第三周周报

本周完成了模型的搭建，并在现有的数据集上进行了测试。

第三周的前半部分主要还是对自行搭建的网络debug, 希望网络能够跑通。与以往直接用别人现成的模型的经历不同，手工搭建网络还是会遇到很多问题：比如说遇到了参数始终不更新，损失函数在不同的`batch-size`下的值相差极大，后来发现这些问题都不是网络自身的问题，而是搭建网络的过程中出现了错误，在解决了这些问题后，网络跑通了。

第三周的后半部分是改进网络，并且在现有的数据集上测试了网络的性能。

1. 标准化步骤放入网络

上一周的构想是像以前模型的一样，把标准化步骤放在数据预处理模块，而不是放在网络中但是计算出来的结果表示这样处理还是存在着极大的问题的。最直观的问题就是，标准化之后的数据放入网络中计算出的损失函数没有现实意义，跟原论文中的误差直接用角度表示存在差异，也就没有可比性。因此把标准化的步骤又加入到网络。因为论文中的标准化的步骤没有现成可以使用的框架，所以这一部分的处理是自写的。

第一次写标准化的处理中没有考虑到效率的影响，发现模型运行极慢（400s一个epoch），后来改进了这部分的函数，运行速度有了较大的提升(10s一个epoch)：

标准化网络：

```
    def norm_diff_test(self, x):
        input_temp1 = (x[:, :, 0: 19]).clone().detach().cuda()
        #input_temp1 = torch.tensor(x[:, :, 0: 19]).cuda()
        input_temp2 = (x[:, :, 1: 20]).clone().detach().cuda()
        #input_temp2 = torch.tensor(x[:, :, 1: 20]).cuda()
        input_temp2 = input_temp2 - input_temp1
        x_max = torch.max(torch.abs(input_temp2), 2)[0].cuda() + 1e-8
        x_max = torch.repeat_interleave(x_max.unsqueeze(dim=2), repeats=19,
dim=2).cuda()
        input_temp2 = input_temp2 / x_max

        return input_temp2
```

逆标准化网络：

```
    def inv_norm_diff_test(self, x, out):
        out_inv = out.view(len(out), 3, 10).clone().detach().cuda()
        x_max = torch.max(torch.abs(out_inv), 2)[0].cuda()
        temp1 = torch.tensor(x_max * out_inv[:, :, 0],
dtype=torch.float32).cuda()
        x = torch.tensor(x, dtype=torch.float32).cuda()
        out_inv[:, :, 0] = x[:, :, -1] + temp1
        for i in range(9):
            out_inv[:, :, i + 1] = out_inv[:, :, i] + x_max * out_inv[:, :, i +
1]
        out_fin = out_inv.view(len(out), 30).clone().detach().cuda()

        return out_fin
```

发现在写逆标准化网络的过程中，其实有一些实现细节是在论文中没有被提到的：

$$\vec{\xi}_{t}^{pred} = \vec{\xi}_{t} + \sum_{k=t-\tau/\Delta t}^{t} \hat{\vec{\xi}}_{diff,\,t}^{pred} \cdot \max(|\vec{\xi}_{diff}|).$$

在做逆标准化的过程时，我们的公式如下：

$$\bar{\xi}_{t+0.1s*n}^{pred} = \bar{\xi}_{t} + \Sigma_{i=1}^{n}(\bar{\xi}_{diff,t}^{t} + i * 0.1s) * max(|\bar{\xi}_{diff}|)\ n = 1, 2, \cdots 10$$

2. FNN网络

之前的FNN网络只有一层，激活函数是`Relu`函数，发现效果很差（预测值几乎全为0），后来有加了一层网络（或者去掉Relu）效果变得正常。

3. 预测角度

利用现在的数据量在训练时发现预测结果与真实结果**相差`8-9`度**，由于数据量较少，可能存在着过拟合的情况，当然也和网络本身现在存在的缺陷存在着较大的关系。

现在正在使用的网络代码：

https://github.com/Zeng-WH/Head-Motion-Prediction/tree/main/%E7%AC%AC3%E5%91%A8/%E6%A0%87%E5%87%86%E5%8C%96%E7%BD%91%E7%BB%9C%E7%9F%A9%E9%98%B5

4. 关于数据

公司要回了之前的成品耳机，目前将较为简陋的半成品耳机给我们测数据。更换耳机前后，相同场景下测出的数据是有差别的（比如说我们头看着正前方，原来那个耳机测出来的角度可能是一百多度，现在的耳机是二百多度快三百度），公司的人解释说这是硬件差异，可能由于陀螺仪装的姿态位置不同。由于我们的数据有标准化的步骤，所以影响应该不大。

```
84
$EAINS,22,3111,862,-224,-669496,-148014,417313,596418*9FC8
$EAINS,23,3111,862,-224,-669367,-147803,416974,596852*26B3
$EAINS,24,3112,862,-224,-669330,-147444,416655,597206*242E
$EAINS,25,3113,862,-224,-669404,-146982,416377,597431*0B28
$EAINS,26,3113,862,-224,-669318,-146711,416169,597738*E693
$EAINS,27,3113,861,-224,-669131,-146632,415927,598135*03C5
$EAINS,28,3113,861,-223,-668876,-146553,415716,598587*5BD5
$EAINS,29,3114,861,-223,-668724,-146168,415631,598909*3F3A
$EAINS,30,3114,860,-224,-668624,-145679,415626,599144*988D
$EAINS,31,3114,860,-224,-668618,-145281,415655,599227*62C0
$EAINS,32,3114,860,-224,-668456,-145472,415559,599428*7171
$EAINS,33,3114,860,-223,-668476,-145738,415353,599483*F7BB
$EAINS,34,3114,860,-223,-668620,-146223,414937,599493*4AAD
$EAINS,35,3115,861,-222,-668965,-146670,414427,599351*C041
```

5. 下周安排

打算参考一下其他使用过`CNN+GRU`网络的论文，看看他们具体是怎么设置参数和搭建网络的，尽管应用场景可能不同，但希望能够得到一定的启发。