深度之眼
deepshare.net

本期论文主题:Transformer

导师：Yamada

Attention is all you need

注意力机制是大家需要掌握的

作者：Ashish Vaswani

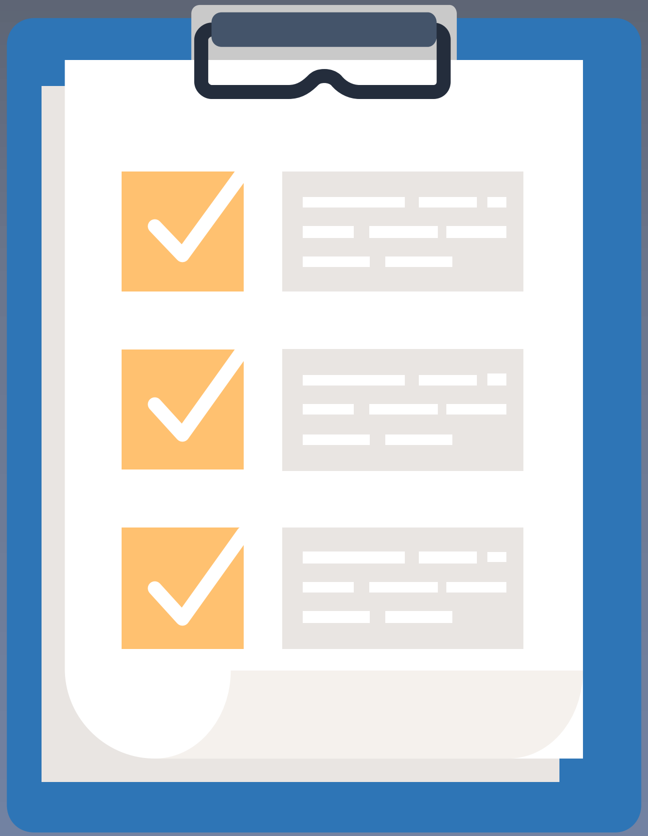
单位：Google

发表会议及时间：NIPS，2017

上节回顾

Review in the previous lesson

深度之眼
deepshare.net



01 Transformer背景介绍

介绍了翻译数据集、翻译衡量指标bleu、transformer的历史意义等等

02 论文泛读

快速浏览整篇论文、介绍了论文的摘要以及各个小标题并且分析了一下论文的实验结果

03 Seq2Seq以及attention回顾

回顾了Seq2Seq的翻译流程以及对attention的公式

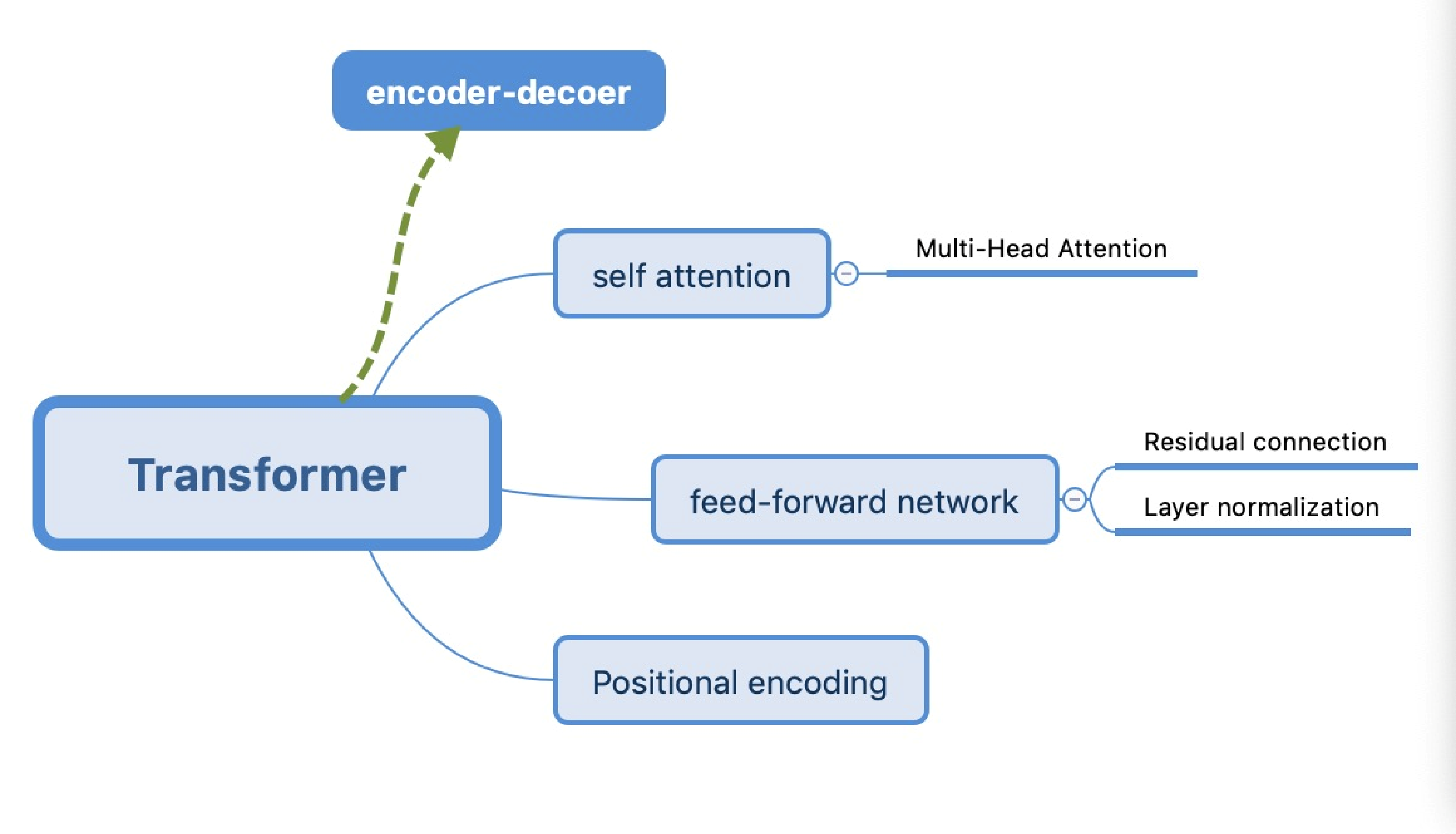
深度之眼
deepshare.net

第二课：论文精读

The second lesson: the paper in detail

目 录

知识树


encoder-decoer
self attention
Multi-Head Attention
Transformer
feed-forward network
Residual connection
Layer normalization
Positional encoding

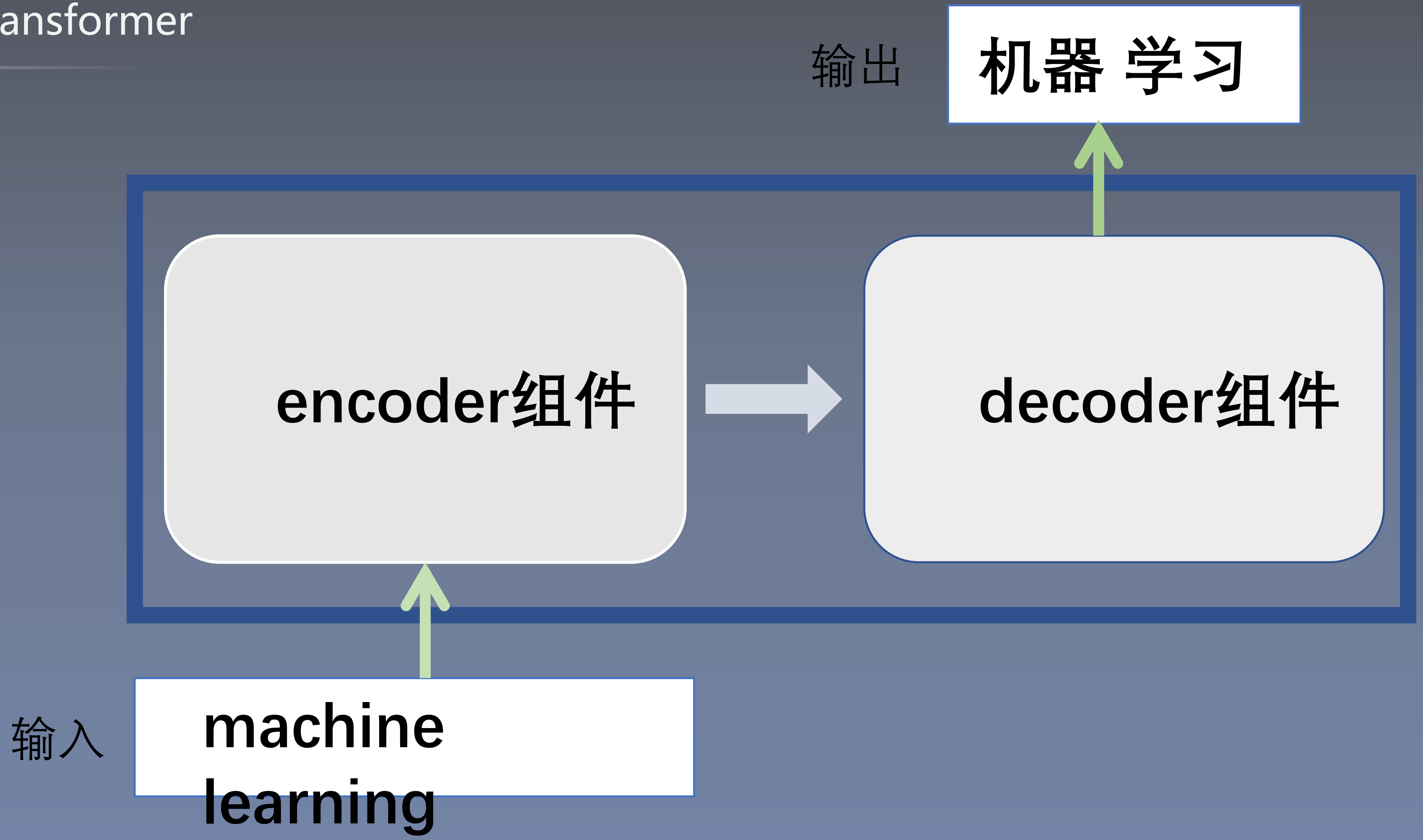
深度之眼
deepshare.net

论文算法模型总览

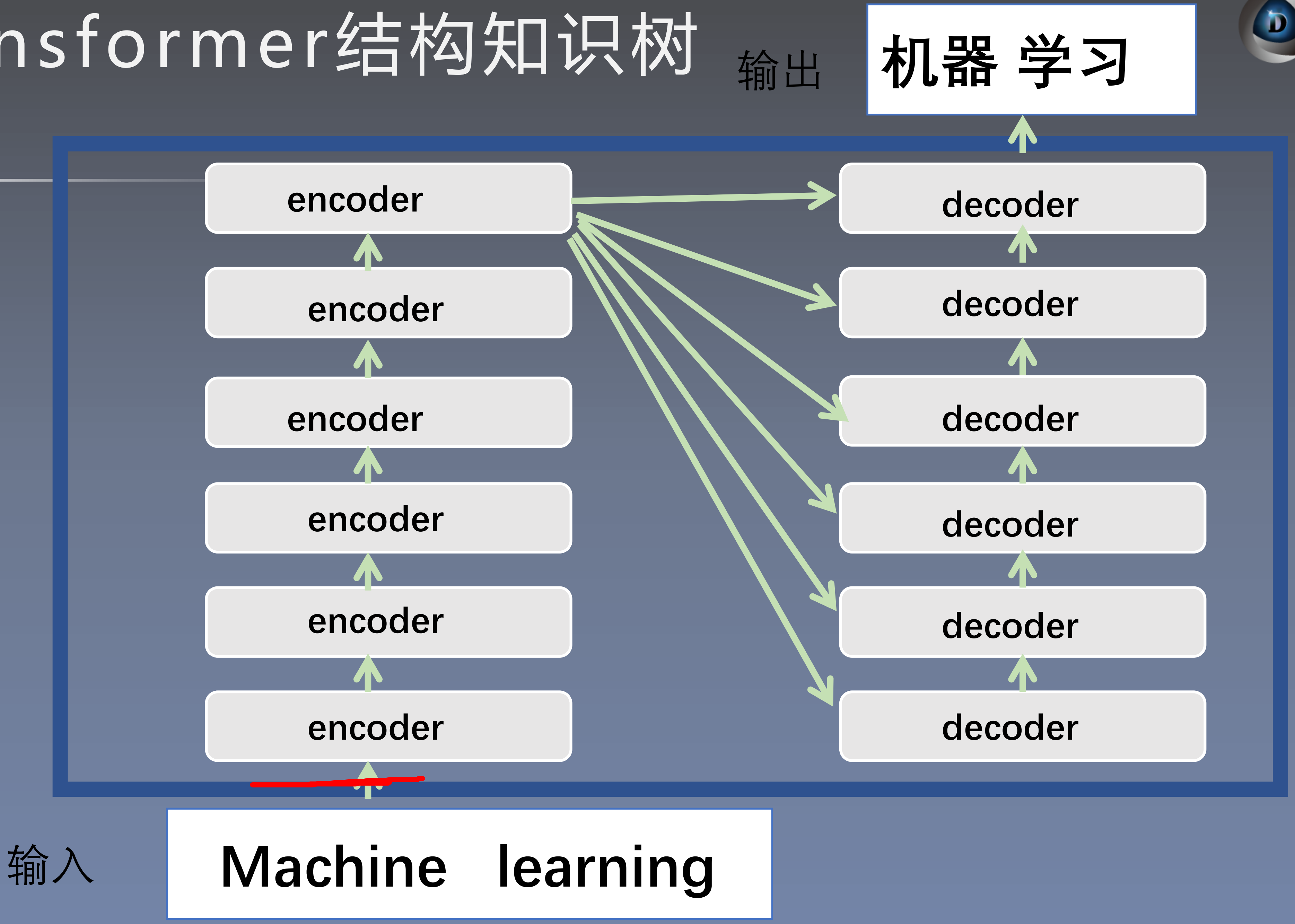
注：这一部分主要介绍在论文改进前的原有模式模型

Transformer结构知识树

Architecture of Transformer

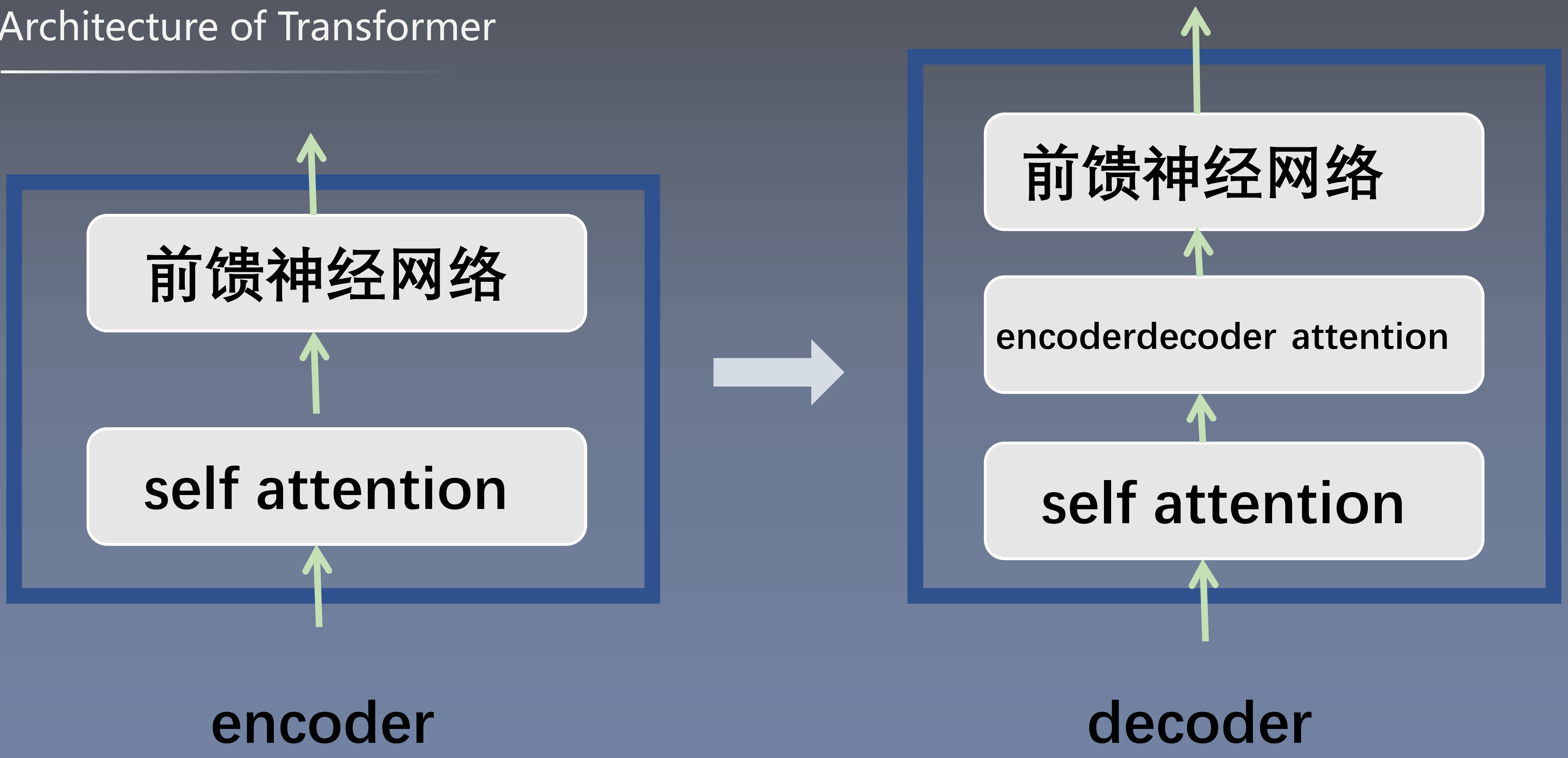


Transformer结构知识树



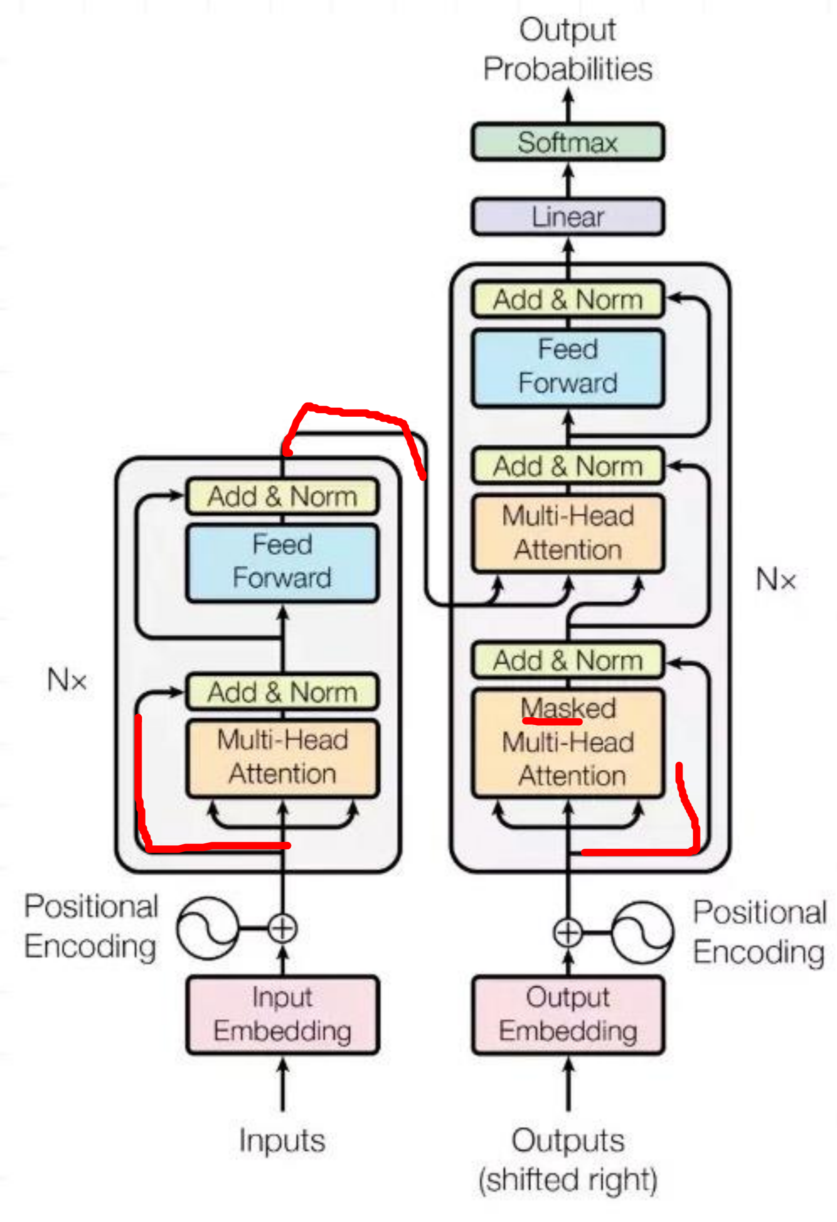
Transformer结构知识树

Architecture of Transformer



Transformer结构知识树

Architecture of Transformer



Transformer结构知识树

Architecture of Transformer

输入

input embedding--> positional encoding

for i in range（6）：

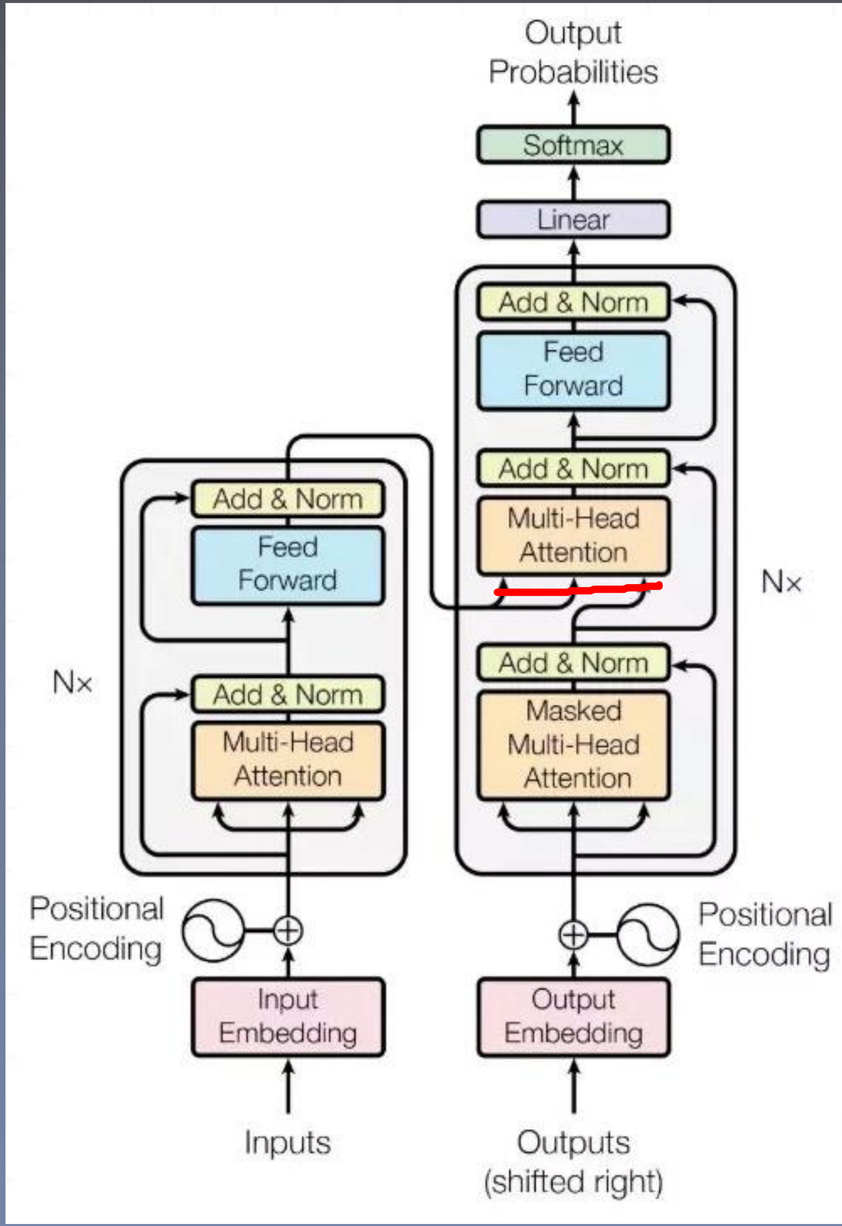
encoder

self attention--> layer normalization
-->feed forward-->layer normalization

for i in range（6）：

decoder

self attention--> layer normalization
-->encoder-decoder attention-->
layer normalization-->feed forward--
>layer normalization



深度之眼
deepshare.net

论文算法模型的细节一

self-attention

Scaled Dot-Product Attention

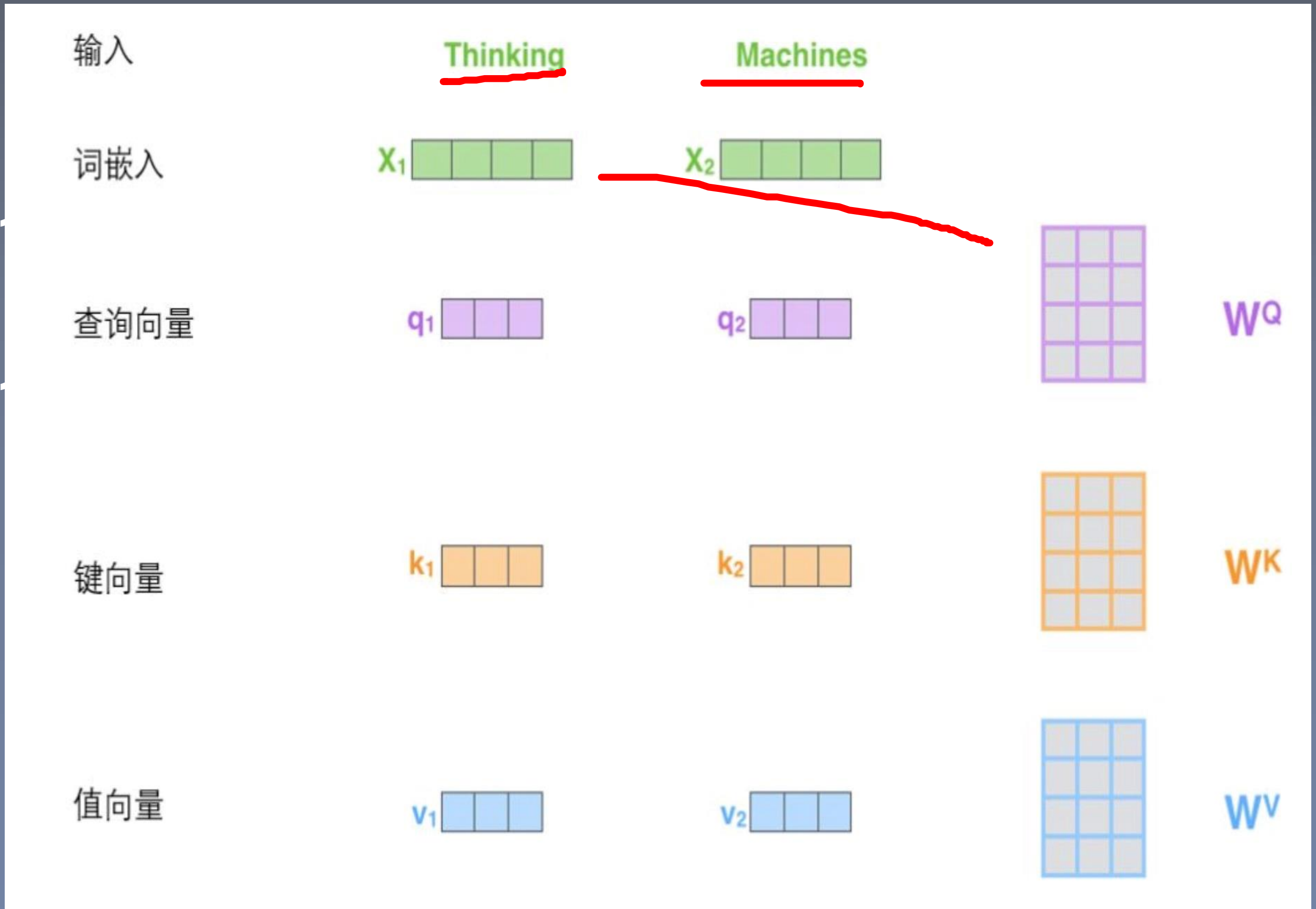
input

输入

x1:[batch_size,1,embedding_dim=51

x2:[batch_size,1,embedding_dim=51

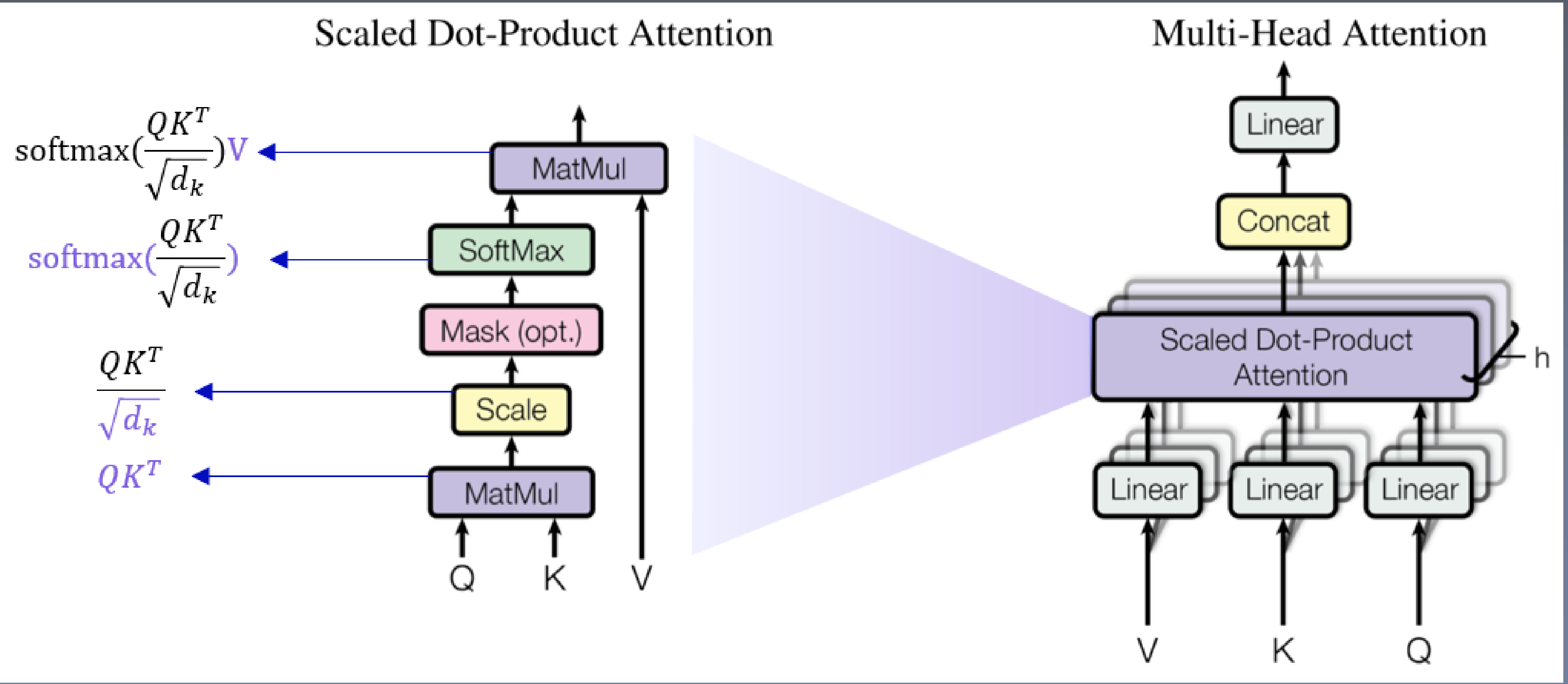
W^Q=[d_model=512,64]
W^K=[d_model=512,64]
W^V=[d_model=512,64]



深度之眼 deepshare.net

Scaled Dot-Product Attention

Architecture



$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

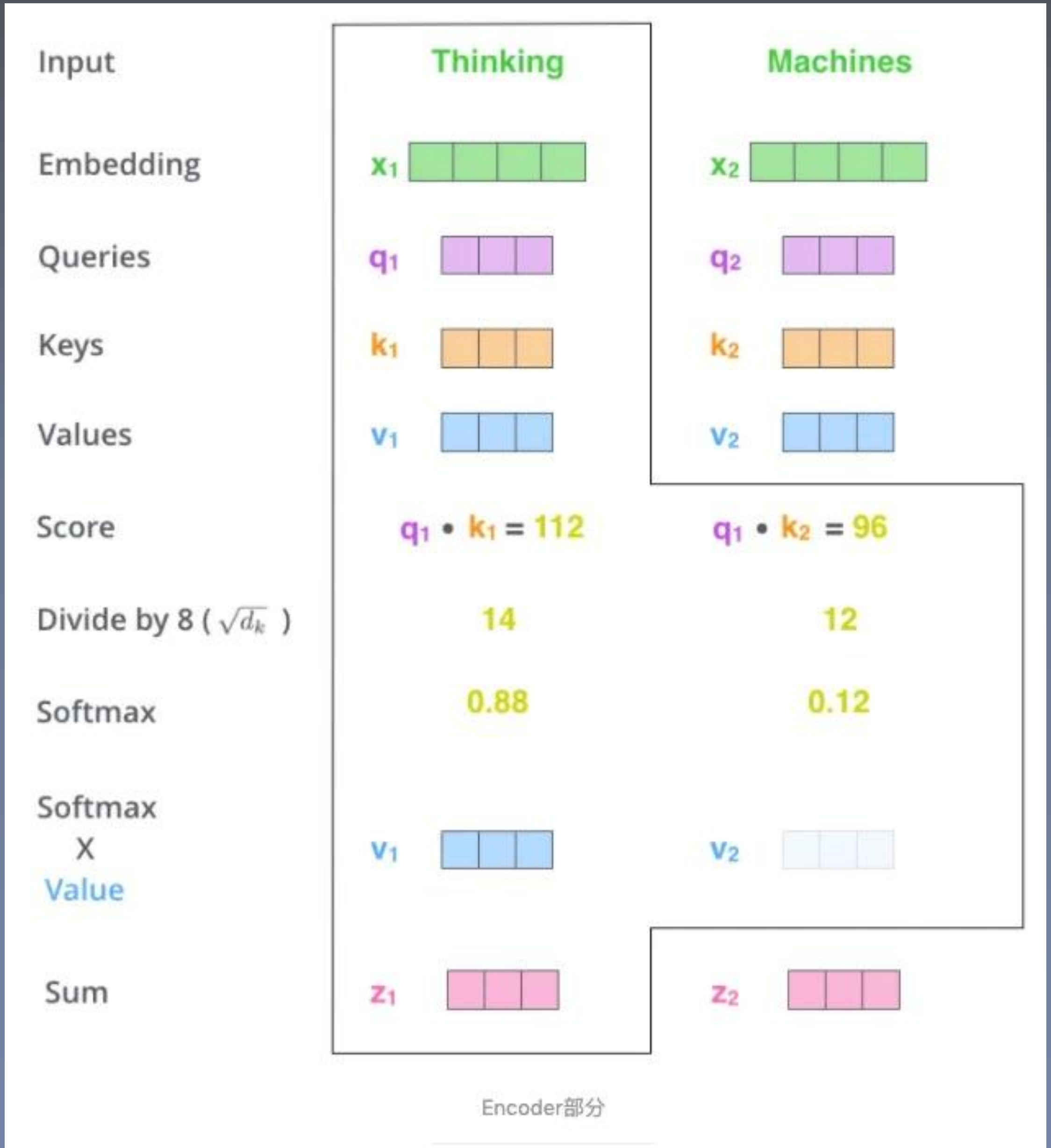
Scaled Dot-Product Attention

Architecture

输入

x1:[batch_size,1,embedding_dim=512]

x2:[batch_size,1,embedding_dim=512]



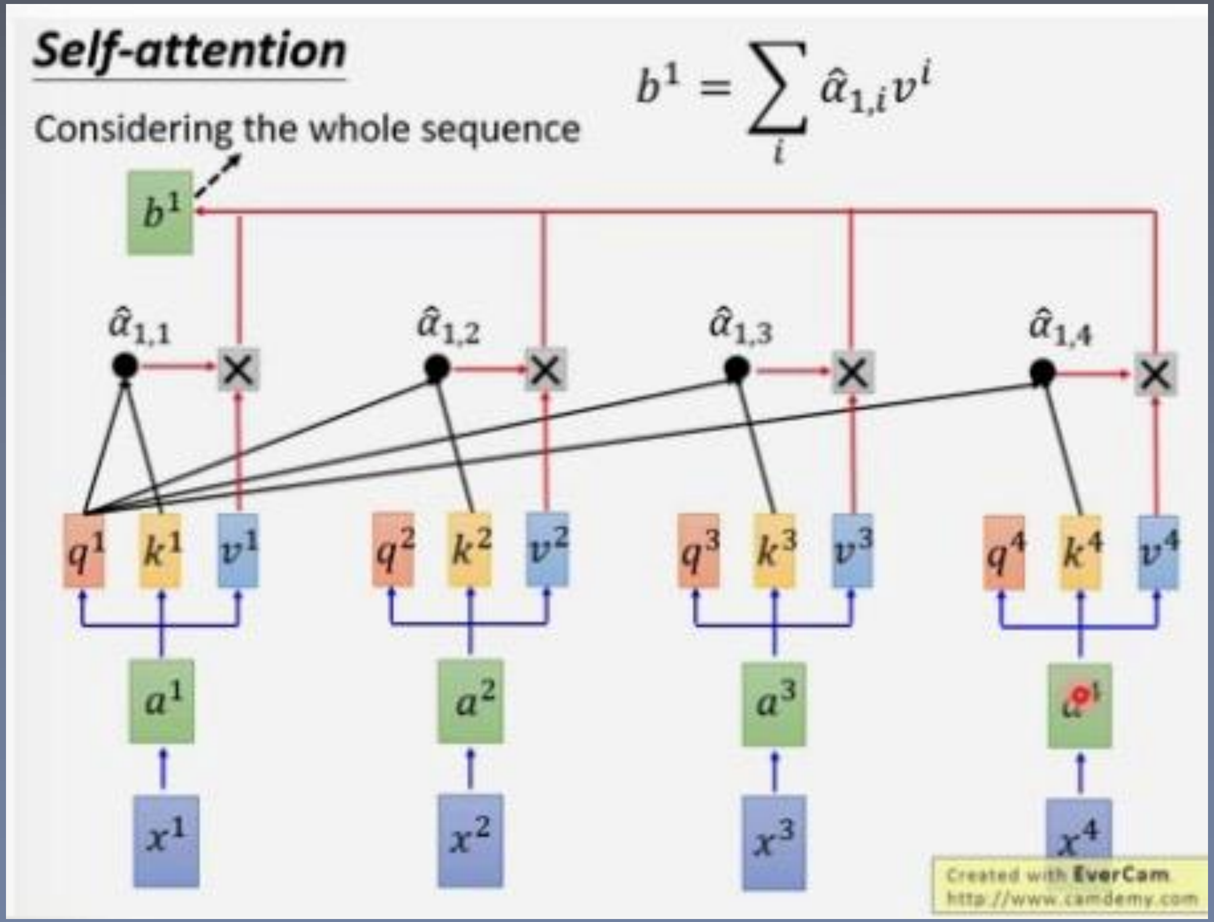
Scaled Dot-Product Attention

Architecture

输入

x1:[batch_size,1,embedding_dim=512]

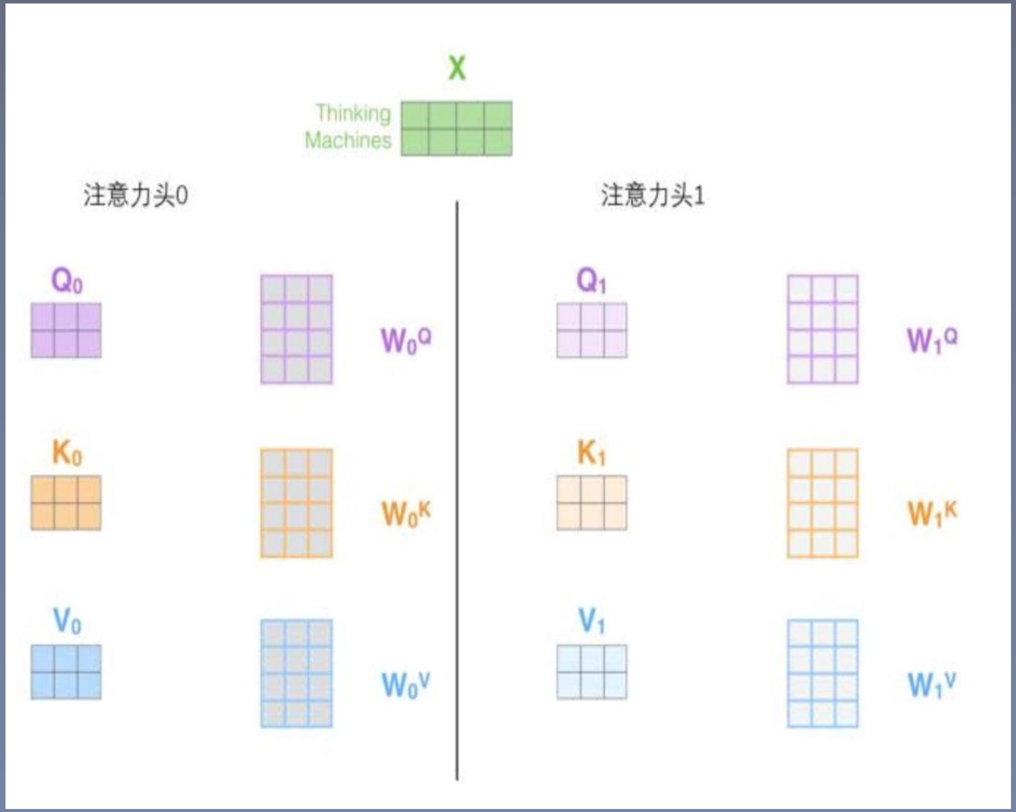
x2:[batch_size,1,embedding_dim=512]

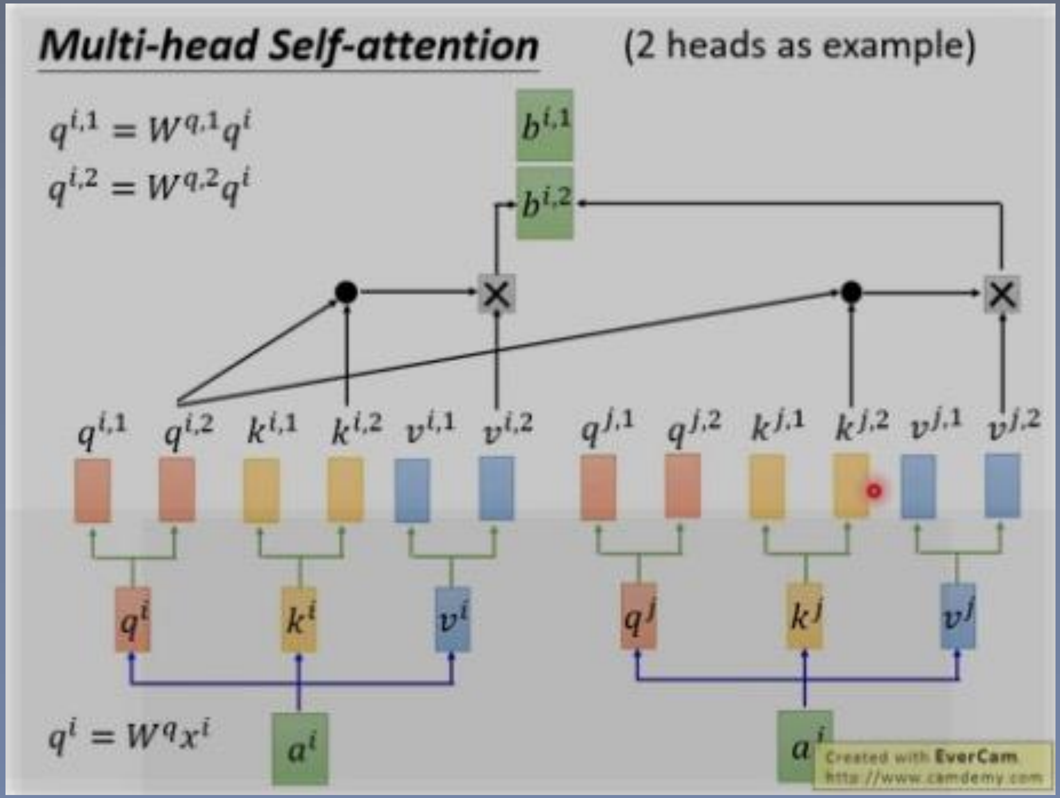


Multi-Head Attention

Architecture

将Q、K、V分成h个头，方便并行计算



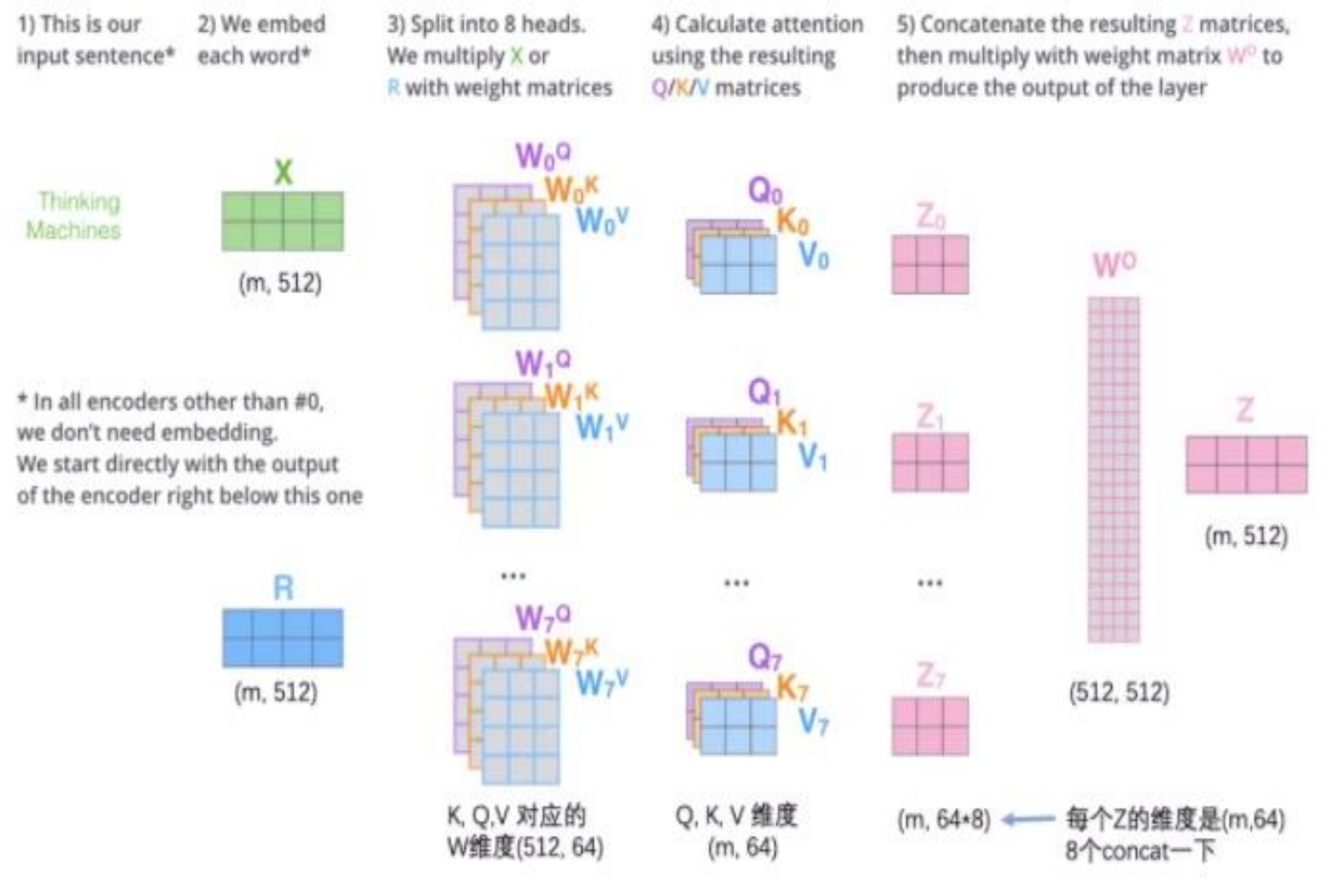


深度之眼
deepshare.net

Multi-Head Attention

Architecture

“多头”注意力机制流程



$$MultiHead(Q,K,V) = Concat(head_1, ..., head_h)W^O$$

$$where \quad head_i = Attention(Q^{W_i^Q}, K^{W_i^K}, V^{W_i^V})$$

内部Attention

Architecture

维度

[batch_size,sequence_length,embedding_dim]

	Quries	Keys	Values
encoder-encoder	encoder-inputs	encoder-inputs	encoder-inputs
encoder-decoder	decoder-inputs	encoder-inputs	encoder-inputs
decoder-decoder	decoder-inputs	decoder-inputs	decoder-inputs

知己

知彼

深度之眼
deepshare.net

论文算法模型的细节二

Feed-Forward Network

Feed-Forward Network

Architecture

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

前馈层：包含两层。1、线性结构，2、卷积结构

x：上一层的输出（一般是self-attention的输出）

W_1、W_2、b_1、b_2都是需要学习的参数

深度之眼
deepshare.net

论文算法模型的细节三

Positional Encoding

Positional Encoding

Architecture

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

i 代表的是当前的维度，pos代表的是当前的位置

$$[\sin(3/10000^{0/128}), \cos(3/10000^{1/128}), \sin(3/10000^{2/128}), \cos(3/10000^{3/128}), \ldots]$$

pos=3，对应的positional encoding

原文讲解

论文算法模型的细节四

Training Details

Mask

Training Details

1、Sequence Mask

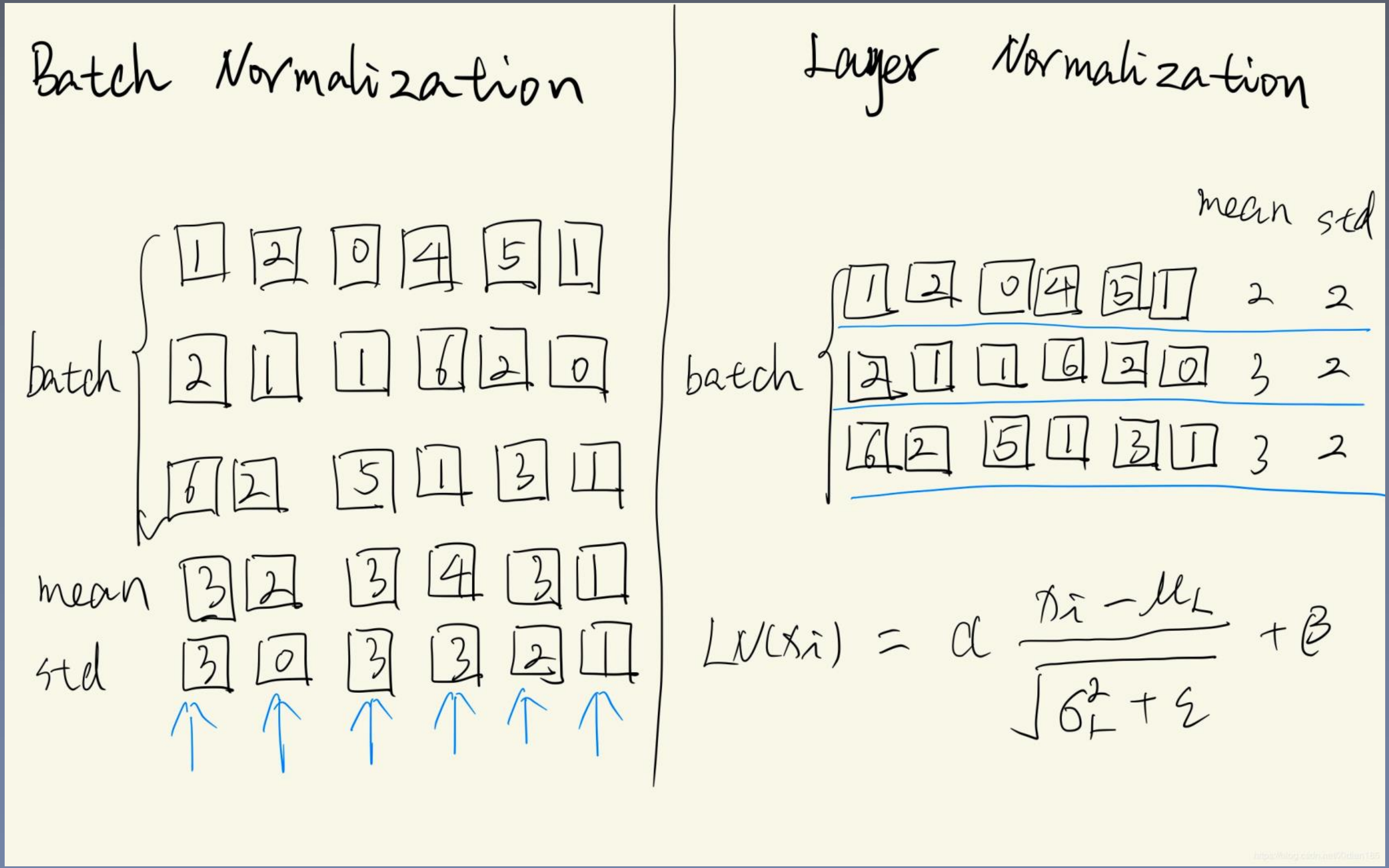
为了防止decoder的时候看到“未来的信息”

2、Padding Mask

attension时处理pad时为0的值

Layer Normalization

Training Details



深度之眼
deepshare.net

实验设置和结果分析

Experiment results

实验设置

Experimental setup

硬件：8块 NVIDIA P100 GPU

优化器：Adam，$\beta_1 = 0.9$, $\beta_2 = 0.98$ and _x000F_ = 10^{-9} .

学习率：

$$lrate = d^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$$

正则化：Dropout、Label Smoothing

实验结果及分析

Results and Discussion

深度之眼
deepshare.net

English-to-German：
比现有最好模型的bleu高出2个点。

English-to-French：
bleu值达到41.0，比单个模型都要高，并且时间上缩减了1/4

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [15]	23.75			
Deep-Att + PosUnk [32]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [31]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [8]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [26]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [32]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [31]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [8]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	$\mathbf{3.3 \cdot 10^{18}}$	
Transformer (big)	**28.4**	**41.0**	$2.3 \cdot 10^{19}$	

深度之眼
deepshare.net

论文总结

论文总结

Summary of the paper

A 关键点

- Self Attention和Soft Attention区别
- Scaled Dot-Product Attention原理
- MultiHead Attention实现

B 小细节

- Mask机制
- Layer Normalization
- 加法attention和dot attention区别

论文总结

Summary of the paper

C

启发点

- 在进行attention机制的时候，对于padd为0的位置可以mask掉
- 可以在模型中添加残差网络结构和layer normalization提高模型效果
- 模型创新的时候，可以添加self attention结构

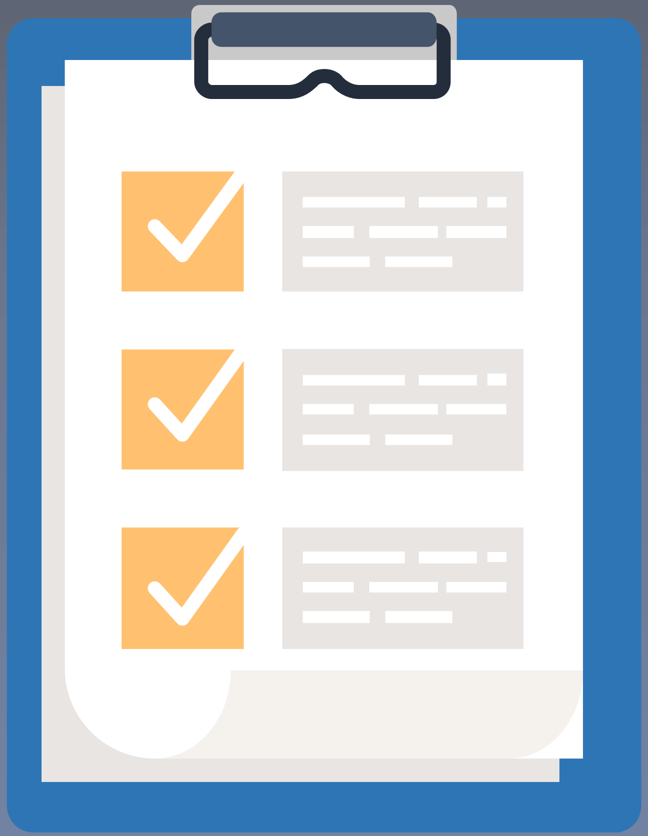
深度之眼
deepshare.net

本课回顾及下节预告

Review in the lesson and Preview of next lesson

本课回顾

Review in the lesson



01 Transformer总体结构

讲解Transformer网络的构成，为啥encoder和decoder组件里面会有6个组成部分，encoder和decoder差别

02 Self Attention结构

Self Attention结构是由Scaled Dot-Product Attention和Multi-head Attention组成，为什么可以实现并行。

03 实验设置及结果分析

网络超参数设置，学习率，batchsize等
实验结果分析对比

04 论文总结

总结论文中创新点、关键点及启发点

下节预告

Preview of next lesson

01 搭建Transformer网络代码介绍

02 介绍Self Attention实现

03 基于翻译数据集训练Transformer模型

下节课前准备

Preview of next lesson

- 再次阅读Transformer论文
- 熟悉Transformer模型结构及数据预处理方式
- 配置PyTorch开发环境
- 下载Transformer代码
- https://github.com/leviswind/pytorch-transformer

结 语

循循而进，欲速则不达也。

深度之眼
deepshare.net

联系我们：
电话：18001992849
邮箱：service@deepshare.net
Q Q： 2677693114

公众号

客服微信