



中国科学技术大学
University of Science and Technology of China

《人工智能数学原理与算法》

第5章 Transformer

5.3 Transformer的典型应用 与跨模态能力

宋彦

songyan@ustc.edu.cn

01

重新理解Token

02

Transformer的典型应用

03

Transformer的跨模态推理

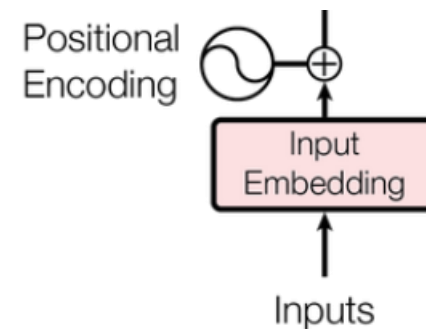
本课重点

□ Transformer有两部分输入

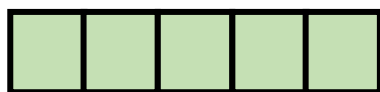
□ Token向量

□ 位置编码

□ 最终输入等于原本token向量和位置编码的和

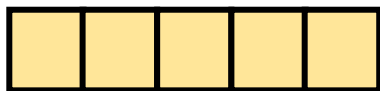


输入向量



=

位置编码



+

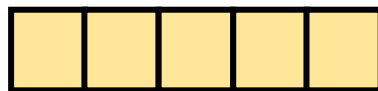
token向量



I



=



+



brought



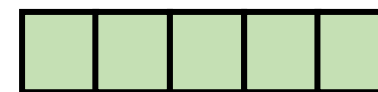
=



+



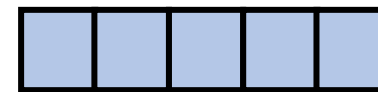
the



=



+



book

- Transformer 的输入是一个向量序列（一个矩阵）
- 这些向量对应的是基本的输入单元 —— Token
 - 上节课讲过Token的向量应具有的特点
- Token是什么？
 - 文本中的一个基本单元（例如，单词、子词等）
 - 图像的一块
 - 基本概念/特征（例如，文本“苹果”是多个概念的组合，例如“红色”、“水果”、“球体”等）
 -

文本的Tokens

□ 词Tokens

例如: " Apple is red" → ["Apple", "is", "red"]

优点: 简单直观

缺点: 词汇量大; 难以理解生词

□ 字Tokens

例子: "Apple" → ["A", "p", "p", "l", "e"]

优点: 可处理任何文本, 包括拼写错误

缺点: 序列较长; 语义含义较少

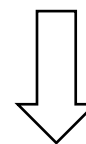
□ 子词Tokens

例子: "unhappiness" → ["un", "happi", "ness"]

优点: 平衡词汇量和灵活性

缺点: 需要标记化算法 (例如 WordPiece、BPE)

"This is an input text."



TOKENIZATION

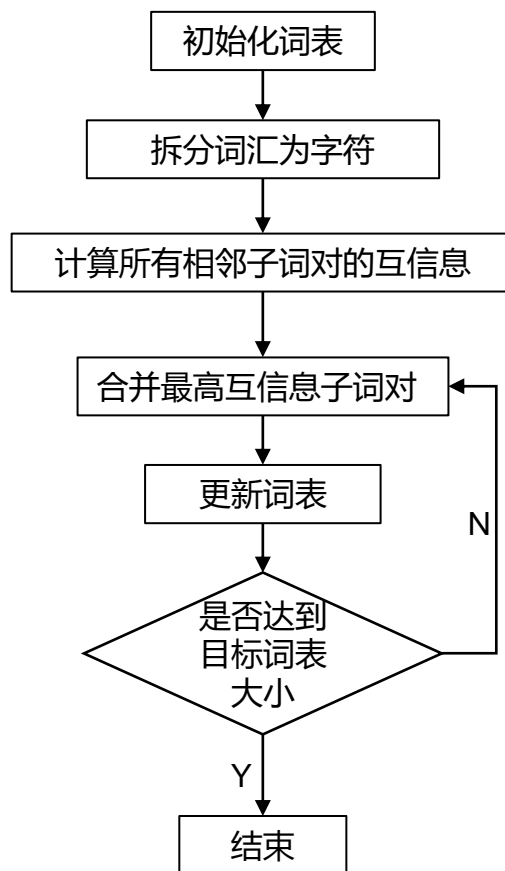
[CLS]	This	is	an	input	.	[SEP]
101	2023	2003	1037	7953	1012	102

切分方法

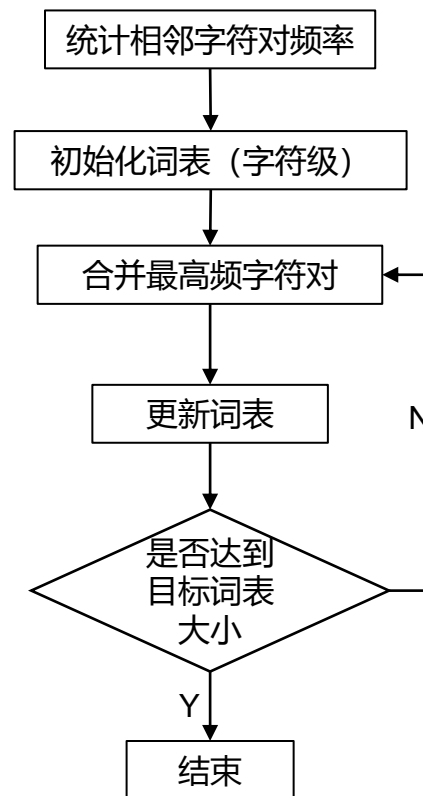
□ 两种典型的序列切分方法

- WordPiece 和 BPE
- 二者均基于统计方法，衡量相邻字符的关联强弱

□ 接下来以BPE为例，介绍Token的切分方法



WordPiece



BPE

字节对编码 (Byte-Pair Encoding, BPE) 切分

□ BPE 是一种将输入序列分割成子词的切分算法

例如: hugs -> hug + s

□ 训练

输入: 语料库 D, 预定义词汇量 N

输出: 一组合并规则 R

从字母表开始, 不断添加新的子词, 直到词汇量达到预定义的大小 N

□ 分词

将单词拆分成字母, 然后应用学习到的合并规则来获取子单词

- 步骤 1: 从字母表 V 开始
- 步骤 2: 对于 V 中的每一对token, 即 w_1 和 w_2 , 获取它们在语料库中的数量
- 步骤 3: 找到计数最高的token对, 将该对添加到词汇表 V 中, 同时向规则集 R 添加合并规则 “ $w_1 w_2 \rightarrow w_1 w_2$ ”
- 步骤 4: 重复步骤 2 和步骤 3, 直到词汇表大小达到 N

□ 语料库:

("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

□ 将语料库拆分成字母以查找字母表

("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12),
("b" "u" "n", 4), ("h" "u" "g" "s", 5)

□ 初始词汇（字母表）是

["b", "g", "h", "n", "p", "s", "u"]

□ 语料库

("hug", 10), ("pug", 5), ("pun", 12), ("bun", 4), ("hugs", 5)

□ 初始词汇（字母表）是

["b", "g", "h", "n", "p", "s", "u"]

□ 获取所有标记对的计数

□ 计数最高的是 (“u”, “g”) 对（出现 20 次）

□ 第一个新标记是 (“u”, “g”) -> (“ug”), 合并规则 “u” “g” -> “ug” 被添加到规则集中

BPE训练

- 第一个新token是 “ug”
- 新词汇表: [“b”, “g”, “h”, “n”, “p”, “s”, “u”, “ug”]
- 第一条新的合并规则是 “ u” “g” -> “ug”
- 规则集合: {“u” “g” -> “ug”}
- 合并语料库中的对

原始语料库

("h" "u" "g", 10), ("p" "u" "g", 5), ("p" "u" "n", 12),
("b" "u" "n", 4), ("h" "u" "g" "s", 5)



新语料库

("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12),
("b" "u" "n", 4), ("h" "ug" "s", 5)

BPE训练

□ 语料库:

("h" "ug", 10), ("p" "ug", 5), ("p" "u" "n", 12),
("b" "u" "n", 4), ("h" "ug" "s", 5)

□ 词汇表: ["b", "g", "h", "n", "p", "s", "u", "ug"]

□ 获取 token 对的数量, 并找出最高的是 ("u", "n") -> "in"

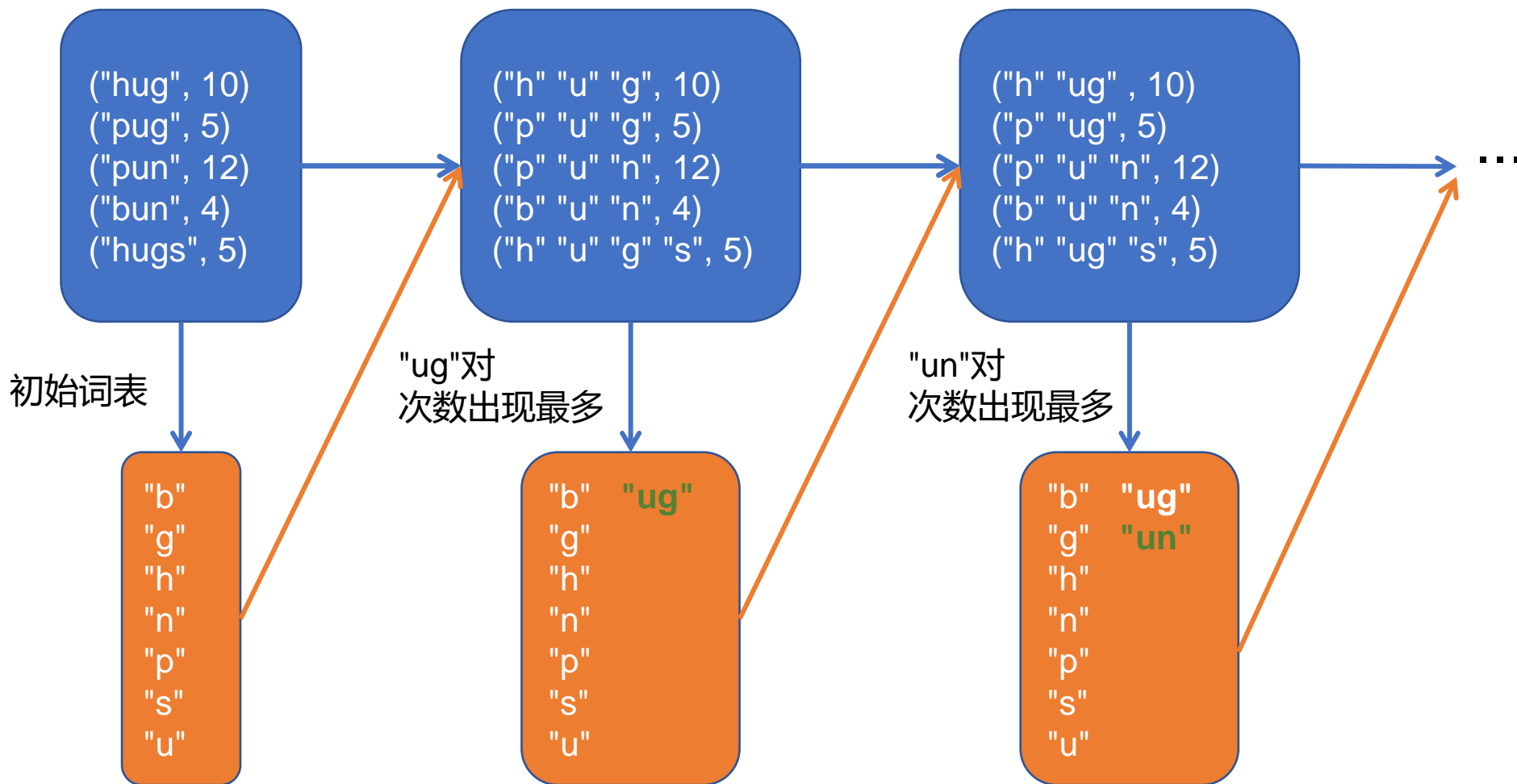
□ 新的词汇、语料库和规则集是Vocabulary: ["b", "g", "h", "n", "p", "s", "u", "ug", "un"]

语料库: ("h" "ug", 10), ("p" "ug", 5), ("p" "un", 12), ("b" "un", 4), ("h" "ug" "s", 5)

合并规则集: {"u" "g" -> "ug", "u", "n" -> "un"}

□ 重复该过程, 直到词汇量达到预定义值

□ 整体流程总结



BPE切分算法

- 步骤 1: 对于一个单词, 将单词拆分成单个字符
- 步骤 2: 按顺序将学习到的合并规则应用于这些拆分
 - 如果剩余部分在词汇表中找不到, 则将其标记为 “[UNK]”。

- 例如:

词表: ["b", "g", "h", "n", "p", "s", "u", "ug", "un", "hug"]

合并规则:

"u" "g" -> "ug"

"u" "n" -> "un"

"h" "ug" -> "hug"

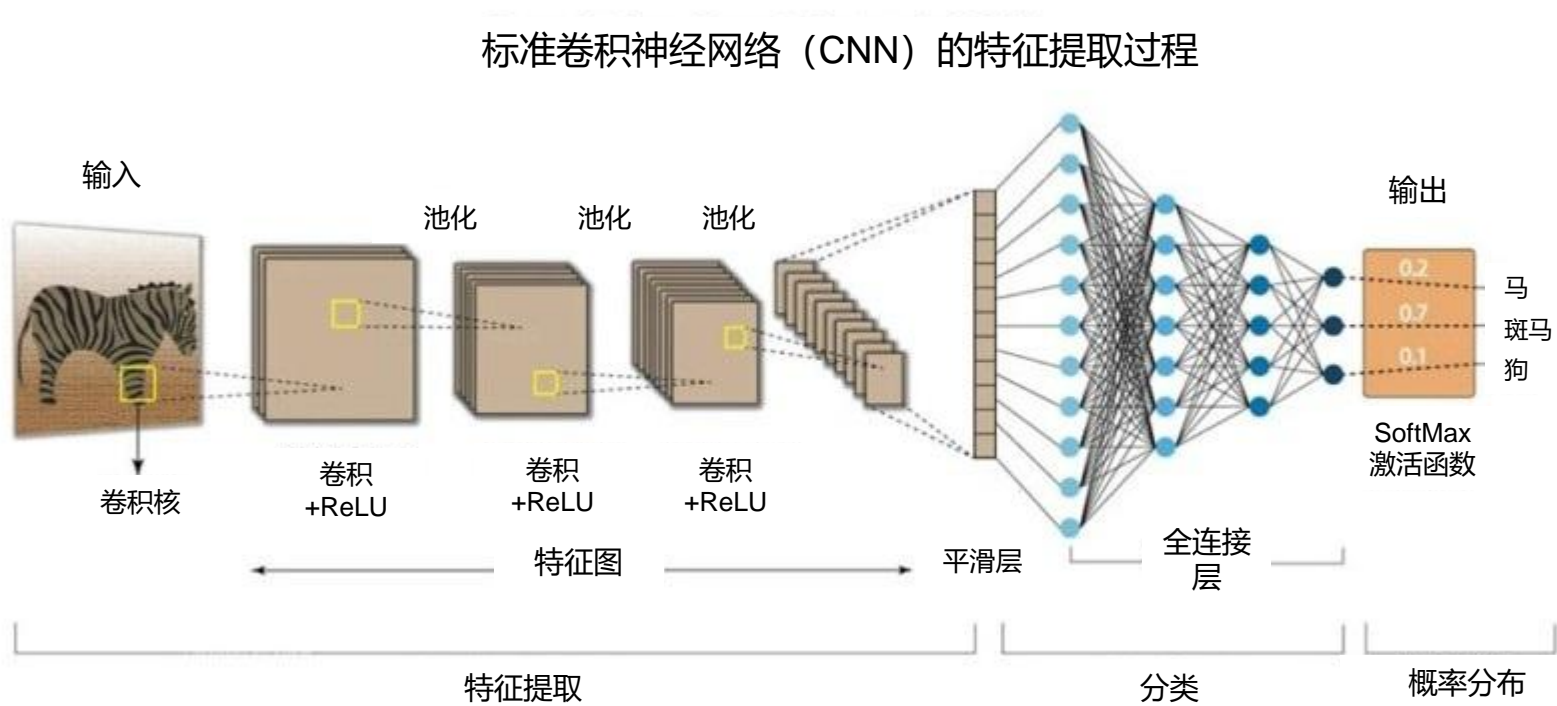
“bug” -> “b” + “u” + “g” -> “b” + “ug”

“mug” -> “m” + “u” + “g” -> “m” + “ug” -> “[UNK]” + “ug”

文本的Tokens

- **文本中的Tokens并不一定对应词，而是可以有不同的选择，例如subword**
 - 中文中的典型Token是字
 - 英文中的典型Token是subword
- **存在多种Tokenization的方法，目前主流方法是基于统计信息的**
 - BPE是一种典型的统计驱动的切分方法
 - 本质上是一种“最有效的信息压缩方法”

□ 传统图像编码方法



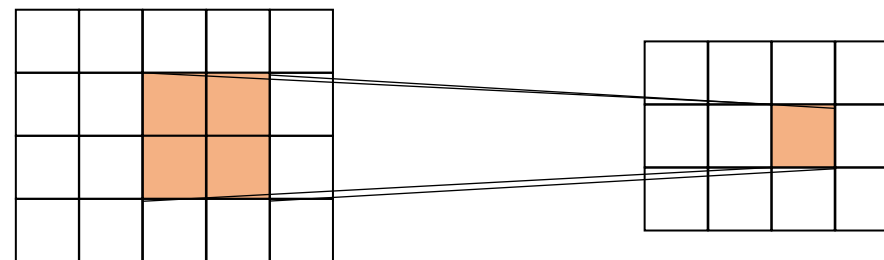
图像中的Tokens

□ 传统图像编码方法的特点（以卷积为例）

- 使用卷积核对二维图像进行压缩
- 输入向量呈二维分布
- 图像编码中缺少位置信息
- ...

□ 这样直接的图像表征方式不适合

Transformer，无法直接使用



图像中的Tokens

□ 因此，引入图像块（Patch）的概念：

定义：将输入图像划分为固定大小、不重叠的小块（称为块）

目的：将二维图像数据转换为与 Transformer 架构兼容的序列格式



图像中的Tokens

□ 块切割的过程

□ 输入图像尺寸: $H \times W \times C$

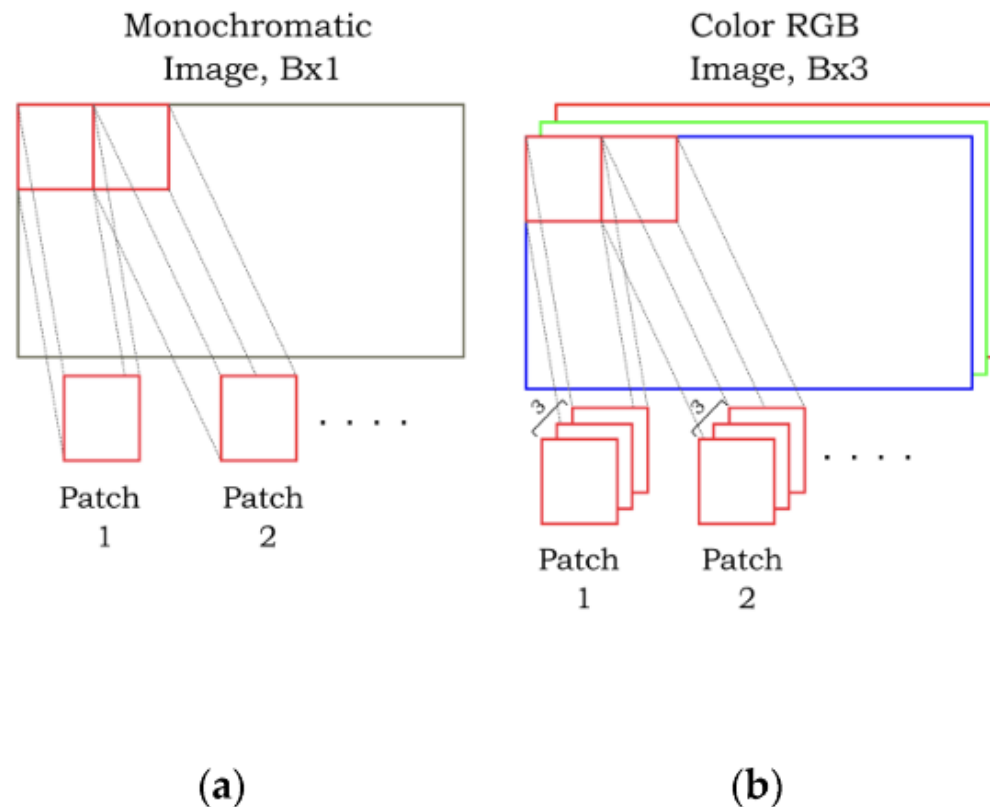
(H: 高; W: 宽; C: 通道数)

□ 块大小: 每个块的大小是 $P \times P$

□ 块的个数: $N = \frac{H}{P} \times \frac{W}{P}$

□ 右图(a)展示单通道patch切分方法
(例如灰度图)

□ 右图(b)展示传统三通道patch切分方法
(例如RGB三通道)



图像中的Tokens

□ 块转换为表征向量的过程

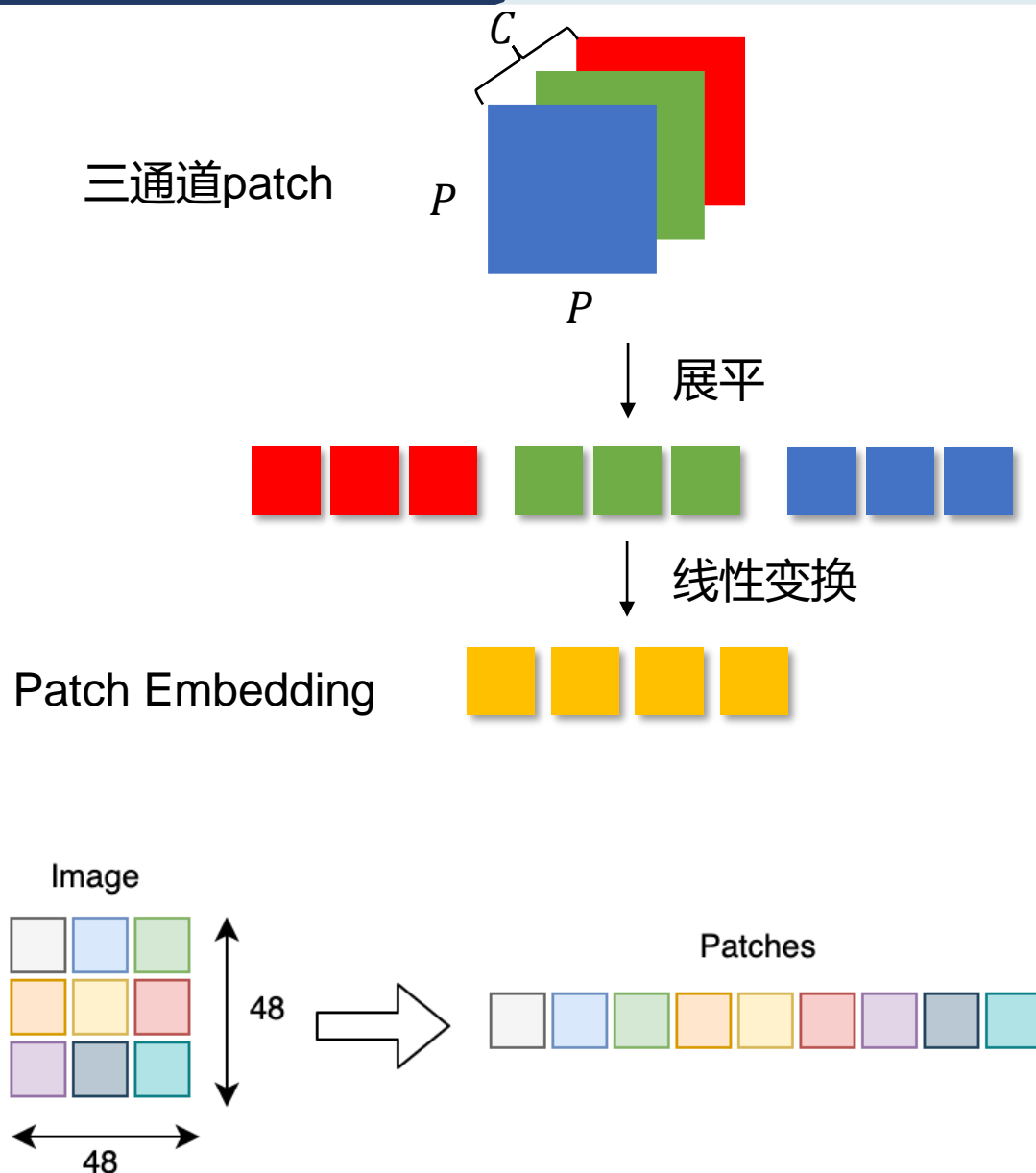
□ 每个patch是一个 $P \times P \times C$ 的张量

□ 将每个patch展平为一个向量：

$$x_i \in \mathbb{R}^{P^2 \cdot C}, i = 1, 2, \dots, N$$

□ 使用一个全连接层将 x_i 映射为patch embedding（相当于文本的token embedding）

□ 将所有 Patch Embedding堆叠，
得到Transformer输入

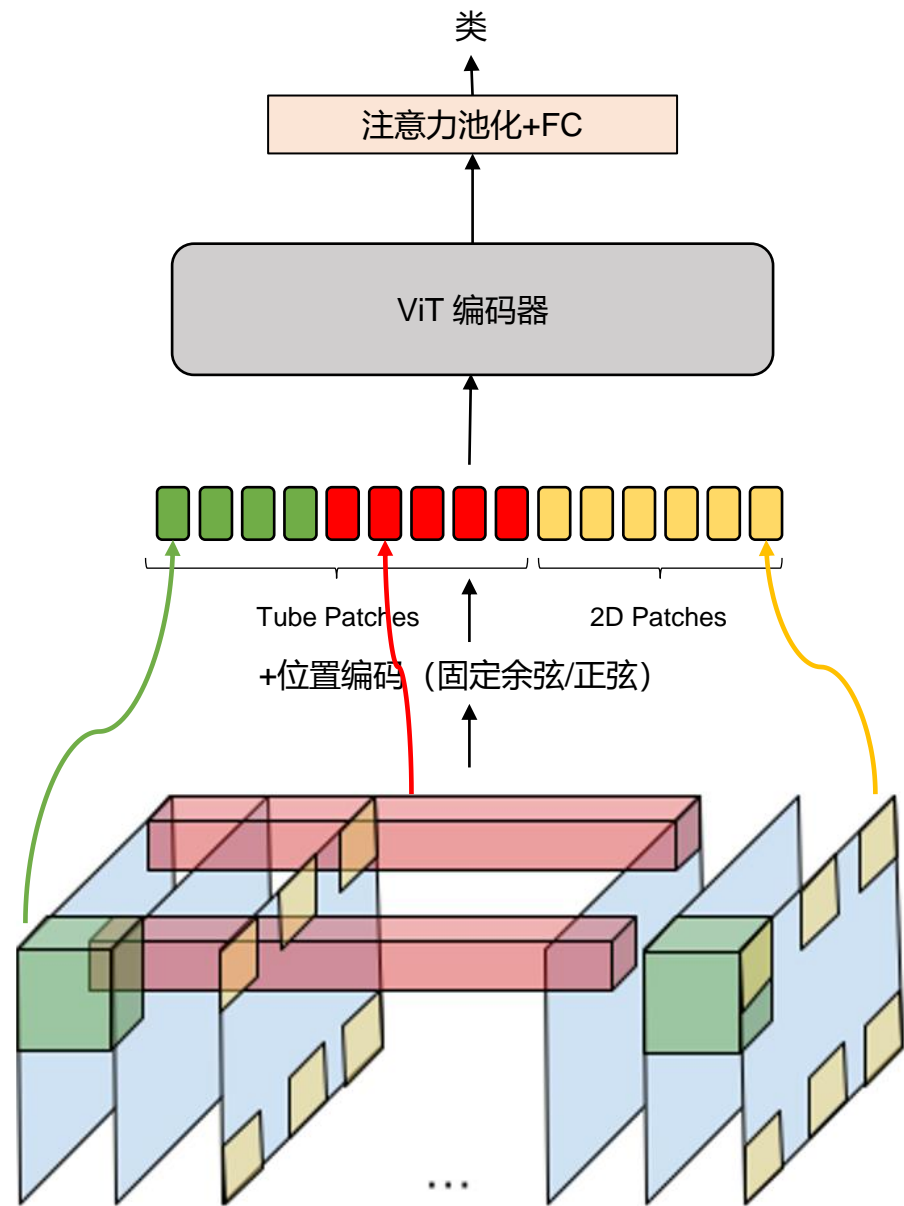


图像的Token总结

- 传统的图像编码方式无法直接应用于Transformer
- 通过将图像切分为小块（Patch），每个小块变换为一个向量，再将所有小块的向量堆叠，实现图像小块的序列化，从而使其可以被Transformer处理
- 与文本类似，序列化后的patch也需要位置编码明确其位置信息

视频中的Tokens

- 视频是图像在时间维度的展开，更为复杂且具备时变特性
- 为了更好地编码视频，除了传统的每张图片的2D Patch，还有Tube Patch（包括短距离与长距离的）



其他种类的Tokens

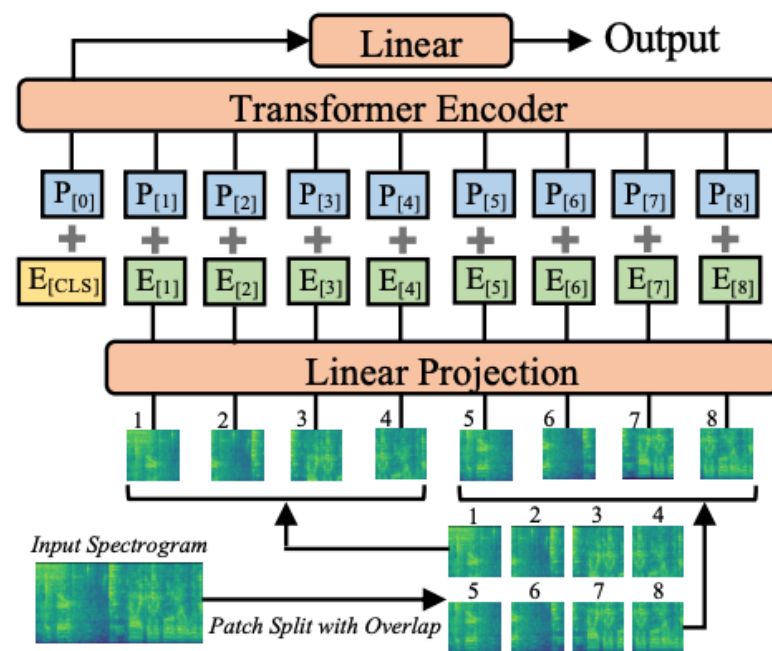
□ DNA序列

- DNA序列被切分为小单元
- 小的碱基片段被输入到Transformer

Sequence 1	ACAATAATAATAAACGG	
Sequence 2	CAATAATAATAAACGG	
	Tokens	Token IDs
<u>k-mer</u>	ACAATA ATAATA ATAACG G	[520, 264, 271, 4103]
	CAATAA TAATAA TAACGG	[2068, 1044, 1075]
<u>BPE</u>	A CAA TAATAATAATAA CGG	[5, 27, 1769, 72]
	CAA TAATAATAATAA CGG	[27, 1769, 72]

□ 音频Patch

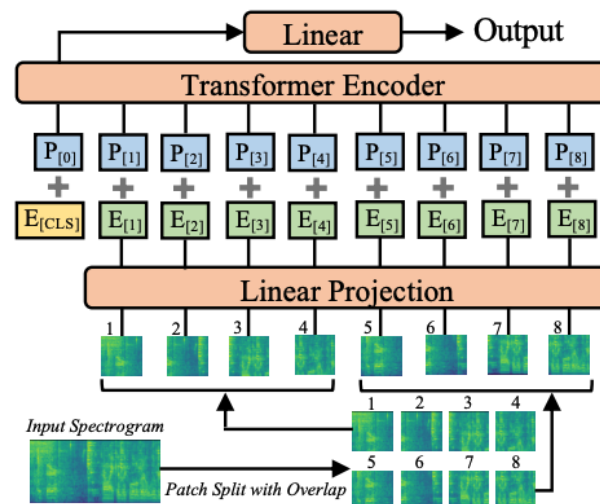
- 音频被表达为频谱图
- 频谱图被按照图像处理



- Token是Transformer可处理的基本可向量化输入单元
- Token不局限于文本，可以根据具体处理的数据形式有多种形态



Sequence 1	ACAATAATAATAACGG	
Sequence 2	CAATAATAATAACGG	
	Tokens	Token IDs
<u>k-mer</u>	ACAATA ATAATA ATAACG G	[520, 264, 271, 4103]
	CAATAA TAATAA TAACGG	[2068, 1044, 1075]
<u>BPE</u>	A CAA TAATAATAATAA CGG	[5, 27, 1769, 72]
	CAA TAATAATAATAA CGG	[27, 1769, 72]



01

重新理解Token

02

Transformer的典型应用

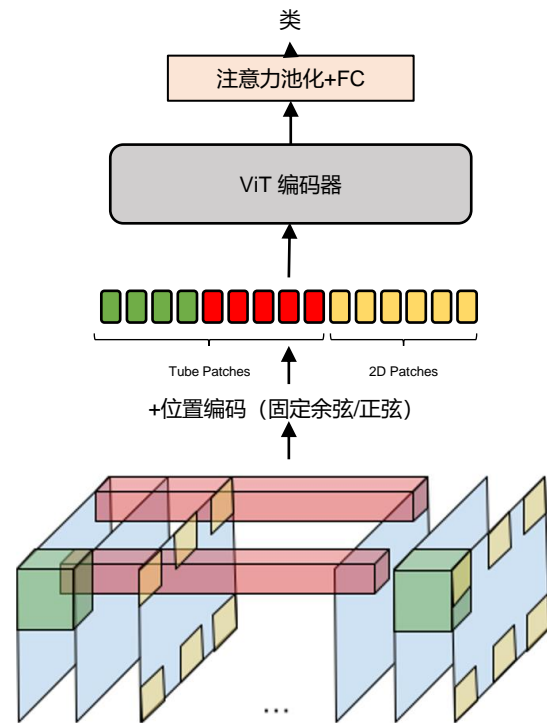
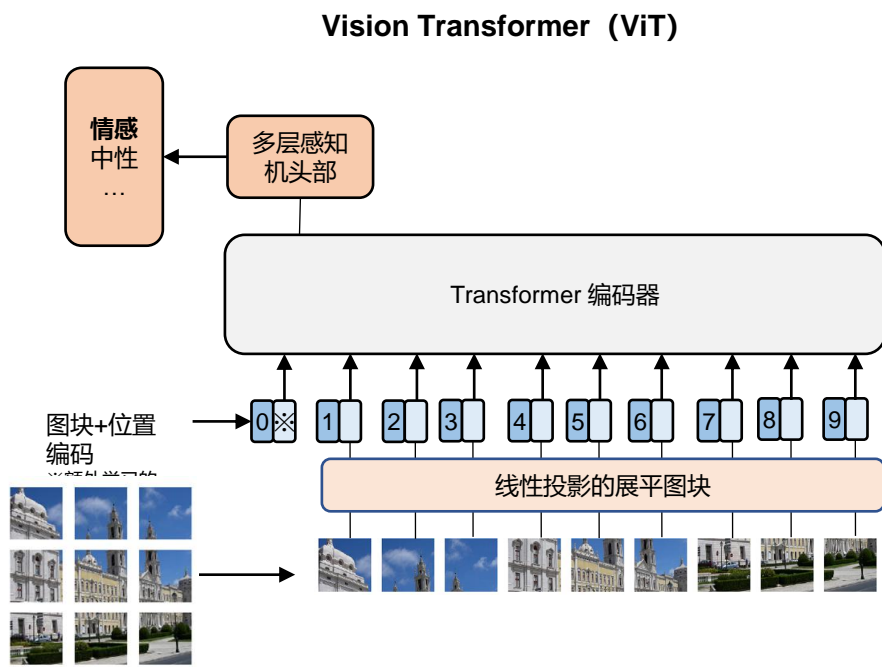
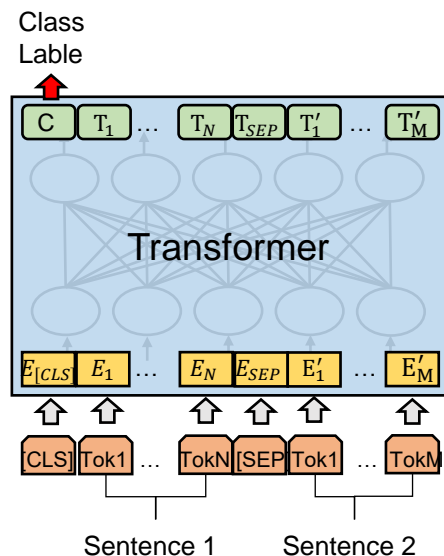
03

Transformer的跨模态推理

本课重点

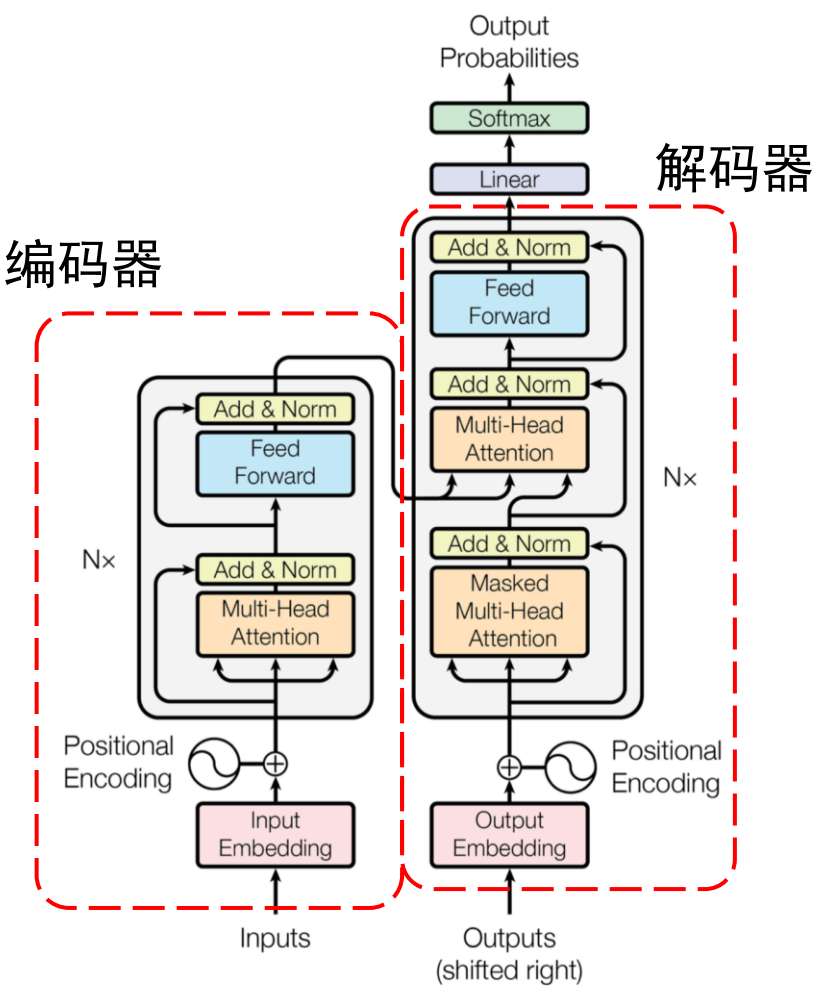
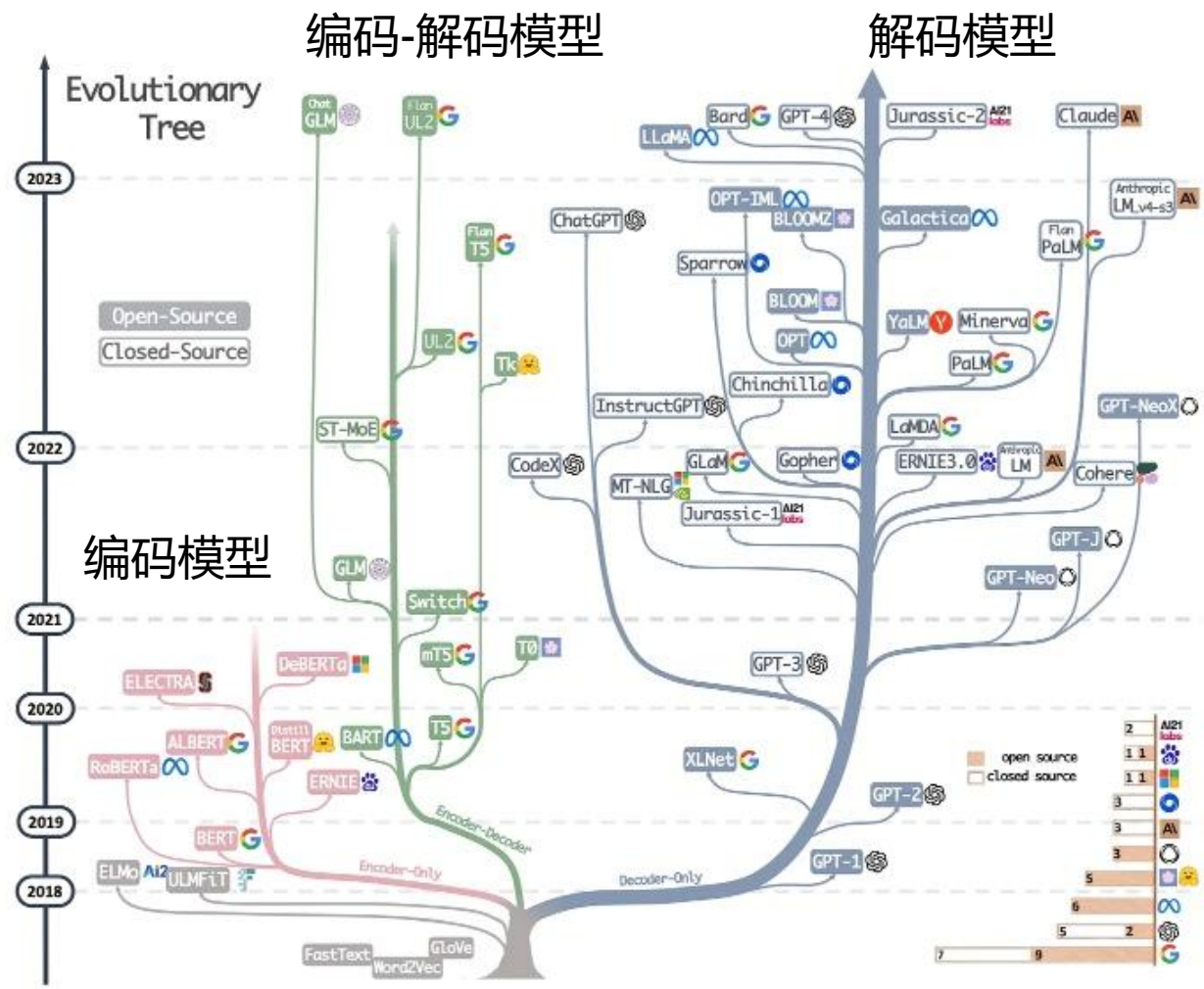
Transformer的应用

□ Transformer优秀的建模机制和高效的特征抽取方式，使其在各个领域都有高效应用（文本、图像、语音等）



Transformer的广泛应用

- Transformer是目前通用表征模型的基本架构



Transformer的应用

- 要在不同应用里面用Transformer，需要考虑数据应用的形态
- 需要根据不同的任务选择合适的Transformer架构
 - 适合Transformer编码器的任务：内容理解等
 - 适合Transformer解码器的任务：内容生成等
- 以文本和图像为例，介绍Transformer的典型应用
 - Transformer在文本处理中的应用（例如情感分析、机器翻译）
 - Transformer在图像处理中的应用（例如图片分类）

Transformer在文本处理中的应用

□ 一般应用流程：

- 将文本切分为Token序列
- 使用Transformer对文本进行编码
- 使用编码的表征进行预测

□ 根据架构不同，Transformer一般有三种应用方法，分别处理不同任务

- 只使用Transformer编码器：一般用于理解任务（例如文本分类或序列标注）
- 使用Transformer编码器+解码器：一般用于生成任务
- 使用Transformer解码器：一般用于生成任务

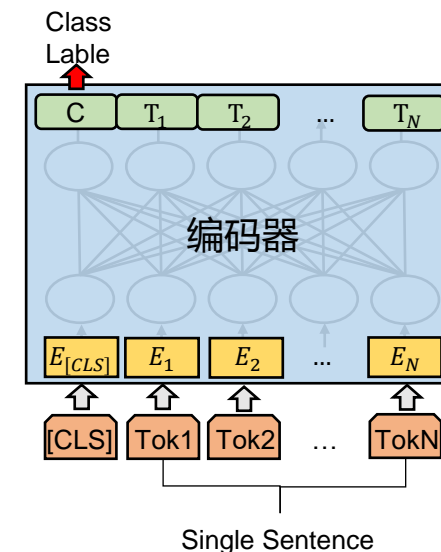
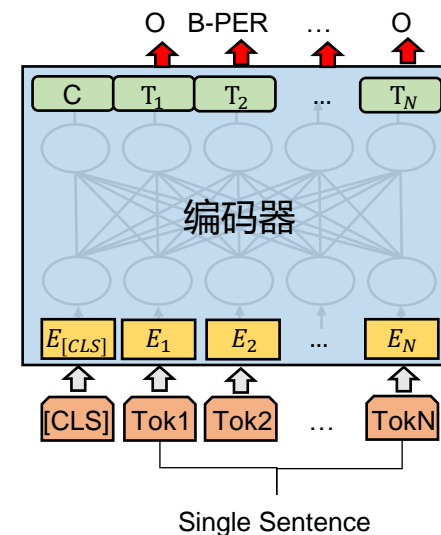
Transformer编码器在文本处理中的应用

□ Transformer编码器在文本处理中的应用流程

- 将文本进行预处理（例如加入句子开始和结束的标记）
- 将文本切分为Token，得到token序列
- 使用Transformer处理Token序列，得到每个Token的表征向量

$$\mathbf{h}_1 \cdots \mathbf{h}_N = \text{Transformer}_{\text{encoder}}(x_1 \cdots x_N)$$

- 将表征进行进一步处理，预测任务标签，主要分两种情况：
 - 序列标注任务
 - 文本分类任务



Transformer编码器应用于序列标注任务

□ 序列标注任务

- 对输入序列中每个元素逐一预测其对应标签
- 即对输入 $x_1 \cdots x_N$, 需要预测对应的 $\hat{y}_1 \cdots \hat{y}_N$, 其中第 n 个输入单元对应的 x_n 对应的输出是 \hat{y}_n

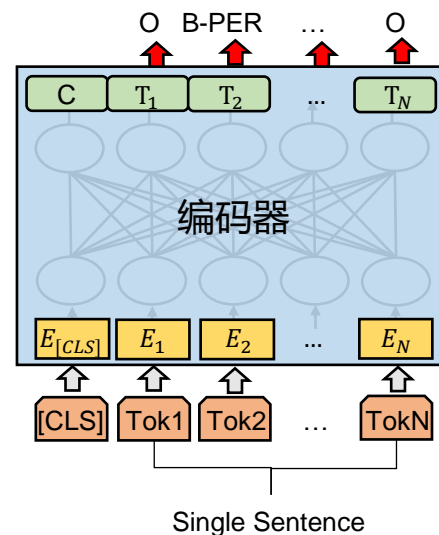
□ Transformer编码器的应用

- 使用Transformer编码器处理输入序列, 得到每个输入的特征向量

$$\mathbf{h}_1 \cdots \mathbf{h}_N = \text{Transformer}_{\text{encoder}}(x_1 \cdots x_N)$$

- 预测每个输入对应的标签

$$\hat{y}_n = \text{softmax}(\mathbf{W}\mathbf{h}_n + \mathbf{b})$$



Transformer编码器的序列标注示例：命名实体识别

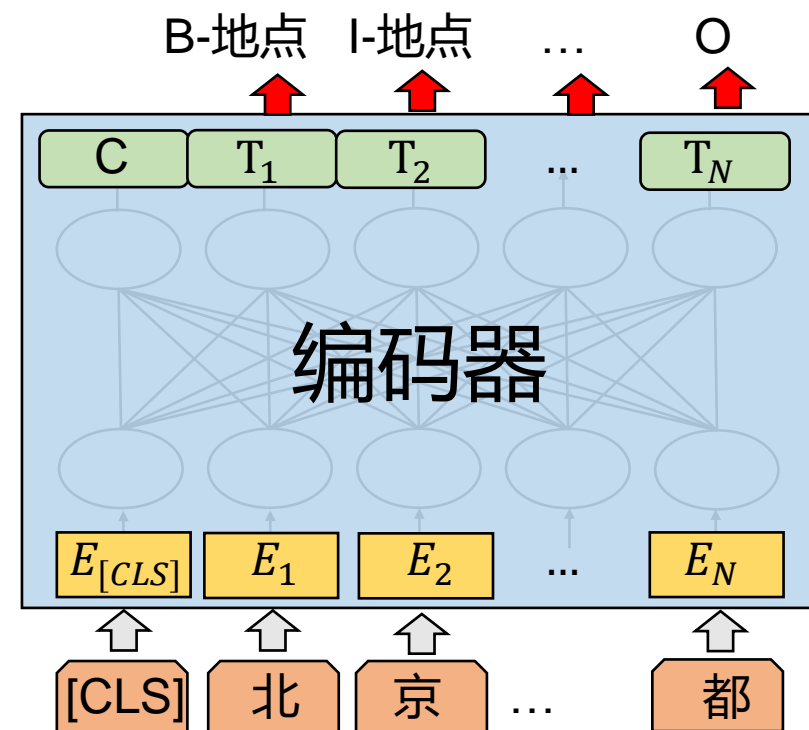
□ 命名实体识别任务：

- 抽取输入中的命名实体并判断其类别标签（例如人物、地点、组织等）
- 例子：句子“北京是我国的首都”中，“北京”是“地点”

□ 命名实体识别是典型的序列标注任务

- 给输入中的 token 分配一个标签，表示其是否是实体的一部分
 - O：不是实体的一部分；
 - B-[类别]：是某个类别实体的开始词（例如“B-地点”）
 - I-[类别]：是某个类别实体的非开始词（例如“I-地点”）
- 例如：“北京是我国的首都”中，“北”、“京”的标签依次是“B-地点”、“I-地点”，其他字的标签为“O”

□ 右图展示编码器用于实体识别的过程



Transformer编码器应用于文本分类任务

□ 文本分类任务

- 对整段文本进行类别预测
- 即对输入 $x_1 \cdots x_N$ ，需要预测一个类别标签 \hat{y}

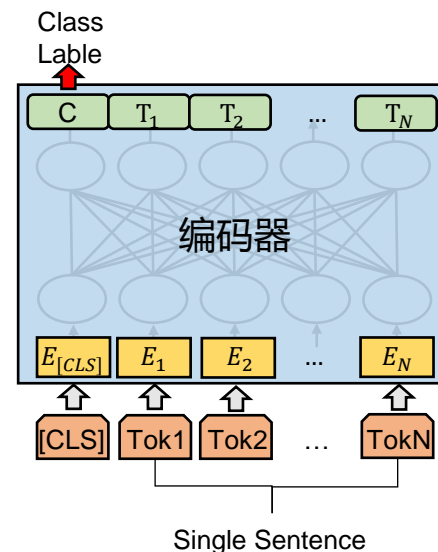
□ Transformer编码器的应用

- 对文本进行预处理，将文本切分为Token序列，并在前后加上表示文本开始与结束的特殊字符 “[CLS]” 和 “[SEP]”，其中 “[CLS]” 记为 $x_{[CLS]}$
- 使用Transformer处理Token序列，得到每个Token的表征向量

$$\mathbf{h}_{[CLS]}, \mathbf{h}_1 \cdots \mathbf{h}_N = \text{Transformer}_{\text{encoder}}(x_{[CLS]}, x_1 \cdots x_N)$$

- 使用全连接层以及softmax函数预测整个文本对应的标签

$$\hat{y} = \text{softmax}(\mathbf{W}\mathbf{h}_{[CLS]} + \mathbf{b})$$



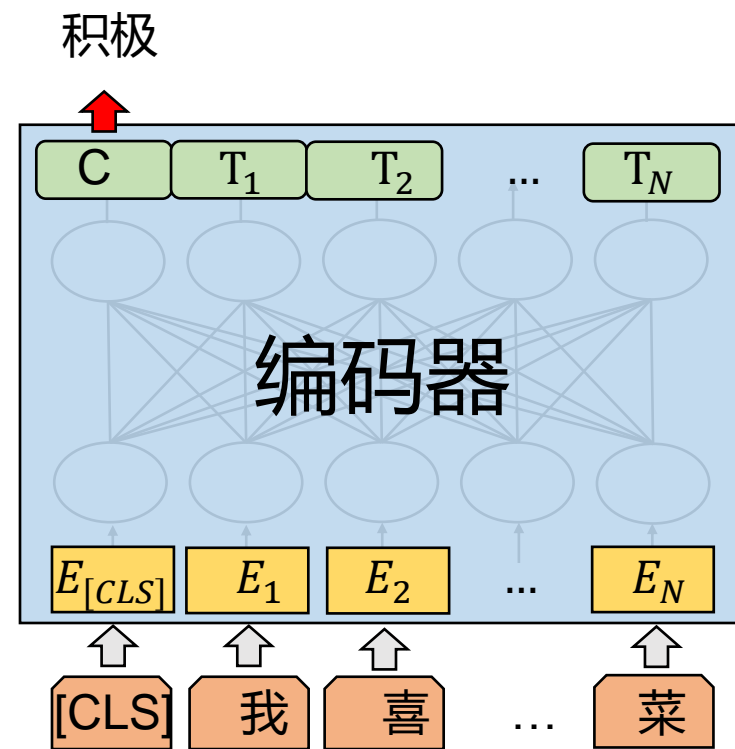
Transformer编码器的文本分类示例：情感分析

□ 情感分析任务：

- 对文本进行分类，判断其情感类别（积极、中性、消极）
- 例子：“我喜欢这道菜”的情感类别是“积极”

□ Transformer编码器的处理流程

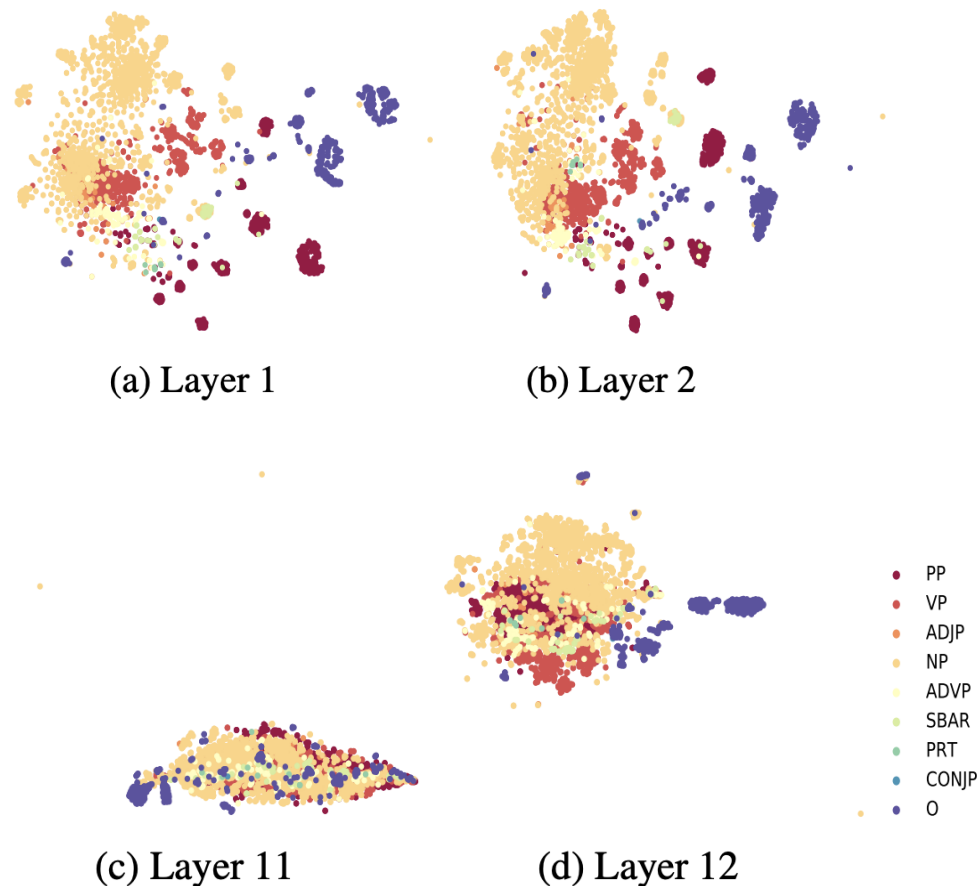
- 使用Transformer编码器对输入序列进行编码
- 将Transformer最后一层得到的，特殊字符 “[CLS]” 对应的表征输入全连接层，并通过softmax激活函数得到情感概率分布，选取概率最高的情感标签作为模型输出



Transformer编码器在文本处理中的应用

□ Transformer编码器学到了什么？

- 从不同层提取文本片段的特征
- 运行聚类算法并可视化结果
- 每个节点对应一个文本片段，颜色表示词性（词类）标签
- Transformer在较低层主要捕捉短语级信息，而在更高层这些信息逐渐被稀释。



Transformer编码器在文本处理中的应用

□ Transformer编码器学到了什么?

Layer	SentLen (Surface)	WC (Surface)	TreeDepth (Syntactic)	TopConst (Syntactic)	BShift (Syntactic)	Tense (Semantic)	SubjNum (Semantic)	ObjNum (Semantic)	SOMO (Semantic)	CoordInv (Semantic)
1	93.9 (2.0)	24.9 (24.8)	35.9 (6.1)	63.6 (9.0)	50.3 (0.3)	82.2 (18.4)	77.6 (10.2)	76.7 (26.3)	49.9 (-0.1)	53.9 (3.9)
2	95.9 (3.4)	65.0 (64.8)	40.6 (11.3)	71.3 (16.1)	55.8 (5.8)	85.9 (23.5)	82.5 (15.3)	80.6 (17.1)	53.8 (4.4)	58.5 (8.5)
3	96.2 (3.9)	66.5 (66.0)	39.7 (10.4)	71.5 (18.5)	64.9 (14.9)	86.6 (23.8)	82.0 (14.6)	80.3 (16.6)	55.8 (5.9)	59.3 (9.3)
4	94.2 (2.3)	69.8 (69.6)	39.4 (10.8)	71.3 (18.3)	74.4 (24.5)	87.6 (25.2)	81.9 (15.0)	81.4 (19.1)	59.0 (8.5)	58.1 (8.1)
5	92.0 (0.5)	69.2 (69.0)	40.6 (11.8)	81.3 (30.8)	81.4 (31.4)	89.5 (26.7)	85.8 (19.4)	81.2 (18.6)	60.2 (10.3)	64.1 (14.1)
6	88.4 (-3.0)	63.5 (63.4)	41.3 (13.0)	83.3 (36.6)	82.9 (32.9)	89.8 (27.6)	88.1 (21.9)	82.0 (20.1)	60.7 (10.2)	71.1 (21.2)
7	83.7 (-7.7)	56.9 (56.7)	40.1 (12.0)	84.1 (39.5)	83.0 (32.9)	89.9 (27.5)	87.4 (22.2)	82.2 (21.1)	61.6 (11.7)	74.8 (24.9)
8	82.9 (-8.1)	51.1 (51.0)	39.2 (10.3)	84.0 (39.5)	83.9 (33.9)	89.9 (27.6)	87.5 (22.2)	81.2 (19.7)	62.1 (12.2)	76.4 (26.4)
9	80.1 (-11.1)	47.9 (47.8)	38.5 (10.8)	83.1 (39.8)	87.0 (37.1)	90.0 (28.0)	87.6 (22.9)	81.8 (20.5)	63.4 (13.4)	78.7 (28.9)
10	77.0 (-14.0)	43.4 (43.2)	38.1 (9.9)	81.7 (39.8)	86.7 (36.7)	89.7 (27.6)	87.1 (22.6)	80.5 (19.9)	63.3 (12.7)	78.4 (28.1)
11	73.9 (-17.0)	42.8 (42.7)	36.3 (7.9)	80.3 (39.1)	86.8 (36.8)	89.9 (27.8)	85.7 (21.9)	78.9 (18.6)	64.4 (14.5)	77.6 (27.9)
12	69.5 (-21.4)	49.1 (49.0)	34.7 (6.9)	76.5 (37.2)	86.4 (36.4)	89.5 (27.7)	84.0 (20.2)	78.7 (18.4)	65.2 (15.3)	74.9 (25.4)

Jawahar, Ganesh, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? ACL-2019

Transformer编码器与解码器在文本处理中的应用

□ Transformer编码器与解码器在文本处理中的应用流程

- 将文本切分为Token，并使用Transformer处理Token序列，得到每个Token的表征向量

$$\mathbf{h}_1 \cdots \mathbf{h}_N = \text{Transformer}_{\text{encoder}}(x_1 \cdots x_N)$$

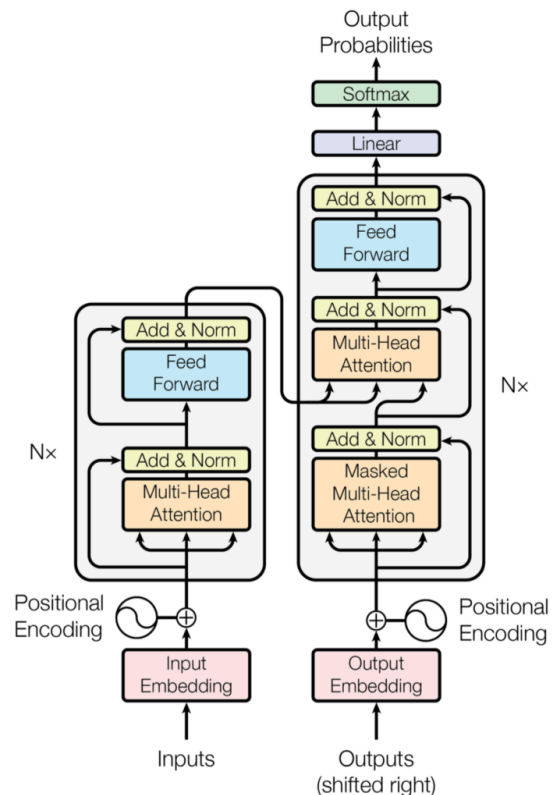
- 在解码过程中， t 时刻，使用Transformer解码器计算输出向量 \mathbf{h}_t

$$\mathbf{h}_t = \text{Transformer}_{\text{decoder}}(\mathbf{h}_1 \cdots \mathbf{h}_N, \hat{y}_1 \cdots \hat{y}_{t-1})$$

- 将输出向量 \mathbf{h}_t 映射到输出的token即 \hat{y}_t

$$\hat{y}_t = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b})$$

- 重复上述解码过程，直至得到终止token（例如<eos>）



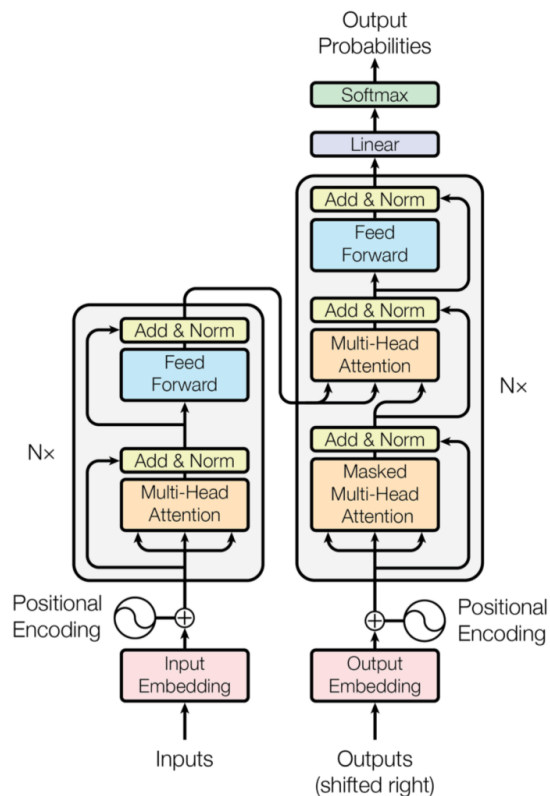
Transformer编码器与解码器在文本处理中的应用

□ 编码器与解码器架构在应用中存在一定局限

- 架构复杂，需要维护独立的编码层与解码层
- 模型拓展难，如增大模型参数量，则需要同时增加编码器与解码器层数
- ...

□ 单纯使用解码器，将原本编码器处理的文本作为解码器端的输入，是一种可行的选择

- 架构更简单，更易于拓展模型规模



Transformer解码器在文本处理中的应用

□ 所有与编码器相关的架构被移除

- 编码器被移除
- 交叉注意力机制被移除

□ Transformer解码器在文本处理中的应用流程

- 在 t 时刻, 使用Transformer解码器计算输出向量 \mathbf{h}_t

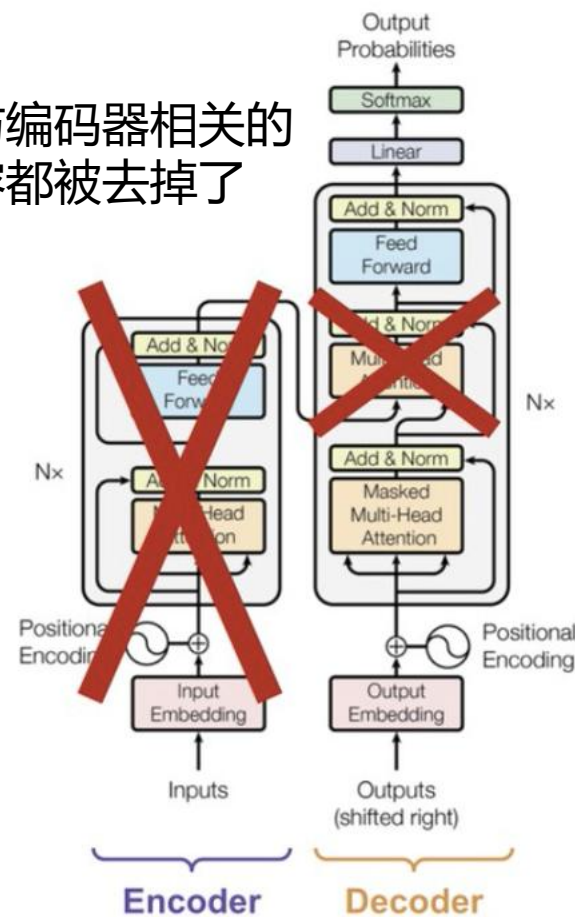
$$\mathbf{h}_t = \text{Transformer}_{\text{decoder}}(x_1 \cdots x_N, \hat{y}_1 \cdots \hat{y}_{t-1})$$

- 将输出向量 \mathbf{h}_t 映射到输出的token即 \hat{y}_t

$$\hat{y}_t = \text{softmax}(\mathbf{W}\mathbf{h}_t + \mathbf{b})$$

- 重复上述解码过程, 直至得到终止token (例如<eos>)

所有与编码器相关的
内容都被去掉了



Transformer解码器在文本处理中的应用

□ 解码器不但可以用于预测下一个Token，也可以用于不同的分类任务

□ 应用流程为

□ 将文本切分为Token，并使用Transformer处理Token序列，输出向量 \mathbf{h}

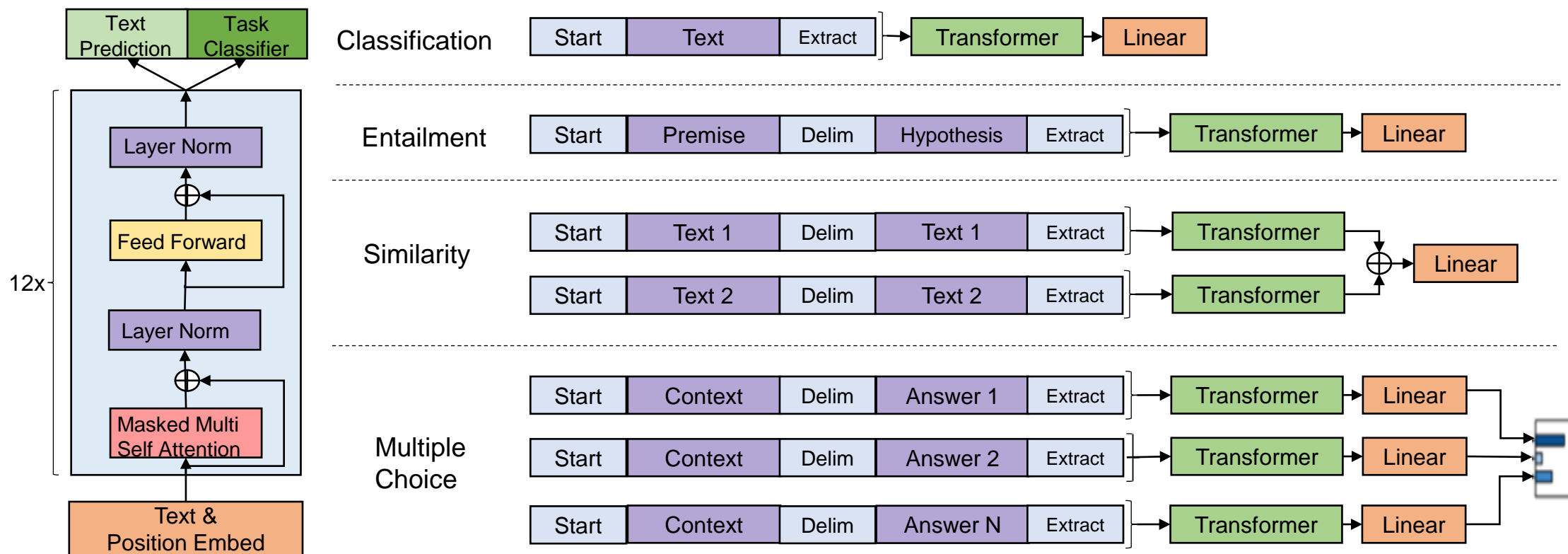
$$\mathbf{h} = \text{Transformer}_{\text{decoder}}(x_1 \cdots x_N)$$

□ 使用全连接层与softmax激活函数，使用输出向量 \mathbf{h} 预测类别标签

$$\hat{y} = \text{softmax}(\mathbf{W}\mathbf{h} + \mathbf{b})$$

Transformer解码器在文本处理中的应用

□ 解码器用于不同分类任务的操作展示



□ 从上述流程看，Transformer解码器的应用流程与Transformer编码器十分相似，
那么他们的区别在哪里？

Transformer解码器在文本处理中的应用

□ 回顾：解码器与编码器最核心的差异

□ 掩码自注意力机制

Masked QK^T V

$A = \{a_{i,j}\}$
 $a_{i,j} = 0 \text{ if } j > i$

$$\begin{bmatrix} a_{11} & 0 & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} = \begin{bmatrix} a_{11}v_1 \\ a_{21}v_1 + a_{22}v_2 \\ a_{31}v_1 + a_{32}v_2 + a_{33}v_3 \\ a_{41}v_1 + a_{42}v_2 + a_{43}v_3 + a_{44}v_4 \end{bmatrix}$$

输出中每个向量，
不依赖其后面向量

□ Transformer解码器中，每个Token只能获取其上文的信息，而编码器可以获取上下文信息

Transformer在文本处理中的应用

□ 不同模型的比较

	优势	局限
编码器	双向上下文信息编码	无法做生成任务
编码器+解码器	可以做理解与生成任务	架构复杂，工程实现存在限制
解码器	可以做理解与生成任务，易于拓展模型大小	单向编码，存在理解局限

□ 典型的Transformer文本处理模型

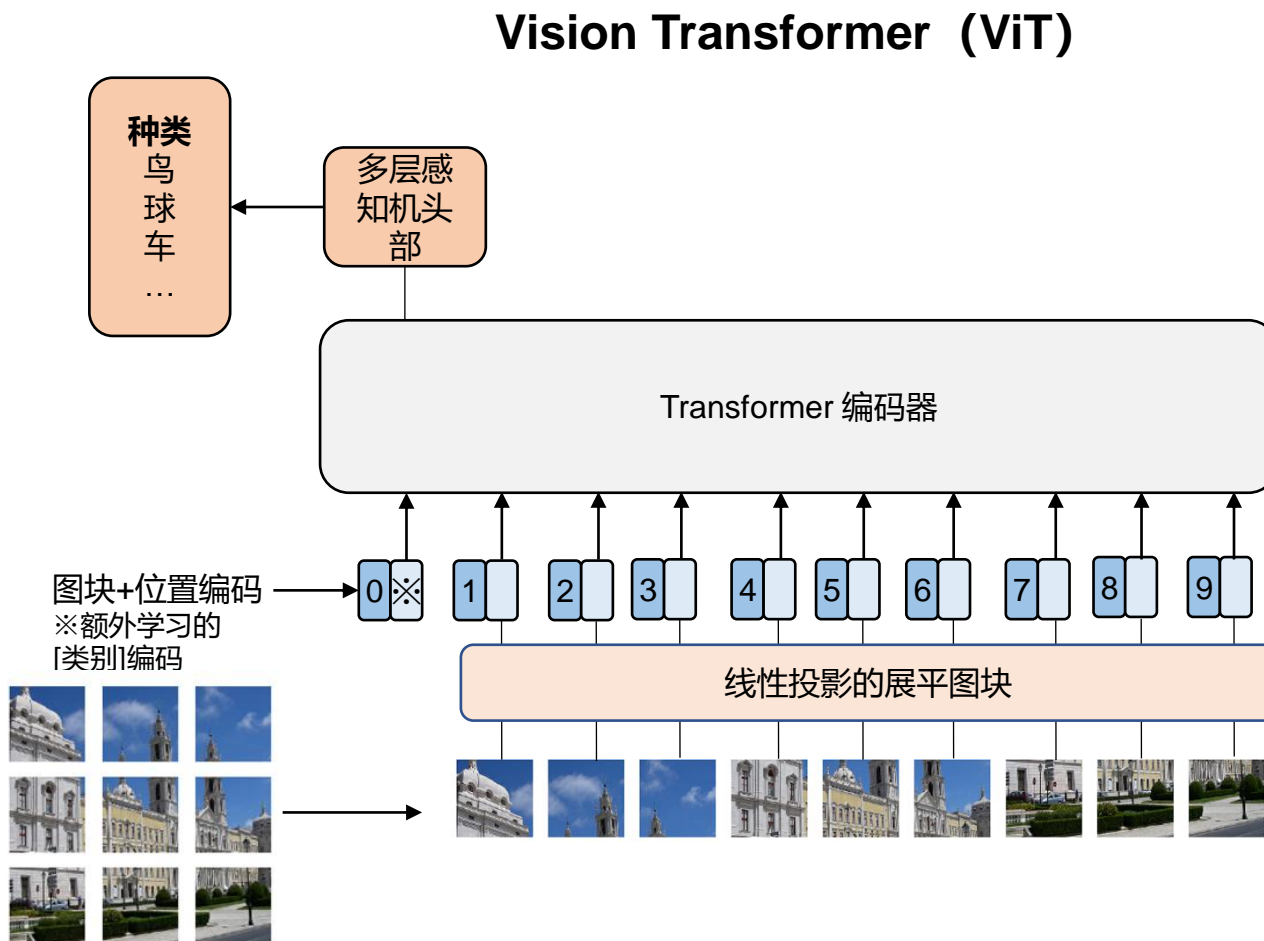
- 编码器模型：BERT等
- 编码器+解码器模型：T5等
- 解码器模型：GPT等

Transformer在图像处理中的应用

- 典型的模型是ViT
- 将patch映射到它们的表征（使用线性投影）

$$\mathbf{z}_i = \mathbf{W}\mathbf{x}_i + \mathbf{b}$$

- Patch表征与位置嵌入相结合

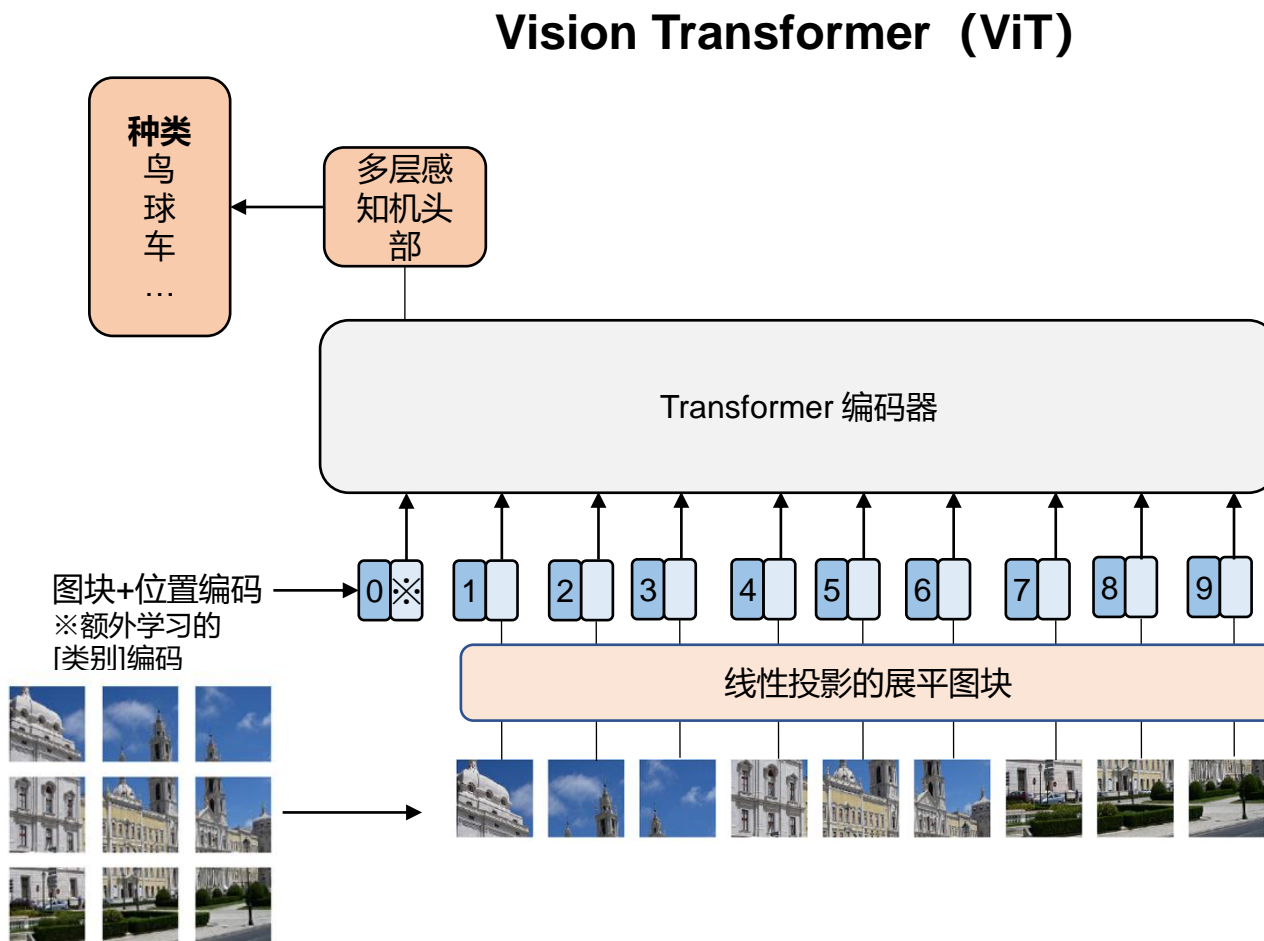


Transformer在图像处理中的应用

- 使用标准的Transformer 编码器计算表征并完成分类（与文本分类类似）

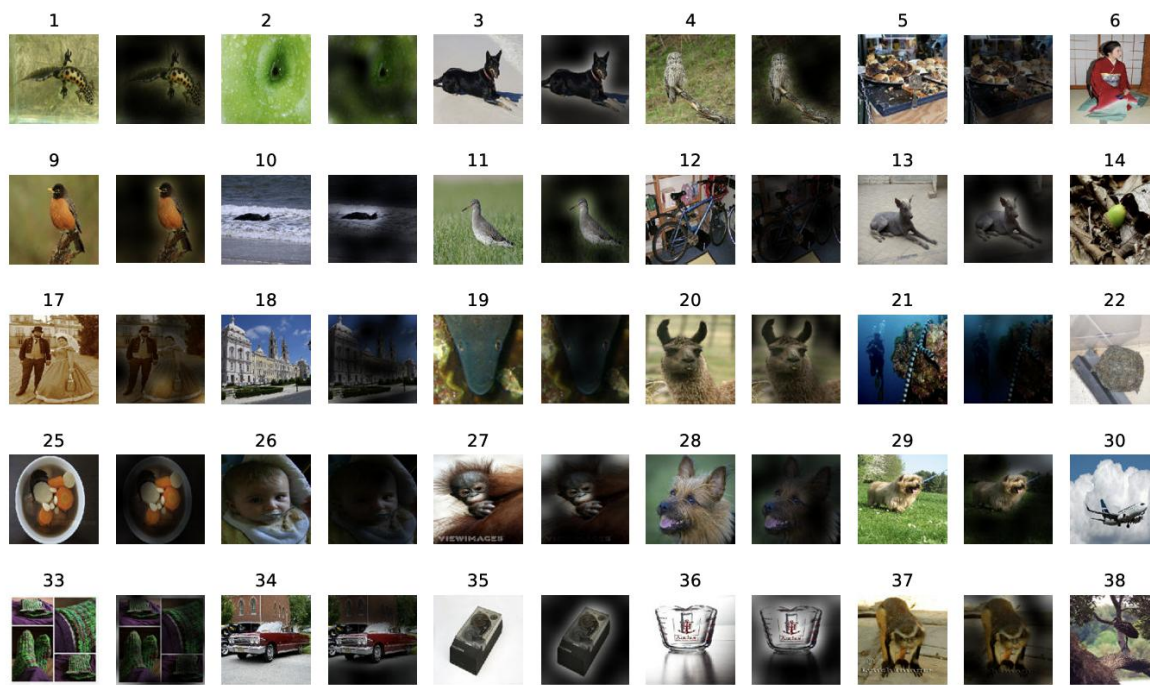
- 应用分为预训练和微调两部分

- 预训练通过在大规模图像分类数据集上训练完整模型
- 微调通过在下游任务继续训练模型



将Transformer应用于图像编码

□ 通过可视化模型注意力强弱，可以发现模型能够比较准确地高亮图像中重要的信息



□ Transformer 可以应用于各种不同类型数据的处理任务

- 文本、图像、音频...

□ Transformer在文本处理上的应用

- Transformer有多种文本处理模型变体（Transformer编码器、编码器+解码器、解码器模型）
- 不同变体有各自的优势以及局限性

□ Transformer在图像处理上的应用

- 以编码器为主的模型架构，可应用于图像分类等任务

01

重新理解Token

02

Transformer的典型应用

03

Transformer的跨模态推理

本课重点

将Transformer应用于跨模态编码

□ 挑战:

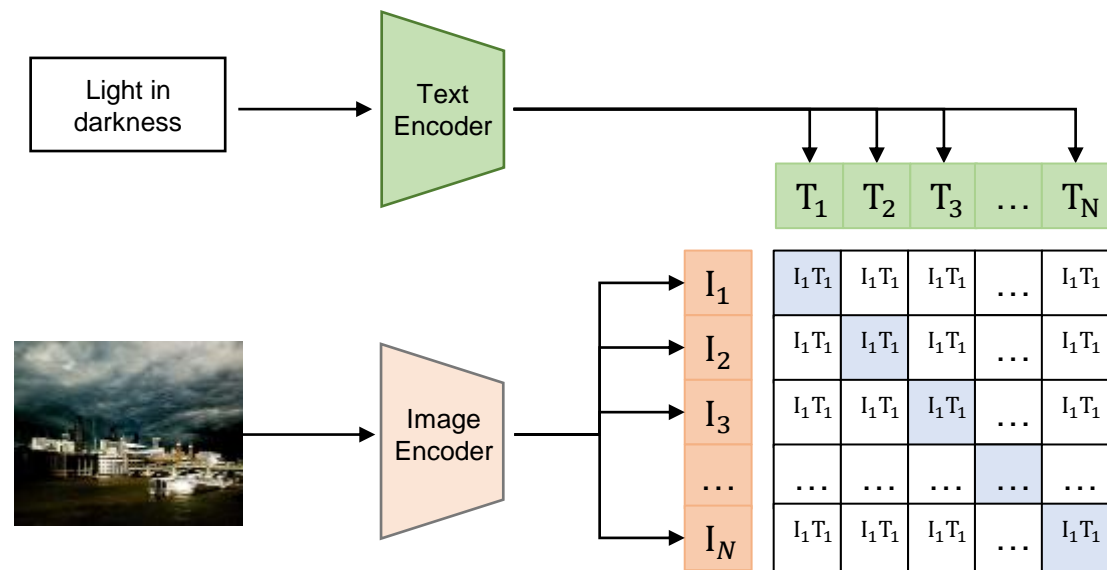
- 图像和文本具有不同的格式
- 视觉特征和文本特征不在同一空间
-

□ 解决方案

- 使用不同的模型将它们编码为矩阵表示
- 设计一种特定的机制来对齐不同的特征
-

Contrastive Language-Image Pre-training (CLIP)

- 输入：图像和文本
- 使用不同的 Transformer 编码器将输入编码为不同的表示形式
- 使用对比预训练来对齐多模态特征



□ 成对计算编码文本特征 (T_n) 和图像特征 (I_n) 之间的相似度，并得到比较矩阵

□ 对于 T_n ，相应的图像特征 I_n 作为正样本，其他图像特征为负样本，反之亦然

□ 最大化对角线上的值

	T_1	T_2	T_3	...	T_N
I_1	I_1T_1	I_1T_2	I_1T_3	...	I_1T_N
I_2	I_2T_1	I_2T_2	I_2T_3	...	I_2T_N
I_3	I_3T_1	I_3T_2	I_3T_3	...	I_3T_N
...
I_N	I_NT_1	I_NT_2	I_NT_3	...	I_NT_N

```
# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T)  #[n, d_t]

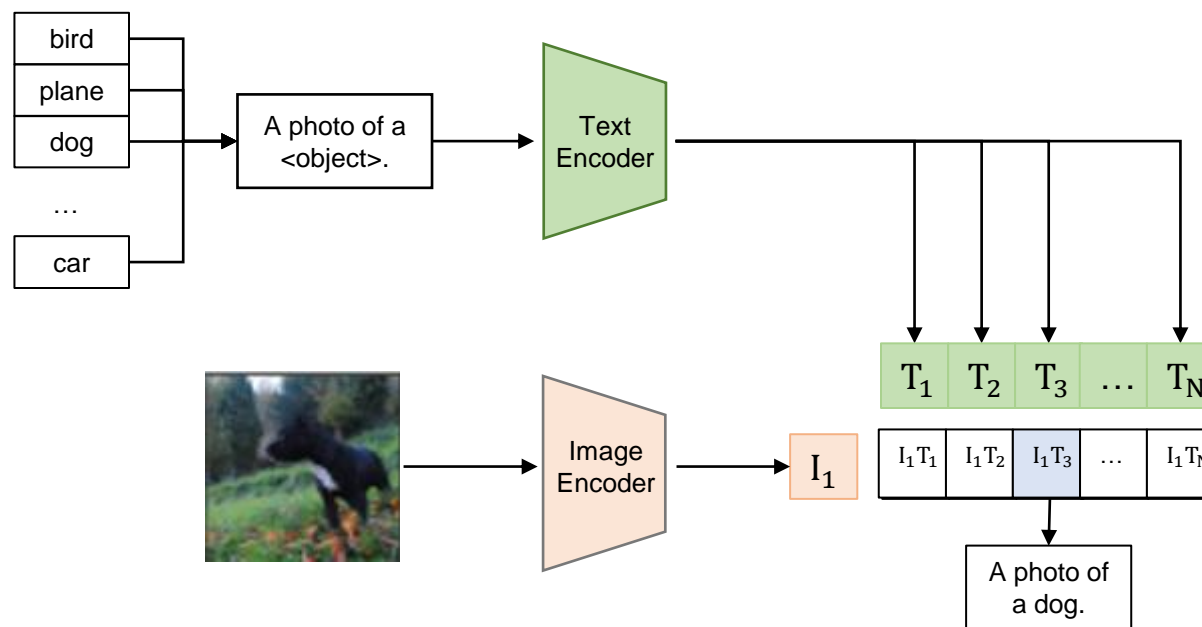
# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss   = (loss_i + loss_t)/2
```

















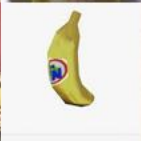







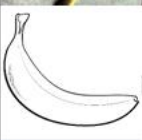


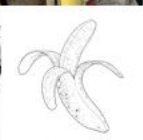



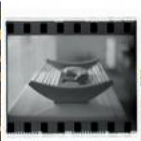




CLIP – 测试

- 学习到的文本编码器通过嵌入目标数据集类别的名称或描述来合成零样本线性分类器
- 每个类别形成一条文本
- 计算图像表征与每条文本的相似度
- 选取相似度最高的文本对应的类别作为图像类别



CLIP – 模型性能

- 右图展示了，将在 ImageNet 上面训练的 ResNet101 和 CLIP 模型应用于其他风格图片（以香蕉为例）的结果
- 展示了 CLIP 的鲁棒性

	Dataset Examples						ImageNet ResNet101	Zero-Shot CLIP	Δ Score
ImageNet							76.2	76.2	0%
ImageNetV2							64.3	70.1	+5.8%
ImageNet-R							37.7	88.9	+51.2%
ObjectNet							32.6	72.3	+39.7%
ImageNet Sketch							25.2	60.2	+35.0%
ImageNet-A							2.7	77.1	+74.4%

CLIP的局限

- CLIP 主要侧重于理解和匹配图像与文本，缺乏创建或修改内容的生成能力
- 针对专门任务对 CLIP 进行微调可能比较麻烦，并且可能无法在不同应用过程中产生最佳性能
- CLIP 对视觉和文本信息的整合虽然有效，但有时可能比较浅层，限制了深度多模态理解
- ...

Bootstrapping Language-Image Pre-training (BLIP)

□ BLIP 是基于 Transformer 编码器和解码器构建的

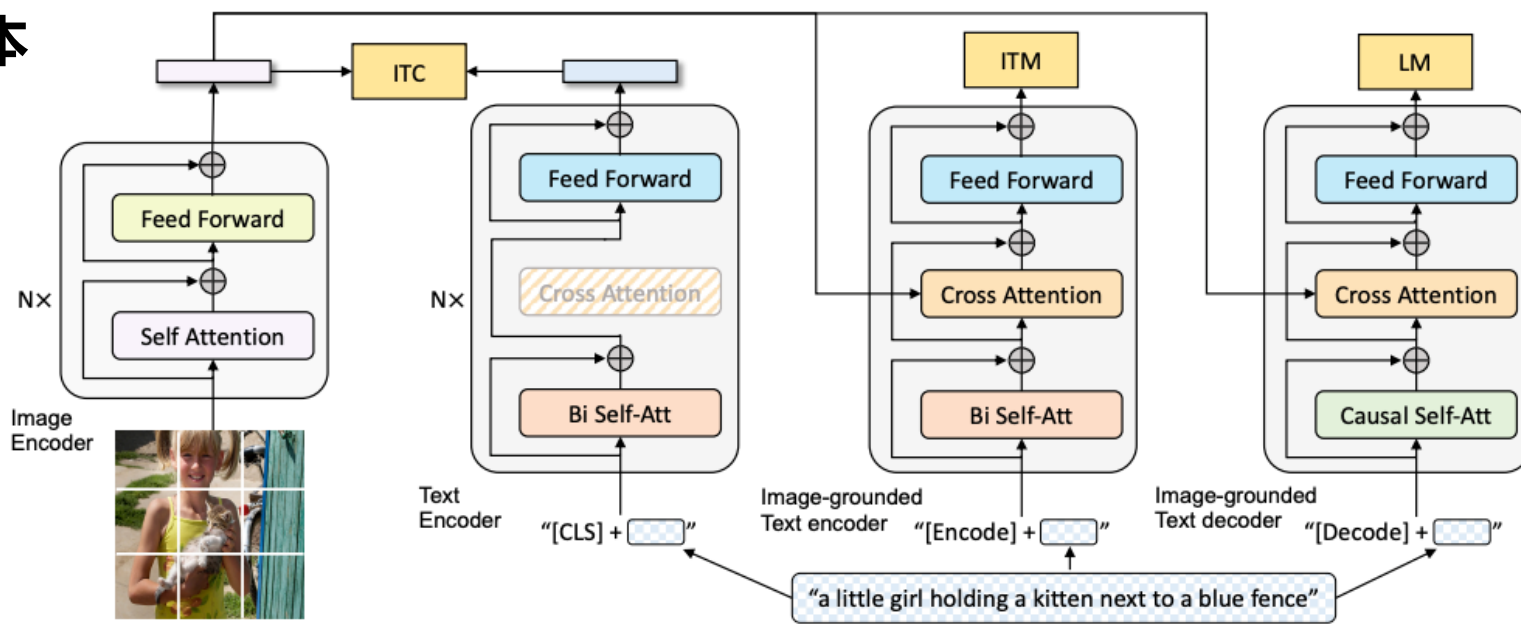
□ 它可以以三种功能运行：

□ 通过图文对比对齐图像和文本

□ 通过图文匹配区分图像和文本
是否匹配

□ 多模态语言建模

□ 训练时，三个任务损失
加权求和作为最终损失



□ BLIP能够生成高质量的图片描述



T_s : "an outdoor walkway on a grass covered hill"



T_s : "the car is driving past a small old building"



T_s : "a moon against the night sky with a black background"



T_s : "an outdoor walkway on a grass covered hill"



T_s : "a tiny white flower with a bee in it"



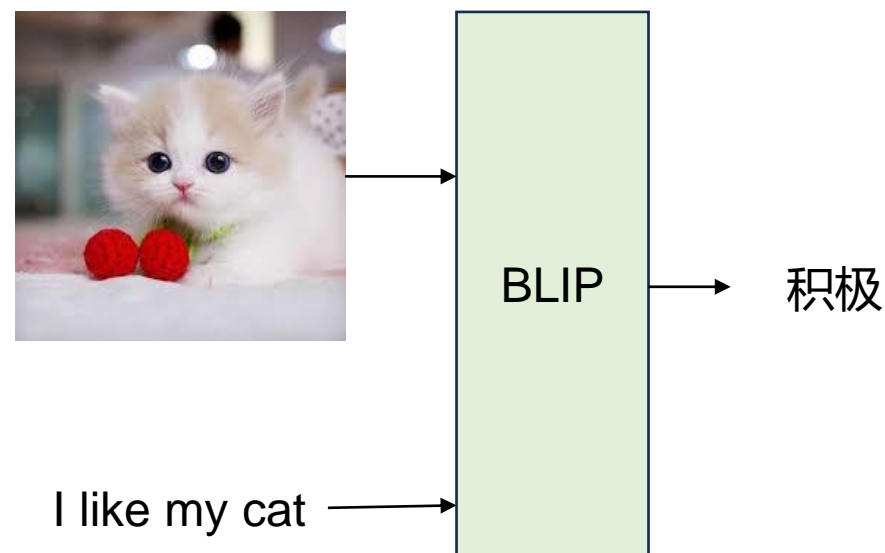
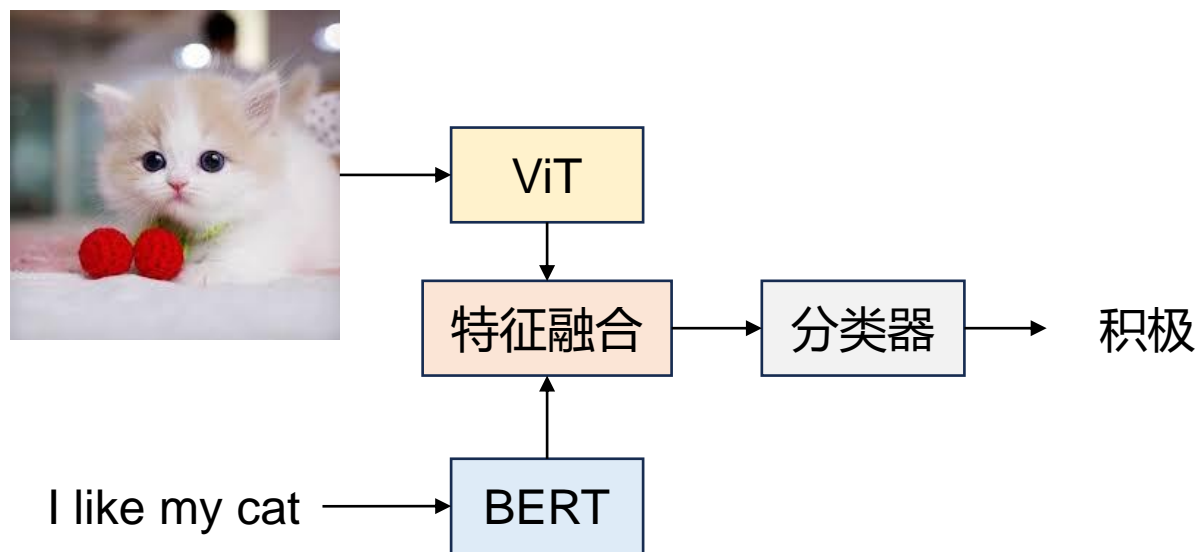
T_s : "some colorful trees that are on a hill in the mountains"

BLIP的优势

- **BLIP 结合了生成功能，使其能够生成描述性文字，并更有效地弥合理解与生成之间的差距**
- **BLIP 的开发旨在提供更大的灵活性和效率，以适应各种特定任务，通过更有效的预训练策略提高性能**
- **BLIP 旨在增强零样本学习能力，确保在更广泛的领域和数据集上实现更稳健的性能，而无需进行大量微调**

示例应用：多模态情感分析任务

- 输入图像以及文本，输出情感极性的分类结果
- 基本做法，使用Transformer模型架构的图像与文本编码器（或者直接使用多模态模型），编码图像与文本，并做特征融合，最后用于预测



示例应用：多模态情感分析任务

□ 效果展示



The calm water and rising sun remind me that every day holds a fresh start and quiet strength.



positive



Somewhere along the road, I lost my way—and now I'm not sure where I'm going, or if it even matters.



negative



My sink looks like this—cracked and forgotten. Kind of like me lately.



negative

- Transformer可以独立应用于图像以及文本处理，也可以同时处理不同模态的信息
- 代表性的多模态Transformer应用
 - CLIP
 - BLIP
- Transformer的典型多模态处理其本质是在学习更好的图文匹配

- **Dan Jurafsky and James H. Martin. Speech and Language Processing (3rd ed. draft); Chapter 10, 13, 14, 15. 2025**
 - 系统介绍了大模型，以及基于Transformer的应用（包括机器翻译、问答、对话系统、信息抽取等）
- **Kevin Clark, Urvashi Khandelwal, Omer Levy, Christopher D. Manning. “What Does BERT Look At? An Analysis of BERT’s Attention.” NAACL-HLT 2019.**
 - 提出基于注意力头的分析方法，揭示 BERT 中不同头关注分隔符标记、固定位置偏移或全局信息等模式；
 - 实验表明部分注意力头能高度对应句法依存关系和共指结构，展示了 self-attention 学到的语言学知识。

- Olga Kovaleva, Alexey Romanov, Anna Rogers, Anna Rumshisky. “Revealing the Dark Secrets of BERT.” EMNLP-IJCNLP 2019.
- 对 BERT 中每个注意力头进行定性和定量分析，发现注意力模式高度重复，模型存在过度参数化问题；
- 通过禁用部分注意力头反而提升性能，说明并非所有头都有用，揭示了 self-attention 的潜在缺陷。

课后思考

- 除了文本、图像的Token，你还能想出哪些类型的Token？
- CLIP与BLIP的共同点和差异有哪些？