流式HybridCNN Batch builder逻辑

1. BB下发的数据

* Batch = 8
* Frame = 64
* Attn\_history\_len = 16
* Attn\_cur\_len = 2
* LowFrame = 4
* conv\_cache\_Len = 2
* multi\_head

|  |  |  |
| --- | --- | --- |
| 数据key | 数据shape | 备注 |
| data | (batch,1,frame+2,Dim) | 原始数据 |
| Convolution\_%i\_input \* 69 | (batch,channel, Dim, con, conv\_cache\_Len) |  |
| Attention\_%i\_q\_input \* 7 | (batch\* multi\_head, Attn\_cur\_len, Dim) |  |
| Attention\_%i\_k\_input \* 7 | (batch\* multi\_head, Attn\_history\_len + Attn\_cur\_len, Dim) |  |
| Attention\_%i\_v\_input \* 7 | (batch\* multi\_head, Attn\_history\_len + Attn\_cur\_len, Dim) |  |
| AttRes\_%i\_input \* 7 | (batch, channel, 1, 2) |  |
| NormMask | (batch,1,frame+2,1) | 数据归一化后Mask |
| SrcMask | (batch, 1, frame, 1) | 原始数据Mask |
| AttnMask | (batch, attn\_history, attn\_history + Frame/ LowFrame) |  |
| AttnConvMask | (batch, 1, Attn\_history\_len) | Attention层卷积mask数据 |
| ConvOutMask | (batch, 1, Frame/ LowFrame) |  |

1. 数据下发的逻辑

2.1 Data

* 每个batch下发需要历史的两帧数据，无未来视野窗
* 首个batch和尾部batch数据需要padding 16帧，但是不作为最后的输出数据
* 尾部batch有效数据长度需要对齐到8帧（然后padding16帧），整体数据补齐到窗长
* 最后额外加一个无效的数据窗，目的是为了让前面未输出的数据帧看到历史，然后进行输出，否则会出现掉帧的问题
  1. NormMask
* Padding数据都需要进行NormMask，将数据进行清零，因为NormLize操作，会将padding的数据转化到非零数据
* 常规有效数据不需要进行NormMask操作
  1. SrcMask
* SrcMask数据不一定需要，原来考虑的是和AttMask共同使用，然后简化AttMask的操作逻辑，暂时先搁置
  1. AttMask
* 首尾batch计算需要考虑无效数据，是否可以考虑通过另外一个Mask的叠加操作，使得简化AttnMask的操作

1. Attn\_mask针对窗的mask可以采用完全固定的表格来设置，并且根据参数的参数可以简单的完成设置
2. SrcMask来标记新来的数据的有效性，并且和历史数据的有效性，形成当前数据维度上的
   1. convMask
   2. convOutMask
3. GraphContext设置

* 由于当前窗的数据，作为上个窗的数据的未来视野，推动上个窗的数据进行输出，因此每次的GraphContext信息需要携带的是上个batch的数据信息，和本次batch的第一个16帧数据的信息（本次可以窗可以输出两个有效帧）