**学习进度**

本周学习任务：

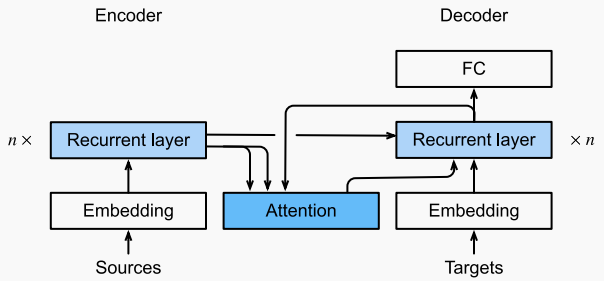
学习李沐老师《动手学深度学习（PyTorch版）》，关于**注意力机制**部分的内容。

**一、Bahdanau注意力**

1. 应知必会

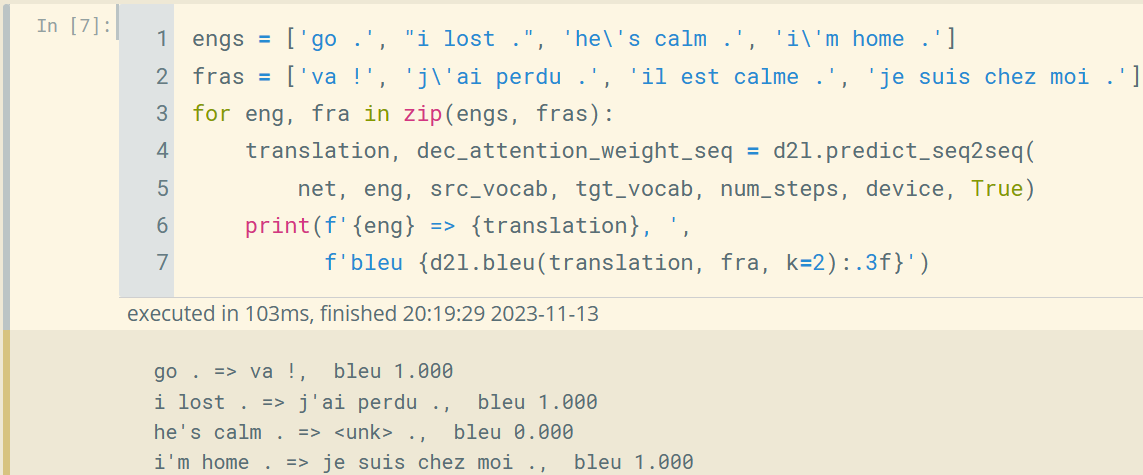
采用编码-解码器结构的Seq2Seq模型，在分时间步的解码过程中，每次都使用相同的编码信息，从而缺乏应对查询发生变化时的灵活性。故此，Bahdanau在分时间步的解码过程中，每次都基于编码器所有时间步生成的隐藏状态，以解码器前步隐藏状态为查询，重构出考虑注意力分配的专门上下文编码信息，作为当前时间步的输入。计算公式如下所示，查询表示解码器的前步隐藏状态，键、值都采用表示编码器全部历史隐藏状态的，指注意力评分函数。





值得注意的是，Bahdanau解码器使用编码器最终时间步输出的隐藏状态，初始化解码器的起始隐藏状态。此外，如果解码器存在多个隐藏层且编码器只有一个隐藏层，通常会将编码器的最终隐藏状态复制多次，用于初始化解码器中每一层的隐藏状态。

1. 实验结果【代码省略，详见：李沐《动手学深度学习（PyTorch版）》P406】



1. 相关思考

**（1）查询Q与键K的长度不同时，如何计算注意力权重矩阵？**

通过点积的方式计算词嵌入向量之间的相似程度，需保证查询与键的长度一致性，即：特征维数上的一致。当查询与键的长度不一致时，可考虑根据多层感知机（MLP）实现一个加性注意力评分类，计算公式为：。其中，可学习的参数是，指代隐藏层单元数。相较于点积注意力评分，加性注意力依赖可学习的权重矩阵，使注意力权重的计算摆脱了同输入特征维度的直接关联，从而可以更灵活地适应不同长度的输入。然而，由于加性注意力评分的计算复杂性，在大规模数据和模型上，缩放点积注意力评分的效率更高。

**（2）在注意力机制中，不定长输入序列的Softmax计算？**

在传统RNN网络的训练过程中，需要保证模型接收等长的输入序列，即：保证训练时间步的一致性。相较于直接等长切分全局文档，更常见的做法是：依据全局文档中句子的长度分布，人为确定输入序列长度；当句子长度小于输入序列长度时，可以在空缺位置上写入填充标记<pad>；当句子长度大于输入输入序列长度时，只能遗憾地进行截断处理。

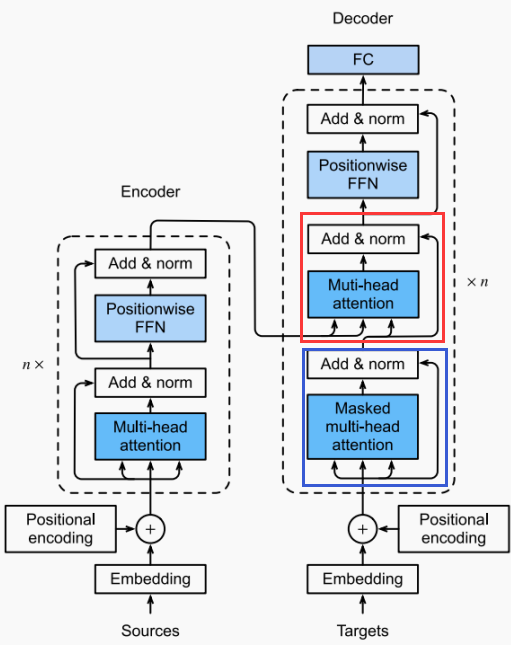
然而，对于有效序列长度不一的等长输入序列而言，在注意力机制softmax计算过程中，词嵌入向量末尾的填充标记<pad>将严重影响输出的概率分布。为此，可以进行经遮掩的Softmax操作（Masked Softmax Operation），依赖事先记忆的输入序列有效长度，将任何超出有效长度位置上的数值都遮蔽重置为0。

**二、Transformer**

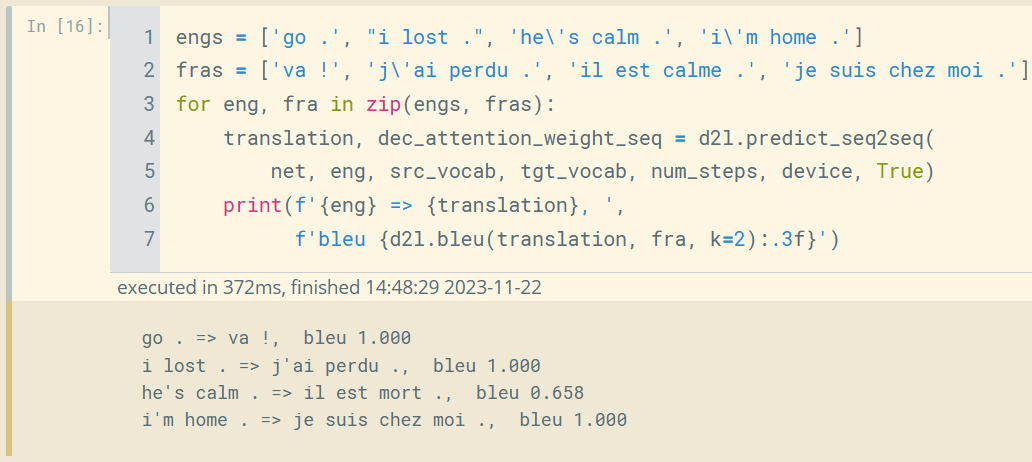
1. 应知必会

相较于前述依赖RNN实现输入重构的自注意力模型，Transformer属于依托编码-解码器结构的纯注意力机制模型，不存在任何卷积或循环神经网络层。在Transformer中，每个计算单元都额外引入了残差连接及层规范化，用于防止网络退化。此外，在Decoder的多头注意力单元中，分步接收经掩码多头注意力单元处理的历史信息作为查询Q，并接收Encoder最终输出的编码信息作为对应的键值K、V，完成并行的注意力分配与多头结果的拼接重塑。值得注意的是，循环神经网络依赖递归特性，天然能极好地处理序列分量的位置信息；对于Transformer而言，则需要通过引入位置编码的方式，使得序列分量额外考虑到位置信息。

总而言之，Encoder负责将输入序列映射为一个考虑全局上下文信息的抽象连续表示，即：Encoder最终输出的编码信息矩阵。Decoder则分步接收历史标签序列段或预测序列段作为查询，依赖编码信息矩阵作为键值，以逐步完成预测序列的生成。



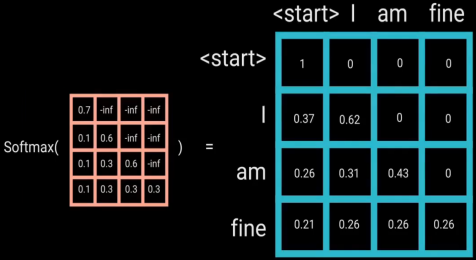
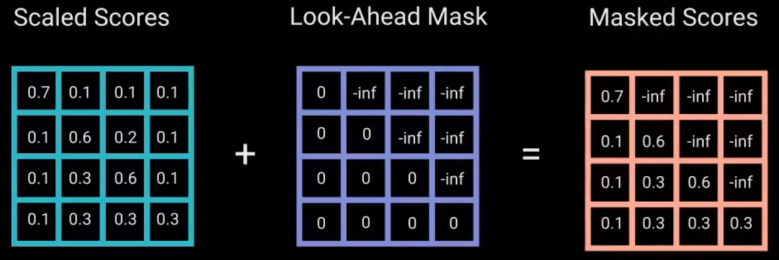
1. 实验结果【代码省略，详见：李沐《动手学深度学习（PyTorch版）》P422】



1. 相关思考
2. **如何理解Masked multi-head attention中的掩码操作？**

Decoder在分步生成预测词元概率分布的过程中，对于一次性输入的标签序列而言，Masked multi-head attention需要事先计算出标签序列词元间的注意力分数（Scaled Scores）。然而，在预测某个序列词元时，注意力不应分配到标签序列中后续位置上的未来信息。

为此，如下图左所示，需将Scaled Scores同Look-Ahead Mask矩阵相加，使Scaled Scores矩阵中右上三角部分的注意力分数取值更替为负无穷大，从而获得对应的掩码注意力分数（Masked Scores）。由此，如下图右所示，经过Softmax处理的Masked Scores，标签序列中的每个分词投射出去的注意力将被限制在历史标签词元集合中。



1. 层规范化还是批次规范化？

层规范化与批次规范化的计算方式相同：先减均值，再除标准差；但规范化的方向不同。对于层规范化而言，取的是同一个样本的不同通道做归一化，逐个样本归一化。对于批次规范化而言，取不同样本中同一个通道的特征做归一化，逐特征维度归一化。在NLP领域，相较于批规范化将不同输入序列中对应位置上的分词逐特征值进行缩放，层规范化则在一个输入序列中的词向量维度上进行缩放处理，这使层规范化能独立地归一化每个分词的特征，从而更适用于处理同一分词中不同词嵌入特征之间的差异。

1. **输入张量X在EncoderBlock中的维数变化**

在一个EncoderBlock中，事先需要将不等长输入序列填充或切割为长度等于seq\_length的等长序列（seq\_length的取值还需要考虑到标签序列的可能长度），故此可假设输入张量X的维数为（batch\_size, seq\_length,vocab\_len）。首先，输入张量X需经过词嵌入处理，将独热编码表示的序列分词映射到低维特征空间中，X的维数变化为（batch\_size, seq\_length,embed\_len）。其次，输入张量X中的输入序列需要添加位置编码，X的维数不变。然后，依赖多头自注意力机制，多尺度重塑X中输入序列不同分量的词嵌入特征值。再后，将引入注意力的重塑X同初始X按权值加和，并完成层规范化处理。最后，输入张量X经过前馈神经网络层并同样进行残差连接及层规范化处理，X转变为编码信息矩阵，且维数变化为（batch\_size, seq\_length, FF\_output\_num）。值得注意的是，由于前馈神经网络模块同样引入了残差连接，需要保证该层输出的FF\_output\_num维数等于embed\_len。故此，在EncoderBlock中，输入张量X自词嵌入层后，维数最终不再发生变化。

1. 如何理解Decoder的运作模式？

在一个DecoderBlock中，掩码多头自注意力处理标签张量X（训练时）或历史预测词元集合（测试时），以获得仅考虑历史注意力的查询张量。此后，在多头注意力单元中接收编码信息矩阵为键值，根据查询张量重塑变形编码信息。之后，通过前馈神经网络层后，还需经过一个全连接层，将新形态的信息张量在特征维数上从embed\_len扩散为vocab\_size，以获得预测词元的概率分布。最后，通过softmax操作获得概率最大的输出词元。

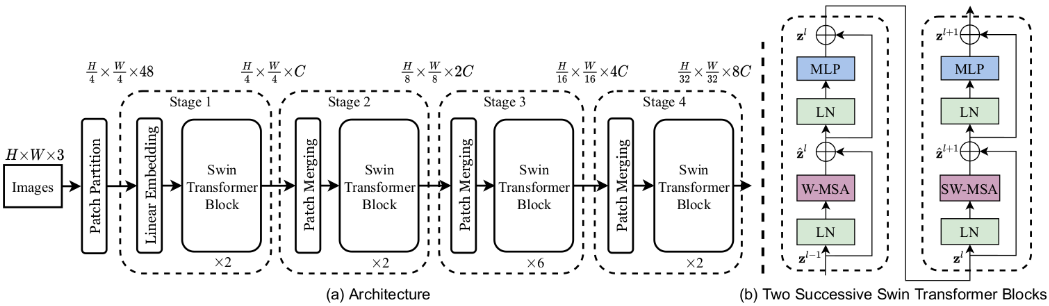
值得注意的是，由于标签序列是已知的，Decoder在训练阶段可以并行计算不同位置上的预测词元，然后根据预测词元与标签词元来计算损失，从而更新网络参数。而在测试时，只有获得了前续预测词元集合，才能继续生成下一个预测词元，只能是串行过程。

**三、Swin Transformer**

1. 基本认知

在CV领域，由于ViT不太擅长处理图像中视觉实例及背景的多变性，且ViT基于全局上下文的注意力机制尤其在处理高分辨率图像时计算效率较低。为此，Swin Transformer创新性地使用非卷积下采样法及分层模型架构，企图模仿CNN实现图像特征的逐层学习，从而提高Transformer解决视觉实例多变性问题的能力。此外，Swin Transformer通过结合窗口多头自注意力（W-MSA）及偏移窗口多头自注意力（SW-MSA）的方式，有效替代了基于全局上下文的多头自注意力机制（MSA），减少了注意力计算量。

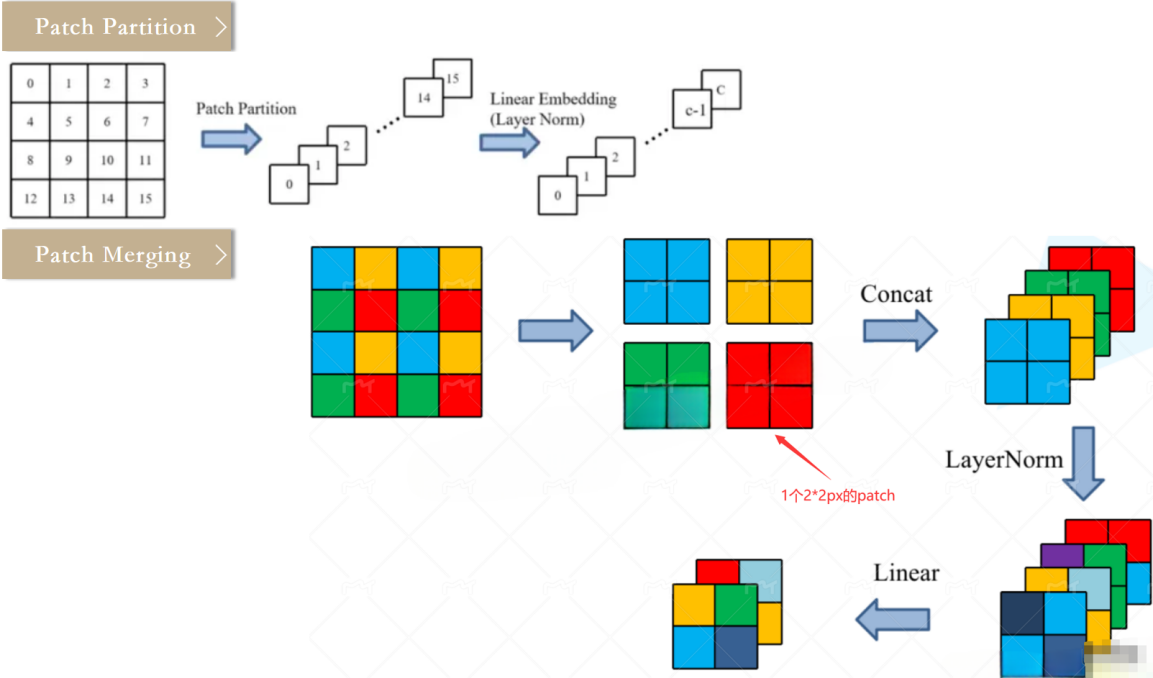
1. 模型架构



Swin Transformer的模型架构如上图所示，可划分为四个不同感受野级别的注意力阶段。

在Stage 1：首先，经过Patch Partition处理可将输入图像切割为指定像素尺寸的Patch，并将Patch们层叠在一起，使得分辨率降低为原来的四分之一且通道数提高了十六倍；然后，经过Linear Embedding层，将特征图中的像素点映射到词嵌入参数空间中，从而将特征图的特征通道维数从48映射为C；最后，经过两个连续的Swin Transformer Block（分别引用了W-MSA、SW-MSA），完成了当前感受野下的注意力计算。而随后的三个阶段都采用了非卷积下采样法Patch Merging及连续的2n个Swin Transformer Block，实现了摆脱卷积层的特征逐层学习及更具并行性的注意力计算。

显而易见的是，Patch Partition + Linear Embedding的实际效果等价于Patch Merging，区别在于：Patch Merging将特征图切分为若干个块后，会依次将每个块中相同位置的像素点拼接起来，从而形成新的Patch。Patch Partition则将特征图切分为若干个块后，直接将它们层叠在一起即可。此外，相较于基于卷积核的卷积下采样操作，非卷积下采样法Patch Merging只不过是分块层叠处理，除了需要线性层控制通道维数外，并不存在其它可学习参数。



1. 相关思考

**（1）为什么使用W-MSA或SW-MSA替代MSA，能够有效减少计算量？**

由于W-MSA与SW-MSA的计算量级基本一致，为简化推理过程，故此只需比较W-MSA与MSA的计算量级即可。

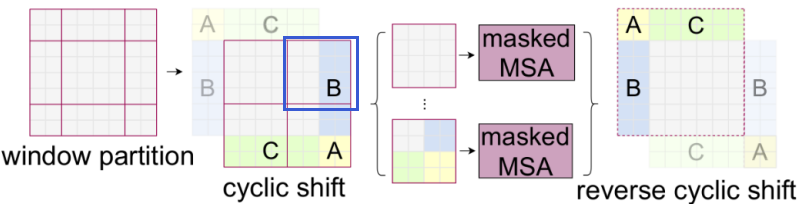
对于MSA而言，若假设Q、K、V的向量长度同特征图的通道维数一致，则Q的计算公式为：，表示将特征图所有像素展平拼接在一起得到的矩阵（共有h\*w个像素，每个像素的通道维数为C）。根据前述公式可知，生成查询Q的计算量为，生成键K及值V同理都是，生成Q、K、V所需的计算量为。

此外，若是缩放点积注意力评分，可忽略除以及softmax的计算量，注意力评分的计算公式为：，所需计算量为。之后，通过注意力评分与V相乘获得最终结果的公式为：，所需计算量为。综上所述，**单头自注意力机制的计算量为**。值得注意的是，多头注意力相比于单头注意力的计算量仅多了一个融合多头注意力结果的参数矩阵，故**MSA的计算量为**。

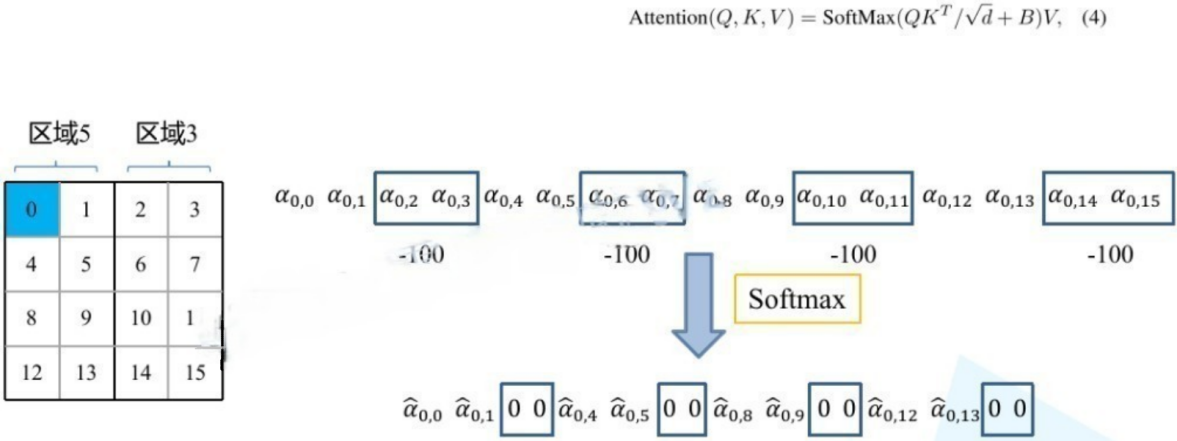
对于W-MSA而言，事先需要在特征图上划分若干个窗口，若窗口的尺寸为，则可获得个窗口。参考前述多头自注意力机制MSA的计算量公式，在窗口内MSA的计算量公式显然为：****，故在全部窗口上**（W-MSA）的计算量为。**显而易见的是，由于（M表示窗口高或宽），W-MSA或者SW-MSA的计算量小于MSA。

1. **为什么需要SW-MSA？**

MSA模块只能在窗口内部展开注意力计算，无法实现窗口间的信息传递。故此，原论文作者提出了Shifted Windows Multi-Head Self-Attention（SW-MSA），通过偏移窗口并填补缺口的方式，完成原始窗口间Patch块的交换，从而达到了窗口间信息传递的目的。具体如下图所示。首先，需要将特征图中的每个窗口都分别向右向下移动个像素点；然后，将新窗口外部的A、B、C填充到窗口内的缺口处；再后，在每个新窗口内部展开掩码MSA计算；最后，撤销先前的偏移和填充操作，恢复到原状。



值得注意的是，上图蓝色框选的窗口内部存在来自不同空间区域的像素块，若直接进行MSA计算，将破坏特征图中不同像素点间的特定位置关系。故此，在此类新窗口内部执行的是Masked MSA计算。其计算过程如下图所示，以计算蓝色高亮处Patch相对于窗口内部所有Patch的注意力分数为例。首先，以蓝色高亮处Patch为查询，计算出窗口内的全局注意力分数；然后，将注意力分数同掩码矩阵相加，以实现将区域3位置上的所有分数都减100；最后，照常通过softmax计算，可以掩码屏蔽掉区域3内部的Patch同蓝色高亮处Patch的注意力分数。



1. Swin Transformer模型在视觉领域的贡献

Swin Transformer作为ICCV2021最佳论文，该模型在目标检测及实例分割等视觉任务中得到了广泛应用。如下图所示，在基于COCO测试集的实例分割任务中，Swin模型变体在2022年仍能夺得冠军宝位。

