初始输入时：



输入：x

N = 2 \* batch\_size, 一帧里面有两个人；

C = 3， channel = 3;

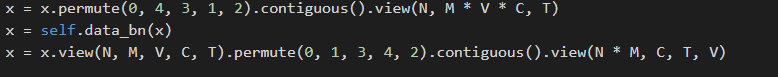
T = 150, #time\_frame = 150;

V = 25, #vertex = 25;

M = 60, #action\_class = 60;

x.size = (2, 3, 150, 25, 60)

维度变换和bn：

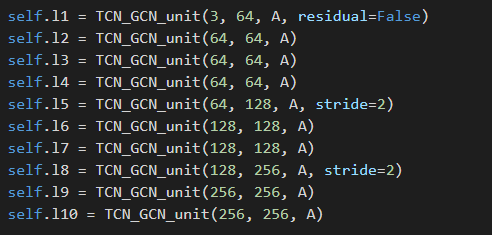


x.size = (2, 60, 25, 3, 150) 合并成 (2, 4500, 150)

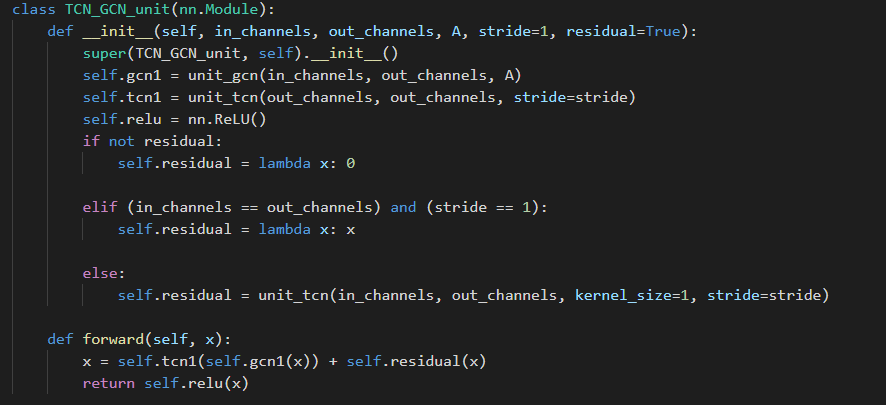
data\_bn = nn.BatchNorm1d (N \* C \* V) 即data\_bn = nn.BatchNorm1d (150) 沿着第2维的方向做bn

再次展开：x.size = (2, 60, 25, 3, 150) -> x.size() = (2, 60, 3, 150, 25) -> x.size = (120, 3, 150, 25)

GCN结构：



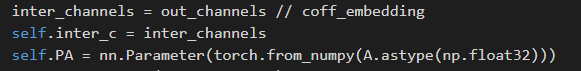
L1层运算：



无残差模块，stride = 1, in\_channel = 3, out\_channel = 64

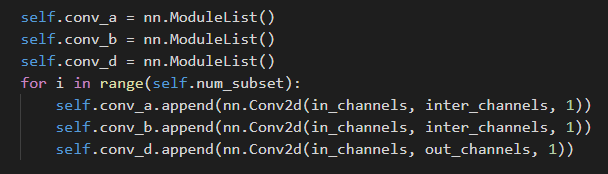
L1.gcn: in\_channel = 3, out\_channel = 64,

input\_x.size = (120, 3, 150, 25)



Inter\_c = 64 // 4 = 16

PA = B, 对应原论文中f\_out = w \* f\_in \* (A + B + C), A为原始邻接矩阵



Conv\_a: conv2d(in\_c = 3, out\_c = 16, kernel = 1, stride = 1), 连续三次

Conv\_b: conv2d(in\_c = 3, out\_c = 16, kernel = 1, stride = 1), 连续三次

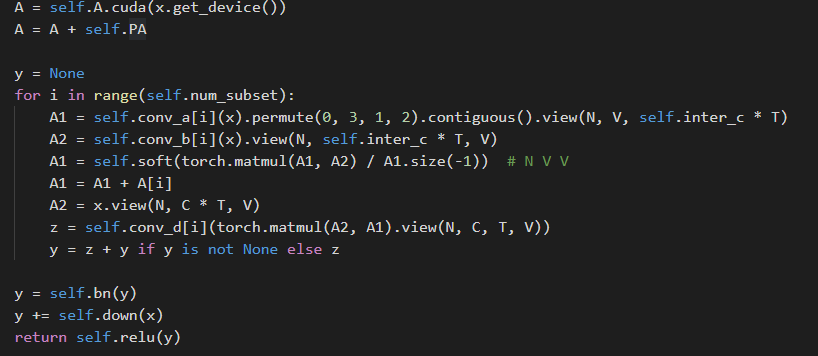
Conv\_d: conv2d(in\_c = 3, out\_c = 64, kernel = 1, stride = 1), 连续三次

Conv\_a的卷积参数size：(1, 3, 1, 1) \* 16 filter \* 3 loops

Conv\_b的卷积参数size：(1, 3, 1, 1) \* 16 filter \* 3 loops

Conv\_c的卷积参数size：(1, 3, 1, 1) \* 64 filter \* 3 loops

Gcn\_forward:



A = A + self.PA: 执行A+ B

Conv\_a[0](x).size = (120, 16, 150, 25) -> (120, 25, 16, 150) -> 60 x (2, 25, 2400) ???

卷积核尺寸：(1, 3, 1, 1), 16个filter, 120\*25\*150\*16 = 7.2M次向量乘加, 21.6M乘&14.4M加, 向量尺寸1x3

Conv\_b[0](x).size = (120, 16, 150, 25) -> (120, 25, 16, 150) -> 60 x (2, 2400, 25) ???

卷积核尺寸：(1, 3, 1, 1), 16个filter, 120\*25\*150\*16 = 7.2M次向量乘加, 21.6M乘& 14.4M加, 向量尺寸1x3

Matmul(A1, A2): 4次 (25, 2400) \* (2400, 25) 的矩阵乘法，2500次向量乘加，6M乘&6M加，向量尺寸1x2400，输出尺寸(4, 25, 25) ???

A1 = Softmax(Matmul(A1, A2)): 得到C矩阵

A1 = A1 +A[i]: 执行A + B + C的操作

A2.size() = 60 x (2, 450, 25) ???