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

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**

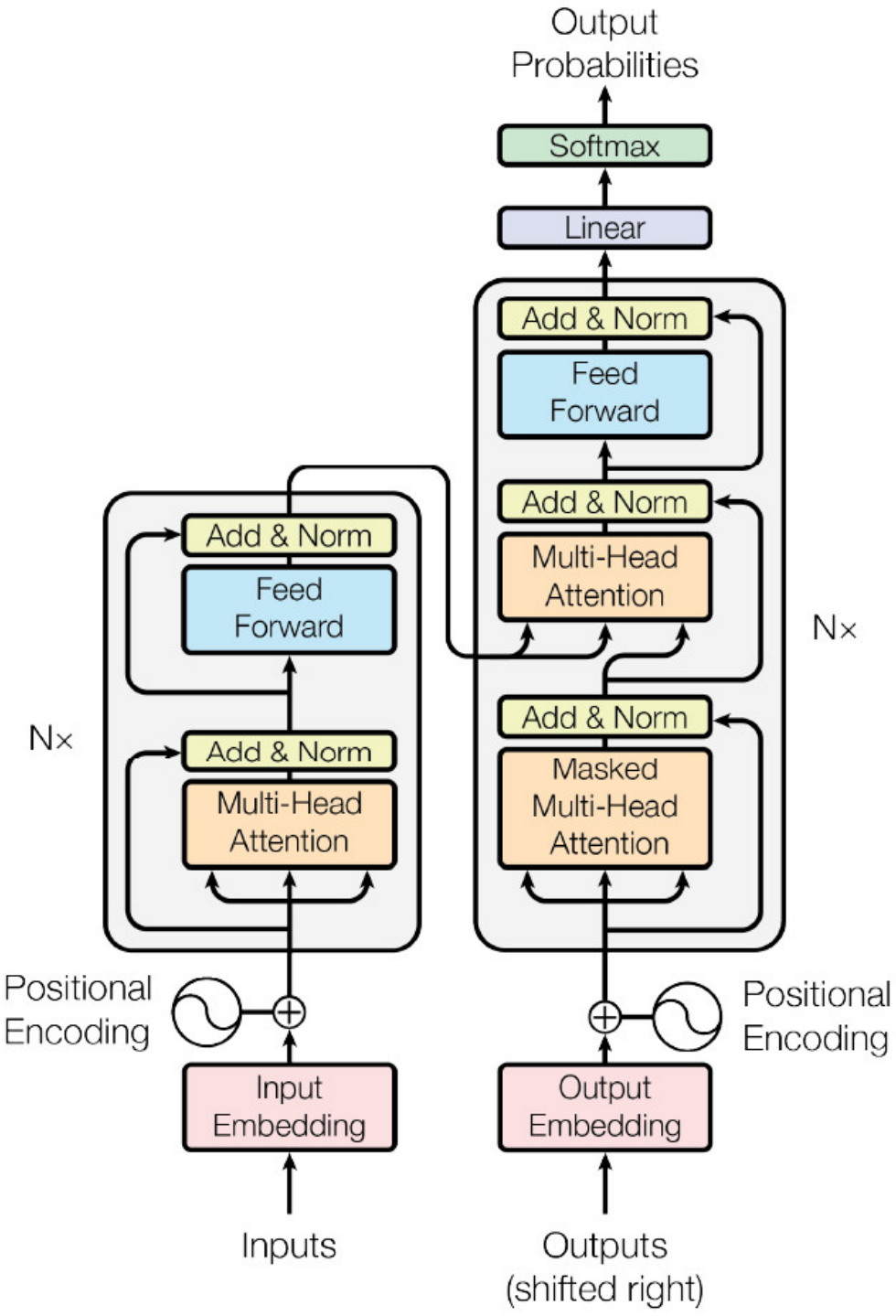

Figure 1: The Transformer - model architecture.

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。

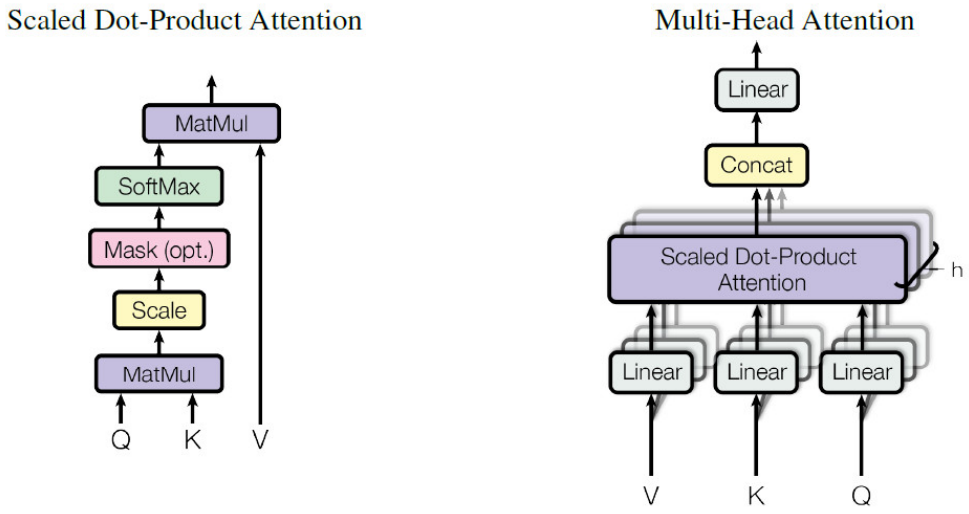

Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

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。

为什么需要加上这个缩放因子呢？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。

我们知道，梯度很小的情况，这对反向传播不利。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。

K, Q, V 具体指代什么？

- 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。

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

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.3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。

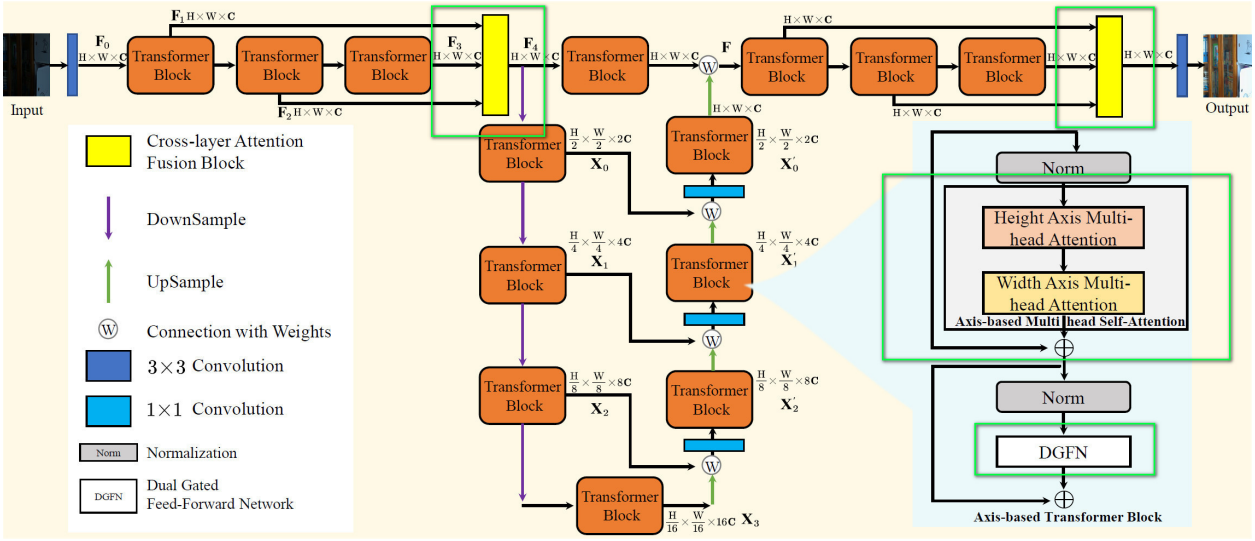

Figure 3: **LLFormer architecture**. The core design of LLFormer includes an axis-based transformer block and a cross-layer attention fusion block. In the former, axis-based multi-head self-attention performs self-attention on the height and width axis across the channel dimension sequentially to reduce 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.4c，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。

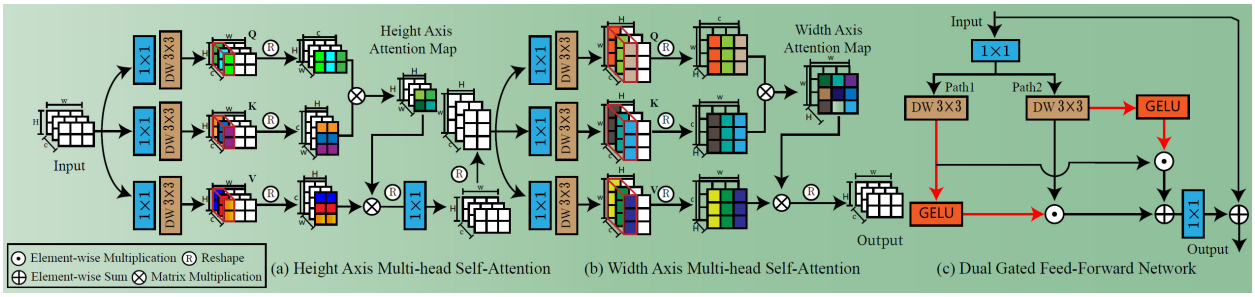

Figure 4: The architecture of our Axis-based Multi-head Self-Attention and Dual Gated Feed-Forward Network. From left to right, the components are Height Axis Multi-head Attention, Width Axis Multi-head Attention, and Dual Gated Feed-Forward Network.

$$
\begin{aligned}
\mathbf{F}, &= \text{A-MSA}\left(\text{LN}(\mathbf{F_{in}})\right) + \mathbf{F_{in}}, \\
\mathbf{F_{out}} &= \text{DGFN}\left(\text{LN}\mathbf{F}'\right)) + \mathbf{F}',
\end{aligned} \tag{4}
$$

其中，DGFN公式如下：

$$
\begin{aligned}
\mathbf{DG} &= \phi\left(W_{3\times3}^{1}W_{1\times1}^{1}\mathbf{Y}\right) \odot \left(W_{3\times3}^{2}W_{1\times1}^{2}\mathbf{Y}\right) \\
&\quad + \left(W_{3\times3}^{1}W_{1\times1}^{1}\mathbf{Y}\right) \odot \phi\left(W_{3\times3}^{2}W_{1\times1}^{2}\mathbf{Y}\right), \\
\hat{\mathbf{Y}} &= W_{1\times1}\mathbf{DG}\left(\mathbf{Y}\right) + \mathbf{Y}
\end{aligned} \tag{5}
$$

Cross-layer Attention Fusion Block

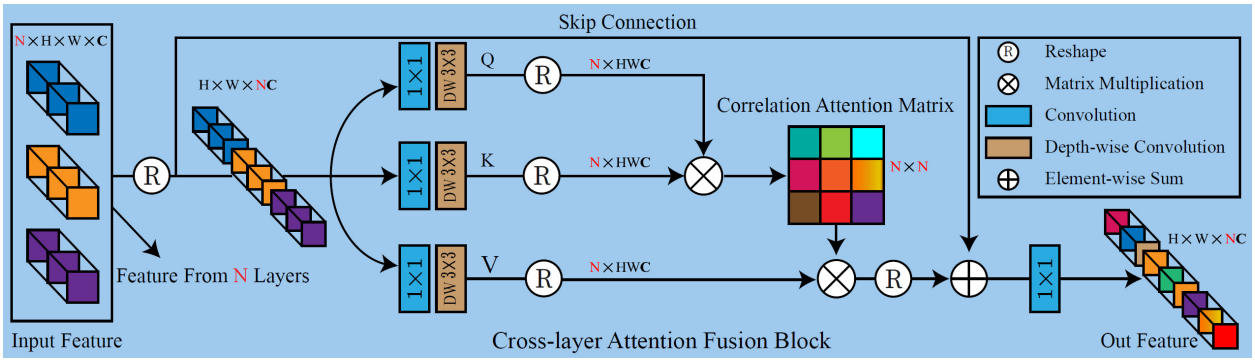

Figure 5: The architecture of the proposed Cross-layer Attention Fusion Block. This block efficiently integrates features from different layers with a layer correlation attention matrix.

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。）

$$
\begin{aligned}
\hat{\mathbf{F}}_{\mathbf{out}} &= W_{1\times1}^{1}\text{Layer\_Attention}(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}) + \hat{\mathbf{F}_{\mathbf{in}}}, \\
\text{Layer\_Attention}(\hat{\mathbf{Q}}, \hat{\mathbf{K}}, \hat{\mathbf{V}}) &= \hat{\mathbf{V}}\text{softmax}\left(\hat{\mathbf{Q}}\hat{\mathbf{K}}/\alpha\right),
\end{aligned} \tag{6}
$$

2.1.3 Result

结果见 Fig.6 和 Fig.7

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 6: 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 7: 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