llama3---RMSNorm源码

首先定义一个RMSNorm类:

```
#写class一定要继承torch.nn.Module
class RMSNorm(torch.nn.Module):
    def __init__(self, dim: int, eps: float = 1e-6):#dim就是输入x的特征维度，限定一定要int类型
        super().__init__()#nn.Module初始化
        self.eps = eps #赋个很小的数，防止除以0
        #nn.Parameter的梯度更新默认是打开的，所以weight也会被加入可训练参数列表中（weight可学习）
        self.weight = nn.Parameter(torch.ones(dim))#参数初始化为长度为dim的全一的tensor

    #x是必须要传入的
    def _norm(self, x):
        #x.pow(2)对x中每个元素平方，mean(-1, keepdim=True)--> -1表示最后一维上求均值，keepdim保证维度不变。结果最后一维的值都相同为均值
        # + self.eps保证传入torch.rsqrt的值大于0，torch.rsqrt先根号再取倒数
        return x * torch.rsqrt(x.pow(2).mean(-1, keepdim=True) + self.eps)

    #在forward中调用self._norm
    def forward(self, x):
        """
        Forward pass through the RMSNorm layer.

        Args:
            x (torch.Tensor): The input tensor.

        Returns:
            torch.Tensor: The output tensor after applying RMSNorm.

        """
        #x先转为float，然后norm，然后再转回x的类型
        output = self._norm(x.float()).type_as(x)
        #乘一个可学习的参数进行scale
        return output * self.weight
```

RMSNorm的使用位置:

```
#将x进行RMSnorm之后再进行attention，然后进行残差连接
h = x + self.attention(
    self.attention_norm(x), start_pos, freqs_cis, mask
)


#ffn时也会进行RMSnorm归一化
out = h + self.feed_forward(self.ffn_norm(h))
return out


#最后的全连接层之前进行RMSnorm
h = self.norm(h)
```

```
output = self.output(h).float()
return output
```