下面是对hybridCNN的开发技术方案和现有流式VGG的技术方案进行评估

1. hybridCNN模型 dfmlp技术方案

|  |  |
| --- | --- |
| 工作内容 | * 根据新模型，**对计算窗的大小逻辑进行修改** * 新模型operator开发，若需要 * 根据新模型，计算json图改动 * **模型一致性对比工作** |
| 风险 | * 数据流调度层BB的开发需要内核部同事支持 |
| 成本 | * 如果模型改动不大，5人日 * 如果模型改动稍大，8人日 |

2. hybridCNN模型 MaxEngine-Lite技术方案

|  |  |
| --- | --- |
| 工作内容 | * **MaxEngine-Lite中开发实现BatchBuilder** * 新模型operator开发，若需要 * 根据新模型，计算json图改动 * **模型一致性工作对比** |
| 风险 | * 移植dfmlp的BB逻辑，在MaxEngine-Lite中开发实现同样功能的BatchBuilder，难度不大，但可能需要一定的调试时间 |
| 成本 | * 如果模型改动不大，8人日 * 如果模型改动稍大，10人日 |

3.基于流式VGG-Trans模型的技术方案

|  |  |
| --- | --- |
| 工作内容 | * MaxEngine-Lite中增加encoder json端后验输出功能 * MaxEngine-Lite对模型初始化逻辑进行小幅修改 |
| 风险 | * encoder端效率不确定，暂时没有测试数据 * 方案没有hybridCNN模型成熟 |
| 成本 | * 3人日 |

* 方案2相比方案1的唯一区别，在MaxEngine-Lite中新增BatchBuilder的工作量
* 方案3基本没有工作量，可以直接使用VGG流式引擎，稍作修改即可，但是模型效果和效率具有不确定，需要验证
* 方案2的工作量稍大，但是可接受，而且方便后期统一维护，包括VGG、RNNT、hybridCNN模型，都可统一使用lite，且持续有维护和更新，且可自行进行小功能更新和开发

综上所述，建议采用方案2

设计评审

1. 工作点

* HybridCNN BatchBuilder的功能
* MaxEngine-Lite引擎中beam search的适配工作
* EDGen引擎Beam search功能适配
* EDGen引擎初始化工作
* HybridCNN资源打包与加载
* HybirdCNN一致性对比工作

1. 工作内容

|  |  |
| --- | --- |
| 工作点 | 具体内容 |
| HybridCNN BatchBuilder的功能 | 1. MaxEngine-Lite引擎VGGEncoder Batch Builder代码阅读 2. MaxEngine-Lite引擎RNN-T encoder Batch builder代码阅读 3. 根据上述两个Batch Builder的开发逻辑，进行改造来适配hybridCNN的BB |
| MaxEngine-Lite引擎中beam search的适配工作 | 1. 阅读当前ED架构下的beam search设计逻辑，尤其是关于信号量的处理逻辑 2. 考虑如何在Batch builder内部加信号量，然后在beam search函数利用信号量进行数据同步 3. 数据同步的一点难点在于   每帧数据是不是需要进行单独的信号量设置  信号量提前发送，会不会导致信号量丢失，引起线程阻塞 |
| EDGen引擎Beam search功能适配 |  |
|  |  |
|  |  |
|  |  |
|  |  |