MLP_model 设计

由 danielnzhou(周亚楠)创建, 最后修改于大约5小时以前

1. 设计背景

在原本的数据中，只用到本局的数据，但是没有用到游戏的历史数据，可能会有部分信息提取不到。

所以采用 frame_sep 的方法，对每一个样本提取之前的历史信息，详见：采用 frame_sep 提取历史信息。

目前 ~~frame_sep = [0, 4, 10, 18, 30]~~ ，更新为 frame_sep = [0, 15, 30]

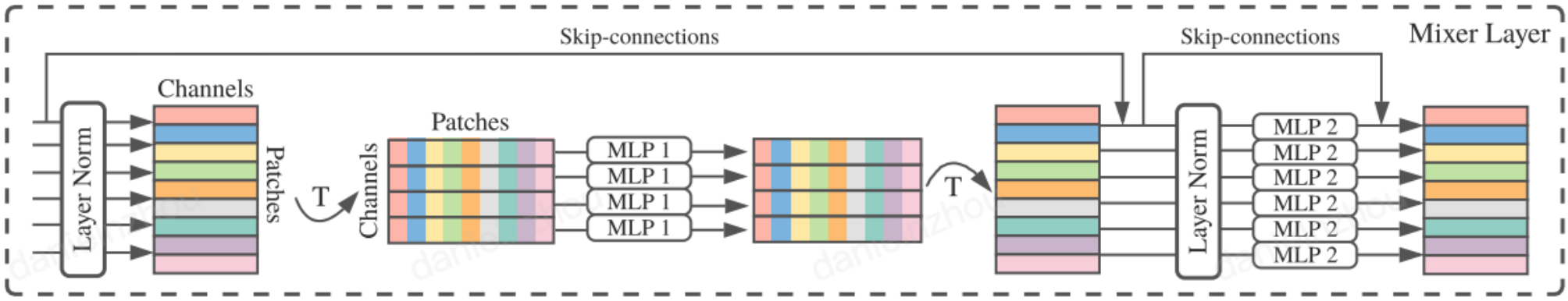
原本计划采用 moving_average 的方法进行加权，给予当前帧，即 sep=0 的时候较大的权重，离当前帧越远的帧权重越低。

现在采用可学习的加权方法，让神经网络自己学习之前的权重。

2. 网络设计思路

因为现在的数据分为两个维度，即时间历史维度（用H表示）和变量维度（用W表示）

根据 MLP-Mixer 和 separate CNN 的思路，需要学习到两个维度上的信息，所以在设计网络的时候添加了横向和纵向两个 MLP 全连接层；



在 width 维度上，宽度会从 209 变为 100 变为 50 变为 1，一共有两个隐藏层，可能会有有些深，容易过拟合。

在 history 维度上，采用 Transformer 中 Feed-Forward Network 的方法，也就是 ResNet 的思路，希望网络能够学到历史信息中的变化趋势；

所以在 history 维度上，数据经过 linear_t 从 H 变换成 1 个维度，在和之前的数据拼合在一起，所以是 (H + 1)个维度

使用了两层 BatchNorm，避免梯度消失，感觉可以不用加；【经过查看变量，batchnorm 层的参数好像都是 1 】

Dropout 使用了 0.2，减少过拟合；

激活函数 使用 relu；

3. 网络代码

```
class ClassifyModelDim3_v4(nn.Module):
    def __init__(self, input_dim):    # input_dim = (3, 209)
        super().__init__()

        self.H = input_dim[0]         # Hist Data 现在用 3
        self.W = input_dim[1]         # Width 目前是 209
        self.W_2 = (self.H+1)*100     # W_2 = 400
        self.linear_t = nn.Linear(self.H, 1)
        self.linear1 = nn.Linear(self.W, 100)
        self.linear2 = nn.Linear(self.W_2, 100)
        self.linear3 = nn.Linear(100, 50)
        self.linear4 = nn.Linear(50, 1)

        self.sigmoid = nn.Sigmoid()
        self.dropout = nn.Dropout(0.2)

        self.bn2_1 = nn.BatchNorm1d(self.W_2)
        self.bn2_2 = nn.BatchNorm1d(100)
        self.bn2_3 = nn.BatchNorm1d(50)

    def forward(self, x):

        x_t = x.permute(0, 2, 1)      # (N, H, 209) - (N, 209, H)
        x_t = self.linear_t(x_t)      # (N, 209, H) - (N, 209, 1)
        x_t = self.dropout(x_t)
        x_t = x_t.permute(0, 2, 1)
        x = torch.cat((x, x_t), 1)    # (N, H, 209) - (N, H, 209)
        x = self.linear1(x)           # (N, 4+1, 209) - (N, H+1, 100)
        x = torch.relu(x)
        x = self.dropout(x)
        x = x.view(x.shape[0], -1)    # (N, H+1, 100) - (N, (H+1)*100)
        x = self.bn2_1(x)
        x = self.linear2(x)           # (N, (H+1)*100) - (N, 100)
```

```
        x = torch.relu(x)
        x = self.dropout(x)
        x = self.bn2_2(x)
        x = self.linear3(x)           # (N, 100) - (N, 50)
        x = torch.relu(x)
        x = self.dropout(x)
        x = self.linear4(x)           #  (N, 50) - (N, 1)
        out = self.sigmoid(x)
        return out
```

现在是先由 209 - 100 然后在进行残差连接

我在想，是不是要先进行维度拼接，然后再进行 209 -100 维度变化比较好

这个可以探索一下