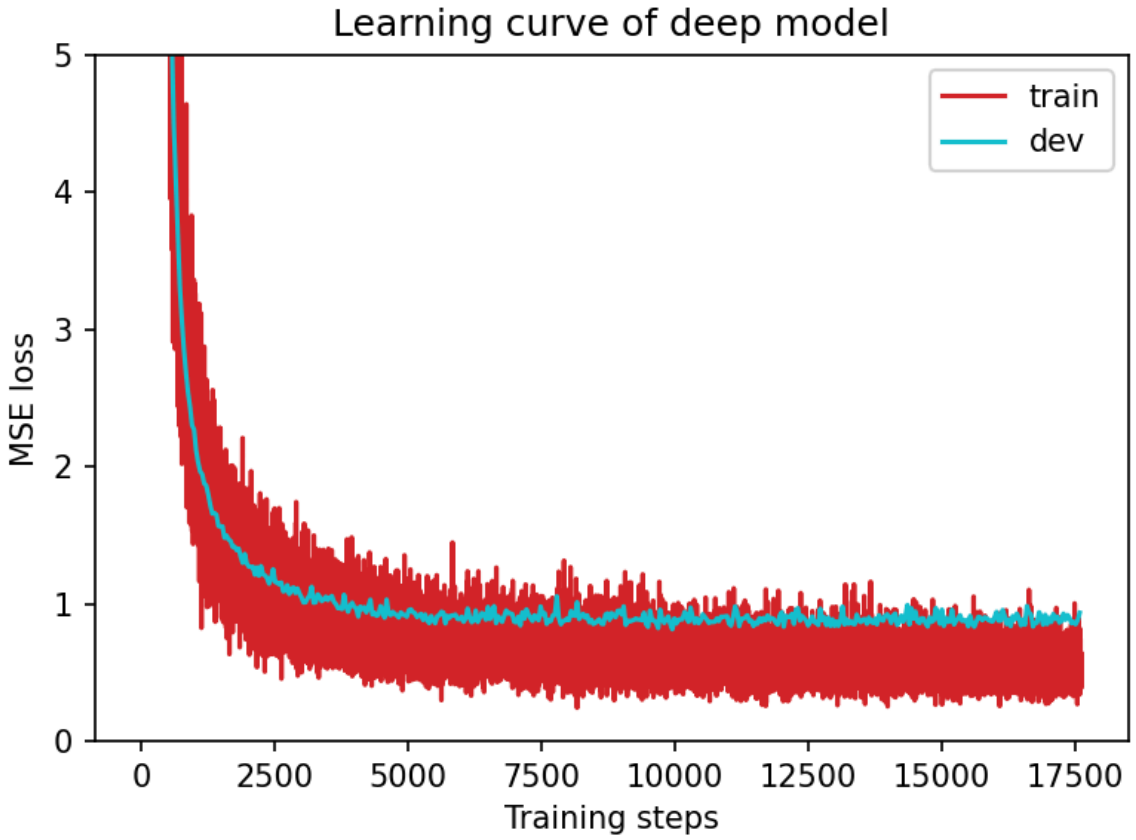
上面是对数据的初始化结果。

下面进行进一步优化，简单调整一些参数。

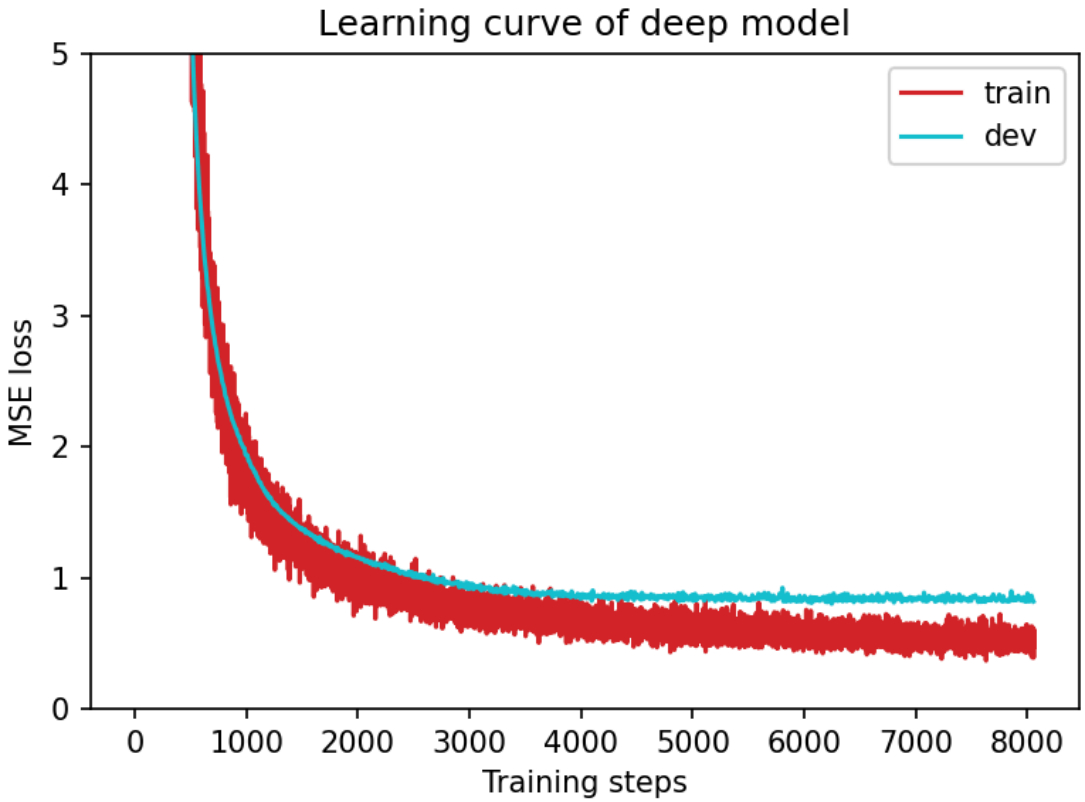
这是最初的参数

```
config = {
'n_epochs': 3000,#最大训练轮次
'batch_size': 270,#批次大小
'optimizer':'SGD',#优化器类型
'optim_hparas': {#优化器超参数
'lr': 0.001,#学习率
'momentum': 0.9# SGD动量
},
'early_stop': 200,#早停阈值（200轮无改善则停止）
'save_path':'models/model.pth'#模型保存路径
}
```

首先经过我的搜索，听从建议不再使用SGD优化器，而采用相较而言更稳定且收敛更快的Adam。然后进一步将batch的数量调低，增加梯度估计的多样性。



但是可以看到效果甚至不如第一次，学习率也只有0.8146。再将batch改为原来的270.结果学习率为0.8015.



将batch改为300，最终得到的loss为0.7744，还是不如之前。再改为280的loss为0.7615.

下一步再单独修改早停阈值为300轮，最终的结果还是0.7602。后面再降低学习率，提高最大训练轮次和早停阈值，最终loss也并未进步。到这里单独调整参数这里感觉已经不用。

经过上网查询得知，batch变换相对的学习率也应该变换，我这里将batch降低到64，相应的学习率也应该降低。这里将学习率定为5e-4，得到的结果如下图，可以看到loss减少的同时，总的训练轮次也减少。

```
Saving model (epoch =  409, loss = 0.7559)
Saving model (epoch =  441, loss = 0.7431)
Saving model (epoch =  469, loss = 0.7360)
Saving model (epoch =  534, loss = 0.7325)
Saving model (epoch =  692, loss = 0.7300)
Saving model (epoch =  844, loss = 0.7242)
```

然后就去调整早停阈值，先将其减为100，发现

```
Saving model (epoch =  469, loss = 0.7360)
Saving model (epoch =  534, loss = 0.7325)
```

发现loss增加，说明停早了，那么再一次修改为150，还是为0.7325.而将早停阈值改为180时，loss就为0.7242.后面再将阈值改为400也没用提升，于是就将阈值固定为180。

在这里我又想到了Adam，于是将优化器设为Adam，并且进行训练获得的loss为0.7293，于是我又将阈值提升到了300，效果并不理想，于是将阈值改为400，最终loss为0.7174

```
Saving model (epoch =  832, loss = 0.7409)
Saving model (epoch =  888, loss = 0.7376)
Saving model (epoch =  934, loss = 0.7293)
Saving model (epoch = 1305, loss = 0.7289)
Saving model (epoch = 1339, loss = 0.7266)
Saving model (epoch = 1376, loss = 0.7174)
```

最终的参数为：

```
config = {
'n_epochs': 4000,#最大训练轮次
'batch_size': 64,#批次大小
'optimizer':'Adam',#优化器类型
'optim_hparas': {#优化器超参数
'lr': 5e-4,#学习率
#'momentum': 0.9            # SGD动量
},
'early_stop': 400,#早停阈值（200轮无改善则停止）
'save_path':'models/model.pth'#模型保存路径
}
```

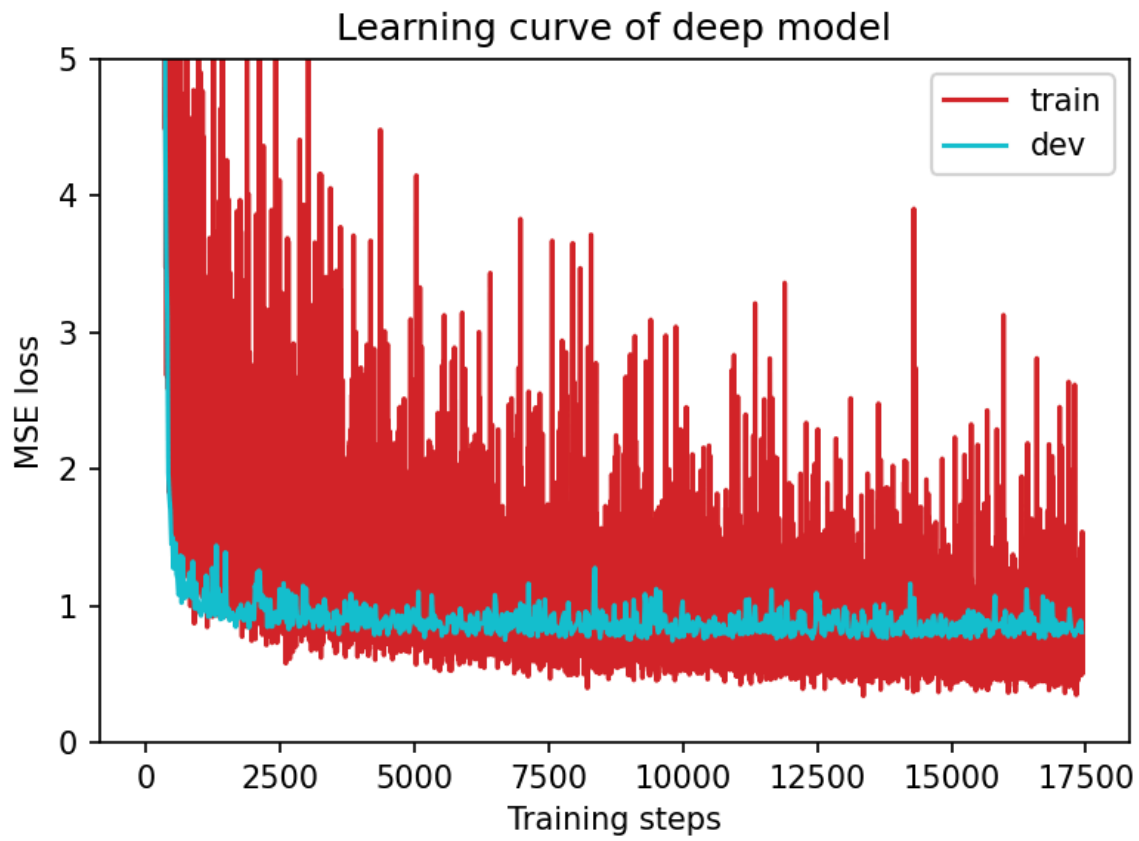
第二步，我决定从网络结构入手。

最初的网络结构为

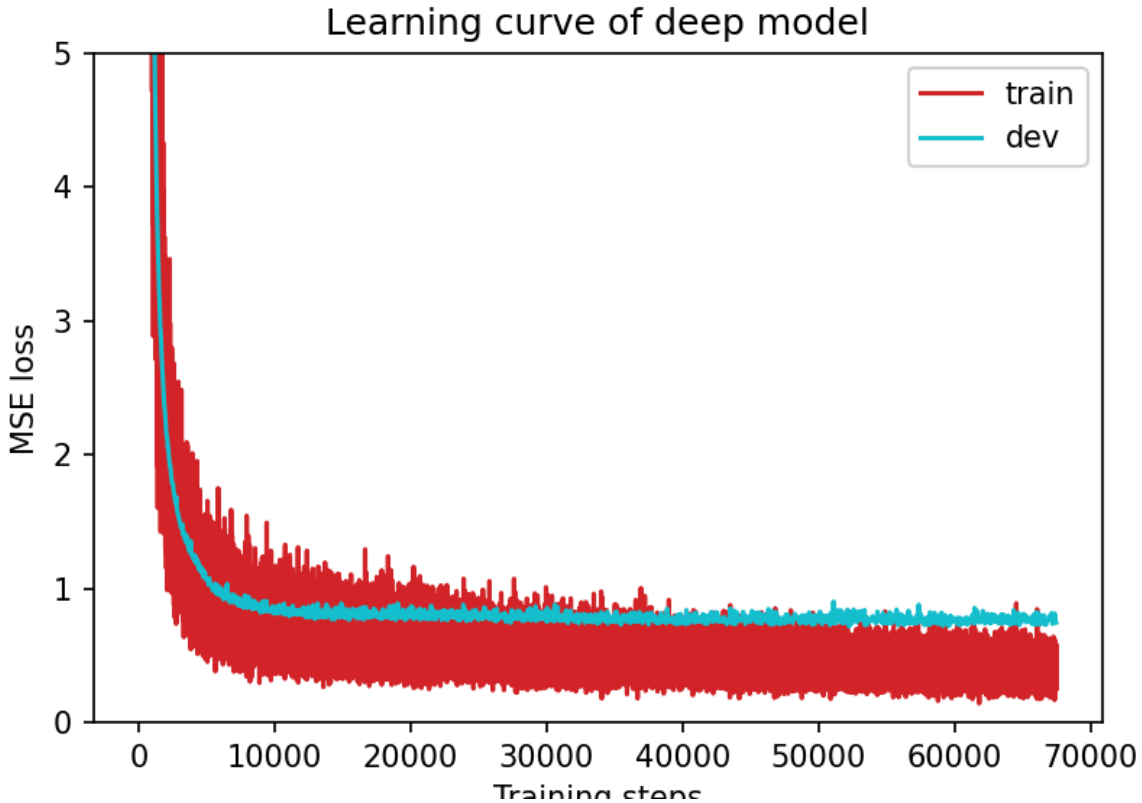
```
self.net = nn.Sequential(
    nn.Linear(input_dim, 64),
    nn.ReLU(),
    nn.Linear(64, 1)
)
```

然后首先先加一层ReLU层，最终loss没有变化。

因为不清楚如何改结构，这里采用ai提供的建议，改为多层，但是出现了训练集损失明显波动，即使更改了batch和学习率依旧改观不是很明显，并且loss进一步升高，例如：



上图是修改过batch和学习率的结果，相较于直接套用ai的结构loss波动较小，但是对于之前的一层波动太过于剧烈。下图是一层ReLU的图：



二层ReLU和一层没什么区别。至此对于结构方面目前还不清楚该如何优化

第三步，使用部分特征

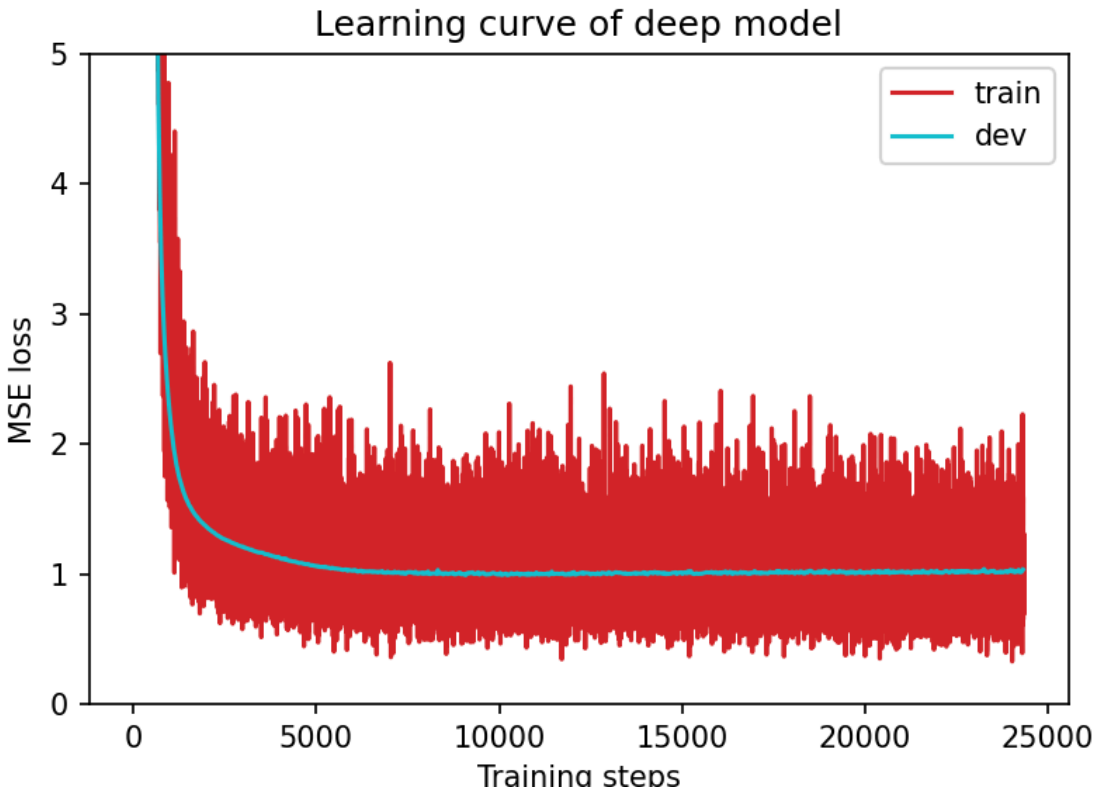
修改代码为：

```
if nottarget_only:#是否仅使用部分特征
feats = list(range(93))#使用全部特征
else:
#TODO: Using 40 states & 2 tested_positive features (indices = 57 & 75)
#pass
feats = list(range(40)) + [57, 75]#只使用前40个state特征和2个测试结果特征
```

进一步改变target_only为true

可以看到结果并不理想：

```
Saving model (epoch =  196, loss = 1.0029)
Saving model (epoch =  197, loss = 1.0023)
Saving model (epoch =  206, loss = 1.0022)
Saving model (epoch =  214, loss = 0.9999)
Saving model (epoch =  224, loss = 0.9972)
Saving model (epoch =  240, loss = 0.9909)
```



毕竟机器学习的目的是选择最有效、最能反应目标变量的特征，而不是越多越好，并不是所有的特征都对预测有帮助，但是这里结果不好也有可能是我的超参数是相对于全部特征而言，对于局部特征仍需要独立调参。

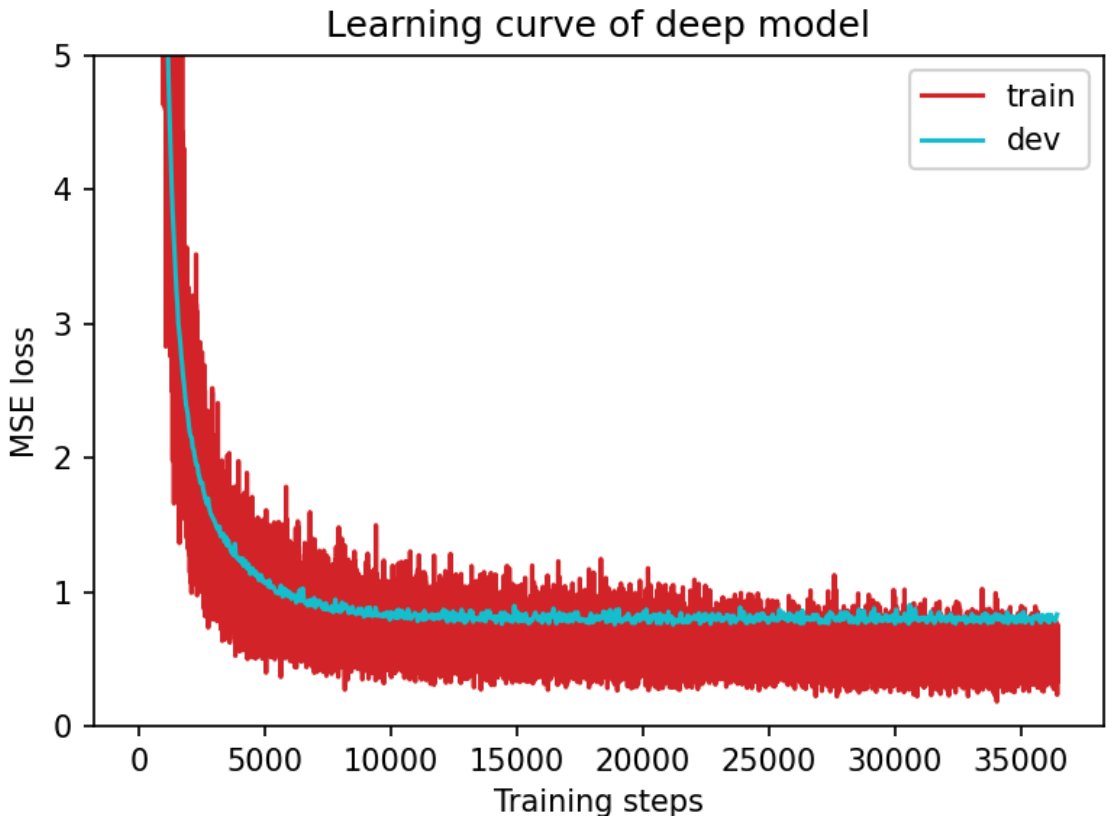
最后优化损失函数，采用正则化。

```
defcal_loss(self, pred, target,l2_lambda=1e-5):
''' Calculate loss '''
    #TODO: you may implement L2 regularization here
#return self.criterion(pred, target)
```

```
mse_loss = self.criterion(pred, target)
# L2正则化：对所有参数的平方和求和
l2_reg = 0.0
forparaminself.parameters():
        l2_reg += torch.norm(param, 2)#计算L2范数
total_loss = mse_loss + l2_lambda * l2_reg#合并损失
returntotal_loss
```

```
config = {
'n_epochs': 4000,#最大训练轮次
'batch_size': 64,#批次大小
'optimizer':'Adam',#优化器类型
'optim_hparas': {#优化器超参数
'lr': 5e-4,#学习率
#'momentum': 0.9             # SGD动量
'weight_decay': 1e-6#优化器自带的权重衰减（辅助正则化）
},
'early_stop': 400,#早停阈值（200轮无改善则停止）
'save_path':'models/model.pth'#模型保存路径
}
```

```
Saving model (epoch =  323, loss = 0.7776)
Saving model (epoch =  341, loss = 0.7699)
Saving model (epoch =  380, loss = 0.7612)
Saving model (epoch =  498, loss = 0.7573)
Saving model (epoch =  559, loss = 0.7562)
Finished training after 960 epochs
```



可以看到最后的结果稍微可以。正则化也是为了让模型在新数据上表现更好，防止过拟合。其最终表现虽不如最佳的loss为0.71左右，但是相较于之前的0.76也是有了一定的提升。