Lab5

3240104875
王耀
2025/8/18

基础任务

我对位置编码的作用有些疑惑：embedding 输出矩阵按照行数已经可以视为位置编码，为什么还需要位置编码？

似乎是因为矩阵运算结果无法直接体现位置信息？毕竟计算机不会逐行分析，所以要通过位置编码来调节矩阵数据？毕竟只是在矩阵乘法，根据行数似乎有些复杂，直接加点位置偏移向量确实更便于运算。

但是 QGA 这里，我直接调用 self.k_proj 不会直接生成许多 KEY 吗？嗯，似乎上面传递给类的参数已经准备好了是可以 QGA 的，毕竟写的是 self.num_key_value_heads * self.head_dim

但是我越写越迷糊，我感觉这么设计 attention 运算的似乎是多个 token 的部分混杂在一起和对应 K 的点积，似乎有些混乱。

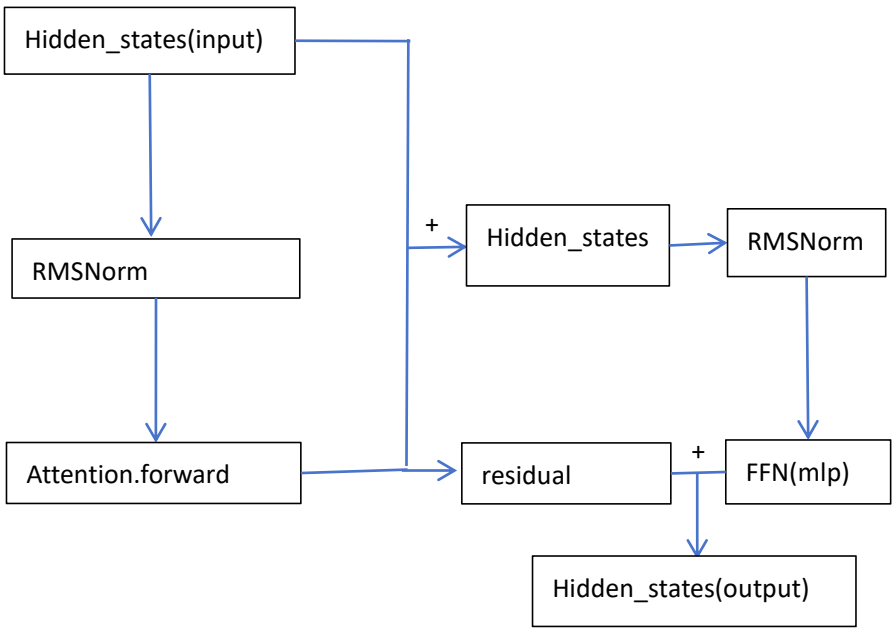
哦。我对多头注意力的机制理解有误，多头注意力本身就是对一个“token”的分段多头计算（对其投影 Q,K），但是这么做的好处是什么呢？

似乎并不是分段计算，只是选用不同的 QKV 投影矩阵？嗯，然后这些矩阵会把原始的输入向量投影到子空间，维度更小，所以可以放在矩阵内被 head_num 个注意力头分割计算，提取特征。

但是为什么我的模型的输出到一半突然停止了？哦。是到了最大输出限制。

不过这个激活函数的作用是什么？仅仅是类似线性的，但是降低<0 的数据的影响吗？似乎能够起到给模型分流信息的作用，重要的就压成正的，不重要的就压成负的。

思考题



Layer.py 的结构如上。初始输入为[batch_size, seq_len, hidden_size]，经过 RMSNorm 后形状不变，经过 attention 计算仍然不变（hidden_size 最后一维是注意力头数量的倍数），

经过 mlp 先变成 batch_size*seq_len*intermediate_size，再返回 hidden_size，故而最终仍然是[batch_size, seq_len, hidden_size]

现在对显存进行预估计算。按照思考题的提示进行计算，那么显存占用应该是 152064*4096(embedding) + 4096*4096(q_proj) + 4096*32*4096/32*2(k_proj+v_proj) + 4096*4096(o_proj) + 4096*11008*3(gate, up, down) + 4096*152064 = 1,431,306,240。思考题的知道说 RMSNorm 也有可训练的参数，难道是偏置向量？如果加入这一部分，那么就是 1431306240 + 4096*2(attention) + 4096*2(decode layer) = 1,431,322,624，最后*2，2,862,645,248

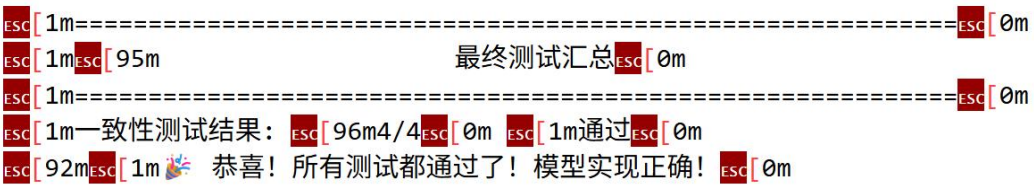
我在 decode layer 类里面调用 parameters 试试看。但是它的元素数量之和输出是 8,190,735,360，这还是没有计算字节数的。不，我是在 model 文件内调用的 parameter，所以要考虑到循环设置的字典，内部有 32 个 att 和 mlp，所以应该是 152064*4096(embedding) + (4096*4096(q_proj) + 4096*32*4096/32*2(k_proj+v_proj) + 4096*4096(o_proj) + 4096*11008*3(gate, up, down) + 4096*4)*32 + 4096*152064 + 4096 = 7,722,240,406。还是有偏差。

经过我的 print，我发现 attention 里面的 self.num_key_value_heads 不是我想象的 32，而是 8。我以为他是和 config 文件一样的 32。原来是 json 文件重置了这个参数。同样的问题也出现在 intermediate 等等参数上面，不再一一列举。而且 norm 计算也有问题，Q 和 K 那里，因为张量维度的置为 4 维，所以 norm 的参数也是计算错误的。

那么重新计算，151936*4096(embedding) + (4096*4096(q_proj) + 4096*8*4096/32*2(k_proj+v_proj) + 4096*4096(o_proj) + 4096*12288*3(gate, up, down) + 128*2 + 4096*2)*36 + 4096*151936 + 4096(output norm) = 8,190,735,360

所以参数占用显存为 16,381,470,720 字节，即 15.26G，略小于我的电脑的内存。

最后将模型正确性证明截图放在下面：



我问了几个其他的问题，这个小模型看起来确实傻傻的。