6月23日-6月30日工作汇报

Ku Jui

June 2023

Contents

1 Pre-Knowledge

1.1 Transformer

在Attention引入以前，seq2seq模型处理机器翻译task最大的问题就是由于对于长距离的信息不能有效的提取和记忆，导致了信息的大量丢失。即使在引入Attention之后，也会因为对关系的捕捉不足，而出现翻译效果不理想。因为在这样的翻译任务中，需要发现的关系有三种：

(1) 源句内部的关系;

(2) 目标句内部的关系;

(3) 源句与目标句之间的关系;

之前的seq2seq模型只捕捉了源句与目标句之间的关系，而忽略了源句、目标句内部的关系，源句内部和目标句内部还是在用RNN，对远距离信息的捕捉能力很差。除了对远距离关系难以学习的不足以外，RNN还有一点不足，那就是训练慢。因为它默认是按时序来进行处理的，一个个单词词从左到右看过去，导致RNN 不能像 CNN 一样，充分利用 GPU 的并行运算优势。

1.1.1 Idea

(1) 刚刚提到了翻译任务存在有三种关系，原来的模型只学到其中一种，即源句与目标句之间的关系，Transformer**引入self-attention的机制将三种关系全部做了学习。**

(2) Transformer**提出了multi-head attention的机制**，分别学习对应的三种关系，使用了全Attention的结构。

(2) 对于词语的位置，Transformer **使用positional encoding机制进行数据预处理**，增大了模型的并行性，取得了更好的实验效果。

1.1.2 Architecture

如Fig.1所示，**Transformer**模型也是使用经典的encoer-decoder架构，由encoder和decoder两部分组成。

上图的左半边用Nx框中，为encoder的一层。**encoder一共有6层这样的结构。**
上图的右半边用Nx框中，为decoder的一层。**decoder一共有6层这样的结构。**
输入序列经过**word embedding**和**positional encoding**相加后，输入到encoder。
输出序列经过**word embedding**和**positional encoding**相加后，输入到decoder。
最后，decoder输出的结果，经过一个线性层，然后计算softmax。

Encoder

encoder由6层相同的层组成，每一层分别由两部分组成：

- 第一部分是一个multi-head self-attention mechanism;
- 第二部分是一个position-wise feed-forward network，是一个全连接层.

两个部分，都有一个**残差连接(residual connection)**，然后接着一个**Layer Normalization**。

Decoder

和encoder类似，decoder由6个相同的层组成，每一个层包括以下3个部分：

- 第一个部分是multi-head self-attention mechanism
- 第二部分是multi-head context-attention mechanism
- 第三部分是一个position-wise feed-forward network

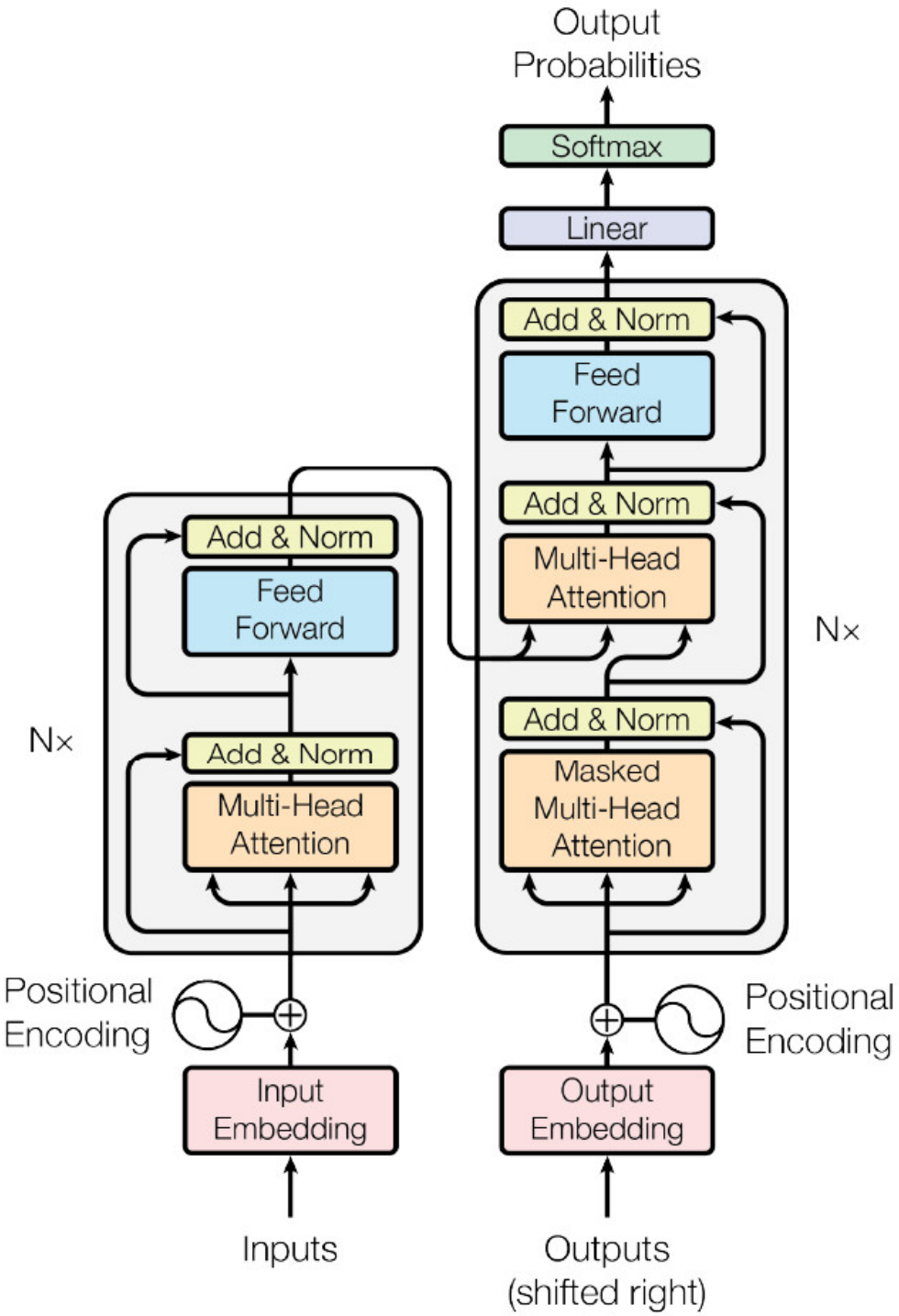

Figure 1: The Transformer - model architecture.

与encoder类似，上面三个部分的每一个部分，都有一个残差连接，后接一个Layer Normalization。

但是，decoder出现了一个新的部分**multi-head context-attention**

1.1.3 Attention Mechanism

Attention是指，对于某个时刻的输出y，它在输入x上各个部分的注意力。这个注意力实际上可以理解为权重。

Scaled dot-product attention

Transformer模型基于乘性注意力(multiplicative attention)，采用**scaled dot-product attention**，即**两个隐状态进行点积。**

$$\mathbf{score}(\mathbf{s_t}, \mathbf{h_i}) = \frac{\mathbf{s_t}^\top \mathbf{h_i}}{\sqrt{n}} \tag{1}$$

其中h_i为输入序列隐状态，s_t为输出序列的隐状态。

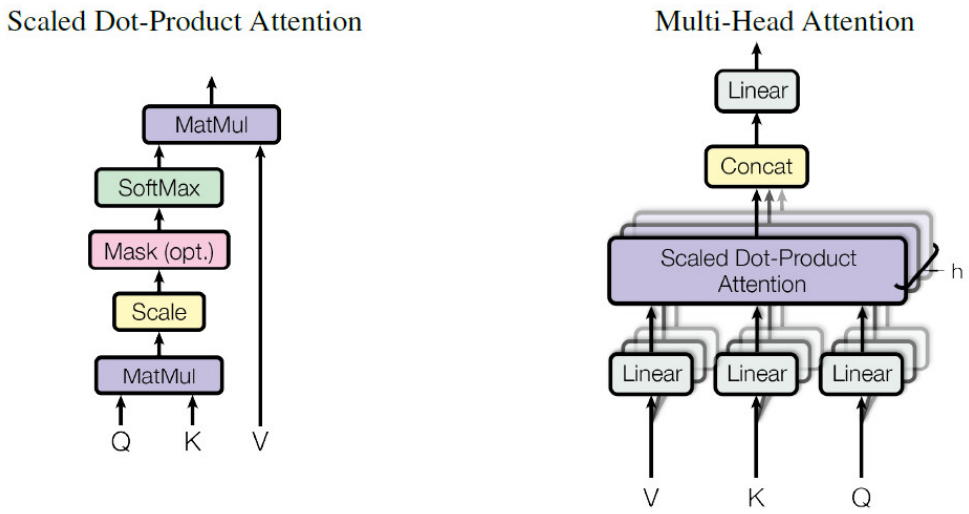

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

self-attention实际上就是，**输出序列就是输入序列!** 因此，计算自己的attention得分，就叫做self-attention!

context-attention是encoder和decoder之间的attention。

从Fig.2(left)看出，Transformer中的attention机制可以被描述为，通过确定Q和K之间的相似程度来选择V，

$$\mathbf{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}})V \tag{2}$$

其中，d_k表示的是K的维度。[1]

Q, K, V具体指代什么？[2]

[1]**为什么需要加上这个缩放因子d_k呢？**论文里给出了解释：对于d_k很大的时候，点积得到的结果维度很大，使得结果处于softmax函数梯度很小的区域。

梯度很小的情况下会对反向传播不利，为了克服这个负面影响，除以一个缩放因子，可以一定程度上减缓这种情况。

[2]https://www.zhihu.com/question/427629601

在Transformer模型中，Q, K, V 是用于执行自注意力机制（Self-Attention）的输入向量。它们分别表示查询(query)，键(key)和值(value)。

- Query(Q)：Query表示查询向量。在自注意力机制中，每个Query向量都会与一组Key和Value向量进行相似度计算，以确定对不同Value向量的注意力权重。可以将Query视为要查询的信息或要关注的特征。
- Key(K)：Key表示被查询信息与其他信息的相关性的向量。键向量用于与查询向量进行点积运算，以计算注意力权重。
- Value(V)：Value表示被查询信息的向量。值向量用于根据注意力权重加权求和的方式来生成最终的上下文表示。

在 Transformer 中，自注意力机制通过将查询向量与所有的键向量进行点积运算，然后将结果经过 softmax 归一化，得到注意力权重。这些注意力权重与对应的值向量进行加权求和，得到最终的上下文表示。

Transformer模型中自注意力机制中Q, K, V的来源、用途以及维度的一些关键特征？

- 在encoder的**self-attention**中，Q, K, V 都来自同一个地方（相等），他们是上一层 encoder 的输出。对于第一层 encoder ，它们就是 word embedding 和 positional encoding 相加得到的输入。
- 在decoder的**self-attention**中，Q, K, V 都来自于同一个地方（相等），它们是上一层 decoder 的输出。对于第一层decoder，它们就是 word embedding 和 positional encoding 相加得到的输入。但是对于 decoder ，我们不希望它能获得下一个 time step （即将来的信息），因此我们需要进行 sequence masking 。
- 在encoder-decoder attention中，Q来自于decoder的上一层的输出，K 和 V 来自于 encoder 的输出，K 和 V 是一样的。
- Q、K、V三者的维度一样，即 $d_q = d_k = d_v$。

Multi-head attention

什么是**multi-head attention**？

如Fig.2(right)所示，作者发现将Q, K, V通过一个线性映射之后，分成 h 份，对每一份进行 **scaled dot-product attention** 效果更好。然后，把各个部分的结果合并起来，再次经过线性映射，得到最终的输出。超参数 h 表示的是 heads 数量[3]。

$$\begin{aligned} &\mathbf{MultiHead}(Q, K, V) = \mathbf{Concat}(\mathbf{head}_1, \ldots, \mathbf{head}_h)W^O, \\ &\mathbf{head}_i = \mathbf{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{3}$$

1.1.4 Layer normalization

- Normalization 有很多种，但是它们都有一个共同的目的，那就是把输入转化成均值为 0 方差为 1 的数据。

[3]论文中的数量默认是8

- Normalization 是一种用于标准化数据的预处理技术[4]。换句话说，在同一范围内具有不同的数据源。在训练之前不对数据进行标准化可能会导致我们的网络出现问题，从而使训练变得更加困难并降低其学习速度。

Batch normalization

Batch normalization是在神经网络层之间而非在原始数据中完成的标准化方法，其主要思想是：**在每一层的每一批数据上进行归一化**。它是按照小批量而不是完整数据集完成的。它可以加快训练速度并使用更高的学习率，使学习变得容易。

可以将Batch normalization的归一化公式定义为：

$$z^N = \left(\frac{z - \mu_z}{\sigma_z} \right) \tag{6}$$

其中μ_z是神经元输出的平均值和σ_z神经元输出的标准差。

我们可能会对输入数据进行归一化，但是经过该网络层的作用后，我们的的数据已经不再是归一化的了。随着这种情况的发展，数据的偏差越来越大，我的反向传播需要考虑到这些大的偏差，这就迫使我们只能使用较小的学习率来防止梯度消失或者梯度爆炸。

Batch normalization的具体做法就是对每一小批数据，在批这个方向上做归一化。如Fig.3所示：

具体Batch normalization是如何应用的？

在Fig.4中展示了一个常规的前馈神经网络，其中x_i是输入、z是神经元的输出、a是激活函数的输出、y是整个网络的输出。

通常，没有Batch normalization的神经元将按Eq.7计算。

$$\begin{aligned} z &= g\left(w, x\right) + b \\ a &= f(z) \end{aligned} \tag{7}$$

$g()$是神经元的线性变换，w是神经元的权重，b是神经元的偏置，$f()$是激活函数。该模型学习参数w和b。添加Batch normalization之后，按Eq.8计算

[4]例如，假设我们有汽车租赁服务。首先，我们希望根据竞争对手的数据预测每辆车的合理价格。每辆车有两个特征：车龄和总行驶公里数。它们的范围可能非常不同，从 0 到 30 年不等，而距离可能从 0 到数十万公里。我们不希望特征在范围上存在这些差异，因为具有较高范围的值可能会使我们的模型产生偏差，从而赋予它们夸大的重要性。

有两种主要方法可以标准化数据。最直接的方法是将其缩放到 0 到 1 的范围：

$$x_{normalized} = \frac{x - m}{x_{max} - x_{min}} \tag{4}$$

x是要标准化的数据点、m是数据集的平均值、x_{max}是最高值、x_{min}是最小值。该技术通常用于数据的输入。大范围的非标准化数据点可能会导致神经网络不稳定。相对较大的输入可能会级联到各层，从而导致梯度爆炸等问题。

用于标准化数据的另一种技术是使用以下公式强制数据点的平均值为 0，标准差为 1：

$$x_{normalized} = \frac{x - \mu}{\sigma} \tag{5}$$

x 是要标准化的数据点、μ是数据集的平均值、σ是数据集的标准差。经过Eq.5后，每个数据点都模仿标准正态分布。拥有正态分布的所有特征，它们不会有偏差，因此，我们的模型会学得更好。

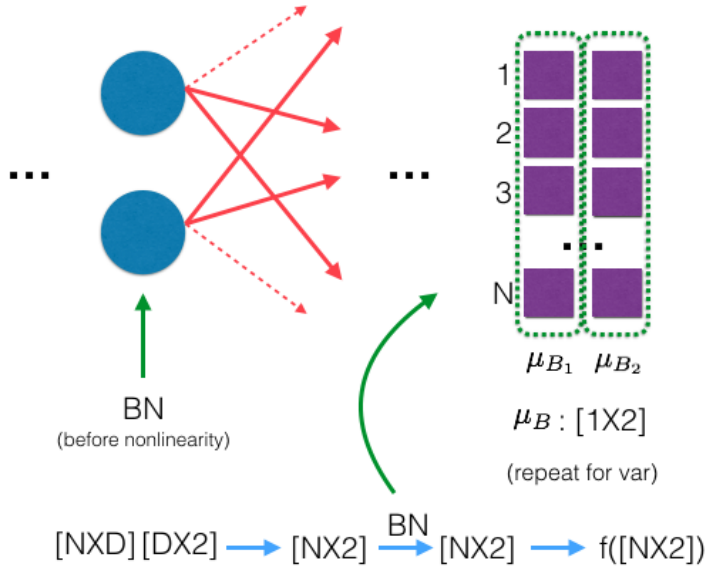

Figure 3: For a given layer, the mean during BN will be 1X. Each training data gets this mean subtracted from it and divided by $\sqrt{(var + \epsilon)}$ and then shifted and scaled. To find the mean and var, we use all the examples in the training batch.

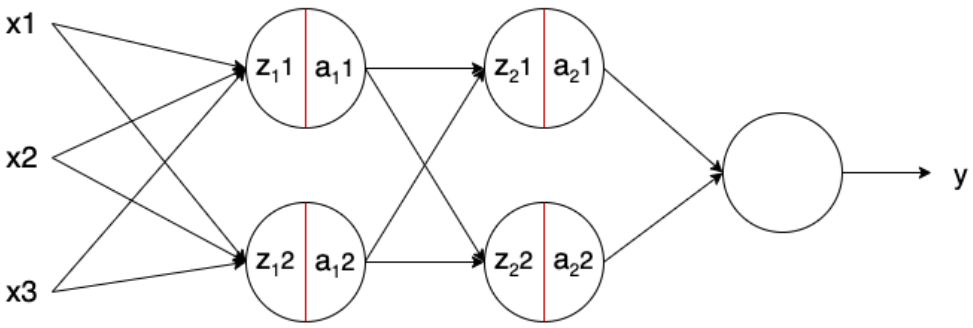

Figure 4: A conventional feedforward neural network.

$$
\begin{aligned}
z &= g\left(w, x\right) \\
BN(z_i) &= \left(\frac{z_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\right) \cdot \gamma + \beta \\
a &= f(BN(z_i))
\end{aligned} \tag{8}
$$

z_i是Batch normalization的输出，μ_B是神经元输出的均值，σ_B是神经元输出的标准差，ϵ是一个小的随机噪声（为了稳定性）。γ为尺度参数，β为位移参数，这两个参数可以从每一个epoch和其他参数（神经元权重）中学习得到[5]。

Batch normalization的好处是：

- **深度神经网络可以训练得更快：**虽然由于前向传播过程中的额外归一化计算和反向传播过程中需要训练的额外超参数，每次训练迭代都会变慢，但它应该收敛得更快；因此，训练总体上应该更快。
- **更高的学习率：**梯度下降通常需要较小的学习率才能使网络收敛。随着网络变得更深，梯度在反向传播过程中变得更小，因此需要更多的迭代。使用批量归一化可以提高学习率，从而提高训练速度。
- **更容易初始化权重：**权重初始化可能很困难，特别是在创建更深的网络时。批量归一化降低了对初始起始重量的敏感性。

Layer normalization

Layer normalization在许多方面与 BN 是类似的，如Eq.9，但是Layer normalization一般多用于循环架构中，如RNN结构。

$$
\mathrm{LN} = \left(\frac{z_i - \mu_L}{\sqrt{\sigma_L^2 + \epsilon}}\right) \cdot \gamma + \beta \tag{9}
$$

从Eq.9中可以看出，Layer normalization作用于每层的每一个样本，其计算特定训练点的特定层的均值和方差。

Layer normalization与 BN 有点不同， LN独立计算每层每个样本的均值和方差，然后使用这些计算值进行 LN 运算(Eq.9)。Layer normalization的过程如Fig.5所示，其在每个样本上计算均值和方差，而不是像BN在每个批次上计算均值和方差。

1.1.5 Mask

Mask 表示掩码，它对某些值进行掩盖，使其在参数更新时不产生效果。Transformer 模型里面涉及两种mask，分别是 padding mask 和 sequence mask 。[6]

Padding mask

因为每个批次输入序列长度是不一样的，因此我们需要对不同长度的序列进行对齐，具体来说，就是给在较短的序列后面填充 0。因为这些填充的位置其实是没有意义的，Attention 机制不应该把注意力放在这

[5]当我们在层上应用Batch normalization时，我们限制输入服从正态分布，这最终将限制网络的学习性能。为了解决这个问题，一般乘以尺度参数γ，并添加位移参数β。

[6]其中，padding mask在所有的scaled dot-product attention里面都需要用到，而sequence mask只有在decoder的self-attention里面用到。

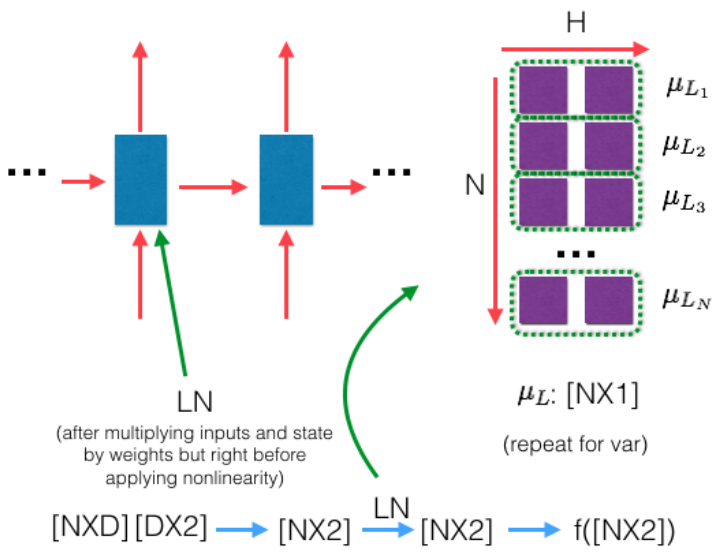

Figure 5: For a given layer, the mean during LN will be NX1. We compute the mean and var for every single sample for each layer independently and then do the LN operations using those computed values.

些位置上，所以具体的做法是，把这些位置的值加上一个非常大的负数(负无穷)，这样经过 softmax 后这些位置的概率就会接近 0，不会影响 Attention 的结果。

Sequence mask

Fig.2(left)为 Multi-head attention中的一个模块，对于decoder中的第一个 Multi-head attention 层，也需要添加一个mask模块进去。这是为了使得decoder看不见未来的信息。也就是说，对于一个序列，在 time_step 为 t 的时刻，我们的解码输出应该只能依赖于 t 时刻之前的输出，而不能依赖 t 时刻之后的输出。因此我们通过增加Mask的方法，把 t 时刻之后的信息给隐藏起来。[7]

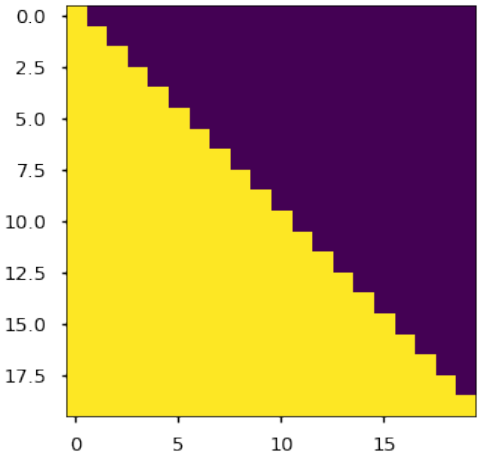

Figure 6: Sequence mask.

[7]具体的做法为：**产生一个上三角矩阵(Fig.6)，下三角的值全为1，上三角的值权威0，对角线也是1**。把这个矩阵作用在每一个序列上，就能达到目的。

1.1.6 Positional encoding

Transformer 摒弃了之前机器翻译任务中常用的 RNN 结构，使得并行性更好。RNN的这一结构天生考虑了词语的先后顺序关系。当 Transformer 模型不使用 RNN 结构时，它就要想办法通过其它机制把位置信息传输到 encoder ，否则可能会出现“所有词语都对了，但是无法组成有意义的语句”的情况。为了解决这个问题，论文使用了 Positional Encoding 对序列中词语出现的位置进行了编码。

在具体实现时，采用正余弦函数来表示Positional encoding

$$\begin{aligned} \mathrm{PE}\left(pos, 2i\right) &= \sin\left(pos/10000^{2i/d_{model}}\right) \\ \mathrm{PE}\left(pos, 2i+1\right) &= \cos\left(pos/10000^{2i/d_{model}}\right) \end{aligned} \tag{10}$$

其中， pos 是指词语在序列中的位置，d_{model}[8]是模型的维度，该公式的意思是：给定词语的位置 pos ，就可以把它编码成d_{model}维的向量。可以看出，在偶数位置，使用正弦编码，在奇数位置，使用余弦编码。[9]

1.1.7 Work embedding

Word embedding 是将 word 看作最小的一个单元，将文本空间中的某个 word ，通过一定的方法，映射或者说嵌入(embedding)到另一个数值向量空间。Word embedding的输入是原始文本中的一组不重叠的词汇，将它们放到一个字典里面，例如：[”cat”,”eat”,”apple”]，就可以作为一个输入。

Word embedding 的输出就是每个 word 的向量表示，变成一个矩阵，如Fig.7所示，通过一个4维向量来表示单词。对于这个二维浮点矩阵，里面的权重都是可以训练的参数，Word embedding 的过程就是如何去构建这个矩阵[10]。

Word embedding 的优点是可以将每一单词映射到一个低维、稠密的实数向量上，向量的每一维代表单词的潜在特征，且该特征能够捕捉到有用的语法和语义属性。词嵌入的最大优势在于我们可以相对快速、有效地根据向量间的距离（比如余弦距离）判断单词的相似性，原因在于语义或语法相似的单词拥有相似的向量表示（见Fig.8）。词嵌入的构建主要依赖于神经网络模型，神经网络能够非常灵活地以线性组合方式表示任意 n-gram 词组，且参数以线性速度增长，相对于基于矩阵和聚类的分布表示，词向量能够捕捉到更多复杂、有用的语义信息。

深度学习 word2vec 是 Google 的一款将词表征为实数值向量的高效工具, 其利用深度学习的思想, 通过训练把对文本内容的处理简化为 k 维向量空间中的向量运算, 而向量空间上的相似度可以用来表示文本语义上的相似度。在实际应用场景中, word2vec 可以用于计算词的相似度、文档的相似度等, 也可以用来对文档进行情感分析等。

深度学习 word2vec 中有 CBOW 和 Skip-gram 两种神经网络框架可用于训练并得到参数, 参数迭代更新后得到的就是每个字的词向量。

---

[8] 论文中默认维度是512

[9] 正弦函数能够表达相对位置信息，主要数学依据见Eq.11

$$\begin{aligned} \sin\left(\alpha+\beta\right) &= \sin\alpha\cos\beta + \cos\alpha\sin\beta \\ \cos\left(\alpha+\beta\right) &= \cos\alpha\cos\beta - \sin\alpha\sin\beta \end{aligned} \tag{11}$$

Eq.11表明，对于词汇之间的位置偏移 k, PE($pos+k$)可以表示为PE(pos)和PE(k)的组合形式，即具备表达相对位置的功能。

[10] 论文中 word embedding 矩阵是一个`vocab_size` × `embedding_size`的二维张量。在论文中，其中`vocab_size`是词典的大小，`embedding_size`是词嵌入的维度大小，论文中维度大小是512。

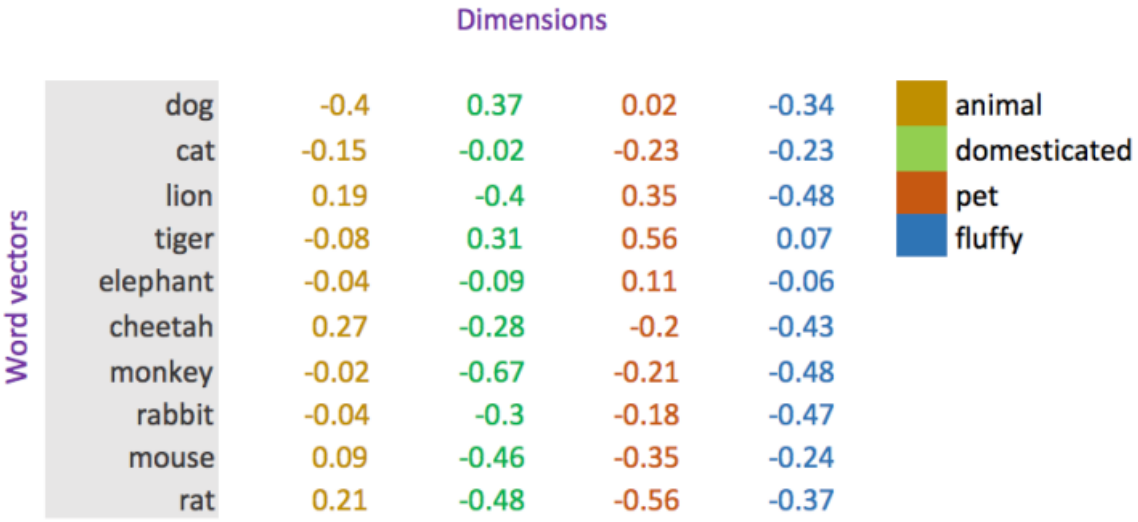

Figure 7: Word embedding.

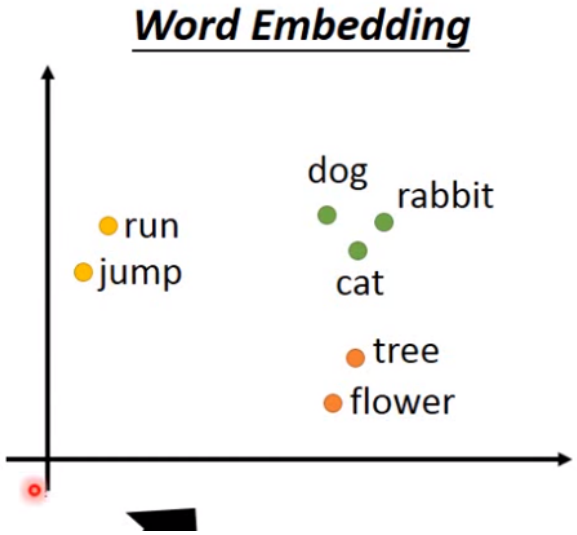

Figure 8: Embedding Later, words with similar meanings will be similar in vector space, and naturally have the effect after clustering, which is a kind of unsupervised learning.

1.1.8 Position-wise Feed-Forward network

除了注意力子层之外，编码器和解码器中的每个层都包含一个完全连接的前馈网络，该网络单独且相同地应用于每个位置。它由两个线性变换组成，中间有一个ReLU激活[11]。

$$\mathrm{FFN}(x) = \max\left(0, xW_1 + b_1\right) W_2 + b_2 \tag{12}$$

其中，W_1和W_2的大小均是 $d_{model} \times d_{ff}$[12]。

Transformer中的FFN全称是 Position-wise Feed-Forward network，重点在于 Position-wise ，区别于普通的全连接网络，这里 FFN 的输入是序列中每个位置上的元素，而不是整个序列，所以每个元素完全可以独立计算，最极端节省内存的做法是遍历序列，每次只取一个元素得到 FFN 的结果，但是这样做时间消耗太大，“分段”的含义就是做下折中，将序列分成 N 段，也就是 N 个子序列，每次读取一个子序列进行 FFN 计算，最后将 N 份的结果拼接，如Eq.13所示。

$$Y = [Y_1; Y_2; \ldots; Y_N] = [X_1 + \mathrm{FFN}(X_1); X_2 + \mathrm{FFN}(X_2); \ldots; X_N + \mathrm{FFN}(X_N)] \tag{13}$$

分段 FFN 只是一种计算上的技巧，计算结果和原始 FFN 完全一致，所以不会影响到模型效果，好处是不需要一次性将整个序列$(batch_size, L, d_{model})$读入内存，劣势是会增加额外的时间开销。

2 Paper reading

2.1 Ultra-High-Definition Low-Light Image Enhancement

2.1.1 Introduce

设备捕捉超高清(UHD)图像和视频的能力对图像处理管道提出了新的要求。

本文考虑了低光亮图像增强(LLIE)的任务，构建了两种不同分辨率的数据集Ultra High Definition Low-Light Image Enhancement(UHD-LOL)，并在不同方法进行基准测试。作者提出一种基于**Transformer**的微光增强方法LLFormer。LLFormer的核心组件是基于轴的多头自注意和跨层注意融合块，显著降低了线性复杂度。

LLFormer的核心设计包括一个基于轴的变压器块(Axis-based Transformer Block)和一个跨层注意力融合块(Dual Gated Feed-forward Network)。在前者中，基于轴的多头自注意在通道维度上依次对高度和宽度轴进行自注意，以降低计算复杂度，而双门控前馈网络采用门控机制来更多地关注有用的特征。跨层注意力融合块在融合不同层中的特征时学习它们的注意力权重。

2.1.2 Innovation

LLFormer 的整体框架如Fig.9所示，可以看出和 Restormer 有些类似。作者改进了三个点：

- (1) Transformer block里面修改了attention；
- (2) Transformer block里修改了FFN；
- (3) 添加了cross-layer attention。

[11] $ReLU: \ f(x) = \max(0, x)$

[12] d_{ff}少则2040，多则4096。全连接层一直是神经网络中的参数量大户，Transformer也不例外。

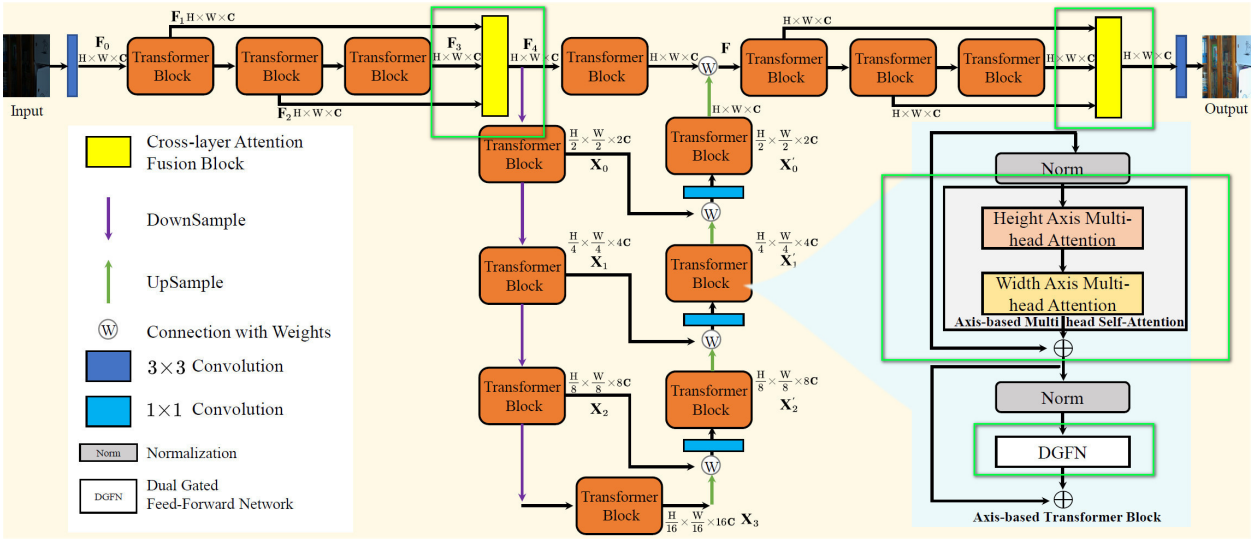

Figure 9: **LLFormer architecture**. The core design of LLFormer includes an axis-based transformer block and a cross-layer attention fusion block. In the former, axis-based multi-head self-attention performs self-attention on the height and width axis across the channel dimension sequentially to reduce the computational complexity, and a dual gated feed-forward network employs a gated mechanism to focus more on useful features. The cross-layer attention fusion block learns the attention weights of features in different layers when fusing them.

Axis-based Transformer Block

Transformer在图像修复中应用的难点在于计算复杂度高，在Q和K计算相似性时，对于输入为(C,H,W)的特征需要进行$(C,H,W)\times(C,H,W)$的矩阵运算。论文使用基于轴的多头注意力，因此，作者分为两个步骤，第一步相似性计算的是$H\times H$，叫做 `height-axis attention` 。第二步相似性计算的是 $W\times W$，叫做 `width-axis attention` 。(这里可以对比 Restormer，只是在C这个维度计算相似性)。。

Dual Gated Feed-forward Network(DGFN)

在Restormer中，FFN有两个分支，其中有一个分支上使用GELU激活对另一个分支添加门控。在该文中，如图Fig.10c，作者在FFN中引入双门控机制，提出双门控前馈网络 DGFN 来捕获特征中的更重要的信息。FFN两个分支都使用 GELU 激活，再互相给另外一个分支添加门控增强非线性建模能力。

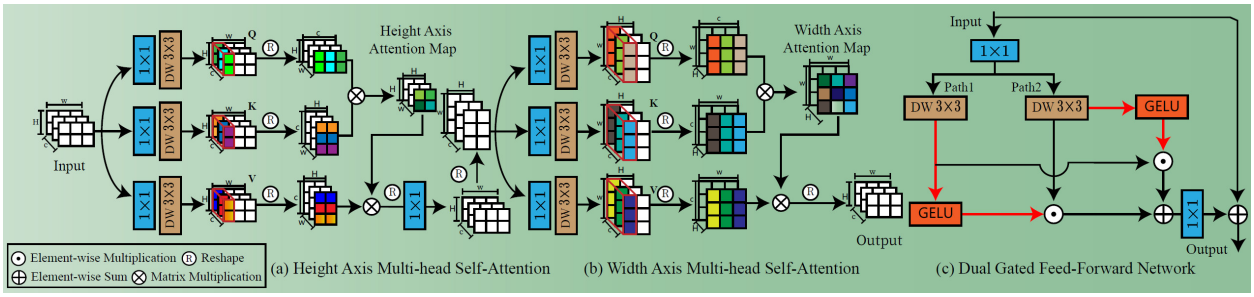

Figure 10: The architecture of our Axis-based Multi-head Self-Attention and Dual Gated Feed-Forward Network. From left to right, the components are Height Axis Multi-head Attention, Width Axis Multi-head Attention, and Dual Gated Feed-Forward Network.

$$
\begin{aligned}
\mathbf{F}, &= \text{A-MSA}\left(\text{LN}(\mathbf{F_{in}})\right) + \mathbf{F_{in}}, \\
\mathbf{F_{out}} &= \text{DGFN}\left(\text{LN}\mathbf{F}'\right)) + \mathbf{F}',
\end{aligned} \tag{14}
$$

其中，DGFN公式如下：

$$
\begin{aligned}
\mathbf{DG} &= \phi\left(W_{3\times3}^{1} W_{1\times1}^{1}\mathbf{Y}\right) \odot \left(W_{3\times3}^{2} W_{1\times1}^{2}\mathbf{Y}\right) \\
&\quad + \left(W_{3\times3}^{1} W_{1\times1}^{1}\mathbf{Y}\right) \odot \phi\left(W_{3\times3}^{2} W_{1\times1}^{2}\mathbf{Y}\right), \\
\hat{\mathbf{Y}} &= W_{1\times1}\mathbf{DG}\left(\mathbf{Y}\right) + \mathbf{Y}
\end{aligned} \tag{15}
$$

Cross-layer Attention Fusion Block

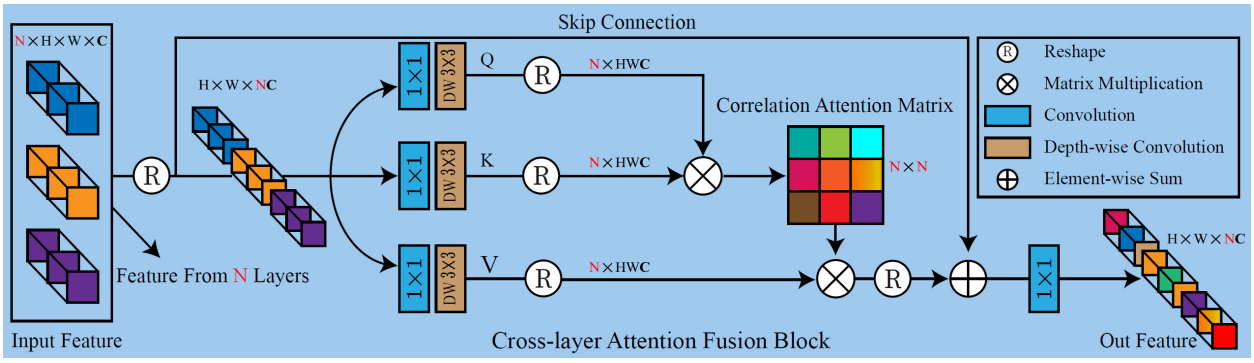

Figure 11: The architecture of the proposed Cross-layer Attention Fusion Block. This block efficiently integrates features from different layers with a layer correlation attention matrix.

网络一般有多层，但大多方法没有考虑层与层之间特征的关联，限制了表示能力。论文使用cross-layer attention获取不同层特征间的相关性并融合。从论文整体架构图中可看到，网络输入有三个Transformer block，能得到三个特征输出。论文通过cross-layer attention 运算，计算一个3x3的相似性矩阵，给输入的特征进行加权，从而达到强调重要特征、抑制不重要特征的作用。（不同层的激活是对特定类别的响应，并且可以使用子注意力机制自适应地学习特征相关性。）

$$
\begin{aligned}
\hat{\mathbf{F}}_{\mathbf{out}} &= W_{1\times1}^{1}\text{Layer\_Attention}(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}) + \hat{\mathbf{F}_{\mathbf{in}}}, \\
\text{Layer\_Attention}(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}) &= \hat{\mathbf{V}}\text{softmax}\left(\hat{\mathbf{Q}}\hat{\mathbf{K}}/\alpha\right),
\end{aligned} \tag{16}
$$

2.1.3 Result

结果见Fig.12 和 Fig.13

3 个人工作进展

3.1 思考

(1) 以后的工作中如果有涉及可以尝试使用双门控来增强局部信息的提取能力，在有依据的情况下可以将注意力计算分解为多步，通过多步计算来控制计算量。

Methods	UHD-LOL4K				UHD-LOL8K			
	PSNR ↑	SSIM ↑	LPIPS ↓	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MAE ↓
input images	11.9439	0.5295	0.3125	0.2591	13.7486	0.6415	0.3104	0.2213
BIMEF† (Ying, Li, and Gao 2017)	18.1001	0.8876	0.1323	0.1240	19.5225	0.9099	0.1825	0.1048
FEA† (Dong et al. 2011)	18.3608	0.8161	0.2197	0.0986	15.3301	0.7699	0.3696	0.1700
LIME† (Guo, Li, and Ling 2016)	16.1709	0.8141	0.2064	0.1285	13.5699	0.7684	0.3055	0.2097
MF† (Fu et al. 2016a)	18.8988	0.8631	0.1358	0.1111	18.2474	0.8781	0.2158	0.1258
NPE† (Wang et al. 2013)	17.6399	0.8665	0.1753	0.1125	16.2283	0.7933	0.3214	0.1506
SRIE† (Fu et al. 2016b)	16.7730	0.8365	0.1495	0.1416	19.9637	0.9140	0.1813	0.0975
MSRCR† (Jobson, Rahman, and Woodell 1997)	12.5238	0.8106	0.2136	0.2039	12.5238	0.7201	0.4364	0.2352
RetinexNet‡ (Wei et al. 2018)	21.6702	0.9086	0.1478	0.0690	21.2538	0.9161	0.1792	0.0843
DSLR‡ (Lim and Kim 2020)	27.3361	0.9231	0.1217	0.0341	21.9406	0.8749	0.2661	0.0805
KinD‡ (Zhang, Zhang, and Guo 2019)	18.4638	0.8863	0.1297	0.1060	17.0200	0.7882	0.1739	0.1538
Z_DCE§ (Guo et al. 2020)	17.1873	0.8498	0.1925	0.1465	14.1593	0.8141	0.2847	0.1914
Z_DCE++§ (Li, Guo, and Loy 2021)	15.5793	0.8346	0.2223	0.1701	14.6837	0.8348	0.2466	0.1904
RUAS△ (Liu et al. 2021b)	14.6806	0.7575	0.2736	0.1690	12.2290	0.7903	0.3557	0.2445
ELGAN△ (Jiang et al. 2021)	18.3693	0.8642	0.1967	0.1011	15.2009	0.8376	0.2293	0.1713
Uformer⋆ (Wang et al. 2022b)	29.9870	0.9804	0.0342	0.0262	28.9244	0.9747	0.0602	0.0344
Restormer⋆ (Zamir et al. 2022)	36.9094	0.9881	0.0226	0.0117	35.0568	0.9858	0.0331	0.0195
LLFormer⋆	37.3340	0.9889	0.0200	0.0116	35.4313	0.9861	0.0267	0.0194

Figure 12: Benchmarking study on the UHD-LOL4K and UHD-LOL8K subsets. †; ‡; §;△ and ? indicate the traditional methods, supervised CNN-based methods, unsupervised CNN-based methods, zero-shot methods and transformer-based methods. The top three results are marked in red, blue and purple, respectively. Input

Methods	LOL				MIT-Adobe FiveK			
	PSNR ↑	SSIM ↑	LPIPS ↓	MAE ↓	PSNR ↑	SSIM ↑	LPIPS ↓	MAE ↓
BIMEF (Ying, Li, and Gao 2017)	13.8752	0.5950	0.3264	0.2063	17.9683	0.7972	0.1398	0.1134
FEA (Dong et al. 2011)	16.7165	0.4784	0.3847	0.1421	15.2342	0.7161	0.1949	0.1512
LIME (Guo, Li, and Ling 2016)	16.7586	0.4449	0.3945	0.1200	13.3031	0.7497	0.1319	0.2044
MF (Fu et al. 2016a)	16.9662	0.5075	0.3796	0.1416	17.6271	0.8143	0.1204	0.1194
NPE (Wang et al. 2013)	16.9697	0.4839	0.4049	0.1290	17.3840	0.7932	0.1320	0.1224
SRIE (Fu et al. 2016b)	11.8552	0.4954	0.3401	0.2571	18.6273	0.8384	0.1047	0.1030
MSRCR (Jobson, Rahman, and Woodell 1997)	13.1728	0.4615	0.4350	0.2067	13.3149	0.7515	0.1767	0.1993
RetinexNet (Wei et al. 2018)	16.7740	0.4250	0.4739	0.1256	12.5146	0.6708	0.2535	0.2068
DSLR (Lim and Kim 2020)	14.9822	0.5964	0.3757	0.1918	20.2435	0.8289	0.1526	0.0880
KinD (Zhang, Zhang, and Guo 2019)	17.6476	0.7715	0.1750	0.1231	16.2032	0.7841	0.1498	0.1379
Z_DCE (Guo et al. 2020)	14.8607	0.5624	0.3352	0.1846	15.9312	0.7668	0.1647	0.1426
Z_DCE++ (Li, Guo, and Loy 2021)	14.7484	0.5176	0.3284	0.1801	14.6111	0.4055	0.2309	0.1539
RUAS (Liu et al. 2021b)	16.4047	0.5034	0.2701	0.1534	15.9953	0.7863	0.1397	0.1426
ELGAN (Jiang et al. 2021)	17.4829	0.6515	0.3223	0.1352	17.9050	0.8361	0.1425	0.1299
Uformer (Wang et al. 2022b)	18.5470	0.7212	0.3205	0.1134	21.9171	0.8705	0.0854	0.0702
Restormer (Zamir et al. 2022)	22.3652	0.8157	0.1413	0.0721	24.9228	0.9112	0.0579	0.0556
LLFormer	23.6491	0.8163	0.1692	0.0635	25.7528	0.9231	0.0447	0.0505

Figure 13: Comparison results on LOL and MIT-Adobe FiveK datasets in terms of PSNR, SSIM, LPIPS and MAE. The top three results are marked in red, blue and purple, respectively. Same as (Zamir et al. 2020), we consider images from expert C for the MIT-Adobe FiveK dataset.

(2) Self-attention Module 到底在 CV 能做什么？

4 下周工作计划

(1) 继续了解Transformer结构原理，去了解Mask和Positional Encoding的部分。

(2) 了解的Transformer模型是通用模型，而且应用在NLP领域，具体细化到CV领域，会有领域差异，目前已经了解到的应用到CV领域的Transformer模型，有ViT，DeiT，Swin，对它们进行一个初步的了解

References