你真的懂deepseek v2？懂MLA？

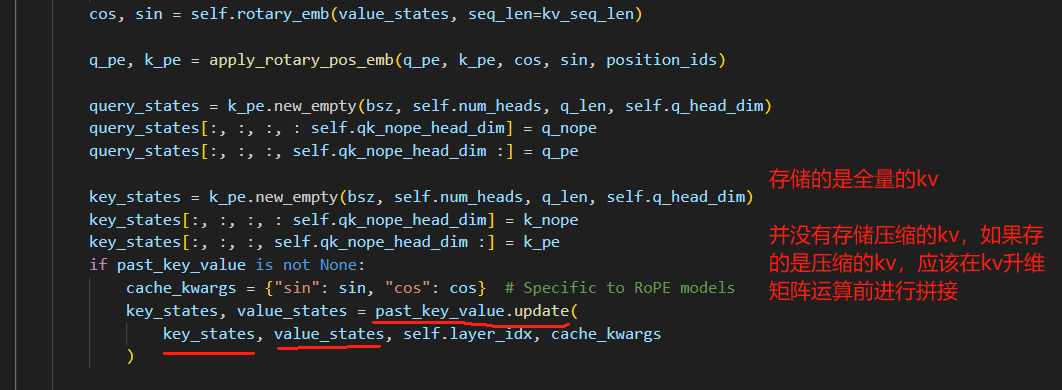
先说结论：

1. 236b的模型，在hf形式下的model.py和vllm的实现中，模型的kv cache并没有节省，反而大大增加。
2. 最终的MLA ≈ MHA + MQA （约等于的意思一会解释QAQ）
3. 不要喷我！伟大的深度求索肯定自己内部的实现是“真正的MLA”，有自己的推理框架。我只是分享下开源的方法

解释：

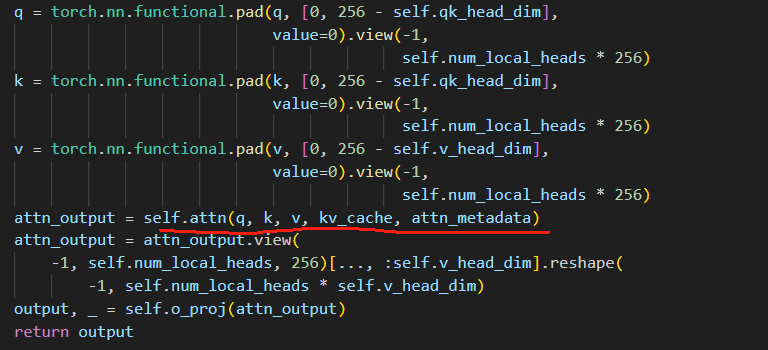
1. 先说为什么没有压缩kv cache。

Hf的代码



Vllm的代码

Vllm中封装的比较深，attn一般用flash-attn，kv cache的更新和attn的主要计算都封装在flash-attn的c++代码中。但也可以看出，kv cache存的也是全量kv



存储全量kv cache为什么大大增大了kv cache呢？

下面是236b模型中的config

"hidden\_size": 5120,

"v\_head\_dim": 128,

"qk\_nope\_head\_dim": 128,

"qk\_rope\_head\_dim": 64,

"num\_attention\_heads": 128

k的head\_dim是128 + 64 = 192

kv都是128个头，k和v的dim变为了128 \* 192 = 24576和 128 \* 128 = 16384

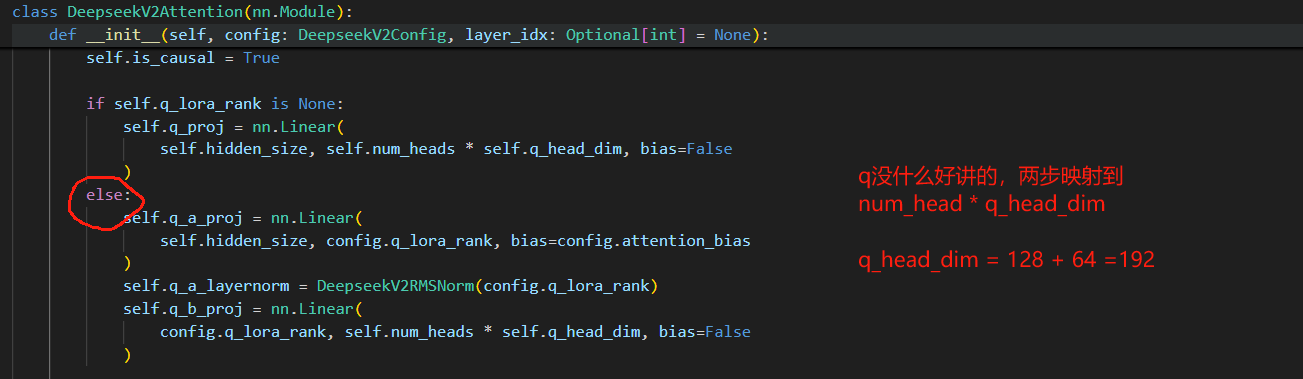
（一会会具体讲MLA参数设置）

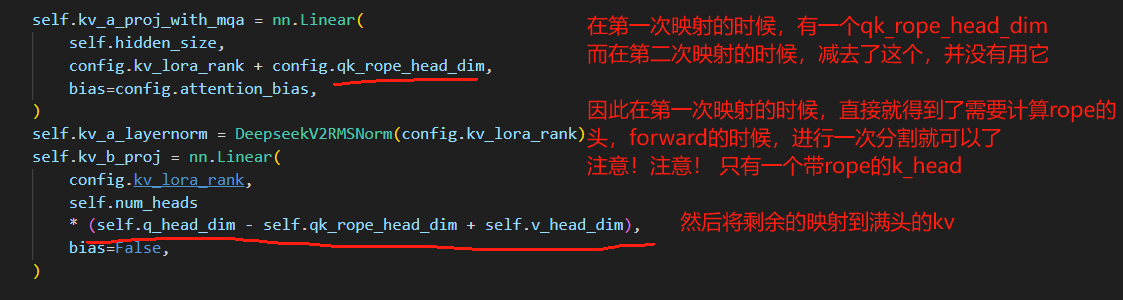
比如llama 8b，hidden\_size是4096，用GQA的话（num\_head=32, kv\_head=8）,

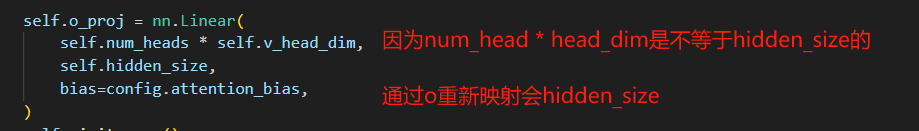
kv的dim都是4096 / 32 \* 8 = 1024

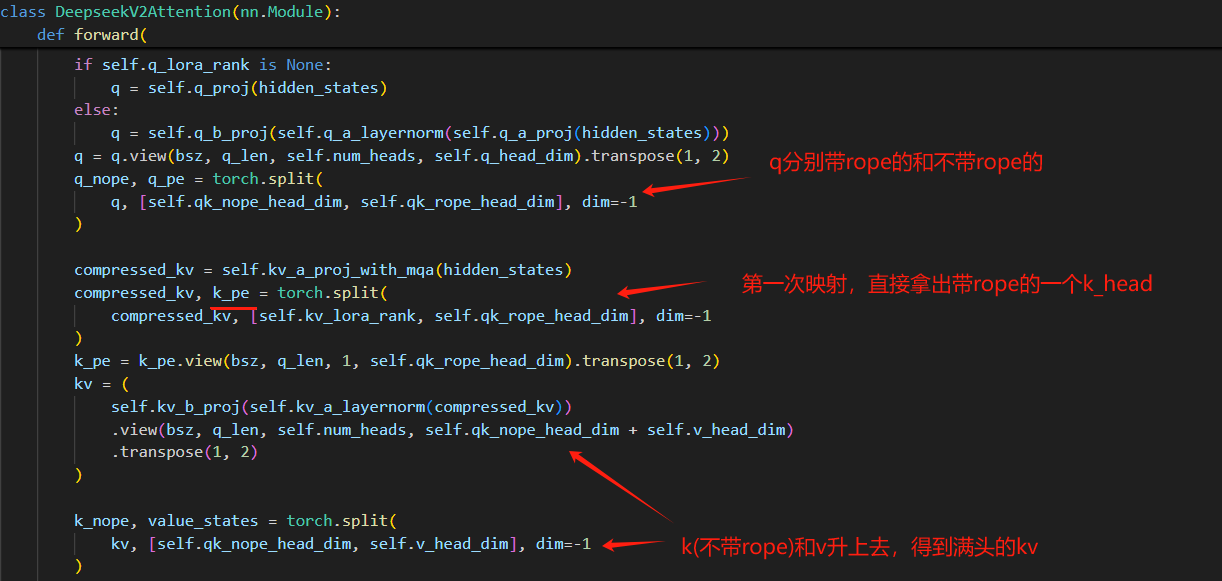
两者的hidden\_size差不多吧(5000和4000，参数那么多是因为ep\_size=160)，但是单个token的kv cache花费量大大增加

1. MLA ≈ MHA + MQA









因此最终q和k相乘的时候，前128维是无rope的MHA，后64维做的是MQA，约等于是因为这毕竟是先拼接再乘，而不是分开乘在拼接（勉强约等于吧，误喷）

