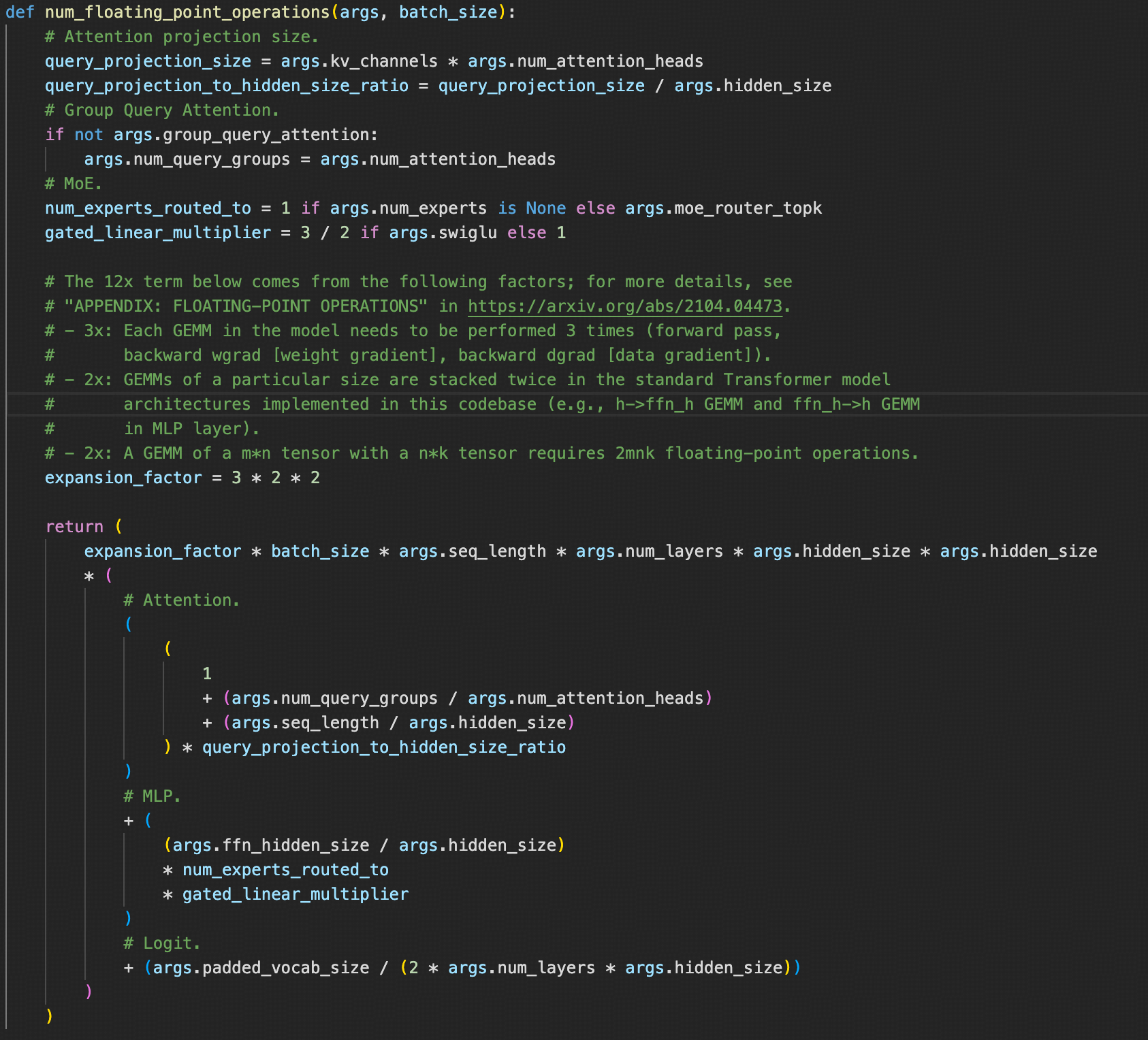
Megatron计算浮点计算量的公式



基础前提：

1. 矩阵A(m,n)与矩阵B(n,k)相乘，矩阵中每个元素都是需要经过n次乘法和n-1次加法，因此计算量是2\*m\*k\*n

2. 一次反向传播的计算量大约是正向的两倍，因为不仅要算权重的梯度，还要算一些中间激活值的梯度

3. 假设矩阵Wq是D\*D的，D时hidden\_size

4. 复杂度主要体现在矩阵乘法上面，例如norm操作，或者softmax操作都是元素级的操作，会比矩阵乘法少一个量级，因此忽略不计

符号

bs: batch\_size大小

N：模型层数

L：句子长度

D：隐藏层维度hidden\_size

r1：num\_key\_head / num\_head

r2: ffn\_hidden\_size / hidden\_size

gate: 使用gate 则为1.5，不使用则为1

vocab\_size：词表大小

K: 路由的专家个数（top\_k），默认是1

Attention复杂度（每层前向）

hidden\_state映射到Q，K，V分别是：

2LD2， 2r1LD2，2r1LD2

计算QK （k会被repeat回D维）

2L2D

计算softmax(QKT)V. （v会被repeat回D维）

2L2D

计算最终输出O=(softmax(QKT)V)Wo

2LD2

加一起，提出LD2

4LD2+4r1LD2+4L2D=4LD2(1 + r1 + L/D)

MLP(一层，前向)

计算hidden\_state -> up\_hidden\_state

2r2LD2

计算up\_hidden\_state -> down\_hidden\_state

2r2LD2

加一起是4r2LD2，如果使用gate的话，又要多个2r2LD2，总共就是6r2LD2

因此就是4r2LD2\*gate

如果是moe架构的话，每个token会过k个专家(注意不是每个token过所有专家)

那么就是4r2LD2\*gate\*k

提出4LD2，最后结果是：

4LD2(r2\*gate\*k)

计算输出层（映射到词表）

2LDV

先把attn(红色)和mlp(蓝色)一层的加一起

4LD2(1 + r1 + L/D + r2\*gate\*k)

乘上模型层数和batch\_size

4bs\*N\*L\*D2(1 + r1 + L/D + r2\*gate\*k)

因为输出层只有一层，将其也融进去需要把层数N也给提出来

4bs\*N\*L\*D2(1 + r1 + L/D + r2\*gate\*k + v/(2\*D\*N))

最后我们加上反向传播的，将上面结果乘3，就是每一步所需的计算量

12\*bs\*N\*L\*D2(1+r1+L/D+r2\*gate\*k+v/(2\*D\*N))